Federated privacy-preserving collaborative filtering for on-device next app prediction

Saiapin, Albert; Balitskiy, Gleb; Bershatsky, Daniel; Katrutsa, Aleksandr; Frolov, Evgeny; Frolov, Alexey; Oseledets, Ivan; Kharin, Vitaliy

DOI
10.1007/s11257-024-09395-0

Publication date
2024

Document Version
Final published version

Published in
User Modeling and User-Adapted Interaction

Citation (APA)
Saiapin, A., Balitskiy, G., Bershatsky, D., Katrutsa, A., Frolov, E., Frolov, A., Oseledets, I., & Kharin, V. (2024). Federated privacy-preserving collaborative filtering for on-device next app prediction. *User Modeling and User-Adapted Interaction*, *34*(4), 1369-1398. https://doi.org/10.1007/s11257-024-09395-0

User Modeling and User-Adapted Interaction (2024) 34:1369–1398
https://doi.org/10.1007/s11257-024-09395-0

Federated privacy-preserving collaborative filtering for on-device next app prediction

Albert Saiapin[1,2] · Gleb Balitskiy[1] · Daniel Bershatsky[1] · Aleksandr Katrutsa[1,3] · Evgeny Frolov[1,3] · Alexey Frolov[1] · Ivan Oseledets[1,3] · Vitaliy Kharin[4]

Received: 11 June 2023 / Accepted in revised form: 10 January 2024 / Published online: 28 March 2024

Abstract

In this study, we propose a novel SeqMF model to solve the problem of predicting the next app launch during mobile device usage. Although this problem can be represented as a classical collaborative filtering problem, it requires proper modification since the data are sequential, the user feedback is distributed among devices, and the transmission of users' data to aggregate common patterns must be protected against leakage. According to such requirements, we modify the structure of the classical matrix factorization model and update the training procedure to sequential learning. Since the data about user experience are distributed among devices, the federated learning setup is used to train the proposed sequential matrix factorization model. One more ingredient of our approach is a new privacy mechanism that guarantees the protection of the sent data from the users to the remote server. To demonstrate the efficiency of the proposed model, we use publicly available mobile user behavior data. We compare our model with sequential rules and models based on the frequency of app launches. The comparison is conducted in static and dynamic environments. The static environment evaluates how our model processes sequential data compared to competitors. The dynamic environment emulates the real-world scenario, where users generate new data by running apps on devices. Our experiments show that the proposed model provides comparable quality with other methods in the static environment. However, more importantly, our method achieves a better privacy-utility trade-off than competitors in the dynamic environment, which provides more accurate simulations of real-world usage.

✉ Aleksandr Katrutsa
amkatrutsa@gmail.com

1 Skolkovo Institute of Science and Technology, Bolshoy Boulevard 30, Moscow, Russia 121205

2 Delft University of Technology, Delft, The Netherlands

3 AIRI, Moscow, Russia

4 An Independent Expert, Moscow, Russia

Keywords Collaborative filtering · Federated learning · Differential privacy · Matrix factorization

1 Introduction

Today smartphones collect a lot of personal data. Based on the collected data, mobile OS can predict the next app the user will run. This internal service helps to manage available resources on the device better. For example, mobile OS can offload some apps from memory and pre-load new ones in the background to save computational resources or prolong battery life. Such gains are important to improve the user experience of using the device. An example of incorporating such a system in practice is presented in Parate et al. (2013), where the gain from using such a system is shown for many users' daily activities.

This study considers the next app prediction problem and proposes a novel approach to solve it. The proposed approach is based on two ingredients. The first part is a new Sequential Matrix Factorization (SeqMF) model that extends the existing federated matrix factorization approach (Ammad-Ud-Din et al. 2019; Minto et al. 2021) to sequential learning setup (Frolov and Oseledets 2023). The second ingredient is a novel privacy mechanism to anonymize data that are sent from devices to the remote server. While it is possible to solve the next app prediction problem separately on each device and avoid the transfer of user data to the remote server, previous studies in recommender systems (Ekstrand et al. 2011) suggest that users typically exhibit common patterns in their behavior. Therefore, utilizing collective behavioral data may help improve the quality of the predictive model. This, in turn, requires information exchange with a remote server to enable collaborative filtering (CF) setup.

As a basis for our work, we use a special form of the federated learning setup (Abdul-Rahman et al. 2020) adapted for the matrix factorization model. This idea was proved to be appropriate for solving general collaborative filtering problems (Ammad-Ud-Din et al. 2019) and was recently equipped with additional privacy protection (Minto et al. 2021). The key distinction of our work is that we focus on the on-device next app prediction problem that significantly differs from the general collaborative filtering problem. This instructs using sequential data for model training and evaluation. In addition, we revisit the problem of privacy-preserving federated learning and propose an improved QHarmony mechanism for the anonymization of transferred data.

Hence, *the main contributions* of our work are twofold:

- We propose a computationally efficient collaborative filtering model SeqMF for the next app prediction problem. The model is based on a sequence-aware matrix factorization technique and is trained in a federated learning setup (Sect. 4.1).
- We equip the SeqMF model with a new Local Differential Privacy (LDP) algorithm based on the modified Harmony mechanism (Sect. 5.2). We demonstrate that it has a better privacy-utility trade-off compared to competitors (Sect. 6.5).

Paper organization.
The structure of the paper is organized as follows. We first introduce the related work on how recommender systems are used for the next app prediction and related privacy

mechanisms in Sect. 2. We then discuss the general problem statement and the feature of the next app prediction problem in Sect. 3. Then we elaborate on the new sequence-aware matrix factorization model and its federated learning for the next app prediction in Sect. 4. We introduce the novel ϵ-LDP QHarmony privacy mechanism in Sect. 5.2. In Sect. 6 we describe the computational experiments and show the results in Sect. 6.5. Finally, we summarize the obtained results in Sect. 8.

2 Related work

In previous studies, the on-device app prediction problem was considered from different perspectives. In particular, a series of studies consider local models without collaborative information. Shin et al. (2012) adopted a naive Bayesian model based on the context data like GPS, Bluetooth, and Wi-Fi. This approach works individually on devices, avoiding privacy concerns; however, gathering context data can be energy-consuming. The method from Changmai et al. (2019) also works locally on devices, represents user intentions as nodes in vector space, and finds the prediction through the neighborhood search and sequence matching algorithms. Baeza-Yates et al. (2015) learned the Bayesian network to predict the next app based on the observed history. The network is constructed from the spanning tree for the graph based on the vector representation of apps (nodes) and conditional mutual information between them (edges). This approach addresses the cold-start problem for users and apps and implicitly uses the data from all users to compose features for apps. Privacy-related issues and details on aggregating users' data are ignored and not discussed. To speed up previous methods that use some features for apps or context, Liao et al. (2013) introduced a feature selection algorithm that reduces the history log size and the prediction time. Further, only the selected personal features are fed to the kNN classifier that generates the next app. All computations are performed locally, only on users' devices, and ignore collaborative information. Several nearest neighbors-based methods were shown to provide great flexibility for the next app prediction problem. These models use both contextual and sequential data to construct a model. A major drawback of all such methods is that they are mostly designed to work locally on devices without data exchange with a remote server. Thus, they cannot enable the collaborative setup that is supposed to improve the performance of a recommender system and the quality of the next app prediction.

Another series of works considers collaborative models. Zhang et al. (2013) built a Bayesian network locally relying on context to predict the next app. In addition, the server part is introduced to update the client model, transfer statistics, etc. However, the authors do not include collaborative information in the proposed model and ignore any privacy-related issues. Xu et al. (2013) developed a framework that generates individual per-user classifiers based on context and session data. One major benefit of this framework is that per-user models are aware of community-level behavior based on user similarity that enhances their local predictions. However, the authors confirm that privacy and scalability are major limitations of their model.

The general collaborative filtering formulations of the next item prediction problem were developed and studied as well. Rendle et al. (2010) suggested combining

matrix factorization and Markov chain models. The advantage of this approach is that each transition is influenced by transitions of similar users, similar items, and similar transitions. Thus, the quality of the final transition is higher. He and McAuley (2016) proposed to combine long-term user preferences and short-term sequential dynamics which lead to more accurate predictions. However, the authors use modified stochastic gradient descent to update only global model parameters. Ludewig and Jannach (2018) gave a broad overview of models for session- and sequential-based recommendations from simple baselines like Association Rules to complex neural models like GRU4Rec (Hidasi et al. 2015). The main issue of the general-purpose collaborative filtering methods mentioned above is that they ignore privacy-related issues and therefore cannot be directly used to solve the next app prediction problem.

The federated learning paradigm adapted for the collaborative filtering task was recently explored by Ammad-Ud-Din et al. (2019), Minto et al. (2021). The authors proposed a special hybrid optimization scheme for a *non-sequential* model that is trained in a federated learning setup and incorporates personal data protection. The advantages of this approach are the usage of collaborative data for model training that improves model performance and federated learning to maintain privacy. However, the main downside of these works is the lack of sequential user preferences that could be a valuable source of information for the next app prediction task (Ludewig and Jannach 2018). In our work, we employ a special form of the federated learning setup adapted for the matrix factorization model (Ammad-Ud-Din et al. 2019; Minto et al. 2021). Moreover, our model is based on a sequence-aware matrix factorization technique; therefore, our model satisfies three requirements: proper processing of sequential data, supporting federated learning setting, and robustness to the privacy-preserving method for data transfer.

In addition, many deep learning-based models were developed for the task of sequential recommendation. A comprehensive survey of such models is presented in Fang et al. (2020). In particular, study (Han et al. 2021) suggests the DeepRec model, which is based on the GRU RNN architecture and does not support sending user data to a server. Only fine-tuning of the base model from the server with local data is supported. This approach avoids privacy-related concerns and ignores collaborative information. Also, study (Ouyang et al. 2022) considers the local model based on the hierarchical graph attention mechanism to capture user interests evolving over time. This model constructs embeddings based on a dynamic usage graph network and extracts from these embeddings the next app prediction. Here, the authors do not consider any privacy-related issues. And, the work (Zhao et al. 2019) constructs a complicated deep learning model with an attention layer and dual-DNN submodule which takes into account the temporal context and simultaneous fitting apps' and users' embeddings. The training of such models is very challenging (Gusak et al. 2022). Also, this model is still local and much more complicated than the proposed SeqMF model. Thus, the applicability of deep learning collaborative models in real-world scenarios remains an open problem since such models require federated learning support and robustness to the selected privacy mechanism.

Since we focus on the privacy of recommendations, we summarize related work on different privacy mechanisms. To protect sensitive user data during transfer in the collaborative filtering setup, the LDP mechanism is used, since it does not assume

the trusted third party (e.g., server) in contrast to Centralized Differential Privacy (CDP) mechanism (Dwork et al. 2006b, 2014, 2006a). Typically, the LDP mechanism encrypts the sensitive user data on a device and only after that transfers them to the remote server. Therefore, the encryption stage has to be computationally and memory efficient.

The most famous deployment of the LDP mechanism is the RAPPOR algorithm (Erlingsson et al. 2014). It is based on the Randomized Response mechanism (Warner 1965), which later found its application in many LDP algorithms. However, this method is not appropriate for gradient-based optimizers since it works with categorical data only. For real-valued data, the Laplace mechanism (Dwork et al. 2014) was adopted. According to this approach, each user adds noise from the Laplace distribution to sensitive data before transmission to a remote server. This simple scheme guarantees users' data privacy but causes a high drop in utility if the dimension of the transmitted data is large. Then, in (Duchi et al. 2016) it was proposed to apply the Randomized Response (RR) technique to perturb users' real-valued data while ensuring an unbiased estimation of aggregated update. In Nguyên et al. (2016), the authors showed that in some cases (Duchi et al. 2016) mechanism does not preserve privacy. To address the revealed issues, they proposed a simple Harmony mechanism, where a user selects only one value from the transmitted matrix and applies the modification of the RR mechanism to obtain the perturbed value. Although it has an asymptotically better privacy-utility trade-off than the Laplace mechanism, in practice, with a finite number of users, the utility drop is very high (Minto et al. 2021). Also, in (Chen et al. 2020) it was suggested to use Kashin representation of the transmitted data, random sampling, and quantization before applying RR. The construction of the Kashin representation evenly distributes information about transmitted data over a new basis. Thus, less amount of information vanishes after random sampling. For now, asymptotically, this method achieves the best trade-off between utility, communication cost, and privacy budget. However, it has limitations for usage on devices due to high computational complexity. To address the issues discussed above, we propose the QHarmony mechanism, which achieves experimentally better privacy-utility trade-off while keeping the gradient perturbation procedure computationally efficient.

3 Problem statement

Let M be the number of users and N be the total number of apps available for installation. From the users' history, we can compose the interaction binary matrix $\mathbf{A} = [a_{ij}]$ of size $M \times N$ such that $a_{ij} = 1$ if the i-th user has run the j-th app. Formally, assume we have a set of the installed apps indices A_u on a user u device such that $|A_u| \ll N$ and an *ordered* set of the used apps $\mathsf{H}_u^{(t)} = \{i_1, i_2, \ldots, i_p\}$ such that $i_k \in A_u$, which we call history of the used apps by the moment t. The order of the used apps is based on the corresponding timestamp. Since matrix $\mathbf{A}$ is constructed from this historical data, it also depends on the parameter t which we omit to simplify notation.

Our main task is to predict the next app that will be run on the particular user's device. Here, we see the sequential nature of the given data that is crucial for the

high performance of the trained model. Then we have to obtain a function that maps the available history or its part to the next app index $i^* \in A_u$. Following previous studies (Hu et al. 2008; Ammad-Ud-Din et al. 2019), we construct the relevance score function r that depends on unknown embeddings of apps $\mathbf{q}_i,\ i = 1, \ldots, N$ and of users $\mathbf{p}_u,\ u = 1, \ldots, M$. The higher the value of $r(\mathbf{q}_i, \mathbf{p}_u)$ is, the more relevant app i to user u is. Therefore, the index of the predicted next app for current user u is the index of the maximum element of $r(\mathbf{q}_i, \mathbf{p}_u)$ over all candidate apps $i \in A_u$.

However, we need to obtain the proper embedding vectors $\mathbf{p}_u$ and $\mathbf{q}_i$ from the observed history collected in matrix $\mathbf{A}$. The classical approach to finding these embeddings is to solve the following optimization problem (Hu et al. 2008):

$$\frac{1}{2}\sum_{u,i} c_{ui}(a_{ui} - r(\mathbf{q}_i, \mathbf{p}_u))^2 + \frac{\lambda}{2}\sum_i \|\mathbf{q}_i\|_2^2 + \frac{\lambda}{2}\sum_u \|\mathbf{p}_u\|_2^2 \to \min_{\{\mathbf{q}_i, \mathbf{p}_u\}}, \tag{1}$$

where c_{ui} measures a confidence in observing a_{ij} and $r(\mathbf{q}_i, \mathbf{p}_u) = \mathbf{q}_i^\top \mathbf{p}_u$.

After computing optimal embeddings $\{\mathbf{q}_i^*, \mathbf{p}_u^*\}$, we can suggest n apps as candidates for a particular user as follows:

$$\text{toprec}(u; n) = \arg\max_{i \in A_u}^{n} \left(r(\mathbf{q}_i^*, \mathbf{p}_u^*)\right), \tag{2}$$

where $\arg\max_{i \in A_u}^{n} (f(i)) = (i_1^*, i_2^*, \ldots, i_n^*)$ is the ordered set of apps from A_u such that if $f(i_k^*) \geq f(i_l^*)$, then $k < l$.

The standard method to solve problem (1) is the alternating least squares (ALS) approach that updates $\mathbf{q}_i$ and $\mathbf{p}_u$ (Hu et al. 2008). However, since every user stores its own interaction history $\mathsf{H}_u^{(t)}$ locally on the device and all users have apps from a predefined set of available apps, it is natural to exploit the *federated learning paradigm* (Ammad-Ud-Din et al. 2019; Minto et al. 2021) to solve problem (1). In this paradigm, users' embeddings $\mathbf{p}_u$ are updated locally on devices in a parallel asynchronous manner and apps' embeddings $\mathbf{q}_i$ are updated globally in the remote server. To perform such updates, we have to support the transmission of data from users' devices to the server and vice versa. Since sending data from a user to a third-party server can lead to personal data leaks, the proper privacy algorithm must be incorporated into the training procedure. Moreover, the relevance score function r has to be constructed to operate with sequential data corresponding to the history of interactions with apps. Taking into account these requirements, we present a proper modification of problem (1) in the next section.

4 Federated learning of the sequence-aware matrix factorization model

In this section, we design a new objective that incorporates knowledge about users' sequential behavior into the framework of the objective function in problem (1). After that, we present the details of the federated learning approach to get optimal users'

and apps' embeddings. This approach combines the ALS step for updating users' embeddings locally and the gradient-based update of the apps' embeddings in the remote server.

4.1 Sequence-aware matrix factorization for the next app prediction

4.1.1 Objective function for sequential data

Let $\mathbf{P} \in \mathbb{R}^{M\times d}$ and $\mathbf{Q} \in \mathbb{R}^{N\times d}$ be matrices composed row-wisely from users' and apps' embeddings, respectively. Since the users' history is distributed among the devices, we rewrite objective function in problem (1) in a user-oriented form:

$$L(\mathbf{P}, \mathbf{Q}) = \frac{1}{2}\sum_{u} \left\|\mathbf{r}(\mathbf{Q}, \mathbf{p}_u) - \mathbf{a}_u\right\|^2_{\mathbf{C}_u} + \frac{\lambda}{2}\left(\sum_{u}\|\mathbf{p}_u\|_2^2 + \|\mathbf{Q}\|_F^2\right) \to \min_{\{\mathbf{Q},\mathbf{p}_u\}}, \quad (3)$$

where $\mathbf{a}_u$ is a binary indicator vector of length N with value a_{ui} in the i-th position, the matrix $\mathbf{C}_u \in \mathbb{R}^{N\times N}$ is diagonal, where the i-th element on the diagonal equals to c_{ui}, respectively. Also, we use the following notations $\|\mathbf{x}\|^2_{\mathbf{W}} = \mathbf{x}^\top\mathbf{W}\mathbf{x}$ and $\|\mathbf{Q}\|^2_F$ is squared Frobenius norm of matrix $\mathbf{Q}$. The form of problem (3) highlights that vectors $\mathbf{p}_u$ are distributed over users' devices. This form is further used to derive the analytical equation for updating every $\mathbf{p}_u$ independently and to introduce the stochastic gradient update for matrix $\mathbf{Q}$ since the exact gradient is the sum of items over a set of users.

The standard matrix factorization model (Ammad-Ud-Din et al. 2019) computes item relevance scores as $\mathbf{r}_{MF}(\mathbf{Q}, \mathbf{p}_u) = \mathbf{Q}\mathbf{p}_u$ that does not capture the sequential nature of the available data. To capture the user's sequential patterns, we introduce the following relevance score function $\mathbf{r}$:

$$\mathbf{r}(\mathbf{Q}, \mathbf{p}_u) = \mathbf{Q}\mathbf{p}_u + \mathbf{h}_u(\mathbf{Q}) = \mathbf{Q}\mathbf{p}_u + \operatorname{diag}\left(\mathbf{S}_u\mathbf{Q}\mathbf{Q}^\top\right), \quad (4)$$

where the design matrix $\mathbf{S}_u$ encodes relative frequency weights corresponding to transition to app i from some previous app j in user u history. We follow the developments in Ludewig and Jannach (2018), where the relevance score function is represented as a sum of global relevance $\mathbf{r}_{MF}$ and relevance based on the sequential property of data. The second term in Eq. 4 depends only on the local transition statistics and therefore captures sequential behavior.

Elements of $\mathbf{S}_u$ are calculated based on user u interaction history $\mathsf{H}_u^{(t)}$ according to the following equation, which represents how frequent pair of apps (i, j) appears sequentially compared to the total number of appearance of the i-th app:

$$(\mathbf{S}_u)_{ij} = \frac{\sum_{k=2}^{|\mathsf{H}_u^{(t)}|} \mathbb{1}[j \to i]\mathbb{1}[i^u_{k-1} = j]\mathbb{1}[i^u_k = i]}{\sum_{k=2}^{|\mathsf{H}_u^{(t)}|} \mathbb{1}[i^u_{k-1} = i]},$$

$$\mathbb{1}[m = n] = \begin{cases} 1, & m = n \\ 0, & \text{otherwise,} \end{cases} \qquad \mathbb{1}[j \to i] = \begin{cases} 1, & (j, i) \in \mathsf{H}_u^{(t)} \\ 0, & \text{otherwise,} \end{cases} \tag{5}$$

where $(j, i) \in \mathsf{H}_u^{(t)}$ indicates that both i and j belong to the history $\mathsf{H}_u^{(t)}$ and i immediately follows j. For example, let a user u has the following history of interactions: $\mathsf{H}_u^{(t)} = (a, b, c, a, a, b, a, c)$. Then, the resulting matrix $\mathbf{S}_u$ is of the size 3×3 and has the following form:

$$\mathbf{S}_u = \begin{array}{c} \begin{matrix} a & b & c \end{matrix} \\ \begin{pmatrix} \frac{1}{4} & \frac{2}{4} & \frac{1}{4} \\ \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{2} & 0 & 0 \end{pmatrix} \end{array} \begin{matrix} a \\ b \\ c \end{matrix} \tag{6}$$

For instance, consider the (a, b)-th element of this matrix equal to $2/4$ since there are 2 sequential co-occurrences of (a, b) and 4 occurrences of a in history. Thus, we extend the approaches that consider only interaction frequencies and repeated events from users' interaction logs (Jannach et al. 2017).

The i-th element of $\mathbf{h}_u^{(t)}(\mathbf{Q})$ represents the average similarity between the target app i and other apps weighted by the corresponding elements of the matrix $\mathbf{S}_u$ which encodes the sequential history (He and McAuley 2016):

$$\left(\mathbf{h}_u^{(t)}\right)_i = \sum_{j=1}^{N} \left(\mathbf{S}_u^{(t)}\right)_{ij} \mathbf{q}_i^\top \mathbf{q}_j. \tag{7}$$

Here we explicitly use the superscript (t) to highlight that the duration of the considered period is important. To improve capturing the global dynamics in app usage frequencies, one has to restrict a period duration. It can be done using a time window of the most recent events, e.g., several weeks or months depending on the available data. Smaller t improves computational performance while larger t may help discover longer-range correlations between events that do not immediately follow one another.

Thus, we have introduced the novel relevance score function in Eq. 4 appropriate for sequential data and modified classical loss in the Matrix Factorization model. The introduced relevance score considers only the most recent events according to the predefined time window, which is a hyperparameter. Since this hyperparameter captures the underlying structure of the available data, it can be estimated with the grid search or any other hyperparameter tuning approach (Akiba et al. 2019). We refer to the resulting objective function and relevance score function as Sequential Matrix Factorization model or **SeqMF**.

4.1.2 Inference in SeqMF model

The relevance score function (see Eq. 4) is designed to facilitate proper learning of app co-occurrence statistics from a history of user actions. However, prediction rule based on Eq. 4 incurs high computational cost prohibitive in the inference step. Therefore, in the inference step, we propose the following more lightweight relevance score function

for user u and the i-th app installed in the user device:

$$\hat{r}(\mathbf{q}_i, \mathbf{p}_u; s_u) = \mathbf{q}_i^\top \mathbf{p}_u + \mathbf{q}_i^\top \mathbf{q}^{(s)}, \qquad \mathbf{q}^{(s)} = \sum_{i_k \in s_u} \mathbf{q}_{i_k}, \tag{8}$$

where $s_u = (i_1, \ldots, i_l), i_k \in A_u$ is a set of the latest l apps user u interacted with. The first item $\mathbf{q}_i^\top \mathbf{p}_u$ is equal to the i-th element of the corresponding first item from Eq. 4, and the second item $\mathbf{q}_i^\top \mathbf{q}^{(s)}$ is the approximation of the second item in Eq. 4 that is responsible for capturing local patterns of user behavior. Thus, we reduce the computational cost of the inference step while preserving the main property of the relevance score function simultaneously.

4.1.3 Confidence-based weighting coefficients

The idea of the confidence-based weighting approach is to assign proper importance weights to discrepancies between predicted and ground-truth values of both observed and unobserved interactions during the training of a model. That way one can expect to learn better embeddings and capture underlying behavioral patterns more accurately. Popular weighting schemes (Hu et al. 2008) assign small uniform weights on unobserved data and higher data-dependent weights on observed entries. Such schemes proved to be effective in practice for standard CF tasks (Hu et al. 2008; Li et al. 2018), where any available item can be recommended to any user. However, preliminary experiments showed that standard weighting schemes did not work in our setting. The possible reason is the next app can be selected *only* from the pool of installed apps rather than all possible apps.

Therefore, following the ideas on frequency-based weighting (He et al. 2016), we propose the following expression for confidence weights:

$$c_{ui} = \frac{d_{ui}^\gamma}{\sum_j d_{uj}^\gamma}, \tag{9}$$

where d_{ui} is a launch frequency of item i in the history of user u and $\gamma > 0$ is hyperparameter. Note that by varying values of γ, one can either make the model more biased toward highly frequently launched apps (higher γ) or more balanced (lower γ), see He et al. (2016).

4.2 Federated learning of SeqMF model

Since the history of users' interactions is stored locally in users' devices it is natural to exploit the federated learning approach to solve problem (3). The federated learning approach specified for CF setup consists of two major intercommunicating blocks (Ammad-Ud-Din et al. 2019): apps' embeddings $\mathbf{Q}$ located at the remote server and user embeddings $\mathbf{p}_u,\ u = 1, \ldots, M$ stored locally at the user devices (one vector per device). The most recent version of matrix $\mathbf{Q}$ is distributed to the devices.

Then, each device can update the corresponding user embedding using newly collected behavioral data and generate more accurate predictions. With some periodicity, some data from users' devices are sent back to the remote server to update the matrix $\mathbf{Q}$, and the process starts over.

The key ingredient of this scheme is the difference in the methods of updating users' and apps' embeddings. As observed in Ammad-Ud-Din et al. (2019), the default ALS scheme to update matrix $\mathbf{Q}$ requires sending the entire user interactions history to the server, which poses a direct threat to data privacy. To mitigate this problem, the authors use a hybrid optimization scheme that combines ALS updates for users' embeddings and gradient-based updates for apps' embeddings.

According to the ALS scheme, matrix $\mathbf{P}$ is updated in a row-wise manner as follows:

$$\mathbf{p}_u = \left(\mathbf{Q}^\top \mathbf{C}_u \mathbf{Q} + \lambda \mathbf{I}\right)^{-1} \mathbf{Q}^\top \mathbf{C}_u \left(\mathbf{a}_u - \mathbf{h}_u(\mathbf{Q})\right). \tag{10}$$

We highlight that every user embedding $\mathbf{p}_u$ is updated *privately on a user device* and *in parallel* to support a distributed nature of federated learning. Also, note that during such an update the influence of the long-term preferences encoded in $\mathbf{a}_u$ is compensated with the short-term sequential dynamics represented by $\mathbf{h}_u(\mathbf{Q})$. This is precisely the behavior we planned to achieve.

In contrast, the matrix $\mathbf{Q}$ is updated on the remote server with a gradient-based optimizer. For example, classical gradient descent reads as follows:

$$\mathbf{Q} := \mathbf{Q} - \beta \frac{\partial L}{\partial \mathbf{Q}}, \tag{11}$$

where $\beta > 0$ is a learning rate, $\frac{\partial L}{\partial \mathbf{Q}}$ is full gradient of the introduced loss function (see Eq. 3) computed as follows:

$$\frac{\partial L}{\partial \mathbf{Q}} = \sum_{u=1}^{M} \mathbf{F}(u) + \lambda \mathbf{Q}, \qquad \mathbf{F}(u) = \mathbf{D}_u \mathbf{e}_N \mathbf{p}_u^\top + \left(\mathbf{D}_u \mathbf{S}_u + \mathbf{S}_u{}^\top \mathbf{D}_u\right) \mathbf{Q}, \tag{12}$$

where $\mathbf{D}_u = \text{diag}\left(\mathbf{C}_u \left(\mathbf{r}(\mathbf{Q}, \mathbf{p}_u) - \mathbf{a}_u\right)\right)$, $\mathbf{e}_N$ is a vector of ones of size N and $\text{diag}(\mathbf{w})$ is a diagonal matrix with vector $\mathbf{w}$ in the diagonal.

To compute $\frac{\partial L}{\partial \mathbf{Q}}$ on a server side we need only values $\mathbf{F}(u)$ from Eq. 12. These values can be gathered from clients on the remote server. However, in practice, not all the clients may send $\mathbf{F}(u)$ by request. This issue is treated by using a stochastic gradient estimator:

$$\frac{\partial L}{\partial \mathbf{Q}} \approx \sum_{u \in U_b} \mathbf{F}(u) + \lambda \mathbf{Q}, \tag{13}$$

where U_b is a set of users that can transmit $\mathbf{F}(u)$. After that, a stochastic optimizer like SGD with momentum (Qian 1999) or Adam (Kingma and Ba 2014) can be used to update matrix $\mathbf{Q}$ in the remote server.

Although the described hybrid scheme of federated learning requires transmitting $\mathbf{F}(u)$ from users' devices to the remote server, these data do not contain explicit user interaction history, so it is less sensitive. Nevertheless, it is still susceptible to certain attacks, so we make this transmission private. We aggregate the anonymized data in the remote server and use the perturbed gradient estimate $\tilde{\mathbf{F}}$ to update apps' embeddings $\mathbf{Q}$. To ensure stronger privacy of the transmitted data, we propose a new privacy-preserving mechanism described in the next section.

4.3 Computational complexity

Since our framework works in a federated learning setup, operations performed on devices have to be computationally and memory efficient. In this section, we present a detailed analysis of the on-device operations' complexity and estimate their runtime on a particular mobile CPU. Our estimates of the number of operations are based on the exact counting of arithmetic operations and linear algebra algorithms described in Golub and Van Loan (2013).

Time complexity.

First, according to Eq. 10, users' embeddings are updated as solutions of the corresponding $d \times d$ linear systems with matrices $\mathbf{Q}^\top \mathbf{C}_u^{(t)} \mathbf{Q} + \lambda \mathbf{I}$ and the right-hand sides $\mathbf{Q}^\top \mathbf{C}_u^{(t)} \left(\mathbf{a}_u - \text{diag}\left(\mathbf{S}_u^{(t)} \mathbf{Q} \mathbf{Q}^\top \right) \right)$. So, the number of operations to perform a single user embedding update is $T_u = T_m + T_r + T_s$, where T_m is the number of operations to compute the matrix of the linear system, T_r is the number of operations to compute the right-hand side and T_s is a number of operations to solve the linear system. Taking into account that the matrix $\mathbf{C}_u^{(t)}$ is diagonal and the diagonal element is nonzero if the corresponding app is installed on user u device, the computing of the matrix $\mathbf{Q}^\top \mathbf{C}_u^{(t)} \mathbf{Q} + \lambda \mathbf{I}$ requires $T_m = 2d^2 N_u + d$ operations, where N_u is the number of apps installed on the user u device. Next, computing the right-hand side as a result of matrix by vector product is performed from right to left and requires $T_r = 2dN_u^2 + 2N_u d + N_u + N_u + 2dN_u$ operations. Finally, since the computed matrix is symmetric and positive-definite, we can use Cholesky factorization to solve the target system. This approach to solving our linear system requires $\frac{d^3}{3} + \frac{3d^2}{2} + \frac{7d}{6}$ operations to factorize the matrix and $2d(d-1)$ operations to solve two intermediate triangular linear systems. Thus, the total number of operations to update user embedding is

$$T_u = T_m + T_r + T_s, \quad T_m = 2d^2 N_u + d, \; T_r = 2dN_u^2 + 4N_u d + 2N_u,$$
$$T_s = \frac{d^3}{3} + \frac{3d^2}{2} + \frac{7d}{6} + 2d(d-1).$$

Second, consider computing item of the gradient $\mathbf{F}(u)$ in Eq. 12 and denote T_g as the total number of operations to compute it. Taking into account structures of matrices $\mathbf{D}_u$ and $\mathbf{S}_u$, one can estimate T_g as follows

$$T_g = 3N_u + 3dN_u + 2d(N_u + N_u^2) + 2N_u d^2 + 4N_u^2.$$

Note that we estimate the exact number of operations or slightly overestimate it, not just asymptotic complexity. This estimate induces the runtime from the CPU performance given below.

On-device memory consumption.

Consider a particular device that has 12 GB RAM. Assume that OS takes 2 GB. Then, based on Eqs. 10 and 12 we can calculate how much data a device should store. We start with the data that are constantly stored on the device:

- History of apps usage stored in $\mathbf{a}_u$ requires $8N_u$ bytes
- Sequential data about apps usage stored in matrix $\mathbf{S}_u$ requires no more than $8N_u^2$ bytes;
- User-specific item embeddings $\mathbf{Q}$ requires $8dN_u$ bytes
- Confidence coefficients $\mathbf{C}_u$ requires $8N_u$ bytes
- User embedding $\mathbf{p}_u$ requires $8d$ bytes.

Firstly, to solve linear system given in Eq. 10 a device should have the following data:

- Matrix of linear system $\left(\mathbf{Q}^\top \mathbf{C}_u \mathbf{Q} + \lambda \mathbf{I}\right)$ requires $8d^2$ bytes;
- Right-hand side $\mathbf{Q}^\top \mathbf{C}_u \left(\mathbf{a}_u - \mathbf{h}_u(\mathbf{Q})\right)$ requires no more than $8(N_u + d)$ bytes;

Computing a gradient item locally through Eq. 12 requires the following terms:

- $\mathbf{D}_u$ requires $8(N_u + d)$ bytes
- $\mathbf{F}_u$ requires $8(Nd + dN_u)$ bytes.

As a result, if a user has $N_u = 80$ installed apps, the item catalog consists of $N = 2 \cdot 10^6$ items, and item embeddings are of size $d = 128$, then storing necessary data on the device constantly requires approximately 0.000126 Gb. In order to update user embedding $\mathbf{p}_u$ one would need approximately 0.000124 Gb. In order to calculate the gradient $\mathbf{F}(u)$ one would need approximately 1.91 Gb. Therefore, in this case, the memory capacity of a device is enough.

From time complexity and memory consumption estimates to numbers on a particular device.

Consider one of the currently high-performance smartphones with the following CPU characteristics:

- 1×3.36 GHz Cortex-X3
- 2×2.8 GHz Cortex-A715
- 2×2.8 GHz Cortex-A710
- 3×2.0 GHz Cortex-A510

We can estimate the number of operations this device can do per second from these characteristics. In particular, the number of floating-point operations per second (FLOPS) can be estimated as FLOPS = #cores $\cdot$ #(cycles/second) $\cdot$ 4, where the last multiplier 4 is due to the double precision format. Assume we reserve the last two cores for OS purposes. Therefore, we can roughly estimate the performance of the considered device in terms of FLOPS as $F = 4 \times 10^9 \cdot (3.36 + 5.6 + 5.6 + 2.0) \approx 66.24 \times 10^9$.

Then, assume that a user has installed $N_u = 80$ apps and apps' embeddings are of size $d = 128$. In this case, we can estimate the runtime on the considered device as $T_{run} = (T_u + T_g)/F = 0.076 + 0.065 = 0.141$ ms.

5 Privacy mechanism in the SeqMF model

Federated learning assumes the presence of a remote server and users whose data is used to train the model. However, it is almost impossible to guarantee the security of the server from violating private protocols or using inference attacks. Therefore, in our work, we consider the remote server to be an untrusted party. In other words, to ensure the protection of user-sensitive data, it is necessary to minimize the probability of reconstruction of their contents on a remote server.

In general, there are several approaches to protect sensitive data: secure Multi-Party Computation (MPC) (Yao 1986), Homomorphic Encryption (HE) (Reyzin et al. 2021), Trusted Execution Environment (TEE) (Zheng et al. 2021; Li et al. 2019; Costan et al. 2016), and Differential Privacy (DP) (Dwork et al. 2006b, 2014, 2006a). However, TEE imposes a large number of restrictions on the hardware level, thereby severely limiting its usage in real systems. MPC and HE significantly increase the computational complexity or require the trusted third party (Kairouz et al. 2021) and, thus, are not suitable for mobile devices with untrusted remote servers. So, further, we consider only the local differential privacy (ϵ-LDP) technique, which gives strong privacy guarantees without large additional overhead on computational complexity and communication cost (Kairouz et al. 2021). Below we provide a formal definition of the ϵ-LDP mechanism for the reader's convenience.

Definition 1 A randomized mechanism A is ϵ-locally differentially private (ϵ-LDP) if and only if for any two inputs $v,\ v'$ and any output $y \in range(\mathsf{A})$ the following holds:

$$\mathbb{P}(\mathsf{A}(v) = y) \leq e^{\epsilon}\mathbb{P}(\mathsf{A}(v') = y), \tag{14}$$

where ϵ is called *the privacy budget*. The smaller the privacy budget is, the better the protection of sensitive data is.

Also, the ϵ-LDP mechanism does not require a trusted third party, unlike central differential privacy algorithms (CDP) (Dwork et al. 2014).

5.1 Local differential privacy

Incorporating a privacy mechanism into the model based on matrix factorization is straightforward and natural. Figure 1 illustrates the scheme of private communications between one client and the remote server. In this scheme, user embeddings $\mathbf{p}_u$ and the history of users' actions $\mathsf{H}_u^{(t)}$ are stored only on the device to preserve the privacy of sensitive information. However, to update the apps' embeddings in matrix $\mathbf{Q}$ users compute gradients $\mathbf{F}(u)$ with private data. Transmission of these gradients to the remote server can lead to the leakage of private data. Thus, to maintain privacy throughout

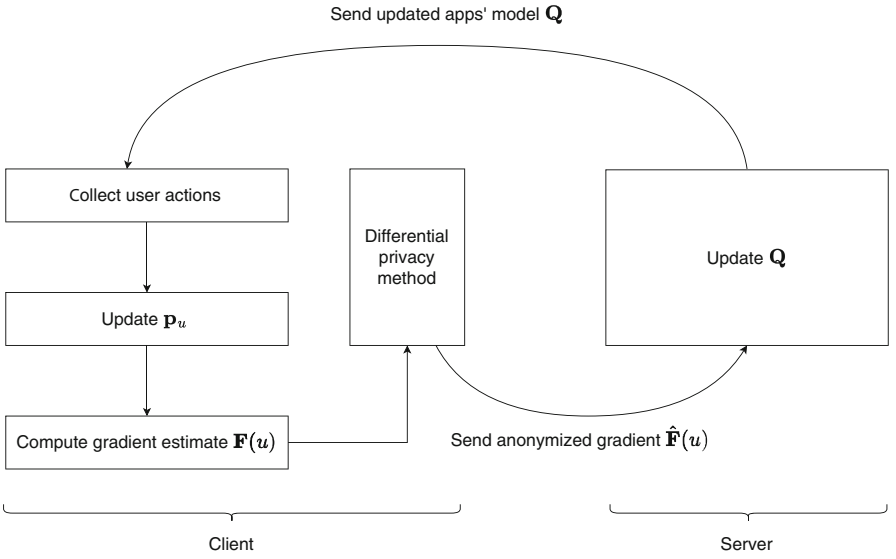

Fig. 1 The scheme of communications between server and clients and what data transmission is performed

the learning process the additional privacy-preserving mechanism is used before the transmission of the gradients.

The privacy-preserving matrix factorization methods have already been considered in studies. In particular, to perturb the gradients before transmission, the k-Harmony mechanism was incorporated in gradient aggregation privacy-preserving algorithm (Minto et al. 2021; Shin et al. 2018). Below we discuss the drawbacks of this mechanism and propose its modification for our setting called *QHarmony mechanism*. In addition, we discuss the Laplace mechanism (Dwork et al. 2014), the Kashin mechanism (Chen et al. 2020), and their drawbacks in the next section.

5.2 QHarmony mechanism

Consider gradient matrix $\mathbf{F}(u) = [f_{ij}(u)] \in \mathbb{R}^{N\times d}$ from user u, where N is a number of apps, d is a dimension of the embeddings. Further, we assume that $f_{ij}(u) \in [-1, 1]$, $i \in \overline{1, N}$, $j \in \overline{1, d}$, $u \in \overline{1, M}$. Since we can normalize the computed gradients, this assumption easily holds. The perturbed gradient matrix $\hat{\mathbf{F}}(u)$ is computed by the ϵ-LDP mechanism A from $\mathbf{F}(u)$ on the device. Then, a user transmits $\hat{\mathbf{F}}(u)$ to the remote server with privacy guarantees. On the server side, the received perturbed gradients $\hat{\mathbf{F}}(u)$, $u \in U_b$ are aggregated and, if needed, the additional transformation is performed on the aggregated update. According to the standard aggregation scheme for mean estimation, $\bar{\mathbf{F}} = [\bar{f}_{ij}]$ is computed as follows:

$$\bar{\mathbf{F}} = \frac{1}{|U_b|} \sum_{u\in U_b} \hat{\mathbf{F}}(u). \tag{15}$$

This way of aggregating gradients aims to reduce the noise from the received gradients $\hat{\mathbf{F}}(u)$ and therefore a sufficiently large number of users M is needed. However, in our setup the number of users is approximately the same as the number of elements in the gradient matrix; thus, the baseline ϵ-LDP mechanisms are not appropriate for such a setting. In particular,

- k-Harmony mechanism makes perturbed gradients sparse, since only k elements are transmitted from every user. The choice of k is based on the requirements of low privacy budget ϵ and high privacy budget per transmitted element $\frac{\epsilon}{k}$. The higher the latter budget is, the less noisy perturbed gradient element $\hat{f}_{ij}$ is. Therefore, k should be small, e.g., $k < 10$. To compute $\bar{f}_{ij}$ only approximately $\frac{kM}{Nd} \approx k$ values are summed and since k is small the remote server cannot estimate $\bar{f}_{ij}$ sufficiently accurately.
- According to the Laplace mechanism (Dwork et al. 2014), the variance of the gradient perturbation is equal to $\frac{4N^2d^2}{\epsilon^2}$, which is a large number for low privacy budget, e.g., $\epsilon < 3$.
  At the same time, the standard deviation of aggregated gradient element $\bar{f}_{ij}$ is $\frac{2\sqrt{2}Nd}{\epsilon M} \approx \frac{2\sqrt{2}}{\epsilon}$. In the case of a low privacy budget, this standard deviation is comparable with the element $\bar{f}_{ij}$. Therefore, the estimate $\bar{\mathbf{F}}$ is inaccurate with the Laplace mechanism, too.
- Although the Kashin mechanism has the lowest asymptotic variance upper bound for the aggregated gradient $\bar{\mathbf{F}}$ compared with other mechanisms, it has excessive computational costs. Also, if the hidden constant in the asymptotic estimate is large, the Kashin mechanism gives an inaccurate estimate $\bar{\mathbf{F}}$.

Thus, in the considered setting, the baseline privacy-preserving mechanisms estimate the average gradient presented in Eq. 15 with a large variance, that leads to poor performance. To improve overall performance and preserve moderate computational cost, we modify the k-Harmony mechanism as follows. Since this mechanism can be considered as a quantization of $\bar{\mathbf{F}}$, we take the aggregation approach from the quantization methods. Thus, our privacy mechanism consists of the following steps. Firstly, user samples uniformly k different pairs of indexes (i_l, j_l), where $i_l \in \overline{1, N}$, $j_l \in \overline{1, d}$ and $l \in \overline{1, k}$. Then, user assigns zeros to all elements of $\hat{\mathbf{F}}(u)$ except $\hat{f}_{i_l j_l}(u)$, $l \in \overline{1, k}$, which are computed in the following way:

$$\mathbb{P}[\hat{f}_{i_l j_l}(u) = x] = \begin{cases} \frac{f_{i_l j_l}(u)(e^{\epsilon/k}-1)+e^{\epsilon/k}+1}{2(e^{\epsilon/k}+1)}, & \text{if } x = 1 \\ -\frac{f_{i_l j_l}(u)(e^{\epsilon/k}-1)+e^{\epsilon/k}+1}{2(e^{\epsilon/k}+1)}, & \text{if } x = -1. \end{cases} \tag{16}$$

Equation 16 is a generalization of the corresponding equation for the Harmony mechanism. Here instead of transferring the single element, we transfer k elements at once and replace the transmitted values with ± 1 for further aggregation in the server. In the final step, user transmits k triples $(\hat{f}_{i_l j_l}(u), i_l, j_l)$ and $f_{\max}(u) = \max\limits_{i,j} f_{ij}(u)$ to the remote server. Further, we observe that the aggregated gradient $\bar{\mathbf{F}}$ on the server is a quantized representation of the non-private gradient estimate presented in Eq. 13.

Based on this observation, we replace the baseline aggregated formula given in Eq. 15 with the following expression:

$$\bar{\mathbf{F}} = \max_{u} f_{\max}(u) \sum_{u \in U_b} \frac{\hat{\mathbf{F}}(u)}{\max\limits_{ij} \sum\limits_{u \in U_b} \left[\hat{f}_{ij}(u) > 0 \right]}, \tag{17}$$

where $f_{\max}(u) = \max\limits_{ij} f_{ij}(u)$ and $f_{\max}(u)$ is transmitted to server by every user. We called this new privacy-preserving algorithm *QHarmony* and summarized it for the user and server sides in Algorithms 1 and 2, respectively. Note that the proposed QHarmony algorithm does not induce significant overhead in computational and memory complexity of the considered federated learning framework compared to analysis from Sect. 4.3.

Algorithm 1 QHarmony mechanism: Client side

Require: $\mathbf{F}(u)$ is the computed gradient on the user u device, ϵ is a privacy budget, k is a number of transmitted nonzero values.

Ensure: $\mathcal{I}(u)$ is set of k triples $(\hat{f}_{i_l j_l}(u), i_l, j_l)$ corresponding to each nonzero position in the perturbed gradient $\hat{\mathbf{F}}(u)$ on the u-th user device, $f_{\max}(u)$ is the maximum element of $\mathbf{F}(u)$.

function QHarmonyC($\mathbf{F}(u), \epsilon, k$)
  Compute $f_{\max}(u) = \max\limits_{ij} f_{ij}(u)$
  Initialize: $\mathcal{I}(u) = \emptyset$
  Initialize: $\mathcal{P} = \{i\}_{i=1}^{N} \times \{j\}_{j=1}^{d}$
  for $l = 1, \ldots, k$ **do**
    Sample uniformly at random (i_l, j_l) from $\mathcal{P}$
    Sample random number $b \sim Bernoulli\left(\frac{f_{i_l j_l}(u)(e^{\epsilon/k}-1)+e^{\epsilon/k}+1}{2(e^{\epsilon/k}+1)}\right)$
    $\hat{f}_{j_l i_l}(u) = 2b - 1$
    $\mathcal{P} = \mathcal{P}(u) \,/\, \{(i_l, j_l)\}$
    $\mathcal{I} = \mathcal{I}(u) \cup \{(\hat{f}_{i_l j_l}(u), i_l, j_l)\}$
  end for
return $\mathcal{I}(u)$, $f_{\max}(u)$
end function

6 Experiments

6.1 Data

Experiments were carried out on two mobile app usage datasets that are publicly available. LSApp dataset (Aliannejadi et al. 2021) is a collection of cross-app mobile search queries where the authors collected sequential app usage events of the users. It contains $N = 87$ unique apps from $M = 292$ users. In App Usage dataset (Yu et al. 2018), each entry contains an anonymized user identification, timestamps of HTTP request or response, etc. It contains $N = 2000$ unique apps from $M = 1000$ users. We

Algorithm 2 QHarmony mechanism: Server side

Require: For every user $u \in U_b$, $\mathcal{I}(u)$ is a set of k triples $(\hat{f}_{i_l j_l}(u), i_l, j_l)$ corresponding to each nonzero element in the perturbed gradient $\hat{\mathbf{F}}(u)$ from the u-th user, $f_{\max}(u)$ is the maximum element of $\mathbf{F}(u)$.
Ensure: $\bar{\mathbf{F}}$ is the resulting perturbed gradient in the remote server for the matrix $\mathbf{Q}$ update.

function QHarmonyS($\mathcal{I}(u), f_{\max}(u), u \in U_b$)
  Initialize: $\mathbf{S} = \{0\}^{N \times d}$, $\mathbf{Z} = \{0\}^{N \times d}$
  for $u \in U_b$ **do**
    for $(\hat{f}_{i_l j_l}(u), i_l, j_l) \in \mathcal{I}(u)$ **do**
      $s_{i_l j_l} = s_{i_l j_l} + \hat{f}_{i_l j_l}(u)$ ▷ Aggregate gradients
      $z_{i_l j_l} = z_{i_l j_l} + [\hat{f}_{i_l j_l}(u) > 0]$
    end for
  end for
  $z_{\max} = \max_{ij} z_{ij}$
  $\bar{\mathbf{F}} = \frac{\max_u f_{\max}(u)}{z_{\max}} \mathbf{S}$
return $\bar{\mathbf{F}}$
end function

noticed an excessive redundancy in both considered datasets: the same app's launch event can occur multiple times in less than 3 s. We believe, this is non-representative of real user behavior and collapse such cases into a single event.

Moreover, we split user actions into sessions based on a time interval between consequent app launches. If in the current session, the time delay after the last seen event exceeds a predefined threshold (15 min), a new session starts. The average number of sessions by users for LSApp and for App Usage is 109 and 58, respectively. Session splitting is only used during evaluation and does not affect the training process. During training, we consider only the sequential nature of data and represent all user history as a long session.

6.2 Evaluation methodology

6.2.1 Top-*n* prediction

The main goal is to predict the next app that is available on a user's device, not in the entire app store. Hence, we restrict the possible predictions to the apps installed on the user's device. For example, if a user history contains interactions with 5 unique apps, we will require a model to generate prediction scores for these 5 apps only. Then, for top-n recommendations with $n = 3$, we select 3 apps with the highest predicted scores among those 5 apps. Note that the number of apps $|A_u|$ in user u history can be lower than n. Hence, we compute top-min($|A_u|, n$) predictions and still assign the result to top-n metrics.

6.2.2 Evaluation protocol

Within each session, we utilize *iterative revealing scheme* (Ludewig and Jannach 2018) for evaluating the quality of predictions. This scheme means that for a particular user u and session s we go iteratively through apps in s and predict the next app taking

all the previous apps into account. Hence, for a session of length l we successively generate $l - 1$ predictions. We use both relevance and ranking-related metrics like hit rate (HR@n) (Zhang et al. 2018), mean reciprocal rank (MRR@n) (Nikolakopoulos et al. 2015), and normalized discounted cumulative gain (NDCG@n) (Nikolakopoulos et al. 2015), where n corresponds to the number of recommended items. Since our setting slightly differs from the standard one, we modify these metrics respectively but preserve their properties. In particular, the averaging is performed not only by the number of users but also by sessions.

6.3 Methods for comparison

Matrix Factorization. This is the original non-sequential MF-based approach proposed in Ammad-Ud-Din et al. (2019).

Sequential Rules. We use a Sequential Rules (SR) predictor (Ludewig and Jannach 2018) based on the co-occurrence frequency of apps going immediately one after another in user history. This is a direct modification of the standard item-to-item approach based on the Markov Chain assumption. After the statistics of such pair-wise sequential co-occurrences are collected, we use them directly without normalization to assign prediction scores for the next item. Note that there are two possible variations of the SR model depending on the way data are collected. The model can be learned either in a collaborative or in an isolated (on-device) manner. We denote the former as *SR* and the latter as *SR-od*.

Most Recently Used. MRU predictor assigns scores based on the recency of items in the current session. To comply with evaluation protocol (see Sect. 6.2), items that do not belong to requested items get zero scores. The nonzero scores are lower for older items and higher for newer ones, which pushes recent session items to the top in generated predictions.

Most Frequently Used. MFU predictor computes item popularity scores based on event frequencies for each user individually. These scores are used to select the top most frequently used items for prediction.

Random. Finally, this predictor assigns random scores to all requested items.

It is worthwhile to say that despite there are more complex models like Zhao et al. (2019), Shen et al. (2019) *the main purpose of this study is to develop a privacy-preserving method for the on-device next app prediction that outperforms well-known model in real-world scenario*. Therefore, approaches based on the privacy-preserving federated learning of neural networks are out of the scope of this study and the subject of future work.

6.4 Experimental methodology

6.4.1 Static environment

The static environment (He and McAuley 2016) aims to evaluate how well the SeqMF model is able to process sequential data compared to standard baselines. In particular, in this environment, the standard splitting on the train, test, and validation subsets is

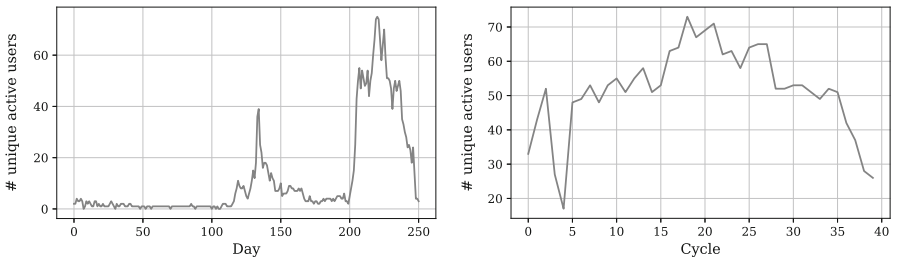

Fig. 2 Comparison of the original statistics from the LSApp dataset (left) and statistics after rearrangement into cycles (right)

used for training and evaluation of the app prediction model. The validation subset is used to tune hyperparameters. In particular, we tune a dimension of users' and apps' embeddings d, the learning rate of optimizer $\beta > 0$ from Eq. 11, regularization coefficient $\lambda > 0$, and γ from Eq. 9. In the LSApp dataset, the train set is the first 222 days, the validation set is the next 7 days, and the test subset is the last 14 days. For App Usage data, we assign the first 6 days to the training subset, the next day is assigned to the validation subset, and the last day is assigned to the test subset.

6.4.2 Dynamic environment

Dynamic environment corresponds to the behavior of the model in a real-world setting, where the model is built into the mobile OS and starts working on a new device. This setting is similar to the pure cold-start problem. To achieve high performance, the model has to adapt to user behavior quickly and start generating reasonable predictions. Dynamic environment assumes that we have pretrained apps' embeddings $\mathbf{Q}$ and randomly initialized users' embeddings $\mathbf{p}_u$ stored on devices. Then embeddings are updated from the online data stream. For example, consider a data stream that is generated over 5 days. In a dynamic environment, we evaluate the initialization of the SeqMF model with the data from the first day. Then, we update embeddings from these data and evaluate the model on the data from the second day and then use the data from the second day to update embeddings. This procedure continues till the fifth day. In the end, we get five numbers that show the dynamic behavior of the model's quality in terms of the chosen metrics.

One more important factor in the dynamic environment setting is the distribution of users' activities over time. If this distribution is unbalanced, then the learning process is unstable due to a noisy gradient estimated on small fractions of data. We observe such a case in the LSApp dataset, see Fig. 2 (left). To make the considered dataset balanced, we artificially rearrange users' actions in the *cycles* such that the number of active users is almost the same in every cycle, see Fig. 2 (right). Each cycle may contain one or several days, depending on user activity. At the same time, the order of users' actions was preserved during such rearrangement. In order to get pretrained apps' embeddings $\mathbf{Q}$, we take the first cycle for training the SeqMF model. The further cycles are composed of a data stream for the dynamic environment.

6.5 Results

6.5.1 Static environment

A comparison of the proposed SeqMF model and competitors described above in the static environment is presented in Table 1 for LSApp and in Table 2 for App Usage data. We highlight every metrics's best and second-best values with bold and underlined font, respectively. We see that SeqMF significantly outperforms its non-sequential predecessor MF on both datasets, which is expected given the sequential nature of the data. However, it underperforms the simple SR-od model on LSApp dataset and SR model on App Usage data in all the metrics. The SR models are known to be generally strong baselines and perform well on various datasets (Ludewig and Jannach 2018). It should be noted that there are at least two reasons for such results. Firstly, there are a lot of repetitive events in the datasets, including long sequences consisting of the same app. Thus, even despite our attempts to filter out such anomalies, they induce a strong bias toward duplicated events. Another reason is that once enough statistical data is collected, the sequential patterns become close to stationary and easier to uncover. Thus, we present comparison results in a dynamical environment that corresponds to the real-world scenarios in Sect. 6.5.2.

6.5.2 Dynamic environment: regular update of the user embeddings is important

In this section, we investigate the impact of long-term and short-term components of the prediction rule presented in Eq. 4 on model performance. We carry out experiments using setup from Sect. 6.4.2.

We consider three regimes of the SeqMF model training: Full, Rare, and Global. In these regimes, apps' embeddings $\mathbf{Q}$ are updated every 10 cycles for the LSApp dataset and every two cycles for the App Usage dataset. In the Full regime, users' embeddings $\mathbf{p}_u$ are updated every cycle. In the Rare regime, users' embeddings $\mathbf{p}_u$ are updated only with the updates of apps' embeddings $\mathbf{Q}$. In the Global regime, users' embeddings $\mathbf{p}_u$ are never updated and remain the same after random initialization. SR-od model is retrained in each cycle from the data collected in previous cycles.

To evaluate the performance of the model in a dynamic environment, we introduce an additional metric which is denoted by δHR@5. This metric is equal to the difference between HR@5 computed from the target training regime and HR@5 computed from the SeqMF model trained in the Full regime since we assume it is the most representative. This difference demonstrates how the model performance can degrade with improper usage of the dynamically updated data on user experience with apps. The smaller δHR@5 is, the worse the model follows the updated data from the clients. Therefore, such a metric evaluates the performance of the considered models in a dynamic regime adequately. In the plots below, we provide the cumulative δHR@5 over the data stream to illustrate the impact of the users' embeddings updates schedule on the SeqMF model performance. We also plot the performance of the SR-od model since it shows high performance in the static environment (see Tables 1 and 2) and uses only the local data from the device. The latter feature makes the SR-od model appropriate in the considered setting which is close to the real-world scenario.

Table 1 Static evaluation results on the LSApp dataset

Models	HR@1	HR@3	HR@5	MRR@1	MRR@3	MRR@5	NDCG@1	NDCG@3	NDCG@5
Random	0.089	0.239	0.365	0.089	0.153	0.181	0.089	0.175	0.227
MRU	**0.618**	0.823	0.849	**0.618**	**0.716**	**0.722**	**0.618**	**0.744**	0.754
MFU	0.403	0.669	0.777	0.403	0.516	0.541	0.403	0.555	0.600
SR	0.610	0.801	0.863	0.610	0.695	0.709	0.610	0.723	0.748
SR-od	0.612	**0.824**	**0.893**	0.612	0.707	**0.722**	0.612	0.737	**0.765**
MF	0.379	0.649	0.784	0.379	0.497	0.527	0.379	0.536	0.591
SeqMF	0.522	0.797	0.871	0.522	0.643	0.660	0.522	0.683	0.713

Table 2 Static evaluation results on the App Usage dataset

Models	HR@1	HR@3	HR@5	MRR@1	MRR@3	MRR@5	NDCG@1	NDCG@3	NDCG@5
Random	0.029	0.093	0.152	0.029	0.056	0.069	0.029	0.065	0.090
MRU	0.393	0.660	0.707	0.393	0.516	0.527	0.393	0.553	0.572
MFU	0.229	0.436	0.546	0.229	0.319	0.344	0.229	0.349	0.394
SR	**0.477**	**0.664**	**0.746**	**0.477**	**0.559**	**0.578**	**0.477**	**0.586**	**0.620**
SR-od	0.414	0.591	0.659	0.414	0.493	0.508	0.414	0.518	0.546
MF	0.227	0.426	0.520	0.227	0.314	0.336	0.227	0.343	0.382
SeqMF	0.402	0.615	0.686	0.402	0.497	0.513	0.402	0.527	0.557

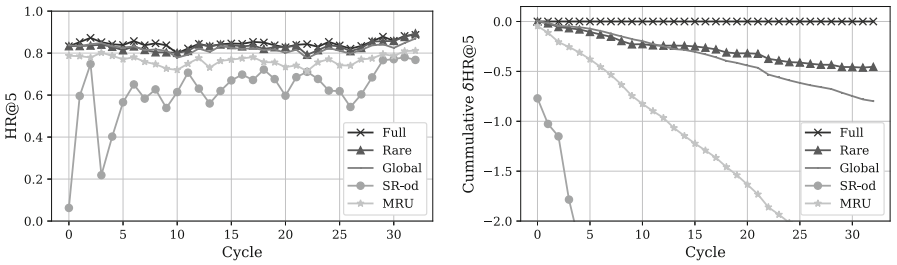

Fig. 3 LSApp dataset. HR@5 metric (left) and cumulative δHR@5 (right) for the considered training regimes of the SeqMF model, SR(od) model and MRU model in the dynamical environment. The SeqMF model in the considered training regimes outperforms the SR-od model and MRU model in the dynamic environment. The right plot shows that regular updates of user embeddings improve performance

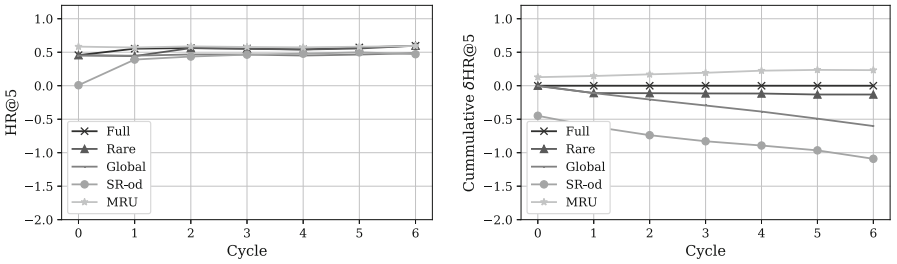

Fig. 4 App Usage dataset. HR@5 metric (left) and cumulative δHR@5 (right) for the considered training regimes of the SeqMF model and SR(od) model in the dynamical environment. The SeqMF model in the considered training regimes outperforms the SR-od model in the dynamical environment only if user embeddings are updated (Full and Rare regimes). The right plot shows that regular updates of user embeddings improve performance. MRU model is slightly more accurate than the SeqMF model in terms of cumulative δHR@5 (right) and almost equal to the SeqMF model in terms of the original HR@5 metric (left)

Figure 3 illustrates that the SeqMF model *outperforms the SR-od model and MRU model on LSApp data in the dynamic environment*. This observation confirms our hypothesis about the reasons for the high performance of SR-based models and the MRU model in the static environment. As we can see, if we do not update users' embeddings on the device at all (Global regime), then the overall performance of the SeqMF model decreases. However, if we add just rare updates of users' embeddings (Rare regime), we can see that the performance increases compared to the SeqMF model in the Global regime. Thus, we demonstrated experimentally that updates of users' embeddings $\mathbf{p}_u$ are essential to preserving the high performance of the SeqMF model.

Figure 4 shows the performance of the SeqMF model in the dynamic environment for the App Usage dataset. From the left plot follows that the performance of the SeqMF model in Full and Rare regimes is very similar. The possible reason for such a trend is that the first cycle contains enough information about patterns in user behavior. Hence, it does not matter how often the users' embeddings are updated for this particular dataset. In addition, the SeqMF model shows similar performance to the SR-od model and slightly less accurate predictions compared to the MRU model in terms of the cumulative δHR@5 metric. This observation indicates that the users from the App

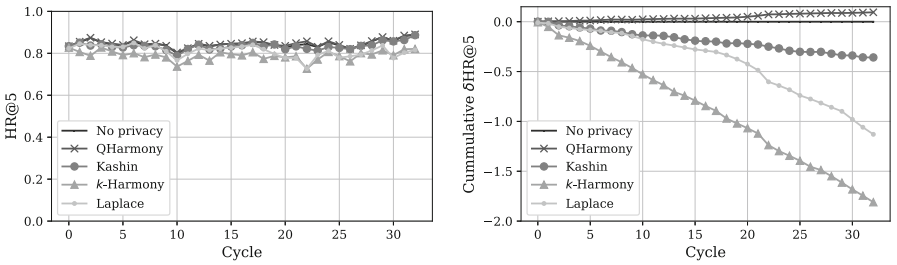

Fig. 5 LSApp dataset. Left: Dependence of the HR@5 on the privacy-preserving mechanisms with privacy budget $\epsilon = 4.5$ incorporated in the SeqMF model. The higher HR@5 is, the better the performance of the model is. Therefore, the QHarmony mechanism provides stronger protection for sensitive users' data. Right: The cumulative δHR@5 presented for 33 cycles confirms that the QHarmony mechanism does not lead to performance decreasing compared to the non-private SeqMF model

Usage dataset may run repeated apps and have specific personal patterns such that collaborative information does not improve prediction quality.

6.5.3 Dynamic environment: QHarmony mechanism outperforms competitors

To analyze the impact of privacy mechanisms on the SeqMF model performance, we use a dynamic environment similar to the previous section and a Full training regime. We compare the performance of the SeqMF model without a privacy mechanism with the same model accompanied by the considered baselines and the proposed QHarmony mechanism.

Figure 5 shows that the SeqMF model with the proposed QHarmony algorithm and privacy budget $\epsilon = 4.5$ has an even higher utility on the LSApp dataset compared to the SeqMF model without any privacy mechanism. The possible reason for such an effect is the well-known implicit regularization property of stochastic gradient observed previously in deep learning studies (Neelakantan et al. 2015). At the same time, we observe the poor performance of the baseline privacy mechanisms except for the Kashin mechanism as was expected in Sect. 5.2. Kashin mechanism still has the worse privacy-utility trade-off and is much more computationally costly than the proposed QHarmony mechanism.

Figure 6 shows that the proposed QHarmony mechanism with budget $\epsilon = 1.1$ preserved almost the same level of HR@5 compared to the non-private SeqMF model on App Usage data. As in the case of the LSApp data, baseline privacy mechanisms have a worse privacy-utility trade-off compared to the proposed QHarmony mechanism. These experiments confirm the effectiveness of the proposed QHarmony mechanism and demonstrate that the performance of the SeqMF model is preserved after incorporating the QHarmony mechanism for both considered datasets.

7 Discussion

The presented above evaluation of the proposed model and privacy-preserving approach to fitting provides the following insights and implications. The introduced

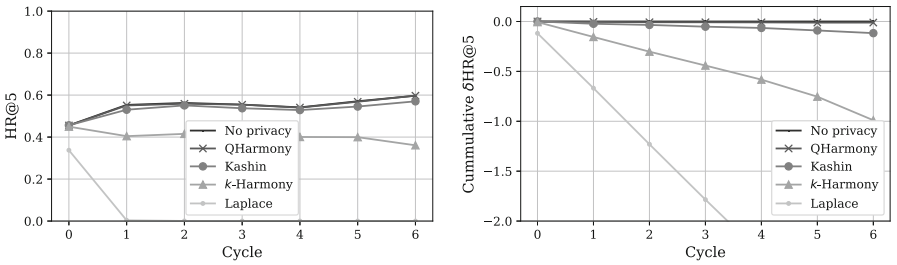

Fig. 6 App Usage dataset. Left: Dependence of the HR@5 on privacy-preserving mechanisms with privacy budget $\epsilon = 1.1$ incorporated to the SeqMF model. The higher HR@5 is, the better the performance of the model is. The QHarmony mechanism provides stronger protection for sensitive users' data. Right: The cumulative δHR@5 presented for 7 cycles confirms that the QHarmony mechanism does not lead to performance decreasing compared to the non-private SeqMF model

modification of the classical Matrix Factorization model treats the sequential nature of the data properly and improves the prediction in the static regime. Also, fitting the proposed SeqMF model in the federated learning manner shows that the stochastic gradient optimizer used for updating the apps' embeddings preserves the prediction quality and does not lead to a significant drop in the prediction quality compared to the on-device models. Moreover, the proposed privacy-preserving QHarmony algorithm induces moderate noise to the gradient estimate and may lead to implicit regularization that even increases the prediction quality in the dynamic regime. Thus, the proposed SeqMF model is appropriate for both federated learning and privacy-preserving setup, which is crucial for real-world applications.

Limitations.

The main limitation of the presented study is the absence of publicly available sufficiently large datasets with run app sequences. Other limitations include long connection breaks between clients and a server that prevent updating global embeddings and the absence of regular patterns in users' interactions with installed apps since this is our main assumption. Also, the on-device memory consumption may be an issue during incorporating our model in a real-world setting. In the next paragraph, we briefly mention what future research is needed to fix this problem.

Future work.

Since our study proposes both modification of the classical matrix factorization model and the corresponding differential privacy algorithm, we describe future work below in both directions. First, the mentioned limitation related to memory consumption can be treated with a proper quantization of the SeqMF model parameters. The effect of such quantization on the SeqMF model performance can be a topic of further research. Second, the modification of the standard matrix factorization model can be further generalized to a tensor factorization model. Additional dimensions can incorporate context data or sequential information like the explicit position of the app in recent history. Also, one can analyze more advanced federated learning methods for this particular problem and derive theoretical convergence guarantees. Third, a detailed theoretical analysis of the proposed QHarmony privacy algorithm can be performed. Such theoretical results could help to control the noise level in the gradient estimation

used to update apps' embeddings. Also, the communication efficiency of the proposed privacy-preserving QHarmony algorithm can be considered as a part of future work to schedule the intensity of the communications with users and avoid redundancy in the received data.

8 Conclusion

In this study, we considered the next app prediction problem and proposed a new sequence-aware matrix factorization model called SeqMF. The model can predict the next app for every user which can be used to optimize the resource consumption on the user's device. The SeqMF model takes into account the sequential nature of data, is accompanied by a new QHarmony mechanism for stronger privacy guarantees, and is trained in the federated learning paradigm. This paradigm means that users' embeddings are stored locally on the user devices and apps' embeddings are the same for all devices and are updated in the remote server. These features make the SeqMF model robust to potential leakage of the transmitted data from user devices to the remote server and help in capturing behavior patterns common for multiple users. To illustrate the performance of our model we carried out experiments on the LSApp and App Usage datasets. In these experiments, the real-world setting of data flow was emulated to evaluate considered models properly. Our experiments demonstrate that the SeqMF model outperforms the standard matrix factorization model and other competitors based on the collected statistics. Also, the proposed QHarmony privacy mechanism shows the best privacy-utility trade-off compared to the competitors and moderate computational complexity. Thus, the proposed SeqMF model accompanied by the QHarmony privacy mechanism provides practical advantages in real-world applications over the previously proposed solutions for the next app prediction task.

Funding This work was supported by the Russian Science Foundation under Grant 22-21-00911.

Declarations

Conflict of interest The authors have no relevant financial or non-financial interests to disclose.

References

AbdulRahman, S., Tout, H., Ould-Slimane, H., et al.: A survey on federated learning: the journey from centralized to distributed on-site learning and beyond. IEEE Internet Things J. **8**(7), 5476–5497 (2020)

Akiba, T., Sano, S., Yanase, T., et al.: Optuna: a next-generation hyperparameter optimization framework. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 2623–2631 (2019)

Aliannejadi, M., Zamani, H., Crestani, F., et al.: Context-aware target apps selection and recommendation for enhancing personal mobile assistants. ACM Trans. Inf. Syst. (TOIS) **39**(3), 1–30 (2021)

Ammad-Ud-Din, M., Ivannikova, E., Khan, S.A., et al.: Federated collaborative filtering for privacy-preserving personalized recommendation system. arXiv preprint arXiv:1901.09888 (2019)

Baeza-Yates, R., Jiang, D., Silvestri, F., et al.: Predicting the next app that you are going to use. In: Proceedings of the Eighth ACM International Conference on Web Search And Data Mining, pp. 285–294 (2015)

Changmai, B.M., Nagaraju, D., Mohanty, D.P., et al.: On-device user intent prediction for context and sequence aware recommendation. arXiv preprint arXiv:1909.12756 (2019)

Chen, W.N., Kairouz, P., Ozgur, A.: Breaking the communication-privacy-accuracy trilemma. Adv. Neural. Inf. Process. Syst. **33**, 3312–3324 (2020)

Costan, V., Lebedev, I., Devadas, S.: Sanctum: minimal hardware extensions for strong software isolation. In: 25th USENIX Security Symposium (USENIX Security 16), pp. 857–874 (2016)

Duchi, J.C., Wainwright, M., Jordan, M.I.: Minimax optimal procedures for locally private estimation. J. Am. Stat. Assoc. **113**, 182–201 (2016)

Dwork, C., Kenthapadi, K., McSherry, F., et al.: Our data, ourselves: privacy via distributed noise generation. In: Advances in Cryptology-EUROCRYPT 2006: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28–June 1, 2006. Proceedings 25, pp. 486–503. Springer (2006a)

Dwork, C., McSherry, F., Nissim, K., et al.: Calibrating noise to sensitivity in private data analysis. In: Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4–7, 2006, pp 265–284. Proceedings 3. Springer (2006b)

Dwork, C., Roth, A., et al.: The algorithmic foundations of differential privacy. Found. Trends® Theor. Comput. Sci. **9**(3–4), 211–407 (2014)

Ekstrand, M.D., Riedl, J.T., Konstan, J.A., et al.: Collaborative filtering recommender systems. Found. Trends® Hum. Comput. Interact. **4**(2), 81–173 (2011)

Erlingsson, Ú., Pihur, V., Korolova, A.: Rappor: randomized aggregatable privacy-preserving ordinal response. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, pp. 1054–1067 (2014)

Fang, H., Zhang, D., Shu, Y., et al.: Deep learning for sequential recommendation: algorithms, influential factors, and evaluations. ACM Trans. Inf. Syst. (TOIS) **39**(1), 1–42 (2020)

Frolov, E., Oseledets, I.: Tensor-based sequential learning via Hankel matrix representation for next item recommendations. IEEE Access **11**, 6357–6371 (2023)

Golub, G.H., Van Loan, C.F.: Matrix Computations. JHU Press, Baltimore (2013)

Gusak, J., Cherniuk, D., Shilova, A., et al.: Survey on efficient training of large neural networks. In: Proceedings of the 31st International Joint Conference on Artificial Intelligence IJCAI-22, Vienna, Austria, pp. 23–29 (2022)

Han, J., Ma, Y., Mei, Q., et al.: DeepRec: on-device deep learning for privacy-preserving sequential recommendation in mobile commerce. In: Proceedings of the Web Conference 2021 (WWW '21), April 19–23, 2021, Ljubljana, Slovenia (2021)

He, R., McAuley, J.: Fusing similarity models with Markov chains for sparse sequential recommendation. In: 2016 IEEE 16th International Conference on Data Mining (ICDM), pp. 191–200. IEEE (2016)

He, X., Zhang, H., Kan, M.Y., et al.: Fast matrix factorization for online recommendation with implicit feedback. In: Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 549–558 (2016)

Hidasi, B., Karatzoglou, A., Baltrunas, L., et al.: Session-based recommendations with recurrent neural networks. arXiv preprint arXiv:1511.06939 (2015)

Hu, Y., Koren, Y., Volinsky, C.: Collaborative filtering for implicit feedback datasets. In: 2008 Eighth IEEE International Conference on Data Mining, pp. 263–272. IEEE (2008)

Jannach, D., Ludewig, M., Lerche, L.: Session-based item recommendation in e-commerce: on short-term intents, reminders, trends and discounts. User Model. User-Adapt. Interact. **27**, 351–392 (2017)

Kairouz, P., McMahan, H.B., Avent, B., et al.: Advances and open problems in federated learning. Found. Trends® Mach. Learn. **14**(1–2), 1–210 (2021)

Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)

Li, H., Diao, X., Cao, J., et al.: Collaborative filtering recommendation based on all-weighted matrix factorization and fast optimization. IEEE Access **6**, 25,248-25,260 (2018)

Li, W., Xia, Y., Chen, H.: Research on arm trustzone. GetMobile Mob. Comput. Commun. **22**(3), 17–22 (2019)

Liao, Z.X., Li, S.C., Peng, W.C., et al.: On the feature discovery for app usage prediction in smartphones. In: 2013 IEEE 13th International Conference on Data Mining, pp. 1127–1132. IEEE (2013)

Ludewig, M., Jannach, D.: Evaluation of session-based recommendation algorithms. User Model. User-Adapt. Interact. **28**(4–5), 331–390 (2018)
Minto, L., Haller, M., Livshits, B., et al.: Stronger privacy for federated collaborative filtering with implicit feedback. In: Proceedings of the 15th ACM Conference on Recommender Systems, pp. 342–350 (2021)
Neelakantan, A., Vilnis, L., Le, Q.V., et al.: Adding gradient noise improves learning for very deep networks. arXiv preprint arXiv:1511.06807 (2015)
Nguyên, T.T., Xiao, X., Yang, Y., et al.: Collecting and analyzing data from smart device users with local differential privacy. arXiv preprint arXiv:1606.05053 (2016)
Nikolakopoulos, A.N., Kalantzis, V., Garofalakis, J.D.: Eigenrec: An efficient and scalable latent factor family for top-n recommendation. arXiv preprint arXiv:1511.06033 47 (2015)
Ouyang, Y., Guo, B., Wang, Q., et al.: Learning dynamic app usage graph for next mobile app recommendation. IEEE Trans. Mob. Comput. (2022). https://doi.org/10.1109/TMC.2022.3161114
Parate, A., Böhmer, M., Chu, D., et al.: Practical prediction and prefetch for faster access to applications on mobile phones. In: Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing, pp. 275–284 (2013)
Qian, N.: On the momentum term in gradient descent learning algorithms. Neural Netw. **12**(1), 145–151 (1999)
Rendle, S., Freudenthaler, C., Schmidt-Thieme, L.: Factorizing personalized Markov chains for next-basket recommendation. In: Proceedings of the 19th International Conference on World Wide Web, pp. 811–820 (2010)
Reyzin, L., Smith, A., Yakoubov, S.: Turning hate into love: compact homomorphic ad hoc threshold encryption for scalable MPC. In: International Symposium on Cyber Security Cryptography and Machine Learning, pp. 361–378. Springer (2021)
Shen, Z., Yang, K., Du, W., et al.: Deepapp: a deep reinforcement learning framework for mobile application usage prediction. In: Proceedings of the 17th Conference on Embedded Networked Sensor Systems, pp. 153–165 (2019)
Shin, C., Hong, J.H., Dey, A.K.: Understanding and prediction of mobile application usage for smart phones. In: Proceedings of the 2012 ACM Conference on Ubiquitous Computing, pp. 173–182 (2012)
Shin, H., Kim, S., Shin, J., et al.: Privacy enhanced matrix factorization for recommendation with local differential privacy. IEEE Trans. Knowl. Data Eng. **30**(9), 1770–1782 (2018)
Warner, S.: Randomized response: a survey technique for eliminating evasive answer bias. J. Am. Stat. Assoc. **60**(309), 63–6 (1965)
Xu, Y., Lin, M., Lu, H., et al.: Preference, context and communities: a multi-faceted approach to predicting smartphone app usage patterns. In: Proceedings of the 2013 International Symposium on Wearable Computers, pp. 69–76 (2013)
Yao, A.C.C.: How to generate and exchange secrets. In: 27th Annual Symposium on Foundations of Computer Science (Sfcs 1986), pp. 162–167. IEEE (1986)
Yu, D., Li, Y., Xu, F., et al.: Smartphone app usage prediction using points of interest. In: Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, vol. 1, no. 4, p. 174 (2018)
Zhang, C., Ding, X., Chen, G., et al.: Nihao: a predictive smartphone application launcher. In: Mobile Computing, Applications, and Services: 4th International Conference, MobiCASE 2012, Seattle, WA, USA, October 11–12, 2012, pp. 294–313. Revised Selected Papers 4, Springer (2013)
Zhang, S., Tay, Y., Yao, L., et al.: Next item recommendation with self-attention. arXiv preprint arXiv:1808.06414 (2018)
Zhao, S., Luo, Z., Jiang, Z., et al.: AppUsage2Vec: modeling smartphone app usage for prediction. In: 2019 IEEE 35th International Conference on Data Engineering (ICDE), pp. 1322–1333. IEEE (2019)
Zheng, W., Wu, Y., Wu, X., et al.: A survey of Intel SGX and its applications. Front. Comput. Sci. **15**, 1–15 (2021)

Albert Saiapin received his Graduate degree from the Skolkovo Institute of Science and Technology (Skoltech) in 2023. He is currently pursuing PhD research focused on Probabilistic Tensor Methods at Delft University of Technology. His main research areas include Recommender Systems, Natural Language Processing, and Tensor Networks. He is exploring various connections between general Artificial Intelligence methods and Tensor Linear Algebra in both industrial and academic research settings. Notably, one of his works was published in 2022 at a leading computer science conference, the Association for Computational Linguistics.

Gleb Balitskiy received the B.Sc. and M.Sc. degrees in applied mathematics and physics from the Moscow Institute of Physics and Technology (MIPT), Russia, in 2018 and 2020, respectively, the M.Sc. degree in data science from the Skolkovo Institute of Science and Technology (Skoltech), Russia, in 2020. He is currently a Ph.D. student at Skoltech. His research interests include information theory, coding theory, privacy, and its application for private and reliable communications.

Daniel Bershatsky received a Graduate degree from the Moscow Institute of Physics and Technology, in 2014, and a Master of Science degree from the Skolkovo Institute of Science and Technology (Skoltech), in 2023, under the supervision of Prof. Ivan Oseledets. He continues working towards a Ph.D. degree at Skoltech. His research interests cover topics in machine learning and efficient algorithms.

Aleksandr Katrutsa is a research scientist at the Skolkovo Institute of Science and Technology (Skoltech) and a senior research scientist at AIRI. He received a Graduate degree from Skoltech in 2016. Aleksandr received a Ph.D. degree in 2019 from the Moscow Institute of Physics and Technology. His research interests include deep learning, optimization, and numerical analysis. He has published research results at the top peer-reviewed journals and conferences, e.g. NeurIPS and IJCAI.

Evgeny Frolov received the Graduate degree from Lomonosov Moscow State University, in 2009, and the Ph.D. degree from the Skolkovo Institute of Science and Technologies (Skoltech), in 2018, under the supervision of Ivan Oseledets. After graduation, he started developing a career as an ICT Industry Professional but returned to academia several years later. He continues working at Skoltech and AIRI as a Research Scientist, where he leads both academic and industrial research projects. His research is concentrated on building better bridges between practical challenges arising in the field of recommender systems and theoretical advances in relevant mathematical disciplines. He is especially attracted by algebraic and geometric methods, but also has broader interests. Some of his work is published at the leading ACM Recommender Systems Conference (RecSys). He is also the co-author of a comprehensive survey on tensor methods in recommender systems.

Alexey Frolov received an M.Sc. degree in computer science from Bauman Moscow State Technical University (BMSTU) in 2010, a Ph.D. degree in mathematics from the Institute for Information Transmission Problems (IITP), Russian Academy of Sciences (RAS), in 2012 and the D.Sc. degree in mathematics from Moscow Institute of Physics and Technology (MIPT) in 2021. He is currently a Full Professor at the Skolkovo Institute of Science and Technology (Skoltech), Moscow, Russia. His research interests include information theory and its applications. He was a recipient of the IEEE GLOBECOM Communication Theory Symposium Best Paper Award in 2020, the Russian Government Award in Science and Technology for Young Scientists in 2016, and the Moscow Government Award for Young Scientists in 2013.

Ivan Oseledets received a Graduate degree from the Moscow Institute of Physics and Technology, in 2006, and the Candidate of Sciences and Doctor of Science degrees from the Marchuk Institute of Numerical Mathematics of the Russian Academy of Sciences, in 2007 and 2012, respectively. He joined Skoltech, in 2013 and AIRI in 2021. His research covers a broad range of topics. He proposed a new decomposition of high-dimensional arrays (tensors)-tensor-train decomposition and developed many efficient algorithms for solving high-dimensional problems. His current research interests include the development of new algorithms in machine learning and artificial intelligence, such as the construction of adversarial examples, the theory of generative adversarial networks, and the compression of neural networks. It resulted in publications in top computer science conferences, such as ICML, NIPS, ICLR, CVPR, RecSys, ACL, and ICDM. He is an Associate Editor of SIAM Journal on Mathematics of Data Science, SIAM Journal on Scientific Computing, and Advances in Computational Mathematics (Springer).

Vitaliy Kharin is an independent expert with broad experience in mobile OS development and embedded recommender systems. He worked in various engineering positions in telecom companies.