Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH

Open Access

A split-federated learning and edge-cloud based efficient and privacy-preserving large-scale item recommendation model

Jiangcheng Qin[1,2], Xueyuan Zhang[1], Baisong Liu[1*] and Jiangbo Qian[1]

Abstract

The combination of federated learning and recommender system aims to solve the privacy problems of recommendation through keeping user data locally at the client device during the model training session. However, most existing approaches rely on user devices to fully compute the deep model designed for the large-scale item recommendation; therefore, imposing high calculation and communication overheads on resource-constrained user devices. Consequently, achieving efficient federated recommendations across ubiquitous mobile devices remains an open research problem. To this end, in this paper we propose an efficient and privacy-preserving federated learning framework which is based on the cloud-edge collaboration for large-scale item recommendation called SpFedRec. In our method, to reduce the computation and communication cost of the federated two-tower model, a split learning approach is applied to migrate the item model from participants' edge devices to the computationally powerful cloud side and compress item data while transmitting. Meanwhile, to enhance the feature representation, the Squeeze-and-Excitation network mechanism is used on the backbone model to optimize the perception of dominant features. Moreover, because the gradients transmitted contain private information about the user; therefore, we propose a multi-party circular secret-sharing chain based on secret sharing for better privacy protection. Extensive experiments using plausible assumptions on two real-world datasets demonstrate that our proposed method improves the average computation time and communication cost by 23% and 49%, respectively. Furthermore, the proposed model accomplishes comparable performance with other state-of-art federated recommendation models.

Keywords Federated learning, Recommendation system, Split learning, Cloud-edge collaboration, Computation and communication compression

Introduction

Nowadays, information technology including storage and computing is viewed as on-demand facilities, denoted as "X as a Provider." Large-scale data owners typically outsource their data to clouds to reduce the cost of data storage and management while enhancing privacy. Even so, data owners do not have direct control over their data stored on remote cloud servers, which raises concerns about their cloud services being illegally acquired or mistreated by cloud service providers (CSPs), particularly for sensitive data such as health records, official records, and email messages etc. Even though many CSPs argue that their cloud services include several security measures such as access control, firewalls, and penetration testing, concerns about the safety and confidentiality of cloud services remain major roadblocks to the widespread adoption of cloud computing [1]. Because of the foregoing, privacy-preserving suggestions have become

*Correspondence:
Baisong Liu
qjc@nbu.edu.cn
[1] College of Information Science and Engineering, Ningbo University, Ningbo 315211, Zhejiang, China
[2] Huzhou Key Laboratory of Sound Resource Data Mining and Intelligent Service Technology, Huzhou University, Huzhou 313000, Zhejiang, China

a research hotspot as user privacy [2] and data security [3] concerns have grown. Under strict regulatory requirements such as GDPR and CPRA [4] federated learning (FL) [5] is regarded as one of the most efficient privacy-preserving machine learning paradigms. Coordinated by a central server, participants of FL cooperatively train a model whereas keeping the training data locally. Since FL can mitigate the systematical privacy hazards of a central machine learning scenario, it fits the need for privacy protection and data safety in recommender systems (RS).

Ammad et al. [6] combined the FL and collaborative filtering and proposed FCF. FCF avoids the centralized collection of users' local training data to ensure privacy and achieves minimal performance loss. However, the gradients uploaded by the client to the server can still expose the user's confidentiality. Following FCF, Lin et al. [7] proposed a federated collaborative filtering recommendation (FedRec) model by explicit feedback. They proposed an obfuscated populating method: (1) Client randomly sampling a set of unrated items list and assigned with false rating values. (2) hybrid with real local historical data to compute the design gradients and upload it to the server, thus avoiding the server-aware local historical interaction. Subsequent research extends the FL-based RS into different application scenarios. Flanagen et al. [8] proposed the first federated multi-view matrix factorization (FED-MVMF) algorithm, which solves the cold-start problem by integrating information from multiple data sources.

Qi et al. [9] proposed the FedNewsRec framework by combining news recommendations with FL, they used a local differential privacy technique to add random noise to the gradients to protect the privacy of the user. Chen et al. proposed a privacy-preserving POI recommendation framework (PriRec) for the POI recommendation scenario, some of the public data including POI description and category information is stored on the server to reduce the communication cost on the client side. Luo et al. [10] proposed an FL-based social recommendation model that utilizes contrastive learning to eliminate heterogeneity of data distribution among users.

Additionally, although several approaches have been planned to enhance the efficiency of FL-based RS, they do not openly report the challenge within large-scale item recommendations. Muhammad et al. [11] proposed the FedFast algorithm, which combines the GMF (Generalized Matrix Factorization) with FL. The FedFast accelerates the convergence of FedGMF by using client-side sampling and enabling parameter sharing among clustering users, which can complete the training with a small number of selected users, circumventing the delay caused by some inefficient clients. Reisizadeh et al. [12] also selected only some clients for model training and utilized low-precision quantization methods to compress the model parameters. Some studies [6, 13, 14] reduce the communication cost between the client and the server by performing multiple iterations of training locally on the client side. However, they do not decrease the absolute workload on a client device.

Qin et al. [15] proposed a novel privacy-preserving recommender scheme framework based on federated learning, which enables the server to build a matching model using explicit feedback from a subset of users but causes privacy exposure. Utmost present works undertake that all client devices are immediately accessible and have sufficient power to contribute in FL training. However, in large-scale item recommendation, FL-based recommendation still faces the challenge of communication and computation limitations from the client device. (1) The RSs in the industry are designed for massive data analysis with large and complex models [16]. Many consumer-grade mobile devices cannot handle the local training task, thus leading to model training failure. (2) large-scale RS often has millions of items to be filtered, and a centralized RS can alleviate this part of the work with the matching mechanism under strict latency requirements. However, in the FL recommendation scenario, since the server cannot directly access the users' data, the vast volume of raw item data has to be transmitted to the user's device, producing considerable communication and calculation overhead. In summary, a major challenge for implementing large-scale item federated recommendations is: How to efficiently compress the computation and communication overhead under privacy protection and enable resource-limited ubiquitous mobile devices to satisfy the performance requirement of federated large-scale item recommendations.

In this study, to overcome the challenge of the FL client-side computation and communication compression while ensuring privacy, we propose a cloud-edge collaboration-based Split Federated Recommendation framework (SpFedRec). The SpFedRec leverages the two-tower recommendation model's unique structure and separates the two independent sub-networks of user and item models based on split learning. The RS server (cloud side) is responsible for maintaining the training and updating of the item model. In contrast, the client device (edge side) maintains the training and update of the user model. Meanwhile, the RS server can provide the required low-dimensional item embeddings to the clients involved in FL training or online prediction, avoiding large-scale item data transfer. With SpFedRec, the client device can complete the similarity matching of cached user embedding and server-provided items embedding for online inferring by lightweight computation. To further improve feature utilization and thus enhance recommendation

performance in a decentralized state of data and models, we utilize the SENet mechanism to learn feature importance dynamically. Currently, many privacy-preserving techniques are introduced into FL, such as secure multi-party computation (MPC) [17], differential privacy (DP) [18], and homomorphic encryption (HE) [19] as the local gradient updates may reveal private user information [20]. Because DP introduces random noises into the global model, affecting its accuracy, and HE requires too much extra computation [21, 22], we opt for MPC, which can offer provable privacy guarantees for multi-party gradient aggregation at a lower computational cost.

This paper concentrates on two useful compression objectives for FL-based RS in particular. (1) Good Communications Compression [23] decreases communication overhead when FL training and online inferring devices are used. (2) Local Computation Reduction [19, 24] decreases computation workload when devices participate in FL training and online inference. The two-tower model is commonly used to pre-filter huge items in large-scale item RS [25–27]. The two-tower model is distinguished by the fact that the user and item features are modeled as separate sub-networks and cached individually [28]. When performing online inferring, only the cached embeddings of users and products must be matched for similarity. It is possible to separate these two independent sub-models, and their training and online inference can be conducted in separate entities. We highlight our research contributions as given below:

- We propose SpFedRec for efficient and privacy-preserving large-scale item recommendation. Apart from preventing users from sharing their private local data, we adopt a practical cloud-edge collaborative split learning approach to enable SpFedRec significantly reduce the computation and communication overhead on resource-constrained participants. SpFedRec can alleviate the critical communication and computation bottleneck for existing FL-based recommenders.
- To defend against the semi-honest server and malicious participants who may infer user data from intermediate information generated by the FL training process, we propose a multi-party circular secret-sharing chain to encrypt the gradient information. We further introduce the obfuscated item request strategy to conceal the label information.
- We conduct extensive experiments on two real-world datasets to verify the effectiveness and efficiency of our proposed models, and results show that SpFedRec outperforms the state-of-art baselines. At the same time, it achieves effective communication and computation reduction.

The rest of this paper is organized as follows. In Sect. "RQ2: Is the computation and communication efficiency of SpFedRec significantly improved compared to the baseline models in the naïve FL setting?", we offer a brief overview of the related work and a few existing recommendation systems. Sect. "RQ3: How do hyper-parameter settings and affect the performance and efficiency of SpFedRec?" introduces the preliminary knowledge and discusses several research methodologies. In Sect. "RQ4: How does our proposed privacy protection mechanism influence the performance and efficiency of SpFedRec?", we delibrate our proposed framework called SpFedRec. In Sect. 5, we present our experimental settings and the obtained results. Finally, we conclude our paper in Sect. 6 and suggest few directions for further research and investigation.

Related work

These days, mobile edge computing as a technological innovation is gaining traction [29], particularly the issue of MEC computing offloading. The majority of current research uses delay, energy usage, a weighted sum of energy usage, and delay as the computing offloading effectiveness indicators. To achieve the optimum solution task scheduling technique for delay-based computational offloading, Liu et al. [30] developed an effective one-dimensional evolutionary algorithm to tackle the challenge of power-constrained delay reduction. Ning et al. [31] suggested an adaptive heuristic allocation of resources for dynamic offloading judgments, taking into account the cooperation of mobile edge computing and cloud computing. Wu et al. [32] developed a computing offloading framework based on nonorthogonal multiple access (NOMA) innovations to reduce the total finish time of all mobile terminal jobs. Zhang et al. [33] combined computing offloading, information caching, and allocation of resources into a single model and created an asymmetric tree structure to reduce the total delay usage of computing tasks. On other hand, communication and computation represent a significant bottleneck in enabling practical applications of cross-device federated learning [4]. The bandwidth of the mobile device's uplink channel is much narrower than the bandwidth of the connection inside the server, and the mobile network is expensive and unreliable. Meanwhile, consumer-grade processors in mobile devices are far less capable than the GPU on deep-learning servers. In deep learning models, the training session on client devices will consume a significant amount of time and power, which is unacceptable to users.

Additionally, most existing FL-based recommendation studies assume that all client devices are immediately available and have sufficient capacity to undertake

the computation of the entire recommendation model [6], which is tough for a mobile device with limited resources, especially since industrial RS are in huge size and have massive amounts of item data. Although several approaches have been planned to enhance the efficiency of FL-based RS, they do not openly address the challenge within large-scale item recommendations. There are some studies on the problem of minimizing total mobile energy usage under the restriction of computational delay. Muhammad et al. [11] proposed the FedFast algorithm, which combines the GMF (Generalized Matrix Factorization) with FL. The FedFast accelerates the convergence of FedGMF by using client-side sampling and enabling parameter sharing among clustering users, which can complete the training with a small number of selected users, circumventing the delay caused by some inefficient clients. Reisizadeh et al. [12] also selected only some clients for model training and utilized low-precision quantization methods to compress the model parameters. Some studies [6, 13, 14] reduce the communication cost between the client and the server by performing multiple iterations of training locally on the client side. However, they do not reduce the absolute workload on a client device. Qin et al. [15] proposed a novel privacy-preserving recommender system framework based on federated learning, which enables the server to build a matching model using explicit feedback from a subset of users but causes privacy exposure.

Even so, the above said privacy and scheme load problems were planned for cloud computing or mobile cloud computing situations. This restriction has motivated our study to discover a solution for the FL client-side computation and communication compaction challenging task while maintaining privacy, and we suggest a split federated recommendation framework based on cloud-edge collaboration (SpFedRec). It proposes the suggested framework for efficient and privacy-preserving large-scale item recommendation. Apart from preventing users from sharing their private local data, we adopt a practical cloud-edge collaborative split learning approach "SpFedRec" that significantly reduces the computation and communication overheads on resource-constrained participants. Moreover, we believe that our proposed SpFedRec model could improve the critical communication and computation bottleneck for existing FL-based recommenders.

Preliminaries

This section introduces some basics and preliminary knowledge about our backbone model, split-learning, and secure multi-party computing.

Backbone model

The standard setting of the two-tower model is 3 layers, fully linked neural network for both the item and user model, i.e., the Deep Semantic Similarity Model (DSSM) [28]. The most important characteristic of the two-tower design is that the user and item model are two independent sub-networks, as shown in Fig. 2(a). The two sub-networks can be cached separately so that the online prediction only requires similarity matchings in memory, which is friendly to the industrial world that values computation efficiency. The two-tower model recommendation structure is extensively utilized in advertising (Facebook) [34], information retrieval (Google) [30], and recommendation (YouTube) [35].

Specifically, the goal of utilizing the two-tower model for large-scale item recommendation is to retrieve a subset of (hundreds of) items for a given user for subsequent ranking. There are two different embedding functions $u(x, \theta_u)$ and $v(y, \theta_v)$, mapping both user $\{x_i\}_{i=1}^{N}$ and item $\{y_j\}_j^M$ to a k-dimensional vector space. The output is the similarity matching of the two embeddings by Eq. 1. The object of training is to learn the parameters θ_u and θ_v based on the training data $D := \{(x_i, y_j, r)\}_{i=1}^{T}$, where the r is the label that indicates if the user interacts with an item.

$$Sim(x, y) = \langle u(x, \theta_u), v(y, \theta_v) \rangle \tag{1}$$

Given a user x, the probability of getting item y from corpus M can be considered as a multi-classification problem, the SoftMax function as illustrate in Eq. 2. And the loss function is a log-likelihood loss as illustrate in Eq. 3, where the T denote the batch size.

$$P(y|x, \theta_u, \theta_v) = \frac{e^{Sim(x,y)}}{\sum_{j=1}^{M} e^{Sim(x,y_j)}} \tag{2}$$

$$L_T(\theta_u, \theta_v) := -\frac{1}{T} \sum_{i \in |T|} r_i \log(P(y|x, \theta_u, \theta_v)) \tag{3}$$

Split learning

Split learning is a deep learning paradigm based on server and client collaboration [36]. Unlike the FL setups that emphasis on data and model distribution, the core idea of split learning is to divide the training and inference process of a deep model by layers and execute them in different entities [37].

The Cloud-Edge collaborative split learning in the U-shape configuration is a privacy-preserving variation, as shown in Fig. 1. The Edge-side includes various mobile devices, such as tablets, mobile phones, and laptops) with limited computational resources but various

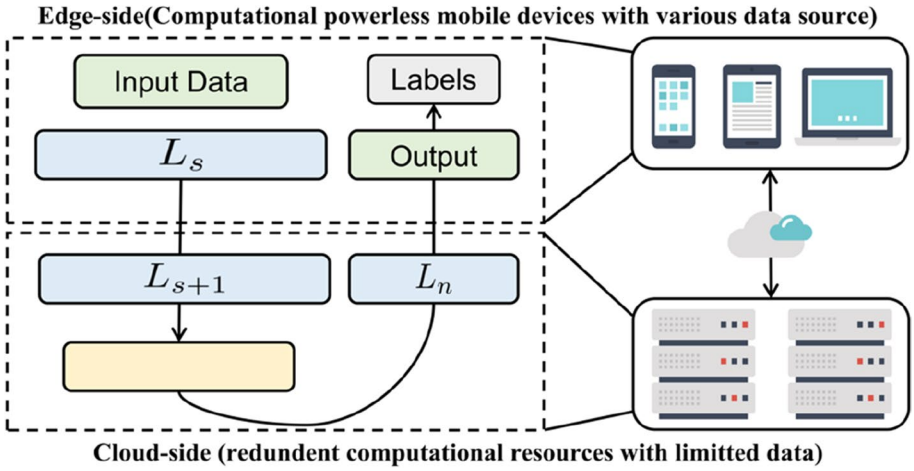

Fig. 1 Cloud-Edge Collaborative Split learning in U-Shape configuration [4]. The L_s denote the edge-side cut layer and the L_n represent the cloud-side output layer

data sources. Edge devices train deep models based on local data to provide better information services to edge-side users. However, due to the limitation of computational power, it cannot complete the local training but requests the cloud server for co-computation. The cloud side provides redundant computing resources as a service to the edge-side devices. Both client-side input data and labels are not transferred to the cloud to ensure edge-side data privacy [38].

Formally, there is a deep model training task including n hidden layers L_i, $i \in [1, n]$. An edge-device needs to train the deep model based on local data D but with insufficient computational capacity. The edge-device begins the model training with forward propagation until it reaches the split layer L_s where $s < n$. The feature mapping of L_s is then encrypted and sent to the cloud for further propagation $L_{s+1} \rightarrow L_n$. The output of the L_n will be sent back to the client for loss computation. The output layer's encrypted gradients will be directed to the cloud and backpropagation is from the L_n to the split layer in a opposite path. The gradients of L_s will be returned to the edge-side, and the rest of backpropagation will be finished. The above steps go through multiple iterations until the model converges.

Based on the cloud-edge collaborative split learning, we can split the hidden layer of a deep learning-based FL training task from a resource-constrained edge device. The cloud server that provides the computation service will handle the split workload, improving the efficiency of FL training and online inferring. To our knowledge, we proposed the first split learning-based federated recommendation framework to improve the computation efficiency and model training durations.

Secure multiparty computing

MPC is an algorithmic protocol based on cryptography to achieve privacy computing, and it is a comprehensive application of a variety of cryptography-based tools [17]. MPC ensures that multiple participants $\{\surd^i\}_{i\in[1,n]}$ can jointly accomplish a specific confidential computation task $f(x_1, x_2, \ldots, x_n)$ without revealing their private data x_i, each party does not receive any additional information other than its inputs and outputs in the end of the task. MPC has been widely used in FL based RS for intermedia parameter secure aggregation, such as PriRec [39] and SeSoRec [40]. Here, we present the Addition MPC protocol by secret sharing that we will use in SpFedRec for secure gradient and BP error signal aggregation.

Denote multiple participants $\{\surd^i\}_{i\in[1,n]}$ wish to compute the sum of their data x_i without revealing their secrets. Each party $\surd^i$ generates n random data $x_i^1, x_i^2, \ldots, x_i^n$ such that they satisfy Eq. 4. Then, send the split secrets to the corresponding party to enable the computation as Eq. 5. Finally, the computation results of Eq. 5 are aggregated to obtain the sum of all parties' data as Eq. 6. With the secret sharing-based MPC protocol, none of the members can access others' private data but can obtain the final sum result. The secrecy of the members' data cannot be broken when the total number of adversarial members is less than $(n-1)$.

$$x_i^1 + x_i^2 + \cdots + x_i^n = x_i, i \in [1, n] \tag{4}$$

$$x_i^* = x_1^i + x_2^i + \cdots + x_n^i, i \in [1, n] \tag{5}$$

$$\mathrm{SUM}(x_1, x_2, \ldots, x_n) = \sum_{i=0}^{n} x_i^* \tag{6}$$

The SpFedRec framework

This section presents the proposed SpFedRec model and its design in detail. We first discuss the problem definition and then systematically explain the overall framework, training steps, and online forecast. We conclude the section with an argument of backbone model variations. We list important notations used throughout the paper in Table 1.

Problem definition

Suppose that the recommendation server has a large-scale item data $V = \{v_1, v_2, \ldots\}$ and a list of clients $C = \{c_1, c_2, \ldots\}$. Each client c_i is a user of recommendation service, his/her personalized feature u_i and interaction history are combined as $D_i = \left\{u_i, \left(v_j, r\right)\right\}_{j=1}^{M}$ stored locally on his/her device. The $u_i \in \mathcal{X}$ and $v_j \in \mathcal{Y}$ are both feature representations of the respective feature space, and r is the interaction label. Let $M_u(u, \theta_u)$ and $M_v(v, \theta_v)$ be the embedding functions for user and item features representations, that can map user and item features to a uniformed low-dimensional dense vector $\widetilde{u}$ and $\widetilde{v}$ for similarity matching, respectively.

The goal of SpFedRec is to collaborate clients from C to jointly learn model parameters θ_u and θ_v in a computationally and communicationally efficient way, while the training data D_i is distributed among the clients' resource-limited mobile device.

Table 1 List of mathematical notations

Notations	Description
u_i, v_j	User i and item j
$M_u(u, \theta_u), M_v(v, \theta_v)$	User and item model
$\widetilde{u}_i, \widetilde{v}_j$	Output of user and item model
t	Training round
θ_u^0, θ_v^0	Initial user and item model's parameters
θ_u^t, θ_v^t	User and item model's parameters at training round t
C^t	Random selected subset of clients at training round t
K	Number of selected clients in each training round
B_k^t	Batch of training set for client k at training round t
V_k^-, V_k^+	Subset of non-clicked and clicked items for client k
$\nabla\theta_u^t, \nabla\theta_v^t$	The gradients of User and item model at training round t
P	Negative sampling rate
η_u, η_v	Learning rate for user and item model update
β_1, β_2, τ	Hyperparameters for FedAdam optimizer
σ, δ	Activation functions
r	Reduction ratio

Framework overview

The overall framework of the proposed SpFedRec model is shown in Fig. 2(c). The Rec Server maintains the large-scale item data V, and N different client device c_i maintains the local user data D_i.

In contrast to the naïve setting of the federated two-tower model shown in Fig. 2(b), the computation of the two-tower model is split into two independent sub-models, the user model and the item model. The edge side client device is responsible for updating and computing the user model, while the powerful Rec Server on the cloud side maintains the item model. The model compression reduces the amount of communication when clients download and upload model parameters and improves the computation efficiency for the model update.

The large-scale item data will be mapped into low-dimensional dense vectors through the server-side item model and provided to the clients. In fact, the lightweight item embedding will further reduce the clients' computation and communication overheads during the training and online inferring phases.

Each client mixes randomly selected negative items with actual samples when sending item embedding requests to the Rec Server (Obfuscated Items Request), ensuring that the Rec Server cannot obtain the natural interaction history of individual users. A secure aggregation server (SAS) is applied to carry out secure aggregation of gradients and back-propagation (BP) error signal by circular secret-sharing chain (Sec 3.3.2). The privacy protection setting is able to against semi-honest adversaries ($< N - 1$) from recover the user's original training data.

In the following subsections, we are going to elaborate on the training process which is embedded to the proposed SpFedRec model.

Split federated learning

At initial stage ($t = 0$), the Rec Server initializes the two independent sub-networks as $M_u\left(u, \theta_u^0\right)$ and $M_v\left(v, \theta_v^0\right)$. Then randomly sample K available clients $C^0 = \left\{c_k^0\right\}_{k=1}^{K}$, distribute the initialized user model $M_u\left(u, \theta_u^0\right)$ to each sampled client for local user model training and SAS for global user model update.

Client-side local training

The first step of local client-side training is requesting items embedding from Rec Server. However, sending the raw item list to the server may leak private user information. Thus, we propose obfuscated item request, a client c_t^k randomly samples a subset of items V_k^- as negative samples

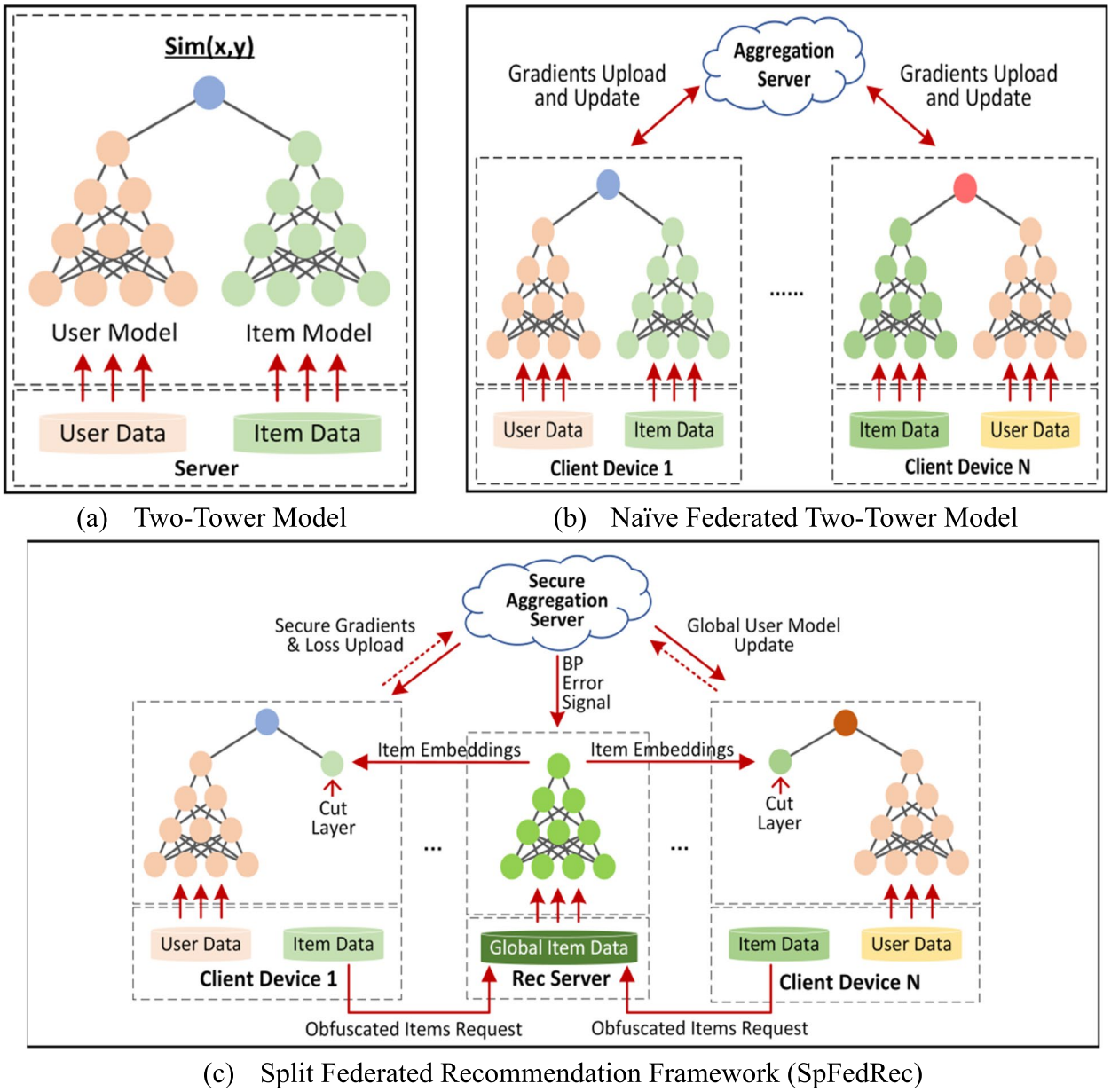

Fig. 2 Illustration of two-tower recommendation model in **a** centralized setting, **b** naïve federated learning setting, and **c** split federated learning framework (SpFedRec)

and combines it with positive samples V_k^+. Denoting the obfuscated item request list as $V_k^t = V_k^+ \cup V_k^- = \{v_{kj}^t\}_{j=1}^T$, where the $T = \left|V_k^t\right|$ as the local batch size. Once received the obfuscated item request, the Rec Server will compute the item embedding by Eq. 7 and returns items embeddings list $\widetilde{V_k^t} = \{\widetilde{v_{kj}^t}\}_{j=1}^T$ to the client device.

$$\widetilde{v_{kj}^t} = M_v\left(v_{kj}^t, \theta_v^t\right), j \in [1, T] \tag{7}$$

While pending the items embeddings from Rec Server, the client c_t^k compute the user embedding u_t^k by Eq. 8. Combined the user embedding with received items embedding as local training set $B_k^t = \{\widetilde{u_t^k}, \left(\widetilde{v_k^t}, r_j\right)\}_{j=1}^T$. Then, the client computes the similarity matching of each training set pairs by Eq. 9 and obtains the average loss by Eq. 10. Denote the local gradients of the user model as $\nabla\theta_{uk}^t$, which are computed as Eq. 11. The local gradients of the user model will be securely uploaded to the SAS for global aggregation by circular secret-sharing chain.

$$\widetilde{u_k^t} = M_u\left(u_k, \theta_u^t\right) \tag{8}$$

$$\mathrm{sim}\left(u_k^t, v_{kj}^t\right) = \frac{\widetilde{u_k^t} \cdot \widetilde{v_{kj}^t}}{\|\widetilde{u_k^t}\| \|\widetilde{v_{kj}^t}\|} \tag{9}$$

$$P\left(v_{kj}^t | u_k^t, \theta_u^t, \theta_v^t\right) = \frac{e^{\mathrm{sim}\left(u_k^t, v_{kj}^t\right)}}{\sum_{j=1}^{T} e^{\mathrm{sim}\left(u_k^t, v_{kj}^t\right)}} \tag{10}$$

$$L_{B_k}^t = -\frac{1}{T}\sum_{j=1}^{T} r_j \cdot \log\left(P\left(v_{kj}^t | u_k^t, \theta_u^t, \theta_v^t\right)\right) \tag{11}$$

$$\nabla\theta_{ku}^t = \frac{\partial L_{B_k}^t}{\partial \theta_u^t} \tag{12}$$

Circular secret-sharing chain

The circular secret-sharing of local user model gradients and loss starts with secret splitting. Each client c_k^t split the gradients of local user model $\nabla\theta_{ku}^t$ as two random matrixes $\left[\nabla\theta_{ku}^t\right]_1$ and $\left[\nabla\theta_{ku}^t\right]_2$ that they satisfy Eq. 13. Split the loss in the same way to satisfy Eq. 14. Then, each client c_k^t delivers part of the secret $\left[\nabla\theta_{ku}^t\right]_2$ and $\left[L_{B_k}^t\right]_2$ to the next client c_{k+1}^t. The last client, c_K^t, will pass his/her part of the secret to the first client c_1^t. The process is under SAS's coordination and forming a circular secret-sharing chain. After received the part of the secret from another client, sum it with the reserved part and obtain a mixed-secrets $\nabla\theta_{ku}^{t\,*}$ and $L_{B_k}^{t\,*}$ by Eq. 15. Finally, during the las step each client uploads the mixed-secrets to the SAS for global model update.

$$\left[\nabla\theta_{ku}^t\right]_1 + \left[\nabla\theta_{ku}^t\right]_2 = \nabla\theta_{ku}^t \tag{13}$$

$$\left[L_{B_k}^t\right]_1 + \left[L_{B_k}^t\right]_2 = L_{B_k}^t \tag{14}$$

$$\nabla\theta_{ku}^{t\,*} = \left[\nabla\theta_{ku}^t\right]_1 + \left[\nabla\theta_{(k-1)u}^t\right]_2, L_{B_k}^{t\,*} = \left[L_{B_k}^t\right]_1 + \left[L_{B_{(k-1)}}^t\right]_2 \tag{15}$$

Global model update

The SAS is responsible for the global user model update. Denote the average gradients of the user model as $\overline{\nabla\theta_u^t}$ and the average loss of model as $\overline{L^t}$, which are computed as Eq. 16.

$$\overline{\nabla\theta_u^t} = \frac{1}{K}\sum_{k=1}^{K} \nabla\theta_{ku}^{t\,*}, \overline{L^t} = \frac{1}{K}\sum_{k=1}^{K} L_{B_k}^{t\,*} \tag{16}$$

The SAS will send the average loss of the model to the Rec Server. Then, SAS will update the global user model by the average gradients of the user model through the FedAdam optimizer [41] as Eq. 17, where the η is the learning rate, β_1, β_2, and τ are hyper-parameters. The updated global user model parameters θ_u^{t+1} are distributed to all clients served for the next round of training.

$$\begin{aligned} m_u^t &= \beta_1 m_u^{t-1} + (1-\beta_1)\overline{\nabla\theta_u^t} \\ \mu_u^t &= \beta_2 \mu_{u-1}^t + (1-\beta_2)\left(\overline{\nabla\theta_u^t}\right)^2 \\ \theta_u^{t+1} &= \theta_u^t + \eta \frac{m_u^t}{\sqrt{\mu_u^t}+\tau} \end{aligned} \tag{17}$$

After received the average loss of the model from SAS, the Rec Server compute the average gradients of item model $\overline{\nabla\theta_v^t}$ by Eq. 18.

$$\overline{\nabla\theta_v^t} = \frac{\partial \overline{L^t}}{\partial \theta_v^t} \tag{18}$$

Similarly, we use the FedAdam optimizer to update the universal entry model as Eq. 19. The efficient global entry model $M_v\left(v, \theta_v^{t+1}\right)$ will provide obfuscated items embedding for next training round.

$$\begin{aligned} m_v^t &= \beta_1 m_v^{t-1} + (1-\beta_1)\overline{\nabla\theta_v^t} \\ \mu_v^t &= \beta_2 \mu_{v-1}^t + (1-\beta_2)\left(\overline{\nabla\theta_v^t}\right)^2 \\ \theta_v^{t+1} &= \theta_v^t + \eta \frac{m_v^t}{\sqrt{\mu_v^t}+\tau} \end{aligned} \tag{19}$$

Finally, update the training iteration as $t = t + 1$ until the model convergence.

Online inferring

After finishing the model's training, the Rec Server will generate item embeddings based on the present global entry design. Denote the item embedding as $\widetilde{V}$ and allocate it to all client devices. When performing online inferring, a client device only needs to compute the resemblance matching of locally cached user embedding $\widetilde{u}$ and each item embedding within $\widetilde{V}$. Transferring low-dimensional dense item embedding will be much less overhead to the client's communication than transferring the original item data.

Model variations

The backbone model is a three-layer, fully linked neural network for both item and user model. However, the standard setting makes insufficient use of contralateral feature information due to the separate training of these two towers. Inspired by [42], we utilize SENet mechanism to improve the perception of core features in the hidden layer, thus enhance their information representation in the similarly matching phase.

The SENet enhanced model architecture for both user and item model is illustrated in Fig. 3. The feature embedding layer can embed the sparse feature of user and item data into dense real-value vector $D = [d_1, d_2, \ldots, d_f]$,

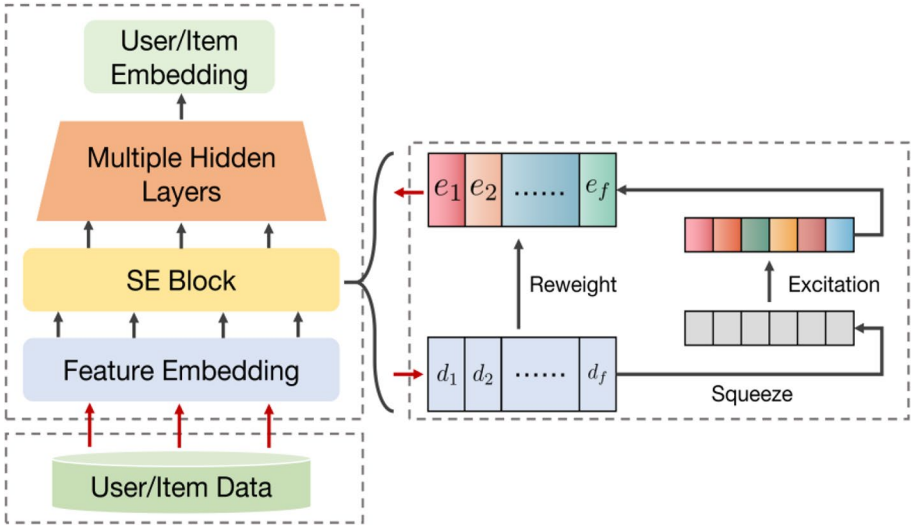

Fig. 3 The SENet enhanced model architecture for user and item model. The SE Block including three phases of feature embedding manipulation, Squeeze, Excitation and Reweight

where f denotes the number of features, $d_i \in R^h$ and $D \in R^{f \times h}$ where h is the embedding size. The feature embedding will be fed into the SE Block, which includes the three steps of feature embedding manipulation highlighted in Fig. 4.

Squeeze

This step compressing the feature embedding in the spatial dimension, apply average pooling to turn two-dimensional feature embedding D into a real number vector $A = [a_1, a_2, \ldots, a_f]$, where a_i is a scalar value and compute by Eq. 20.

$$a_i = F_{sq}(d_i)\frac{1}{h} = \sum_{j=1}^{h} d_{i,j} \tag{20}$$

Excitation

This step captures the weight of feature embedding and represent it as a weight vector S. The weight of feature embedding can be calculated as Eq. 21. The σ and δ are activation functions. $W_1 \in R^{f \times \frac{f}{r}}$, and $W_2 \in \frac{f}{r} \times f$ represent two fully connected layers, where the r is reduction ratio to control capacity and computational cost [43].

$$s_i = F_{ex}(a_i) = \sigma(W_2 \cdot \delta(W_1 \cdot a_i)) \tag{21}$$

Reweight

The final does field-wise multiplication of D and S and outputs the embedding $E = [e_1, e_2, \ldots e_f]$, and $e_i \in R$, which can be calculated as Eq. 22. Since the value range of a is in [0, 1], the new embedding will retain useful features and suppress less useful ones to improve model performance.

$$e_i = F_{re}(d_i, s_i) = d_i \cdot s_i \tag{22}$$

Finally, the user and item embedding are output by the multiple hidden layers compressing several fully connected layers. Let $E^{(0)}$ as the output of SE Block and fed into the hidden layers. The feed-forward process is computed by Eq. 23. The l is the depth of hidden layer, σ is the activation function, $W^{(l)}$ and $b^{(l)}$ as the weight and bias of l-th layer. The $E^{(l)}$ is the hidden layer's output, which also represents the user and item embedding for similarity computation.

$$E^{(l)} = \sigma\left(W^{(l)} \cdot E^{(l-1)} + b^{(l)}\right) \tag{23}$$

In the subsequent experiments, we build DSSM based SpFedRec and SENet enhanced SpFedRec for performance and efficiency evaluation.

Evaluation setup

This section introduces our evaluation settings.

Datasets

To evaluate the performance of the proposed model "SpFedRec" against the state-of-art baseline approaches, we conducted several experiments on two real datasets i.e. MovieLens-1 M [44] and Adressa-1 week [45], as explained as below:

MovieLens-1 M

It contains about 1 million explicit rating data for over 3,000 movies from 6,022 users. It includes movie's metadata (*movieID, title, genres*) and user features including

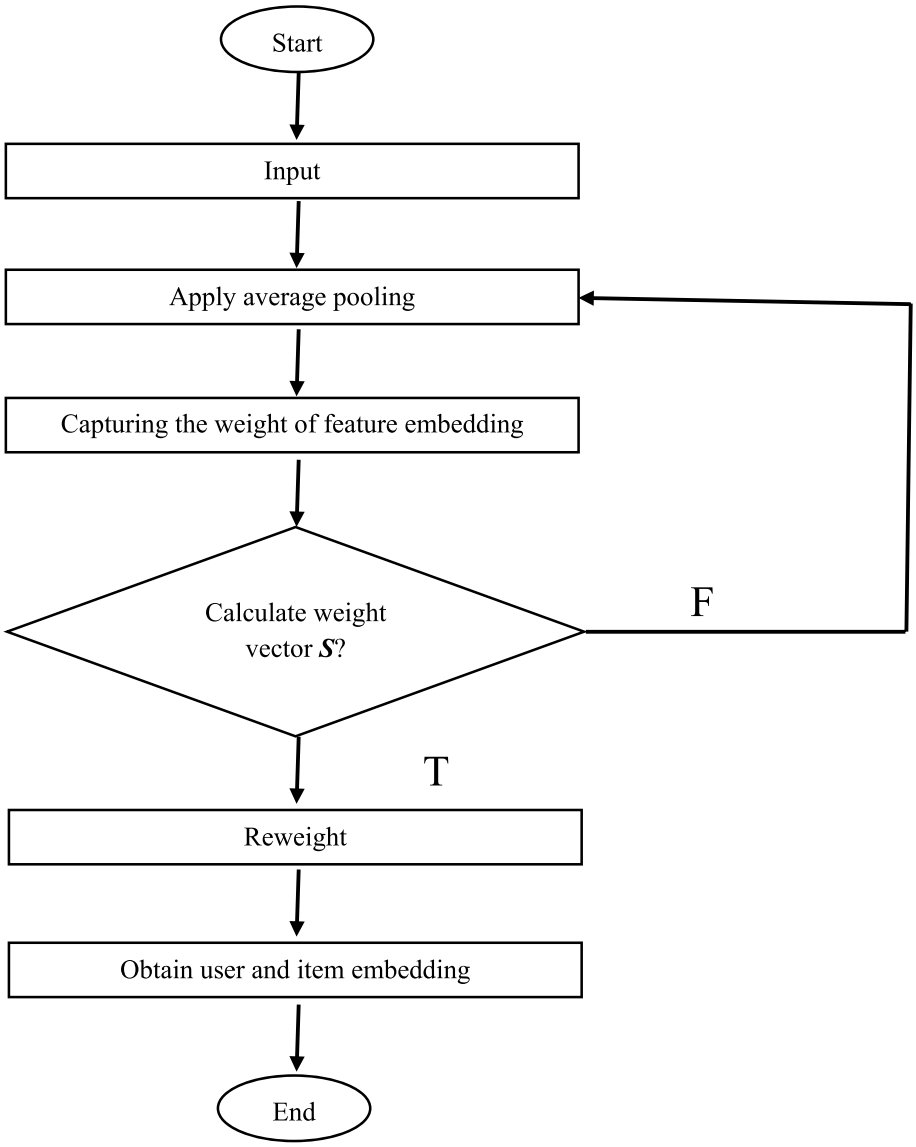

Fig. 4 Flowchart of feature embedding

(*UserID, Gender, Age, Occupation, Zip-code*). Following [46], we converted explicit ratings to implicit ones by treating not rated items as negative implicit feedback.

Adressa-1 week

It is a real-world online news dataset which is released by multiple Norwegian news publishers. This dataset contains up to 2 million clicks from 186,255 users on 14,732 news items. Following [47], we use (*news_id, news_title, key_words, news_body*) as item features. Since Adressa does not contain user features, therefore we randomly sampled five clicked news for the first 2 days as user feature.

To ensure that each client has sufficient interaction records, we removed the users with less than 20 implicit records in both two datasets. For each positive feedback, randomly sample 10 negative interactive items for obfuscated item requests. The statistics of the two datasets are presented in the following Table 2.

Evaluation metrics

The evaluation is in Top-K recommendation, and we verify the performance of SpFedRec and baseline models using Recall@10, NDCG@10, and AUC as evaluation metrics. Recall@10 and NDCG@10 focus on the very

top of recommendation list, while AUC measures overall accuracy.

For communication efficiency evaluation, we evaluate the client's communication size per round of training and online inferring in megabytes. For computation efficiency evaluation, we evaluate time spent per round of training and online inferring in milliseconds. Each experiment is repeated five times independently, and the average results are reported.

Implementation details

We implemented all models including the baseline methods with PyTorch in our numerical experiments. The feature embedding size is set to 20 for MovieLens and 60 for Adressa. For SE Block, the two activation functions are Relu, and the reduction ratio is set to 3. The backbone model with three-layer fully linked neural network and the neuron density for each layer is set as < 256, 128, 128 > . The activation function for individually layer is tanh. K is set as 120 for MovieLens-1 M and 150 for Adress-1 week. To mitigate overfitting, we apply dropout and set the dropout rate as 0.2. The learning rate is 0.0001. We initialize the user and item model parameters by Gaussian distribution with a mean of 0 and a standard deviation of 0.1. We conduct our experiments with dual-NVIDIA 2080Ti GPUs as the server and a MacBook Pro with an M1 chip as the client device.

Experiments

Following the evaluation settings in Sect. "RQ4: How does our proposed privacy protection mechanism influence the performance and efficiency of SpFedRec?", we conduct experiments to evaluate our proposed SpFedRec framework. Specifically, we intend to verify our major claims by answering to the following four research questions:

1. RQ1: How does our method perform compared with baseline models?
2. RQ2: Is the computation and communication efficiency of SpFedRec significantly improved compared to the baseline models in the naïve FL setting?
3. RQ3: How do hyper-parameter settings ρ and K affect the performance and efficiency of SpFedRec?
4. RQ4: How does our proposed privacy protection mechanism influence the performance and efficiency of SpFedRec?

Table 2 Statistics of MovieLens-1 M and Adressa-1 week datasets

Dataset	# Users	# Items	# Clicks	Density (%)	Item Data Size
MovieLens	6,022	3,043	995,154	5.4%	171Kbytes
Adressa	186,255	14,732	2,103,852	0.07%	Gigabyte

Performance evaluation (RQ1)

As the core of SpFedRec model is its capability of performing recommendation, we compare SpFedRec with the following seven baseline methods, including the centralized DSSM, DSSM in naïve setting of FL, and other state-of-art FL recommender systems.

1. DSSM [28]. The centralized version of DSSM is a deep semantic matching model for large corpus item recommendation and information retrieval. The setting of its hidden layer is the same as the backbone model.
2. FCF [6]. The Federated Collaborative Filtering approach is an FL-based matrix factorization for the privacy-preserving recommendation. The client reserves the training data and maintains the update of the user feature vector and the gradient update of the item feature vector.
3. FedMVMF [46]. The Federated multi-view matrix factorization recommendation model utilized side-information from both users and items to improve FL-based CF's performance and alleviate the cold start problem.
4. FedNewsRec [9]. The FedNewsRec is a FL based privacy-preserving recommendation model designed for news recommendation.
5. FedDSSM [48]. The FL based DSSM model in the naïve setting, the client device is accountable for the training of both user and item model.
6. SpFedRec. In this model, we will be using our proposed SpFedRec framework to train the backbone model in a privacy-preserving and efficient manner.
7. SpFedRec-SENet. In this model, we will be using our proposed SpFedRec framework to train the SENet enhanced backbone model in a privacy-preserving and well-organized manner.

The experimental results of all these models are shown in Table 3, and we have the following observations during the numerical experiments:

- Compared to the centralized baseline model, SpFedRec performs worse than the centralized DSSM but within the performance loss of FL [4]. The reason for the performance loss could be the bias introduced by the gradient averaging.

Table 3 Comparison of our proposed models and relevant baselines on MovieLens and Adressa

Models	MovieLens			Adressa		
	AUC	Recall@10	nDCG@10	AUC	Recall@10	nDCG@10
DSSM	76.15 ± 0.91	15.01 ± 0.41	35.84 ± 1.11	69.24 ± 0.18	24.36 ± 0.41	40.17 ± 1.20
FCF	70.56 ± 0.41	10.84 ± 0.08	25.27 ± 0.71	54.49 ± 0.03	17.61 ± 0.06	27.09 ± 1.72
FedMVMF	72.15 ± 0.35	12.03 ± 0.88	27.15 ± 1.31	60.73 ± 0.01	20.91 ± 0.10	30.13 ± 0.58
FedNewsRec	75.81 ± 0.51	15.42 ± 0.01	34.22 ± 0.38	70.95 ± 0.18	26.30 ± 0.01	42.68 ± 1.84
FedDSSM	75.93 ± 0.25	14.63 ± 0.68	33.41 ± 0.54	68.45 ± 1.03	23.11 ± 0.22	39.13 ± 0.48
SpFedRec	75.55 ± 1.21	14.48 ± 1.73	32.96 ± 1.61	68.10 ± 1.25	22.79 ± 1.03	38.65 ± 1.71
SpFedRec-SENet	76.93 ± 0.13	15.54 ± 1.11	36.21 ± 0.75	69.91 ± 0.22	25.43 ± 0.55	41.34 ± 1.01

• Compared to the naive setting of FL based DSSM (FedDSSM), SpFedRec has only a slight performance loss. The use of average error for backward propagation of the item model is probably causing the minor performance loss. This result indicates that training the two-tower model using the SpFedRec framework does not affect the performance too much.
• The SpFedRec-SENet outperforms the state-of-art federated recommender models except less well than the FedNewsRec model on the Adressa dataset. The result demonstrates that our model well handled the task of large-scale item recommendation.
• The SpFedRec-SENet significantly outperforms SpFedRec on both datasets (AUC +2.2% and +4.2% on the two datasets, respectively) and achieves comparable results to the centralized model. The main reason is that SE Block enhances the information utilization of the dominant features while weakening the noise feature, allowing the dominant features to gain more weight in the similarity computation phase.

Efficiency comparison (RQ2)

This subsection presents the efficiency comparison of FedDSSM, SpFedRec, and SpFedRec-SENet on MovieLens and Adressa datasets. The evaluation includes the computation time (training time and online inferring time) by milliseconds and communication overhead (training communication cost and inferring communication cost) by megabytes. The simulation of client computation is conducted on a MacBook pro with the M1 CPU. The total parameter size of DSSM is 0.57 MB and SENet enhanced is 1.03 MB. The computation of each client for each round is performed one by another and reported average score. The simulation results are reported in Table 4. The table highlight that on both datasets, utilizing the SpFedRec framework can significantly improve the efficiency of the federated two-tower model on the client's device computation. Especially on Adressa dataset, it is reducing the total time from 21690 to 574 ms and reducing the total communication from 571.48 MB to 9.37 MB, which yields an improvement of 38 × faster and 60 × less communication overhead. The SpFedRec-SENet, with the addition of the SENet Block computation, improves 23 × faster and 49 × less communication. More specifically, we observed the following impacts during the experiments:

• The communication and computation overhead rise with the size of item data, but the growth rate of SpFedRec is flatter than the naïve FL setting.

Table 4 The efficiency comparison results of FedDSSM, SpFedRec, and SpFedRec-SENet on MovieLens and Adressa

	MovieLens			Adressa		
	FedDSSM	SpFedRec	SpFedRec-SENet	FedDSSM	SpFedRec	SpFedRec-SENet
Device Training	1170 ms	228 ms	449 ms	2570 ms	324 ms	681 ms
Inferring Time	4090 ms	55 ms	55 ms	19120 ms	250 ms	250 ms
Total Time	5260 ms	283 ms	471 ms	21690 ms	574 ms	931 ms
<Improvement>	-	18 ×	11 ×	-	38 ×	23 ×
Training Comm	5.42 MB	1.05 MB	1.73 MB	13.68 MB	2.18 MB	4.40 MB
Inferring Comm	0.14 MB	0.05 MB	0.5 MB	557.80 MB	7.19 MB	7.19 MB
Total Comm	5.56 MB	1.10 MB	1.78 MB	571.48 MB	9.37 MB	11.59 MB
<Improvement>	-	5 ×	3 ×	-	60 ×	49 ×

- In the dataset with a large number of items and dense features (Adressa), SpFedRec more significantly reduces the computational and communication overhead for federated training and online prediction, further validating the effectiveness of SpFedRec in large-scale item recommendation scenarios.
- The overall efficiency comparison results show that SpFedRec is more friendly to the resource-constrained mobile device than the naïve FL setting regarding computation time and communication size.

Hyper-parameter investigation (RQ3)

This subsection studies the influence of negative sampling rate on model performance, and the influence of selected user size on convergence iterations, server aggregation time, device secret sharing time, and device training communication size.

First, we studied the effect of parameter ρ. From the Fig. 5, we observe that the performance of SpFedRec and SpFedRec-SENet improves as ρ increases. The results are consistent with previous works, such as Shen et al. [49], of applying DSSM in a centralized recommendation scenario.

Secondly, we study the effect of parameter K. The amount of users selected to participate in a training round affects the model's convergence efficiency and secure aggregation efficiency. The comparison results are shown in Fig. 6.

As shown in Fig. 6(a), with the increasing of k, the fast the model converges. However, when the number of users involved in a round of training reaches a certain number, the effect on the speed of model convergence gradually decreases.

The impact of k on secure aggregation time (server-side) is shown in Fig. 6(b), the secure aggregation time increase with larger number of users involved per round of training.

From Fig. 6(a) and Fig. 6(b), we can obtain the optimal number of users involved per training round for SpFedRec.

This should be noted that the proposed multi-party circular secret-sharing chain only needs clients share confidences with the member next to him/her, so that the computation time and communication size of secure aggregation (client-size) is not affected by the k as shown in Fig. 6(c) and Fig. 6(d).

Privacy protection mechanism investigation (RQ4)

In this section, we study the effect of privacy protection components: To further investigate the effects of the proposed obfuscated item request (OI) and secure aggregation by secret sharing (SA) on model performance and efficiency, in this section, we conduct an ablation study of these two privacy protection components in SpFedRec. We set SpFedRec-DSSM and SpFedRec-SENet as base model and perform it in the following ways: (1) NO-OI: remove the obfuscated item request from SpFedRec, the client will directly request items' ID of positive samples to the server. The impact of different privacy protection components in SpFedRec on performance and efficiency can be highlighted in Table 5.

The server will mix the negative samples by proportion ρ and return the computed items embedding to the client. (2) NO-SA: remove the secure aggregation from SpFedRec, the gradients and loss of user model will directly

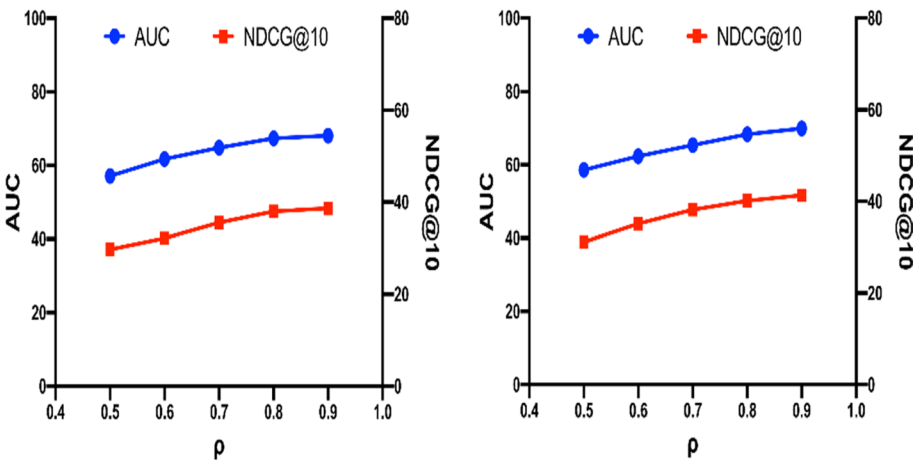

Fig. 5 The performance comparison with different ρ on Adressa

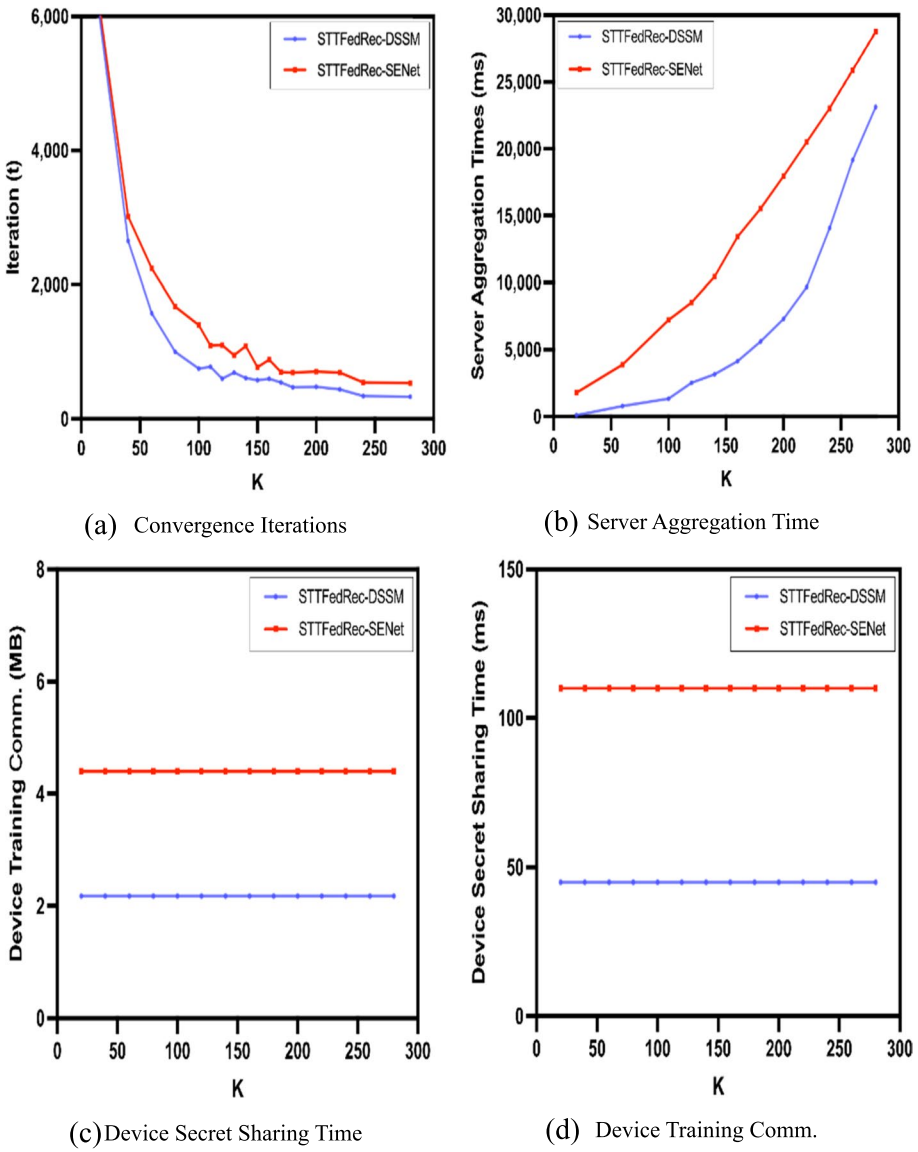

Fig. 6 The effectiveness effect of factor of K on Adressa

Table 5 The impact of different privacy protection components in SpFedRec on performance and efficiency. (Comm. Denote the communication overhead)

Method	MovieLens			Adressa		
	AUC	Training Time	Training Comm	AUC	Training Time	Training Comm
SpFedRec-DSSM	74.83	228 ms	1.35 MB	67.79	324 ms	2.18 MB
NO-OI	74.83	225 ms	1.35 MB	67.79	322 ms	2.18 MB
NO-SA	75.55	182 ms	0.91	68.10	279 ms	1.62 MB
SpFedRec-SENet	76.32	449 ms	1.73 MB	69.15	681 ms	4.40 MB
NO-OI	76.32	444 ms	1.73 MB	69.15	678 ms	4.40 MB
NO-SA	76.93	336 ms	1.02 MB	69.91	569 ms	3.26 MB

send to the recommendation server for gradients average and item model backpropagation. We noted the following observations in Table 5.

- The obfuscated item request does not affect the performance and efficiency of the SpFedRec, and there is only a slight decrease in user device training time, which is mainly caused by the random negative sampling operation. The secure aggregation can increase the computation and communication cost of user devices, and also have slightly impact on recommendation performance.

Now, we assess our suggested scheme's computational cost and memory requirements in terms of the data outsourcing phase. During this phase, we take into account the computation complexity as well as storage overhead of the information outsourcing phase, which is primarily due to the index construction runtime and memory requirements. Figures 7 and 8 show the data outsourcing construction runtime and storage overhead compared to the number of records n and ρ attributes. We can see from these figures that both computation complexity and storage overhead progressively increase with n and ρ.

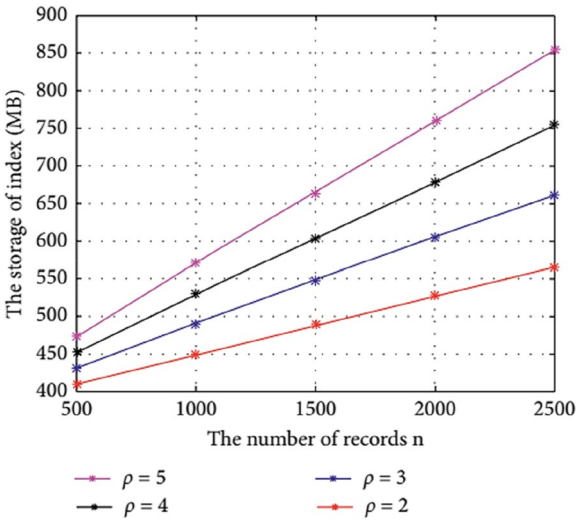

Fig. 8 Comparison of storage overhead and number of records n (where, attributes ρ is between 2 to 5)

Conclusions and future work

In this study, we propose a cloud-edge collaboration based split federated learning framework for the large-scale item recommendation called SpFedRec. In our method, to reduce the computation and communication cost of the federated two-tower model, a split learning approach is applied to migrating the item model from participants' edge devices and compressing item data transmitting. Meanwhile, to enhance the feature representation, the Squeeze-and-Excitation network mechanism is used on the backbone model to optimize the perception of dominant features. Moreover, because the gradients transmitted contain private information about the user, we propose a multi-party circular secret-sharing chain based on secret sharing for better privacy protection. Extensive experiments using plausible assumptions on two real-world datasets demonstrate that our method improves the average computation time and communication cost by 23% and 49%, respectively. Furthermore,

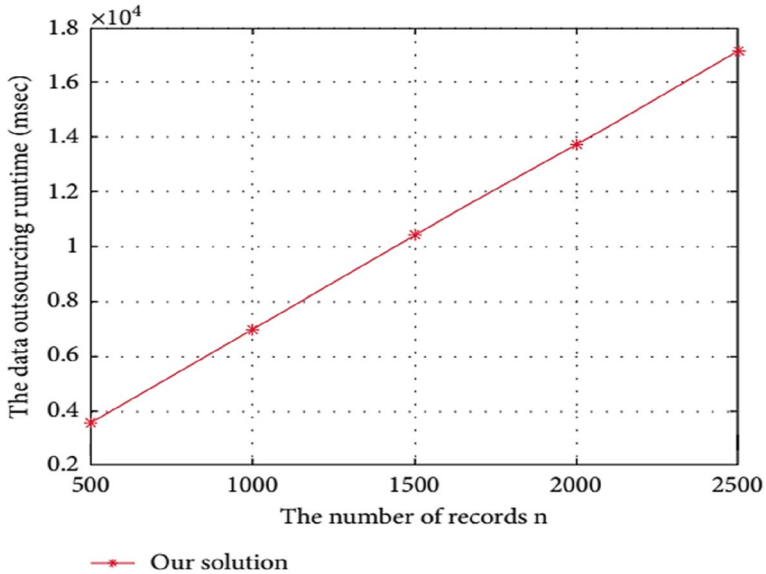

Fig. 7 Comparison of data outsourcing running times and number of the records (where, attribute $\rho = 3$)

the proposed model attains similar performance with the state-of-art federated recommendation models.

In the future, we aim to test the model in a real-time platform and generalize the obtained outcomes. We will implement deep learning model and integrate them with the proposed learning model to improve the efficiency of the recommendation system in terms of training accuracy. This could be achieved through deploying the recommendation system in various modules so that each module runs independently on different machines i.e. edge cloud, and/or remote cloud. We will investigate how the proposed system can be deployed into various modules and then where each module should run in order to improve the training and prediction durations.

Authors' contributions
All authors contribute this paper. The author(s) read and approved the final manuscript.

Authors' information
Jiangcheng Qin was born in Huzhou, Zhejiang, China, in 1993. He received theMaster's degree from Blekinge Institute of Technology, Sweden. Now, he is aPh.D. candidate in College of Information Science and Engineering, NingboUniversity and works at Huzhou University. His research interests areprivacy-preserving recommender systems and federated learning.E-mail:qjc@zjhu.edu.cn
Xueyuan Zhang was born in Ningbo, Zhejiang, China, in 1993. He received theMaster's degree from Ningbo University, P.R. China. Now, he is a Ph.D.candidate in College of Information Science and Engineering, Ningbo University.His research interests are information hiding, data security attack and defensetechniques.E-mail:1711082129@nbu.edu.cn
BaisongLiu was born in 1971. He received the PhD degree from ZhejiangUniversity, China. Now, he works as a professor in College of InformationScience and Engineering, Ningbo University, His research interests includeinformation retrieval and recommendation system, multi-party trusted computingand privacy protection, big data analysis and knowledge organization.E-mail: qjc@nbu.edu.cn
JiangboQian was born in 1974. Now, he works as a professor in College ofInformation Science and Engineering, Ningbo University, His research interestsinclude database management, steaming data processing, multidimensionalindexing and Bloom filter.E-mail: qianjiangbo@nbu.edu.cn

Funding
This work was supported in part by the National Science Foundation of China (No. 61472194), Natural Science Foundation of Zhejiang Province (No. LZ20F020001), Science and Technology Innovation 2025 Major Project of Ningbo (No. 20211ZDYF020036), and the Natural Science Foundation of Ningbo (No. 2021J091).

Availability of data and materials
The datasets generated during and/or analyzed during the current study are available from the corresponding author on reasonable request.

Declarations

Competing interests
The authors declare no competing interests.

Received: 16 November 2022 Accepted: 4 April 2023
Published online: 14 April 2023

References

1. S. Grzonkowski, P. M. Corcoran, and T. Coughlin (2011), "Security analysis of authentication protocols for next-generation mobile and CE cloud services," in Proceedings of the IEEE International Conference on Consumer Electronics, pp. 83–87, Berlin, Germany.
2. Mothukuri V, Parizi RM, Pouriyeh S, Huang Y, Dehghantanha A, Srivastava G (2021) A survey on security and privacy of federated learning. Futur Gener Comput Syst 115:619–640
3. Chai D, Wang L, Chen K, Yang Q (2020) Secure federated matrix factorization. IEEE Intell Syst 36(5):11–20
4. Kairouz P, McMahan HB, Avent B, Bellet A, Bennis M, Bhagoji AN, Bonawitz K, Charles Z, Cormode G, Cummings R, D'Oliveira RG (2021) Advances and open problems in federated learning. Found Trends Mach Learn 14(1–2):1–210
5. McMahan, B., Moore, E., Ramage, D., Hampson, S., y Arcas, B.A (2017) .: Communication-efficient learning of deep networks from decentralized data. In: Singh, A., Zhu, X.J. (eds.) Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017. Proceedings of Machine Learning Research, 54, 1273–1282. PMLR, Fort Lauderdale, USA
6. Ammad-ud-din M, Ivannikova E, Khan SA, Oyomno W, Fu Q, Tan KE, Flanagan A (2019) Federated collaborative filtering for privacy-preserving personalized recommendation system. CoRR abs 1901:09888
7. Lin GY, Liang F, Pan WK et al (2020) FedRec: federated recommendation with explicit feedback. IEEE Intell Syst 2020(36):21–30
8. Flanagan A, Oyomno W, Grigorievskiy A, Tan KE, Khan SA, Ammad-ud-din M (2020) Federated multi-view matrix factorization for personalized recommendations. In: Hutter F, Kersting K, Lijffijt J, Valera I (eds) Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2020, vol 12458. Lecture Notes in Computer Science. Springer, Ghent, Belgium, pp 324–347
9. Qi T, Wu F, Wu C, Huang Y, Xie X (2020) Privacy-preserving news recommendation model learning. arXiv preprint arXiv:2003.09592
10. Luo L, Liu B (2022) Dual-Contrastive for Federated Social Recommendation. In 2022 International Joint Conference on Neural Networks (IJCNN) (pp. 1-8). IEEE
11. Muhammad K, Wang Q, O'Reilly-Morgan D, Tragos EZ, Smyth B, Hurley N, Geraci J, Lawlor A (2020) Fedfast: Going beyond average for faster training of federated recommender systems. In: Gupta R, Liu Y, Tang J, Prakash BA (eds) KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. ACM, Virtual Event, CA, USA, pp 1234–1242
12. Reisizadeh A, Mokhtari A, Hassani H, Jadbabaie A, Pedarsani R (2020) Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization. In International Conference on Artificial Intelligence and Statistics (pp. 2021-2031). PMLR
13. Khan FK, Flanagan A, Tan KE, Alamgir Z, Ammad-ud-din M (2021) A payload optimization method for federated recommender systems. In: Pampín HJC, Larson MA, Willemsen MC, Konstan JA, McAuley JJ, Garcia-Gathright J, Huurnink B, Oldridge E (eds) RecSys '21: Fifteenth ACM Conference on Recommender Systems. ACM, Amsterdam, The Netherlands, pp 432–442
14. Khan A, Marijn ten Thij M, Wilbik A (2022) Communication-efficient vertical federated learning. Algorithms 15(8):273
15. Qin J, Liu B, Qian J (2021) A novel privacy-preserved recommender system framework based on federated learning. In: Li Y, Nishi H (eds) ICSIM 2021: 2021 The 4th International Conference on Software Engineering and Information Management. ACM, Yokohama, Japan, pp 82–88
16. Acun B, Murphy M, Wang X, Nie J, Wu CJ, Hazelwood K (2021) Understanding training efficiency of deep learning recommendation models at scale. In 2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA) (pp. 802-814). IEEE
17. Lindell Y, Pinkas B (2009) Secure multiparty computation for privacy preserving data mining. J Priv Confidentiality 1(1):197
18. Dwork C, Roth A et al (2014) The algorithmic foundations of differential privacy. Found Trends Theor Comput Sci 9(3–4):211–407
19. Lim WYB, Luong NC, Hoang DT, Jiao Y, Liang YC, Yang Q, Niyato D, Miao C (2020) Federated learning in mobile edge networks: A comprehensive survey. IEEE Communications Surveys & Tutorials 22(3):2031-2063

20. Zeng, Q., Lv, Z., Li, C., Shi, Y., Lin, Z., Liu, C., Song, G (2022). Fedprols: federated learning for iot perception data prediction. Appl Intell, 1–13
21. Dwork C, McSherry F, Nissim K, Smith A (2016) Calibrating noise to sensitivity in private data analysis. Journal of Privacy and Confidentiality 7(3):17–51
22. Pulido-Gaytan B, Tchernykh A, Cortés-Mendoza JM, Babenko M, Radchenko G, Avetisyan A, Drozdov AY (2021) Privacy-preserving neural networks with homomorphic encryption: challenges and opportunities. Peer-to-Peer Networking and Applications 14(3):1666–1691
23. Li T, Sahu AK, Talwalkar A, Smith V (2020) Federated learning: Challenges, methods, and future directions. IEEE Signal Process Mag 37(3):50–60
24. Wang X, Han Y, Leung VC, Niyato D, Yan X, Chen X (2020) Convergence of edge computing and deep learning: a comprehensive survey. IEEE Communications Surveys & Tutorials 22(2):869–904
25. Yang J, Yi X, Zhiyuan Cheng D, Hong L, Li Y, Xiaoming Wang S, Xu T, Chi EH (2020) Mixed negative sampling for learning two-tower neural networks in recommendations. In Companion Proceedings of the Web Conference 2020 (pp. 441-447)
26. Wang J, Zhu J, He X (2021) Cross-batch negative sampling for training two-tower recommenders. In: Diaz F, Shah C, Suel T, Castells P, Jones R, Sakai T (eds) SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM, Virtual Event, Canada, pp 1632–1636
27. Cai X, Wang N, Yang L, Mei X (2022) Global-local neighborhood based network representation for citation recommendation. Applied Intelligence, pp.1-18
28. Huang P, He X, Gao J, Deng L, Acero A, Heck LP (2013) Learning deep structured semantic models for web search using clickthrough data. In: He Q, Iyengar A, Nejdl W, Pei J, Rastogi R (eds) 22nd ACM International Conference on Information and Knowledge Management, CIKM'13. ACM, San Francisco, CA, USA, pp 2333–2338
29. Wu M, Tan L, Xiong N (2015) A structure fidelity approach for big data collection in wireless sensor networks. Sensors 15(1):248–273
30. Liu J, Mao Y, Zhang J, Letaief KB (2016) Delay-optimal computation task scheduling for mobile-edge computing systems. In 2016 IEEE international symposium on information theory (ISIT) (pp. 1451-1455). IEEE
31. Ning Z, Dong P, Kong X, Xia F (2019) A cooperative partial computation offloading scheme for mobile edge computing enabled internet of things. IEEE Internet Things Journal 6(3):4804–4814
32. Wu Y, Qian LP, Ni K, Zhang C, Shen X (2019) Delay-minimization nonorthogonal multiple access enabled multi-user mobile edge computation offloading. IEEE Journal of Selected Topics in Signal Processing 13(3):392–407
33. Zhang J, Zhang J, Hu X et al (2019) Joint resource allocation for latency-sensitive services over mobile edge computing networks with caching. IEEE Internet Things Journal 6(3):4283–4294
34. Huang J, Sharma A, Sun S, Xia L, Zhang D, Pronin P, Padmanabhan J, Ottaviano G, Yang L (2020) Embedding-based retrieval in facebook search. In: Gupta R, Liu Y, Tang J, Prakash BA (eds) KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. ACM, Virtual Event, CA, USA, pp 2553–2561
35. Yi X, Yang J, Hong L, Cheng DZ, Heldt L, Kumthekar A, Zhao Z, Wei L, Chi E (2019) Sampling-bias-corrected neural modeling for large corpus item recommendations. In Proceedings of the 13th ACM Conference on Recommender Systems (pp. 269-277)
36. Gupta O, Raskar R (2018) Distributed learning of deep neural network over multiple agents. J Netw Comput Appl 116:1–8
37. Singh A, Vepakomma P, Gupta O, Raskar R (2019) Detailed comparison of communication efficiency of split learning and federated learning. arXiv preprint arXiv:1909.09145
38. Vepakomma P, Gupta O, Swedish T, Raskar R (2018) Split learning for health: Distributed deep learning without sharing raw patient data. arXiv preprint arXiv:1812.00564
39. Chen C, Zhou J, Wu B, Fang W, Wang L, Qi Y, Zheng X (2020) Practical privacy preserving poi recommendation. ACM Transactions on Intelligent Systems and Technology (TIST) 11(5):1–20
40. Chen C, Li L, Wu B, Hong C, Wang L, Zhou J (2020) Secure social recommendation based on secret sharing. arXiv preprint arXiv:2002.02088
41. Reddi S, Charles Z, Zaheer M, Garrett Z, Rush K, Konečný J, Kumar S, McMahan HB (2020) Adaptive federated optimization. arXiv preprint arXiv:2003.00295
42. Huang T, Zhang Z, Zhang J (2019) FiBiNET: combining feature importance and bilinear feature interaction for click-through rate prediction. In Proceedings of the 13th ACM Conference on Recommender Systems (pp. 169-177)
43. Ying Y, Zhang N, Shan P, Miao L, Sun P, Peng S (2021) Psigmoid: Improving squeeze-and-excitation block with parametric sigmoid. Appl Intell 51(10):7427–7439
44. Harper FM, Konstan JA (2015) The movielens datasets: History and context. Acm transactions on interactive intelligent systems (tiis) 5(4):1–19
45. Gulla JA, Zhang L, Liu P, Özgöbek Ö, Su X (2017) The adressa dataset for news recommendation. In Proceedings of the international conference on web intelligence (pp. 1042-1048)
46. Yang F, Wang H, Fu J (2021) Improvement of recommendation algorithm based on collaborative deep learning and its parallelization on spark. Journal of Parallel and Distributed Computing 148:58–68
47. Hu L, Xu S, Li C, Yang C, Shi C, Duan N, Xie X, Zhou M (2020) Graph neural news recommendation with unsupervised preference disentanglement. In Proceedings of the 58th annual meeting of the association for computational linguistics (pp. 4255-4264)
48. Huang M, Li H, Bai B, Wang C, Bai K, Wang F (2020) A federated multi-view deep learning framework for privacy-preserving recommendations. arXiv preprint arXiv:2008.10808
49. Shen Y, He X, Gao J, Deng L, Mesnil G (2014) A latent semantic model with convolutional-pooling structure for information retrieval. In Proceedings of the 23rd ACM international conference on conference on information and knowledge management (pp. 101-110)

Publisher's Note