Successive Point-of-Interest Recommendation With Local Differential Privacy

JONG SEON KIM[1], **JONG WOOK KIM**[2], **(Member, IEEE), AND YON DOHN CHUNG**[1], **(Member, IEEE)**

[1]Department of Computer Science and Engineering, Korea University, Seoul 02841, South Korea
[2]Department of Computer Science, Sangmyung University, Seoul 03016, South Korea

Corresponding author: Yon Dohn Chung (ydchung@korea.ac.kr)

ABSTRACT A point-of-interest (POI) recommendation system performs an important role in location-based services because it can help people to explore new locations and promote advertisers to launch advertisements at appropriate locations. The existing POI recommendation systems require raw check-in history of users, which might cause location privacy violations. Although there have been several matrix factorization (MF) based privacy-preserving recommendation systems, they can only focus on user-POI relationships without considering the human movements in check-in history. To tackle this problem, we design a successive POI recommendation framework with local differential privacy, named SPIREL. SPIREL uses two types of information derived from the check-in history as input for the factorization: a transition pattern between two POIs and the visit counts of POIs. We propose a novel objective function for learning the user-POI and POI-POI relationships simultaneously. We further integrate local differential privacy mechanisms in our proposed framework to prevent potential location privacy breaches. Experiments using four public datasets demonstrate that SPIREL achieves better POI recommendation quality while accomplishing stronger privacy preservation.

INDEX TERMS Point-of-Interest, recommendation system, local differential privacy, matrix factorization.

I. INTRODUCTION

Smartphones have become an integral part of our everyday lives. In particular, smartphones have resulted in people sharing their daily check-in experiences through social network services, such as Facebook, Foursquare, and Instagram. Through these check-in data, it is possible to study the online activities, physical movements, and preferences on the points-of-interest (POI) of users. Accordingly, various location-based services (LBSs) utilize the check-in data to provide the best experiences for their services. Among the various tasks in LBSs, POI recommendation has attracted considerable attention in recent years [1]–[4].

Predicting the subsequent location of a mobile user is important as it can help them to explore interesting and unvisited places. To recommend new POIs for users, most of

the recommendation methods leverage the check-in history of users. As the check-in history embeds rich sequential patterns, it is intuitive to utilize it to decide on the next location of users. However, collecting the check-in data poses serious privacy concerns. For example, an earlier work [5] demonstrated that only four successive location points are sufficient to uniquely identify each individual. Alongside this kind of identity disclosure problem, it is also possible to infer sensitive information through the visited places. For example, [6] showed that they could infer sensitive information (e.g., sexual preferences, religions) through the trace data collected for tracing people infected with COVID-19.

The notion of local differential privacy (LDP) has attracted considerable attention in recent years from many industries owing to its rigorous and provable privacy guarantees. In an LDP setting, each user randomizes the original data in his/her device and sends the perturbed data to the server. As the original data remain within the device of the user, the server

cannot infer the sensitive information of the user, regardless of the background knowledge. Accordingly, many global IT companies, including Google [7], Apple [8], Microsoft [9], and Samsung [10], have already adopted LDP mechanism for data collection from their clients.

Several earlier works adopted differential privacy (DP) in the recommendation system for preserving the privacy of users [11]–[13]. Hua *et al.* [11] proposed a recommendation system based on the centralized DP (CDP) model. However, in the CDP model, the server is assumed to be trusted, and the original data of the user are collected at the recommendation server directly. Zhang *et al.* [12] proposed a recommendation system based on a personalized CDP model. Similarly, they assumed that the recommender is trusted, and the feedback of each user was randomly sampled and sent to the server. Finally, Shin *et al.* [13] proposed a recommendation system under LDP. In their work, the LDP mechanism was utilized by involving a stochastic gradient descent (SGD) step for updating the latent factors, where each user sends perturbed gradients to the recommendation server.

The existing privacy-preserving recommendation methods have two limitations in their solutions. First, the recommenders are not always trusted; thus, the check-in data from the users should not be submitted in their original form. However, [11] and [12] assumed that the recommenders are trusted. Second, all the existing works [11]–[13] are single-domain (SD) recommendation systems that focus on the user-POI relationship only. If we build a POI recommendation system based on the user-POI relationship alone, the system will highly recommend unattractive POI candidates without considering the sequential characteristics of the POI.

In this paper, we propose a novel private POI recommendation framework called SPIREL (**S**uccessive **POI RE**commendation with **L**DP). SPIREL recommends the next POI candidates by considering the preferences on POIs as well as the preferred transitions among POIs while preserving the privacy of users. To do this, SPIREL uses the matrix factorization (MF) [14] among many collaborative filtering (CF) techniques. SPIREL exploits two types of implicit interactions from the check-in history of the user. First, to represent the POI sequences, SPIREL uses transition patterns modeled with a first-order Markov chain. Specifically, each user is required to record a movement between two successive POIs. Second, SPIREL further requires to extract the visit counts for each POI from the check-in history of the user, which reflects the user-POI relationship. Subsequently, SPIREL jointly learns the user-POI and POI-POI relationships based on the visit counts and transition patterns, respectively.

The state-of-the-art privacy-preserving MF-based recommendation system under LDP [13] is designed for explicit feedback data, where users denote their "like" or "dislike" through a fixed scale rating. Accordingly, [13] focuses on reducing the amount of noise added on gradients without modeling the user preferences. On the other hand, lacking explicit ratings is the main problem in POI recommendation.

One way is to guess the preferences through the number of check-ins that only reveal positive interactions between users and POIs. However, visit counts alone cannot deeply explain the users' preferences for the next POIs. Thus, we measure POI transition preferences across users by mapping the estimated frequency of transition patterns to a confidence score. Moreover, we revise the objective function to integrate the confidence scores, such that a user's preference for the next POI is modeled as an inner product between visit counts and the confidence scores.

The main contributions of our work are summarized as follows.

1) To map the implicit feedback to the preferences of users, we sample a single POI-POI transition pattern from each user and estimate the frequency of whole transition patterns under LDP. Then, we transfer the estimated frequency of transition patterns into a confidence score to model the POI transition preferences.

2) During the learning process, our proposed framework simultaneously factorizes both user-POI and POI-POI matrices. To achieve this, we developed a new objective function and an optimization framework under LDP. Note that the existing methods are SD recommendation systems that can factorize only one user-POI matrix, which has limits in integrating additional knowledge from other domains.

3) In our framework, the users do not have to disclose their current location to receive the next POI candidates. When compared to most LBSs requiring the current location of the user, SPIREL is a significantly safer framework for successive POI recommendation.

4) We conducted extensive experiments using four public datasets and demonstrated that SPIREL improves the accuracy and quality of recommendations significantly compared to the existing privacy-preserving recommendation systems.

The remainder of this paper is organized as follows. In Section II, we explain the background knowledge. In Section III, we formally define the problem and describe the challenges. In Section IV, we propose our SPIREL framework for successive POI recommendation. Section V demonstrates the performance of SPIREL on public datasets. Section VI reviews the related work. Finally, in Section VII, we conclude this paper.

II. PRELIMINARIES
A. MATRIX FACTORIZATION
MF is used as a CF algorithm in recommendation systems [14]. In recent years, owing to its accurate prediction, many industries have adopted it for personalized advertisement targeting. MF decomposes the user-item matrix into two smaller matrices to discover the unobserved relationship between users and items. Each decomposed matrix embeds the latent factors of the user/item, which simplifies the complicated user/item characteristics. Then, by multiplying the two latent matrices, we can predict the unobserved

user-item relationships. In this study, we assume that the latent factors of the item represent the characteristics of POIs.

The major challenge involved in MF is the optimization process for finding the mapping between users and items. Let us assume that the elements of the user-item matrix indicate the observed preference scores of users on items. Then, the objective of MF is to determine the latent factors of the user and item, whose product becomes similar to the observed preference scores. To be specific, MF attempts to reduce the error between the observed preference scores and the predicted scores calculated over an inner product of the two latent factors. While minimizing these errors, the latent factors are fitted to uncover the preference scores of items that are not rated.

Formally, MF is defined as follows. Suppose there are m users and n items. We can denote r_{ij} as the preference score of user i for item j. Then, r_{ij} can be approximated by calculating the inner product $r_{ij} \approx \mathbf{u_i}^\mathsf{T}\mathbf{v_j}$. The objective of MF is to minimize the error between r_{ij} and $\mathbf{u_i}^\mathsf{T}\mathbf{v_j}$. In Table 1, we have listed the set of notations used throughout this paper. Unless otherwise stated, we assume that all vectors are column vectors.

TABLE 1. Notations.

Notation	Meaning
m	number of users
n	number of POIs
d	size of latent factors
$\mathbf{u_i} \in \mathbb{R}^d$	profile vector of user i
$\mathbf{v_j} \in \mathbb{R}^d$	profile vector of POI j
$U \in \mathbb{R}^{m \times d}$	user profile matrix
$V \in \mathbb{R}^{n \times d}$	POI profile matrix
$P \in \mathbb{R}^{m \times n}$	user-POI matrix
$Q \in \mathbb{R}^{n \times n}$	POI-POI matrix
$P_{i,*}$	ith row vector of matrix P
$P_{*,j}$	jth column vector of matrix P

Generally, as users rate only a small set of items, we have a considerably limited number of observed scores. Thus, the user-item matrix P is significantly sparse, which implies that most of the elements in P are unknown. Therefore, many of the recent studies performed optimization with only the observed scores while avoiding overfitting by introducing a regularization term. Specifically, MF attempts to minimize the following objective function.

$$\mathcal{L} = \sum_{(i,j)\in P}(r_{ij} - \mathbf{u_i}^\mathsf{T}\mathbf{v_j})^2 + \lambda(\sum_{i=1}^{m}||\mathbf{u_i}||^2 + \sum_{j=1}^{n}||\mathbf{v_j}||^2) \quad (1)$$

Here, the user profile vector $\mathbf{u_i}$ and item profile vector $\mathbf{v_j}$ are represented by a d-dimensional vector. Furthermore, λ is a regularizer used for avoiding the overfitting problem. In summary, by minimizing the mean square error over the known preference scores, we can predict the unobserved user-item relationships.

There are primarily two optimization techniques to minimize the objective function: (1) SGD and (2) alternating least square (ALS). SGD first predicts r_{ij} and computes associated prediction error for each observed case. Then, SGD derives the gradient from the corresponding latent factors $\mathbf{u_i}$ and $\mathbf{v_i}$, which indicates the direction of the greatest rate of increase of the objective function. The gradient is multiplied with a learning rate γ, which determines the change in the profile vector with respect to the gradient. Subsequently, SGD performs steps in the inverse direction of the gradient to minimize the objective function. We describe the update rule of the user profile vector below (the item profile vector can be updated similarly).

$$\nabla_{\mathbf{u_i}}\mathcal{L} = \sum_{(i,j)\in P} -2\mathbf{v_j}(r_{ij} - \mathbf{u_i}^\mathsf{T}\mathbf{v_j}) + 2\lambda\mathbf{u_i} \quad (2)$$

$$\mathbf{u_i} = \mathbf{u_i} - \gamma \cdot \nabla_{\mathbf{u_i}}\mathcal{L} \quad (3)$$

Next, we briefly introduce ALS. It is difficult to optimize $\mathbf{u_i}$ and $\mathbf{v_j}$ jointly, as minimizing Equation 1 is a non-convex problem. One way to solve this problem is to minimize the equation by fixing $\mathbf{u_i}$ and $\mathbf{v_i}$ in an alternative manner. For example, consider the item profile vector as a constant and calculate the derivative of Equation 1 with respect to $\mathbf{u_i}$. Then, set the derivative to zero and solve the quadratic equation to obtain the update rule of $\mathbf{u_i}$ as follows.

$$\frac{\partial\mathcal{L}}{\partial\mathbf{u_i}} = -2\sum_{j}(r_{ij} - \mathbf{u_i}^\mathsf{T}\mathbf{v_j})\mathbf{v_j}^\mathsf{T} + 2\lambda\mathbf{u_i}^\mathsf{T}$$

$$0 = -(P_{i,*} - \mathbf{u_i}^\mathsf{T}V^\mathsf{T})V + \lambda\mathbf{u_i}^\mathsf{T}$$

$$\mathbf{u_i}^\mathsf{T} = P_{i,*}V(V^\mathsf{T}V + \lambda I)^{-1} \quad (4)$$

In terms of convergence, ALS is faster than SGD and generally completed the learning process in the first twenty iterations [15]. SGD requires more iterations to converge; however, SGD can reach a lower mean square error than that of ALS. We combine both SGD and ALS to minimize the loss function of SPIREL, and the combined method can integrate the advantages of both techniques and complement the shortcomings of them.

B. TRANSFER LEARNING

CF algorithms rely on different types of input data to estimate the preference scores of users on items. A common approach is to extract the preference scores directly from explicit feedback (e.g., movie rating) or to estimate the scores from implicit feedback (e.g., purchase history). Generally, using explicit feedback leads to more accurate results as a user reveals exact preferences for items. In our framework, the input data is a check-in history that provides only positive samples (the places the users had visited), which implicitly reflects the preferences of POIs.

Most of the recommendation systems are *SD recommendation systems*, which only focus on one domain while ignoring various interactions between users and items. In several real-world applications, user-item relationships are recorded over time as event sequences; moreover, the check-in history

is one such example. Accordingly, when directly building the POI recommendation system over the SD recommendation system, the natural idea is to utilize the visit counts to learn interests of users on POIs. However, such methods easily suffer from the data sparsity issue as the users typically visit only a few locations.

In recent years, the notion of *cross-domain (CD) recommendation* [16], which leverages additional knowledge learned from other domains to improve the target domain recommendation task, has emerged. The CD recommendation systems are known to overcome the data sparsity issue in the SD recommendation system. One of the widely studied approaches is *transfer learning*. Previous studies [17]–[19] introduced the transfer learning approach to MF. The primary idea of using transfer learning within MF is to share the latent factors between the source domain and the target domain, assuming that the shared latent factors have similar characteristics.

In our work, to assist recommending the next POIs from the user-POI domain modeled with the visit counts, we integrated the confidence score for each POI transition pattern derived from the LDP mechanism. Consequently, we transferred the knowledge learned from POI transition patterns to a user-POI domain, assuming that the latent factors of the POI can bridge the static and dynamic preferences of the POI.

C. LOCAL DIFFERENTIAL PRIVACY (LDP)
LDP is a rigorous mathematical definition of privacy [20], which is used to preserve the location privacy of users in our study. Previously, DP [21] was employed in a centralized setting, which assumed the case of a trusted data curator. In the central setting, each user submits their original data to the trusted data curator, and the curator randomizes the aggregated results to guarantee the privacy of the involved users. However, in the real world, we cannot guarantee that the data curator is always trusted. Hence, we adopted a local version of DP, which can preserve the privacy of users under an untrusted data curator.

Specifically, LDP requires the following setting. There exists an untrusted server and m users. A user i holds a data item x_i in his/her mobile device and the server wants to know the aggregated result of $x_1, x_2, \cdots, x_m$. To guarantee the privacy, each user i is allowed to perturb x_i to obtain a noisy version of x_i, say x_i'. Then, the server receives x_i' instead of x_i, and calculates the aggregation result of $x_1', x_2', \cdots, x_m'$. While perturbing x_i, LDP requires a high probability that the server cannot infer the original value x_i from the perturbed value x_i'. The probability is decided by a privacy parameter ε, which controls the level of privacy guarantee. Specifically, LDP is defined as follows.

Definition 1: ε**-LDP** A randomized mechanism $\mathcal{A}$ satisfies ε-LDP for any two input values $x_1, x_2 \in Domain(\mathcal{A})$ and any possible output value x' of $\mathcal{A}$, if the following condition is satisfied

$$Pr[\mathcal{A}(x_1) = x'] \le e^\varepsilon Pr[\mathcal{A}(x_2) = x'].$$

Moreover, as LDP is also a variant of DP, LDP satisfies the composition theorem [22]. We introduce the sequential composition theorem that we utilized in our work as follows.

Theorem 1: **Sequential Composition** We assume an algorithm $\mathcal{F}$ consists of k LDP mechanisms $(\mathcal{A}_1, \cdots, \mathcal{A}_k)$, where each satisfies $(\varepsilon_1, \cdots, \varepsilon_k)$-LDP, respectively. Then, $\mathcal{F}$ guarantees $\sum_{i=1}^{k} \varepsilon_i$-LDP.

Because of the composition theorem, we usually refer to ε as a privacy budget. Specifically, to guarantee ε-LDP, each LDP mechanism should use a part of ε, and the sum should be no more than ε.

III. PROBLEM DEFINITION
A. SUCCESSIVE POINT-OF-INTEREST RECOMMENDATION
We first define the notion of *POI*, *check-in* and *check-in history* as follows.

Definition 2: **POI** A POI is defined as a specific location point uniquely identified with identifier p_i.

Definition 3: **Check-in** A check-in is defined as a tuple (u, p_i^t), if user u visited POI p_i at timestamp t.

Definition 4: **Check-in history** A check-in history of user u over T timestamps is defined as a sequence of check-ins represented as $CH_u = \{(u, p_x^1), \cdots, (u, p_y^T)\}$.

Based on Definition 2, 3, and 4, we can define the successive POI recommendation problem as follows. Given a user u and his/her check-in history $CH_u = \{(u, p_x^1), \cdots, (u, p_y^T)\}$, the objective of the recommendation system is to recommend a set of POIs that the user u is likely to visit at timestamp $T+1$.

B. CHALLENGES
Our problem setting is challenging. In particular, our objective is to build a recommendation system under an untrusted server while ensuring the privacy of the check-in data. Moreover, when it comes to handling implicit feedback, existing privacy-preserving MF-based recommendation systems have a problem in interpreting non-negative values (in our case, visit counts) into useful preference values. The details of our challenges are as given below.

1) *What information can be shared for CD successive POI recommendation and how can it be shared?* Existing studies on privacy-preserving recommendation systems [11]–[13] are suitable for SD scenarios with explicit feedback data. Considering that check-in actions are implicit feedback, we should model proper preference values from check-in data and integrate the preference values into MF.

2) *How can the check-in data be collected in a privacy-preserving manner?* A previous study [5] showed that only four location points are sufficient to identify most people. Therefore, people are hesitant to submit their check-in data, if the recommendation server is untrusted.

3) *How can the recommendation system be updated privately?* To accurately estimate the preference on the POIs, we should optimize the objective function of the proposed model. However, we cannot use the widely
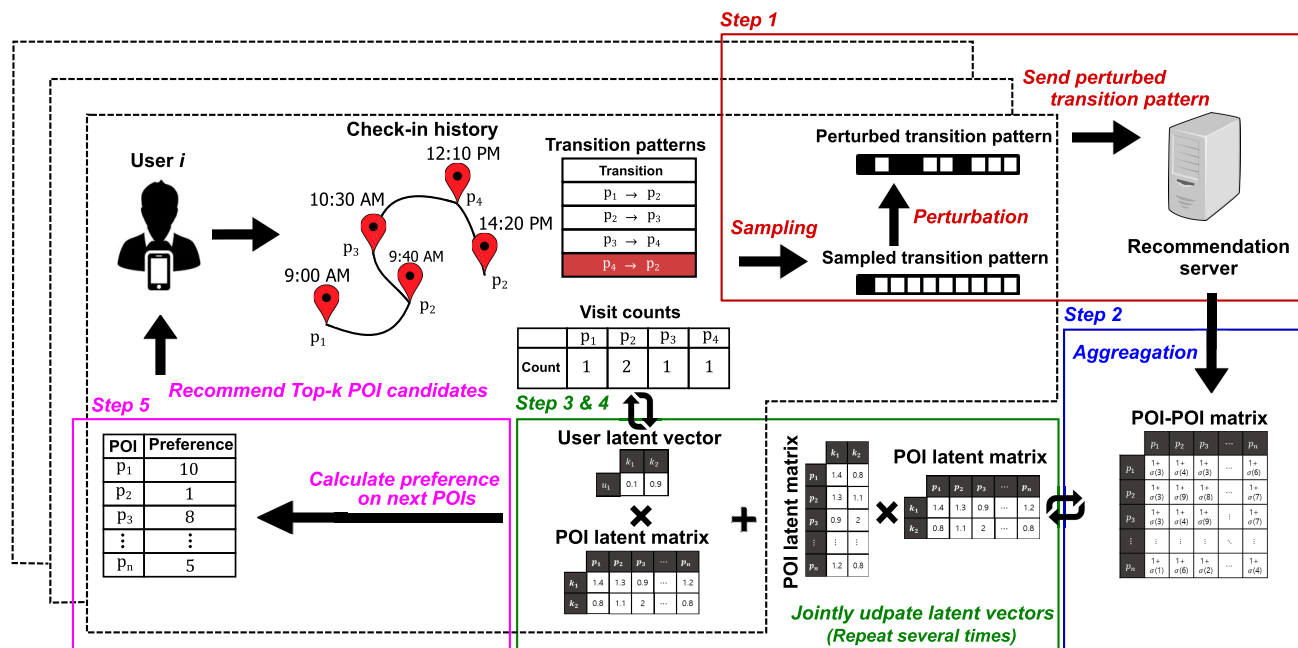
FIGURE 1. Overview of SPIREL framework.

used DP-SGD technique [23] that is designed for the centralized setting because an untrusted server can exploit the local gradients to reconstruct the input data of the model [13].

4) *How can the next POI candidates be received without disclosing the current location?* Although the collection of check-in data and the model updating is done in a privacy-preserving manner, the framework is not completely secure if the users who wish to receive services submit their exact location.

IV. PROPOSED METHOD

A. OVERVIEW OF OUR FRAMEWORK

In this section, we introduce our SPIREL framework illustrated in Figure 1. In this section, we will use the terms "user" and "participant" interchangeably. In our proposed framework, we assumed that m participants exist, each having their individual check-in histories, and an untrusted recommendation server. Note that the check-in data is a type of implicit feedback and it provides only positive samples (visited POIs). Thus, if we only use the visit counts to model the interactions between users and POIs, the user-POI matrix is extremely sparse as the users typically visit only a small number of POIs.

To infer the interest of a user in the next POI based on the current location, the proposed framework leverages additional information that reveals the preferences of transition between POIs. For this purpose, we used the notion of first-order Markov chain to model the sequential transition pattern between two consecutive POIs. For example, a user in Figure 1 has four transition patterns ($p_1 \rightarrow p_2, p_2 \rightarrow p_3, p_3 \rightarrow p_4, p_4 \rightarrow p_2$).

Figure 2 graphically illustrates the transfer learning approach in SPIREL. Here, the user-POI matrix P represents the POI preference of users based on the visit counts, and the POI-POI matrix Q is learned from the collected POI transition patterns. Then, the POI latent matrix V is shared and used to connect both the user-POI matrix P and POI-POI matrix Q. Thus, the visit counts and the transition patterns are integrated into the learning process of MF. Ultimately, our primary idea is that the knowledge learned from the transition patterns (matrix Q) can help increase the next POI recommendation accuracy in the target domain (matrix P). Accordingly, the objective function of SPIREL should be revised to learn P and Q simultaneously compared to Equation 1. We describe our novel objective function as follows.

$$\mathcal{L}_{SPIREL} = ||P - UV^\mathsf{T}||^2 + ||Q - VV^\mathsf{T}||^2 + \lambda(||U||^2 + ||V||^2) \quad (5)$$

Here, we explain the process flow of our proposed framework.

- **Step 1.** Each of the m participants independently samples a single transition pattern between two consecutive POIs. Then, the participant perturbs the sampled transition pattern in their local device and submits the perturbed transition pattern to the server.
- **Step 2.** After receiving the perturbed transition patterns from all participants, the server builds a POI-POI matrix that approximately reflects the global transition trend.
- **Step 3.** The server performs the learning process, which minimizes the objective function (Equation 5). First, the users update their profile vector using ALS. Then,
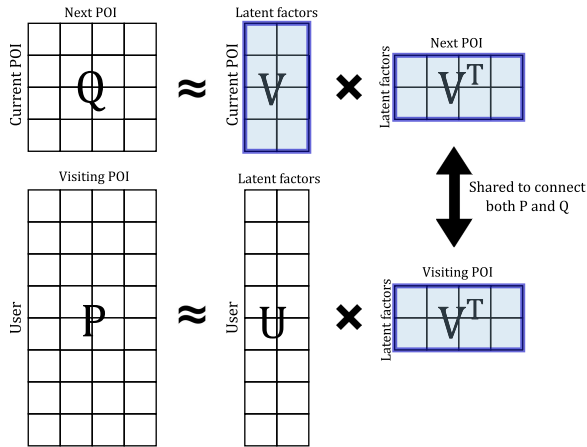
FIGURE 2. Transfer learning in SPIREL framework.

each user perturbs his/her local gradients of POIs and submits them to the server.

- **Step 4.** The server aggregates the perturbed gradients received from the users and updates the POI latent matrix using SGD.
- **Step 5.** The server sends the learned POI latent matrix to each user. Then, the users can calculate the preference score of the next POIs by themselves.

In the following section, we will explain each of these steps in detail.

B. TRANSITION PATTERN AGGREGATION UNDER LDP

1) PARTICIPANTS (Step 1)

Suppose there are n POIs and the check-in history consists of T check-ins. Then, the possible number of check-in histories is n^T. A naive solution would be to directly collect the frequency of each possible check-in history. However, directly computing frequencies over an enormous domain for all possible check-in histories is impractical, even if n and t are very small.

Our idea is to sample one transition pattern from the check-in history and estimate the frequency of sampled transition patterns under LDP to capture the global transition trend. Figure 3 shows an example of the transition pattern aggregation process, where each bit represents one specific transition pattern. We assume that all users share the same POI domain, and each user selects a single transition pattern from his/her check-in history. In this way, we can prevent certain users who have a long sequence of check-ins from dominating the global transition trend. Moreover, it makes the server harder to reidentify the users, as information about only two POIs is reported.

Randomized response (RR) [24] is a basic mechanism for guaranteeing LDP, where a user submits a "yes" or "no" answer according to the result of a coin-flipping process. The purpose of RR is to estimate the actual number of "yes" users. We used the optimized RR (ORR) scheme [25], which minimizes the estimation error. We present the detailed process of ORR in Algorithm 1.

Algorithm 1 Optimized Randomized Response (ORR)

Require: privacy budget ε, binary value $b[i] \in \{0, 1\}$
Ensure: perturbed binary value $\hat{b}[i]$
1: Set $p = \frac{1}{2}$, $q = \frac{1}{e^\varepsilon + 1}$
2: **if** $b[i] = 1$ **then** $\hat{b}[i] \sim Bernoulli(p)$
3: **else** $\hat{b}[i] \sim Bernoulli(q)$
4: **end if**
5: return $\hat{b}[i]$

Algorithm 2 Perturbing Transition Patterns Using ORR (Client Side)

Require: privacy budget ε, the number of POIs n
Ensure: perturbed bit string $\hat{b}$
1: User u randomly samples a transition pattern $p_x^t \rightarrow p_y^{t+1}$ from check-in history
2: User u initializes a length n^2 bit string b
3: **for** $i = 1$ to n^2 **do**
4:      **if** $p_x \rightarrow p_y$ corresponds to the ith position of b **then** $b[i] = 1$
5:      **else** $b[i] = 0$
6:      **end if**
7:      $\hat{b}[i] = ORR(\varepsilon, b[i])$ ▷ Algorithm 1
8: **end for**
9: Send $\hat{b}$ to the server

Algorithm 2 shows the detailed transition pattern perturbation process. In our framework, each user first generates a binary bit string of length n^2, where each bit represents whether the sampled transition pattern corresponds to the specific POI-POI combination. As directly sending the original bit value can cause a location privacy breach, each user perturbs the bit value using the ORR mechanism (Algorithm 1). Here, ORR is invoked n^2 times to collect the support of each transition pattern from all users.

When reporting the transition patterns, the communication cost $O(n^2)$ could be a problem if the domain size of POI is too large. If the communication cost is critical, we could reduce this cost by instead adopting one-bit randomized response methods, such as Hadamard randomized response technique [8], [10], [26] following the analysis in [27].

2) RECOMMENDATION SERVER (Step 2)

The server aggregates the perturbed bit strings and estimates the frequency of each transition pattern. In Algorithm 3, we present the procedure for aggregating the perturbed transition patterns to build a POI-POI matrix Q, which is used as the auxiliary information for the learning process. After receiving the perturbed bit strings from m users, the server adds each bit value to the corresponding element of Q. As the users implement ORR to perturb their transition patterns, the server can estimate the frequency of each transition pattern by calculating $\frac{Q[i][j] - mq}{p - q}$, where p and q are predefined probability in Algorithm 1.
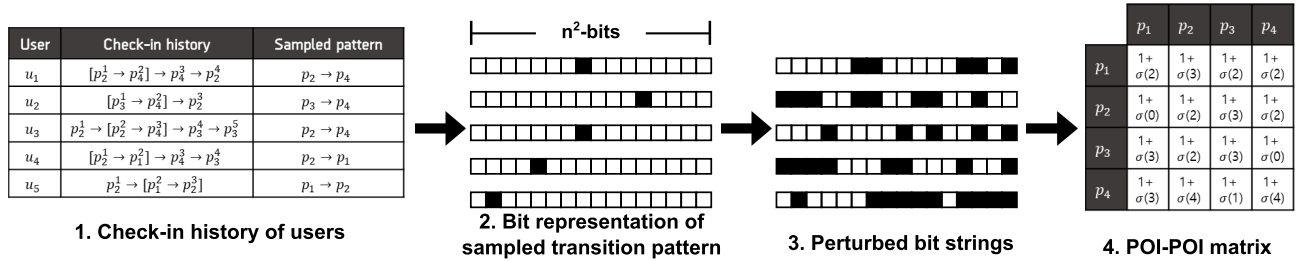
FIGURE 3. Example of point-of-interest (POI) transition pattern aggregation under local differential privacy (LDP).

Algorithm 3 Aggregating Perturbed Transition Patterns (Server Side)

Require: m perturbed bit strings $\hat{b}$, the number of POIs n
Ensure: a POI-POI matrix Q

1: Initialize a $n \times n$ matrix Q whose elements are all zero
2: **for** $i = 1$ to n **do**
3:     **for** $j = 1$ to n **do**
4:         **for** $k = 1$ to m **do**
5:             $Q[i][j] = Q[i][j] + \hat{b}_k[j + n * i]$
6:         **end for**
7:         $Q[i][j] = \frac{Q[i][j] - mq}{p - q}$ ▷ estimate the frequency of each transition pattern
8:         $Q[i][j] = 1 + \sigma(Q[i][j])$ ▷ calculate confidence score of each transition pattern
9:     **end for**
10: **end for**
11: **return** Q

Note that we let each user sample and perturb one transition pattern from his/her check-in history as discussed in Section IV-B1. Thus, it is challenging how to interpret the estimated frequency and map the fake positive feedback to user preference values. Here, we follow an assumption that positive feedback (no matter real or fake) simply indicates 'more preferable' than non-observed feedback [28].

Randomized response methods essentially give more weight to the probability that the truthful answer is delivered as it is. However, we cannot directly use the estimated frequency as preference scores because of the following two reasons. First, the estimated frequency obtained through the randomized response methods can have negative values. If the true count is close to zero, the noisy count can be unbiasedly estimated and thus can be a negative value. In this case, we cannot assume that the negative value indicates that all the participants dislike the corresponding transition pattern. Secondly, the estimated frequency grows linearly with the number of participants. Thus, we should carefully transfer the estimated frequency into another preference scores.

A widely used approach is to use confidence scores as preferences, which represents how much confidence increases with the number of positive feedback. For example, [29] suggested a plausible choice for the confidence score as

$c_{ui} = 1 + \alpha r_{ui}$. Here, r_{ui} indicates the number of actions of user u on item i. The rate of increase in confidence is controlled by α. In our proposed method, we used the sigmoid function and set the (i, j)-th element of matrix Q to $1 + \sigma(Q[i][j])$. The intuitive reasons for using the sigmoid function are as follows. First, the sigmoid function takes all real numbers as the input; thus, it can handle the dynamic range of estimated frequency, including negative value. Second, its output is monotonically increasing and is bounded in the range $[0, 1]$, which can handle any number of participants.

C. PRIVACY-PRESERVING LEARNING UNDER LDP

In this section, we propose a locally private solution to optimize our objective function. After building a POI-POI matrix, the next step is to factorize two matrices to identify the latent factors of users and POIs. Note that our objective function (Equation 5) aims to factorize the user-POI and POI-POI matrices simultaneously. Accordingly, we are required to derive the update rules for Equation 5.

In Section II-A, we introduced two techniques to minimize the quadratic objective function. SPIREL adopts both SGD and ALS to optimize Equation 5. The previous work [13] used SGD only to update both user and item profile vectors. However, SGD typically requires more iterations compared to ALS, and its performance is very sensitive to the choice of the learning rate. There is a limit of maximum iterations in the privacy-preserving recommendation system as the gradient computation at each iteration consumes the privacy budget, which can sometimes achieve an unsatisfactory model.

We first update the user profile vectors using ALS because it converges faster than SGD. Another reason is that each user can independently update their profile vector, which is more suitable for distributed learning setting [15]. Then, we can obtain the ALS update rule for the profile vector of user i as follows.

$$\frac{\partial \mathcal{L}_{SPIREL}}{\partial \mathbf{u_i}} = -2 \sum_j (r_{ij} - \mathbf{u_i}^\mathsf{T} \mathbf{v_j}) \mathbf{v_j}^\mathsf{T} + 2\lambda \mathbf{u_i}^\mathsf{T}$$

$$\mathbf{u_i}^\mathsf{T} = P_{i,*} V (V^\mathsf{T} V + \lambda I)^{-1} \quad (6)$$

As shown in Equation 6, the user profile update rule is the same as in Equation 4. As we assumed the POI profile vectors as a constant, only the user-POI matrix term remains. Here, the server can pre-compute $V(V^\mathsf{T} V + \lambda I)^{-1}$ and send the term
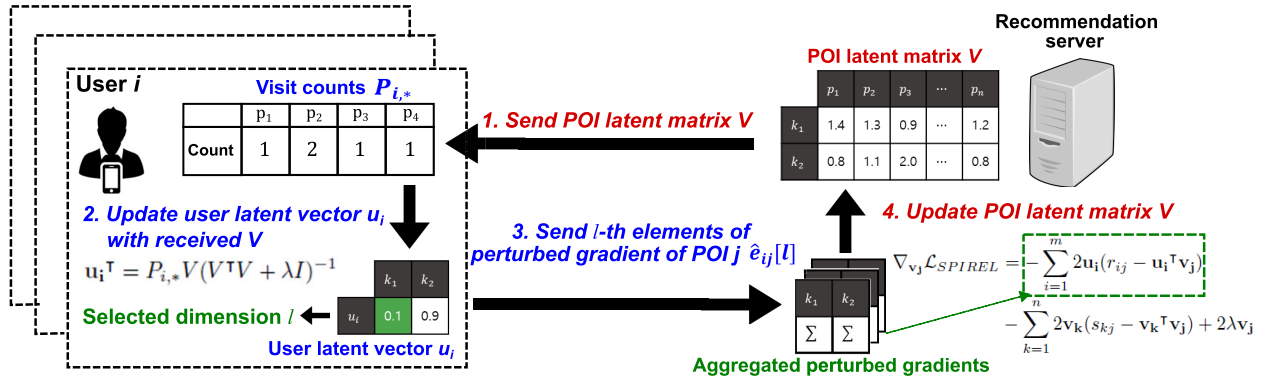
FIGURE 4. Learning process of SPIREL by gradient perturbation under LDP.

to the users. In this way, each user is not required to forward his/her visit counts ($P_{i,*}$) to the server to update their profile vector. Next, we list the ALS update rule for the POI profile vector as follows.

$$\frac{\partial \mathcal{L}_{SPIREL}}{\partial \mathbf{v_j}} = -2 \sum_i (r_{ij} - \mathbf{v_j}^\mathsf{T} \mathbf{u_i}) \mathbf{u_i}^\mathsf{T}$$
$$- 2 \sum_k (s_{kj} - \mathbf{v_j}^\mathsf{T} \mathbf{v_k}) \mathbf{v_k}^\mathsf{T} + 2\lambda \mathbf{v_j}^\mathsf{T}$$
$$0 = -(P_{*,j}^\mathsf{T} - \mathbf{v_j}^\mathsf{T} U^\mathsf{T}) U$$
$$- (Q_{*,j}^\mathsf{T} - \mathbf{v_j}^\mathsf{T} V^\mathsf{T}) V + \lambda \mathbf{v_j}^\mathsf{T}$$
$$P_{*,j}^\mathsf{T} U + Q_{*,j}^\mathsf{T} V = \mathbf{v_j}^\mathsf{T} (U^\mathsf{T} U + V^\mathsf{T} V + \lambda I)$$
$$\mathbf{v_j}^\mathsf{T} = (P_{*,j}^\mathsf{T} U + Q_{*,j}^\mathsf{T} V)(U^\mathsf{T} U + V^\mathsf{T} V + \lambda I)^{-1}$$
(7)

Here, $P_{*,j}$ indicates a column vector of matrix P with index j and s_{kj} indicates the (k, j)-th element of matrix Q. The server requires a user latent matrix U to update the POI profile vector, which can reveal the interactions between users and POIs by multiplying the POI latent matrix V with user latent matrix U. One option is to let users add noise to their profile vector in each iteration of training. However, this method will significantly affect both recommendation accuracy and convergence of learning, since the noise added to the profile vectors can lead to a domino effect of adding noise to the outputs of MF algorithm.

For circumventing the above issue, we instead perturbed the gradients and applied SGD to update the POI profile vectors. In this way, we can bound the noise to affect only the learning process of MF. Specifically, we let each user submit the perturbed gradients of Equation 5, as displayed in Figure 4. Then, the recommendation server aggregates the perturbed gradients and updates the POI profile vectors. Finally, we can rewrite the gradient of the POI profile vector as follows.

$$\nabla_{\mathbf{v_j}} \mathcal{L}_{SPIREL} = - \sum_{i=1}^m 2\mathbf{u_i}(r_{ij} - \mathbf{u_i}^\mathsf{T} \mathbf{v_j})$$
$$- \sum_{k=1}^n 2\mathbf{v_k}(s_{kj} - \mathbf{v_k}^\mathsf{T} \mathbf{v_j}) + 2\lambda \mathbf{v_j} \quad (8)$$

Algorithm 4 Perturbing Gradients (Client Side)

Require: privacy budget ε, gradient value of POI j e_{ij}, the number of POIs n, profile vector size d, a randomly selected dimension l of e_{ij}
Ensure: perturbed gradient $\hat{e}_{ij}[l]$
 1: Project $e_{ij}[l]$ into the range $[-1, 1]$
 2: User i draws a value x from Bernoulli distribution such that $\Pr[x = 1] = \frac{e_{ij}[l] \cdot (e^\varepsilon - 1) + (e^\varepsilon + 1)}{2e^\varepsilon + 2}$
 3: **if** $x = 1$ **then**
 4: $\quad \hat{e}_{ij}[l] = nd \frac{e^\varepsilon + 1}{e^\varepsilon - 1}$
 5: **else**
 6: $\quad \hat{e}_{ij}[l] = -nd \frac{e^\varepsilon + 1}{e^\varepsilon - 1}$
 7: **end if**
 8: **return** $\hat{e}_{ij}[l]$

---

1) PARTICIPANTS (Step 3)

Equation 8 predominantly consists of two terms: $\mathbf{u_i}(r_{ij} - \mathbf{u_i}^\mathsf{T} \mathbf{v_j})$ and $\mathbf{v_k}(s_{kj} - \mathbf{v_k}^\mathsf{T} \mathbf{v_j})$. The recommendation server can calculate the term $\mathbf{v_k}(s_{kj} - \mathbf{v_k}^\mathsf{T} \mathbf{v_j})$ by itself because the value s_{kj} is the confidence scores, as discussed in Section IV-B. The other term $\mathbf{u_i}(r_{ij} - \mathbf{u_i}^\mathsf{T} \mathbf{v_j})$ has a prediction error of the visit counts r_{ij}. In our framework, the participants perturb the term $\mathbf{u_i}(r_{ij} - \mathbf{u_i}^\mathsf{T} \mathbf{v_j})$ to prevent the recommendation server from learning whether a user i visits POI j.

In our framework, we applied the widely used RR method that proposed by Nguyen _et al._ [10] to perturb the user gradients, which was also utilized in [13]. It should be noted that SPIREL is not confined to a specific LDP mechanism; thus, any LDP mechanisms, such as [30] and [31], can be used in our proposed framework. In the experiments, we presented the results using a more accurate LDP mechanism [30] and demonstrated that the error induced by noisy gradients has a limited effect on the performance.

Algorithm 4 shows the procedure of the gradient perturbation process. Each participant first chooses a random value of l and j, thereby sending only a single dimension of the perturbed gradient for a single randomly chosen POI. Because the gradient values are not bounded, we impose a bound on the raw gradients before perturbation by manually clipping

---

Algorithm 5 Update of Point-of-Interest Latent Matrix (Server Side)

Require: the number of POIs n, the number of iterations I, profile vector size d

Ensure: updated POI latent matrix V

1: Divide the users into I groups $\{g_1, g_2, \cdots, g_I\}$
2: **for** $i = 1$ to I **do**
3:    Initialize $\nabla_V \mathcal{L}_{SPIREL} \in \{0\}^{n \times d}$
4:    **for** each user u in group g_i **do**
5:        Selects a random value j from $\{1, 2, \cdots, n\}$
6:        Selects a random value l from $\{1, 2, \cdots, d\}$
7:        Gets perturbed gradient $\hat{e_{uj}}[l]$ ▷ Algorithm 4
8:        $\nabla_V \mathcal{L}_{SPIREL} = \nabla_V \mathcal{L}_{SPIREL} + \hat{e_{uj}}[l]$
9:    **end for**
10:    Compute other terms in Equation 8 and add to $\nabla_V \mathcal{L}_{SPIREL}$
11:    Update V with $\nabla_V \mathcal{L}_{SPIREL}$ using Adam optimizer
12: **end for**
13: **return** V

---

the gradients at each iteration [23]. As the LDP mechanism we use assumes that the input value lies in the range $[-1, 1]$, we project the gradient e_{ij} in $[-1, 1]$. Then, the user generates a noisy gradient $\hat{e}_{ij}$ following the probability distribution of LDP mechanism and submits the perturbed gradient to the server. Note that $\hat{e}_{ij}$ needs to be scaled by a factor of n and d because of sampling l and j respectively in Algorithm 5.

2) RECOMMENDATION SERVER (Step 4)

The server is responsible for aggregating the perturbed gradients from the participants and updating the POI profile vectors. As each user submits a noisy gradient of a randomly chosen POI through Algorithm 4, the server cannot learn about the details of the POIs visited by the user. However, the server can still estimate the sum of perturbed gradients and thus updates the model. We further applied two strategies to update our models efficiently.

(1) Learning with user group Generally, the non-private recommendation systems will stop the learning process when the objective function converges. However, under a privacy-preserving setting, the server continues the learning process during a predefined number of iterations because the participants are supposed to submit perturbed gradients at each iteration. For example, [13] sets a fixed number of iterations i. Then, at each iteration, each user submits a perturbed gradient using $\frac{\varepsilon}{i}$ privacy budget, which guarantees the entire learning process satisfies ε-LDP.

When answering multiple questions, Wang *et al.* [25] proved that partitioning users into groups and assigning one question for each group is better than splitting the privacy budget in terms of accuracy of the aggregated result under LDP. Inspired by this idea, we divided the users randomly into i groups and asked a single user group to submit their perturbed gradients using the entire privacy budget. In this way, each user only participates in one iteration of

the learning process. This approach is also in line with the mini-batch gradient descent concept [32], which the model is updated with a subset of training examples during one iteration.

(2) Stabilizing iterative learning After obtaining a gradient value, the next step is to determine a learning rate that decides how fast the model learns. If the learning rate is set too high, the training path of objective function frequently oscillates and may never converge. On the other hand, if the learning rate is too low, the model will take too many iterations to converge. In our framework, determining the learning rate is more challenging due to the inherent noisy gradients.

Recently, variants of gradient descent methods that have the power of adaptive learning rates were widely applied to train deep neural networks, such as Adagrad [33], RMSprop [34], and Adam [35]. Here, we used Adam optimizer that maintains a weighted moving average of the gradients (Equation 9). Averaging over gradients makes the whole learning process stable and robust to initial learning rate selection.

$$x = \beta_1 x + (1 - \beta_1)\nabla_{v_j}\mathcal{L}_{SPIREL}$$
$$y = \beta_2 y + (1 - \beta_2)(\nabla_{v_j}\mathcal{L}_{SPIREL})^2 \qquad (9)$$

Here, as x and y are typically initialized to zero, the Adam optimizer performs bias correction as follows.

$$\hat{x} = \frac{x}{1 - \beta_1}, \quad \hat{y} = \frac{y}{1 - \beta_2} \qquad (10)$$

Finally, the update rule is provided by the following equation.

$$v_j = v_j - \frac{\gamma}{\sqrt{\hat{y}} + \epsilon}\hat{x} \qquad (11)$$

Here, β_1 and β_2 control the decay rates of the moving averages. Further, γ indicates the learning rate and ϵ is used to avoid division by zero.

D. RANKING (Step 5)

Using the learned profile vector of users and POIs, SPIREL can recommend the next POI candidates while considering the current location of users. Let us assume a user i is in POI j. Then, the preference of next POI k of user i can be calculated as follows.

$$pref_i^k = u_i^\top v_k + v_j^\top v_k \qquad (12)$$

Equation 12 consists of two preferences: the first term, $u_i^\top v_k$, indicates the personal preference for the next POI k and the second term, $v_j^\top v_k$, represents the POI transition preference of POI j to POI k. By calculating the sum of the two preferences, we can recommend the top-k next POIs to users. It should be noted that because the POI profile vectors are publicly known, users are not required to submit their current location to the server. Instead, the recommendation server sends all the POI profile vectors to users, and the users can compute the preferences for all next POIs within their local devices.

E. COMPLEXITY ANALYSIS

In this subsection, we analyze the computational complexity of the participants and the server in the SPIREL framework, respectively.

- **Participants.** Each participant first perturbs the n^2-length bit string. So the time complexity of perturbing bit string is $O(n^2)$. After sending the bit string, each user updates their profile according to Equation 6. Note that the server can precompute the $n \times d$ matrix $V(V^\intercal V + \lambda I)^{-1}$ in advance. Thus, each user simply needs to multiply n-length vector $P_{i,*}$ with $n \times d$ matrix, which requires $O(nd)$ time. Finally, each user calculates the d-length gradient vector in the first term of Equation 8 and perturbs one bit of the gradient vector, which can be finished in $O(d^2)$ time. In summary, the total time complexity of one user is $O(n^2)$.

- **Server.** The server first aggregates the n^2-length bit string from m users and constructs the $n \times n$ POI-POI matrix. The time complexity for building the POI-POI matrix is $O(mn^2)$. Next, the server precomputes the term $V(V^\intercal V + \lambda I)^{-1}$ used for the ALS. Even though matrix conversion of $d \times d$ matrix $(V^\intercal V + \lambda I)^{-1}$ requires $O(d^3)$ time, n is typically larger than d. Thus, we can assume that the server needs $O(nd^2)$ time to compute the entire term. Finally, the server updates the POI latent matrix according to Equation 8. Assume that the maximum number of iterations is i. For each iteration, the server computes the second term of Equation 8, which requires $O(nd^2)$ time. Then, the total time complexity for updating the POI latent matrix is $O(ind^2)$. Since m is much larger than i or d, we can conclude that the total time complexity of the server is $O(mn^2)$.

F. PRIVACY ANALYSIS

In this subsection, we analyze the privacy guarantee of each step in SPIREL. Our framework mainly consists of two perturbation processes, as explained in Sections IV-B and IV-C. According to Theorem 1, to guarantee ε-LDP, the users have to split their privacy budget for each process. In our work, we let users use ε_1 to perturb their transition patterns and ε_2 to perturb a gradient value of selected POI; thus the sum of ε_1 and ε_2 equals to ε.

1) **Step 1.** Each participant prepares a n^2-length bit string, where only a single bit is set according to the sampled transition pattern. Other bits are zero. Each participant invokes ORR mechanism n^2 times to report supports of each transition pattern to the server. As each bit is relevant to the independent transition pattern, the participant can use the entire budget to perturb each bit. Thus, the process of perturbing such binary vectors satisfies ε_1-LDP as analyzed in [25].

2) **Step 2.** The server aggregates the perturbed bit strings received from all participants. Then, the server estimates the supports of transition patterns. Based on the estimated supports, the server computes the

preference score for each transition pattern. Because of DP's post-processing immunity, no further privacy leaks occur in this step.

3) **Step 3.** First, the server transmits the information needed to update the user profile vector through ALS using POI latent matrix. Then, each participant can locally update their profile vector without any transmission. Second, each user submits a perturbed gradient value for a randomly selected POI. Note that each user is involved in a disjoint user group, and only a single user group is assigned to one iteration of the learning procedure. The participants clip their one sampled gradient value to the range [-1, 1] and spend the entire ε_2 budget for perturbation. We can exploit any LDP mechanism in this step, and the privacy guarantee depends on the LDP mechanism used. In our work, we utilize the LDP mechanism proposed by Nguyen *et al.* [10], and it ensures ε_2-LDP.

4) **Step 4.** For each iteration of the learning procedure, the server receives the perturbed gradients from one dedicated user group and updates the POI profile vectors. Note that the participant submits a perturbed gradient for a randomly selected POI and its value is either $\pm nd \frac{e^\varepsilon + 1}{e^\varepsilon - 1}$. Thus, the server cannot infer which POIs are visited by the participants or how much they visited those POIs. Therefore, this step does not involve any privacy disclosure.

5) **Step 5.** The server publishes the updated profile vectors of all POIs to the participants. Then, each participant can compute the preference score of the next POI without disclosing their current position. Since the server does not know the profile vectors of participants, there is no privacy leakage.

V. EVALUATION

In this section, we evaluate the SPIREL method in various settings. All experiments were performed on a server with Intel i9-9900X CPU with 128GB of memory and GeForce RTX 2080 GPU. In all experiments, we averaged the results over 50 runs.

A. EXPERIMENTAL SETUP

1) DATASETS

We used four public datasets: *Gowalla*,[1] *Taxi*,[2] *Yelp*[3] and *Foursquare* [36]. For each of the above datasets, we discard users and POIs with fewer than 10 associated check-ins. Table 2 shows the statistics for all datasets after cleaning. We excluded the last check-in for the evaluation and trained the model with the remaining check-ins for each check-in history. Here, data sparsity is based on $1 - \frac{Check-ins}{Users \cdot POIs}$.

---

[1] https://snap.stanford.edu/data/loc-gowalla.html
[2] https://data.cityofchicago.org/Transportation/Taxi-Trips/wrvz-psew/data
[3] https://www.yelp.com/dataset

TABLE 2. Statistics of four datasets.

Dataset	Users	POIs	Check-ins	Sparsity
Gowalla	9,617	585	95,956	98.29%
Taxi	267,739	526	5,354,780	96.20%
Yelp	97,141	1,075	691,744	99.34%
Foursquare	32,158	2,409	423,675	99.45%

2) METHODS

SPIREL is the first work to integrate the relationship between POIs with that of users and POIs while guaranteeing the LDP. There is no existing privacy-preserving recommendation method that can directly compare with SPIREL. Accordingly, our ultimate goal is to demonstrate the benefits of combining the POI-POI relationship in MF and the privacy-preserving learning process of SPIREL compared to the following five methods.

1) **SD approach** This approach is a single-domain MF-based recommendation system that only considers the visit counts to predict the preference of the next POIs.

2) **SD approach with private learning (SD-PL) [13]** This is a private version of SD, which assumes that users hold visit counts in their local devices. SD-PL optimizes the same objective function of SD except that the users follow the LDP protocol of [13], i.e., gradients are sent to the server after perturbation.

3) **CD approach** This is a non-private version of our framework, which assumes that the server receives the raw check-in history of users. Thus, the CD approach directly uses original visit counts and all transition patterns (not sampled, not perturbed) between two POIs.

4) **SPIREL with piecewise mechanism (SPIREL-PM)** This is a variant of SPIREL that utilizes a piecewise mechanism [30] in the stage of gradient perturbation. Note that the piecewise mechanism is known to obtain higher accuracy than the LDP mechanism [10] used in SPIREL.

5) **Item-based collaborative filtering (ICF) [37]** This approach predicts the next items based on the item similarity score. The item similarity score is derived from the estimated joint frequencies of co-occurrence between two items using the LDP mechanism.

3) METRICS

We employed two evaluation metrics to evaluate the quality of recommendation: hit ratio (HR) and mean reciprocal rank (MRR). HR@k measures the accuracy of the recommendation, which indicates the ratio of users whose latest POI is included in the given top-k POI candidates recommended by the server. MRR is another widely used metric for evaluating the quality of recommendation. For a single recommendation result, the reciprocal rank is $\frac{1}{rank}$, where $rank$ is the position of the correct answer in the recommendation list. We evaluated HR@k and MRR for k in {3, 5, 7, 10}, and the default value of k was set to 5.

4) PARAMETERS

We experimented with five values of privacy budget ε in {0.2, 0.4, 0.8, 1.6, 3.2}, where the default value is 0.8. The regularization parameter λ was set to 10^{-4}. We set the size of the profile vector d to 40 for all methods. The learning rate γ was set to 0.01 for CD, SPIREL, and SPIREL-PM. In SD and SD-PL, γ was set to 0.001. We also considered various values for the maximum number of iterations from 5 to 40 (by default 20) and different privacy budget allocation ratios (we divided the budget equally by default). The parameters related to the Adam optimizer were set to $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$ based on [35] in CD and SPIREL variants. Finally, we used the same parameter setting for ICF in [37].

B. EXPERIMENTAL RESULTS

1) VARYING RECOMMENDATION SIZES

Figs. 5 and 6 show the HR@k and MRR with different recommendation list sizes, respectively. We first compared the non-private methods (SD and CD) to identify the effects of integrating the POI-POI relationship. Overall, we observed the CD approach outperforms on all datasets in HR@k and MRR, except for the *Taxi* dataset. The reason is that the sparsity of *Taxi* dataset is very low compared to other datasets, which allows the SD approach to predict next POIs sufficiently with only the user-POI relationship. Concretely, CD approach improves the SD by 25.82% in HR@k and 22.35% in MRR on average.

Next, we compared SD-PL and SPIREL, which are the private version of SD and CD, respectively. We can observe both methods experience performance drops due to the perturbation process. Nevertheless, SPIREL always outperforms SD-PL, on average, 41.56% and 38.84% in terms of HR@k and MRR, respectively. Another observation is that SPIREL even achieves better performance than SD in *Foursquare* dataset, presumably due to the extreme sparsity of the user-POI domain. Compared to methods that rely on the user-POI relationship alone, the results show the advantages of integrating the confidence score of the noisy POI-POI relationship.

ICF consistently shows the worst performance among all methods. Note that ICF does not consider the users' preferences for POIs. Thus, it is beneficial to unify both the user-POI and POI-POI relationships for modeling users' preferences. Finally, SPIREL-PM achieves similar performance compared to SPIREL. This is due to the situation in which SPIREL should divide the privacy budget, and we will discuss this in detail in Section V-B3.

2) VARYING PRIVACY BUDGETS

We explored the effect of privacy budget ε on SPIREL. Figs. 7 and 8 illustrate the HR@k and MRR of SPIREL over various privacy budgets ranging from 0.2 to 3.2 using boxplot visualization. Each boxplot provides six pieces of information: the maximum value (highest point), 75 percentile (upper hinge of the box), median (line inside the box), 25 percentile (lower hinge of the box), minimum
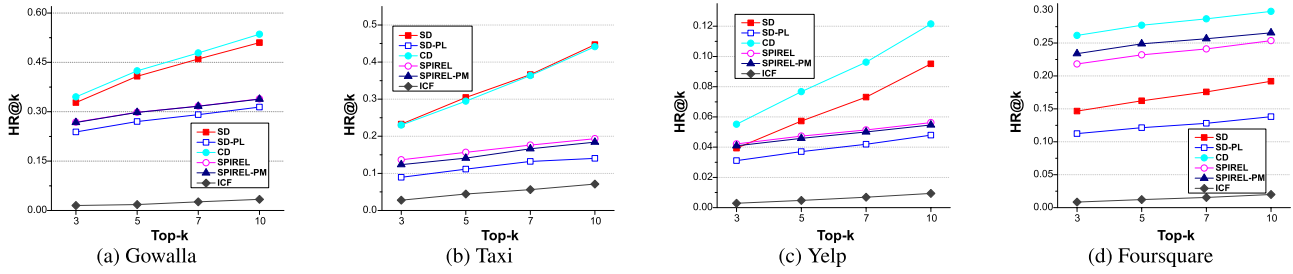
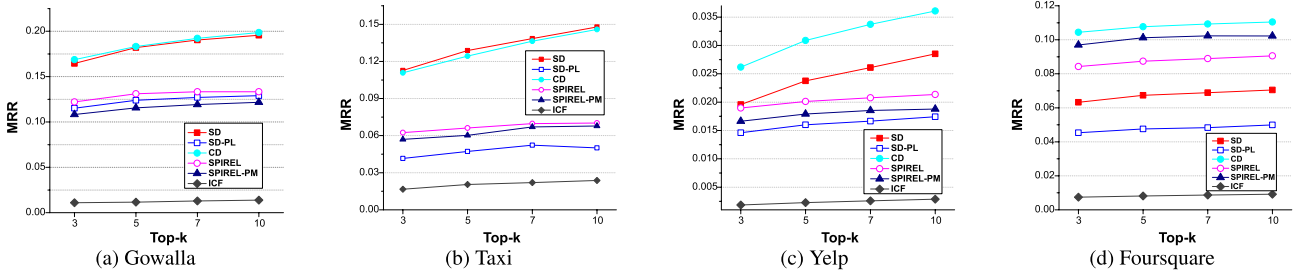**FIGURE 5.** HR@k comparison with varying top-k recommendation sizes.

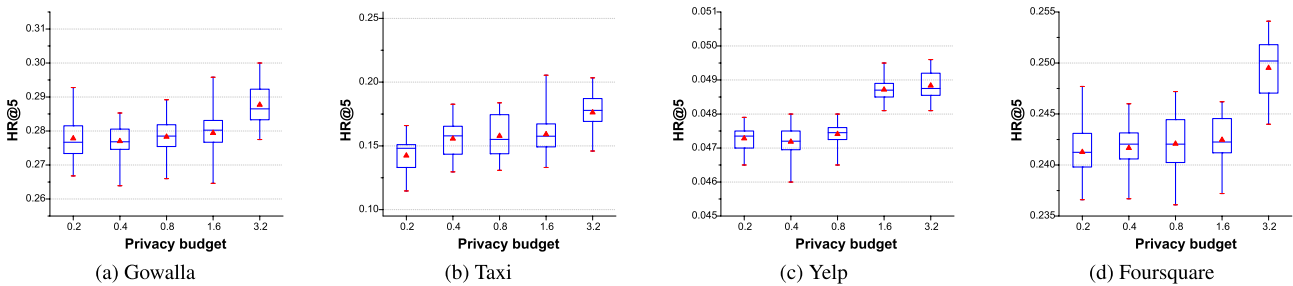**FIGURE 6.** MRR comparison with varying top-k recommendation sizes.

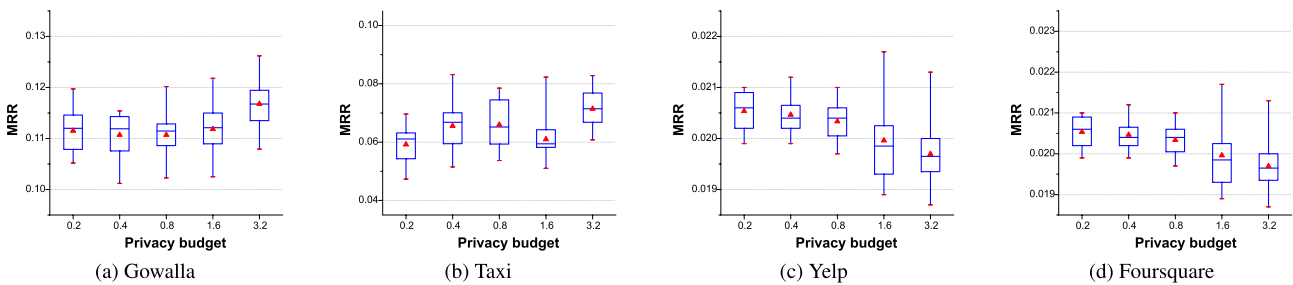**FIGURE 7.** HR@k of SPIREL corresponding to varying privacy budgets.

FIGURE 8. MRR of SPIREL corresponding to varying privacy budgets.

value (lowest point), and arithmetic mean (triangular point). We predominantly focused on the arithmetic mean value of HR@k and MRR. Overall, HR@k improves as ε increases because fewer noises are added to both transition patterns and gradients. On datasets *Yelp* and *Foursuqare*, the MRR of SPIREL does not increase as ε rises. A possible explanation is that these two datasets are highly sparse and have a larger POI domain, which makes SPIREL hard to infer the relative ranking between top-k candidates from the preference scores.

3) VARYING PRIVACY BUDGET ALLOCATION RATIOS

As described in Subsection IV-B and IV-C, each participant of SPIREL should perturb his/her transition patterns and gradients, respectively. Figure 9 shows how the performance of the SPIREL is affected by varying the privacy budget allocation ratios from 1:9 to 9:1 (transition pattern perturbation:gradient perturbation). We observed that the HR@k and MRR show a similar trend for various allocation ratios. However, these trends behave very differently for each dataset. We can
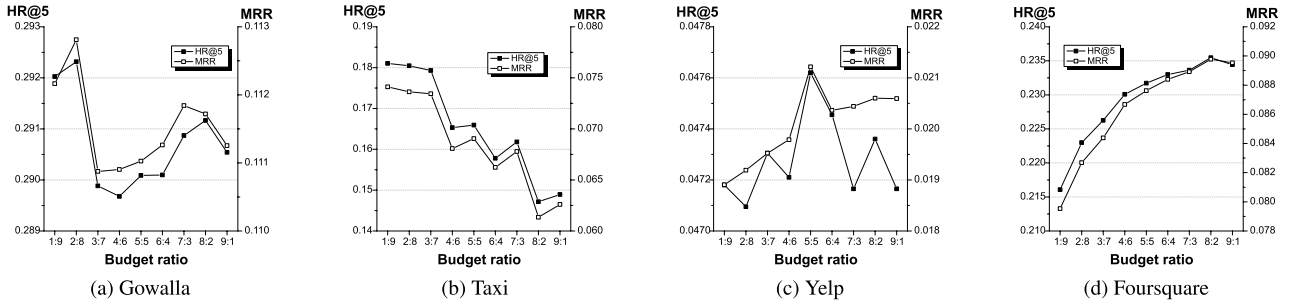
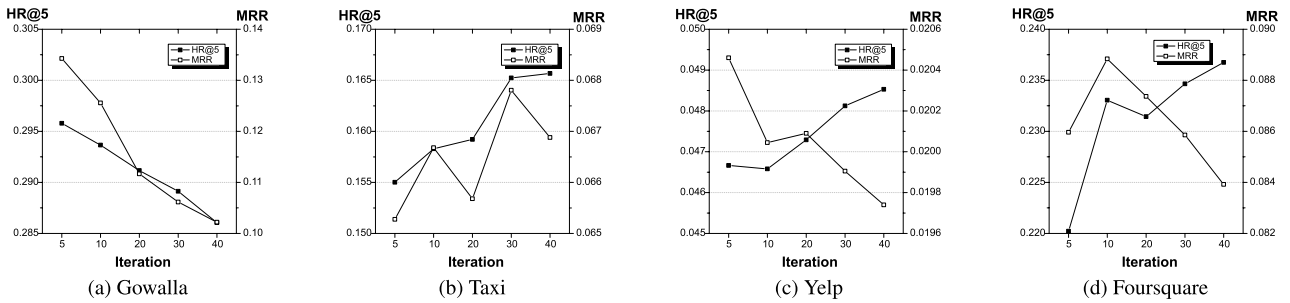**FIGURE 9.** Performance of SPIREL corresponding to varying privacy budget ratios.

FIGURE 10. Performance of SPIREL corresponding to varying numbers of maximum iterations.
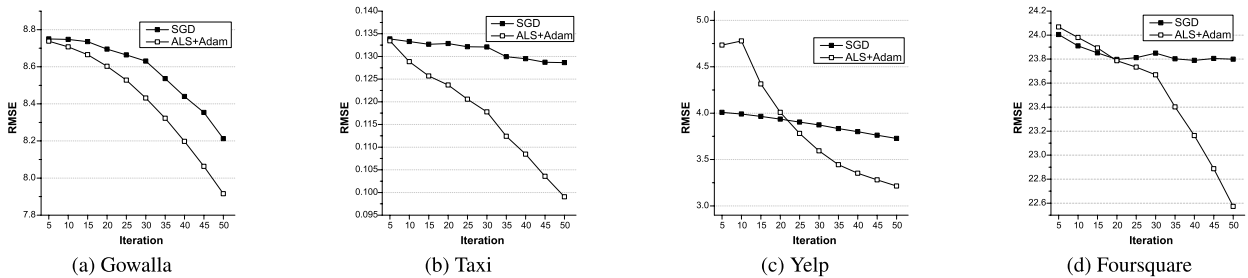

FIGURE 11. Root mean square errors of SPIREL over 50 iterations with or without ALS and Adam optimizer.

conclude that when dealing with a relatively small POI domain (*Gowalla* and *Taxi*), focusing on the gradient perturbation results in higher recommendation performance. On the other hand, SPIREL could benefit from allocating more privacy budget to transition perturbation when the dataset is sparse (*Yelp* and *Foursquare*). It is the reason why the performance gain of SPIREL-PM stands out in Figure 5d, where the other datasets have less room for performance to rise thanks to the piecewise mechanism when equally allocating the privacy budget.

4) VARYING NUMBERS OF MAXIMUM ITERATIONS
The maximum number of iterations is limited in privacy-preserving recommendation systems, as each iteration consumes a portion of the privacy budget. Thus, we should consider both the stability and convergence of learning process, otherwise SPIREL would suffer from underfitting or even cause the loss function to diverge. We analyzed the

effects of the maximum number of iterations on SPIREL in Figure 10. Moreover, we illustrate the root mean square error (RMSE) of loss function during 50 iterations to demonstrate the effects of using both ALS and Adam optimizer in Figure 11.

While the RMSE consistently decreases in the dataset *Gowalla*, the performance of SPIREL decreases as the maximum number of iterations increases. We can conclude that SPIREL suffers from the overfitting problem, and this indicates that SPIREL can reach optimal within five iterations for small dataset. Except for the dataset *Gowalla*, the HR@k of SPIREL consistently increases during the learning process. Again, due to the large POI domain size of the datasets *Yelp* and *Foursquare*, SPIREL experiences performance drops in MRR despite HR@k increases.

Finally, we examined the effects of combining ALS and Adam optimizer compared to naive SGD updates. The results are shown in Table 3. SPIREL obtains a huge

TABLE 3. HR@5 and MRR of SPIREL with SGD and SPIREL with ALS+Adam.

Dataset	Gowalla		Taxi		Yelp		Foursquare	
Metric	HR	MRR	HR	MRR	HR	MRR	HR	MRR
SGD	0.0101	0.0040	0.0122	0.0049	0.0056	0.0022	0.0031	0.0012
ALS+Adam	0.2983	0.1311	0.1569	0.0662	0.0473	0.0201	0.2321	0.0874

performance improvement in all datasets when using our learning framework. We attribute the good performance of SPIREL to merging the advantages of ALS and Adam. We also observed that increasing the learning rate of SGD does not improve the performance because the loss function diverges immediately.

5) Training time comparison

Because of the vast numbers of users and items in collaborative filtering, the recommender model would be inefficient if it takes too much time for training. So, we compared the training time of five algorithms. The training time is reported in Figure 12. Note that the training time includes server-side computations as well as user-side computations. Thus, the actual training time in a real-world setting is less than the results shown in Figure 12. We fixed the maximum number of iterations to 20 for MF-based algorithms.
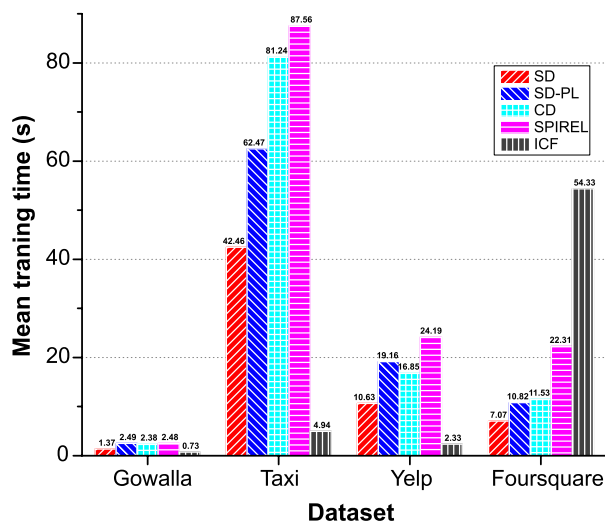

FIGURE 12. Training time comparison on the different datasets.

We observed that SPIREL required the longest training time across all datasets except for the *Foursquare* dataset. This can be explained by the computation time for building the POI-POI matrix for transfer learning. Overall, the privacy-preserving approaches (SD-PL, SPIREL) demonstrated a longer training time than their non-private versions (SD, CD) because of the time for perturbation. Although ICF showed the shortest training time on *Gowalla*, *Taxi*, and *Yelp* datasets, its training time is sensitive to the domain size of POIs due to the nature of comparing every POI-POI pairs to derive the similarity scores. Accordingly, it required the longest training time on the *Foursquare* dataset.

VI. RELATED WORK

The problem of successive POI recommendation has received significant attention recently [1]–[4]. To predict where a user will visit next, we have to consider the relationship between POIs in addition to that between users and POIs. However, the traditional MF-based recommendation methods neglect the sequential patterns and infer the user preferences only based on the places the users visited. To overcome this problem, [39], [40] consider the sequential influences by integrating the Markov chain into MF and obtain good performance on next-item recommendation. Currently, geographical influence is fused with the MF algorithms to enhance the accuracy of POI recommendations [3], [41].

Our research direction was to incorporate the relationship between POIs by adapting the transfer learning approach [17]–[19]. Most transfer learning methods under MF technique utilize relevant data from other domains into the target domain by sharing the latent matrix. For example, the approach called *coordinate system transfer* [17] first learns the user latent matrix U_A from an auxiliary domain and generates the user latent matrix U_T for target domain based on U_A. Further, an extension of this approach was proposed that exploits the implicit binary feedback (e.g., like/dislike) on items to construct a target explicit rating matrix (e.g., 5-star rating) [19].

DP [21] is a rigorous privacy standard; the output of a DP mechanism should not reveal much information specific to any individuals. DP requires a trusted data curator to collect original data from users. Recently, a local version of DP (LDP) was proposed. In the local setting, each user perturbs his/her data and sends the perturbed data to the data curator. As the original data never leaves the device of the users, the LDP mechanisms have the benefit of not requiring a trusted data curator. Accordingly, many companies attempt to adopt LDP while collecting data from the clients for privacy reasons [7]–[10].

Beyond estimating the frequency of a single item, there are several works on discovering item sequences through LDP. [38] found frequent new words by building a noisy prefix tree. [27] and [42] used the estimated frequency of n-gram to identify the frequent new terms. [43] also used the n-gram model to release noisy trajectory. Our work employed the n-gram approach to model the POI sequences. Instead of using the estimated frequency directly, we turn the frequency into a confidence score and map it to the users' preferences.

Table 4 summarizes the important differences between the SPIREL and previous works. To the best of our knowledge, no previous MF-based recommendation systems [11]–[13] exploit the implicit feedback to recommend the next POI.

TABLE 4. Comparison between SPIREL and other privacy-preserving recommendation methods.

Group	Method	DP model	Learning model	Data type
CD	SPIREL	LDP	MF	implicit
SD	[11], [12]	CDP	MF	explicit
	[13]	LDP	MF	explicit
	[37]	LDP	Item-based CF	implicit

Specifically, Hua *et al.* [11] proposed an objective function perturbation method. In their work, a trusted data curator added Laplace noises to the objective function so that the factorized item matrix satisfies CDP. Zhang *et al.* [12] proposed a probabilistic MF-based recommendation system with personalized CDP. They used a random sampling method to satisfy the privacy requirements of different users. Shin *et al.* [13] proposed a MF-based recommendation system under LDP. In their method, users updated their profile vectors using SGD locally and submitted the perturbed gradients to the server. They reduced the amount of noise added on gradients by adopting the dimensionality reduction technique into the item domain. In short, these works focused on reducing the RMSE during the learning process and attempted to predict users' preferences using explicit feedback that has a fixed scale rating.

Some works adopted LDP for building recommendation systems [37], [44], [45]. Guo *et al.* [37] proposed a privacy-preserving item-based CF technique under LDP. Their method predicted the next item based on the similarity score that is estimated through item co-occurrence frequencies. However, employing only the similarity score between items had a limit to predict the preferences of users on POIs. Asada *et al.* [44] proposed a location preference recommendation system with LDP. However, the location preference here indicated whether a user wants to publish information about a specific location or not, which is a different area from recommending the next POI. Gao *et al.* [45] proposed a POI recommendation model using Factorization Machines. They estimated the user preferences by combining a linear model with a high-order feature model and adopted decentralized SGD to train the model in a privacy-preserving manner.

VII. CONCLUSION

In this paper, we proposed a novel successive POI recommendation framework under LDP, named SPIREL. We first collected noisy transition patterns between two consecutive POIs. Then, we calculated a confidence score for each transition pattern for the POI-POI matrix. Subsequently, we assisted the predict the preferences of the next POI in the user-POI domain using the confidence score from the POI-POI domain by transfer learning approach. For preventing privacy leakage from the stage of iterative learning, we integrated LDP mechanisms to collect gradients and combined both ALS and SGD to train SPIREL. By extensive experiments on four public datasets, we demonstrated that SPIREL provides significantly better POI recommendation performance than the existing privacy-preserving recommendation methods.

REFERENCES

[1] C. Cheng, H. Yang, M. R. Lyu, and I. King, "Where you like to go next: Successive point-of-interest recommendation," in *Proc. IJCAI*, 2013, pp. 2605–2611.

[2] X. Liu, Y. Liu, K. Aberer, and C. Miao, "Personalized point-of-interest recommendation by mining users' preference transition," in *Proc. 22nd ACM Int. Conf. Conf. Inf. Knowl. Manage. (CIKM)*, 2013, pp. 733–738.

[3] D. Lian, C. Zhao, X. Xie, G. Sun, E. Chen, and Y. Rui, "GeoMF: Joint geographical modeling and matrix factorization for point-of-interest recommendation," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2014, pp. 831–840.

[4] B. Chang, Y. Park, D. Park, S. Kim, and J. Kang, "Content-aware hierarchical point-of-interest embedding model for successive POI recommendation," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, Jul. 2018, pp. 3301–3307.

[5] Y.-A. de Montjoye, C. A. Hidalgo, M. Verleysen, and V. D. Blondel, "Unique in the crowd: The privacy bounds of human mobility," *Sci. Rep.*, vol. 3, no. 1, pp. 1–5, Dec. 2013.

[6] G. Jung, H. Lee, A. Kim, and U. Lee, "Too much information: Assessing privacy risks of contact trace data disclosure on people with COVID-19 in South Korea," *Frontiers Public Health*, vol. 8, pp. 1–13, Jun. 2020.

[7] Ú. Erlingsson, V. Pihur, and A. Korolova, "RAPPOR: Randomized aggregatable privacy-preserving ordinal response," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2014, pp. 1054–1067.

[8] Differential Privacy Team Apple, "Learning with privacy at scale," *Mach. Learn. J.*, vol. 1, no. 8, pp. 1–25, 2017.

[9] B. Ding, J. Kulkarni, and S. Yekhanin, "Collecting telemetry data privately," in *Proc. NIPS*, 2017, pp. 3571–3580.

[10] T. T. Nguyên, X. Xiao, Y. Yang, S. Cheung Hui, H. Shin, and J. Shin, "Collecting and analyzing data from smart device users with local differential privacy," 2016, *arXiv:1606.05053*. [Online]. Available: http://arxiv.org/abs/1606.05053

[11] J. Hua, C. Xia, and S. Zhong, "Differentially private matrix factorization," in *Proc. IJCAI*, 2015, pp. 1763–1770.

[12] S. Zhang, L. Liu, Z. Chen, and H. Zhong, "Probabilistic matrix factorization with personalized differential privacy," 2018, *arXiv:1810.08509*. [Online]. Available: http://arxiv.org/abs/1810.08509

[13] H. Shin, S. Kim, J. Shin, and X. Xiao, "Privacy enhanced matrix factorization for recommendation with local differential privacy," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 9, pp. 1770–1782, Sep. 2018.

[14] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, Aug. 2009.

[15] W. Tan, L. Cao, and L. Fong, "Faster and cheaper: Parallelizing large-scale matrix factorization on GPUs," in *Proc. 25th ACM Int. Symp. High-Perform. Parallel Distrib. Comput.*, May 2016, pp. 219–230.

[16] M. M. Khan, R. Ibrahim, and I. Ghani, "Cross domain recommender systems: A systematic literature review," *ACM Comput. Surveys*, vol. 50, no. 3, pp. 1–34, Oct. 2017.

[17] W. Pan, E. W. Xiang, N. N. Liu, and Q. Yang, "Transfer learning in collaborative filtering for sparsity reduction," in *Proc. AAAI*, 2010, pp. 230–235.

[18] W. Pan, N. N. Liu, E. W. Xiang, and Q. Yang, "Transfer learning to predict missing ratings via heterogeneous user feedbacks," in *Proc. AAAI*, 2011, pp. 2318–2323.

[19] W. Pan and Q. Yang, "Transfer learning in heterogeneous collaborative filtering domains," *Artif. Intell.*, vol. 197, pp. 39–55, Apr. 2013.

[20] J. C. Duchi, M. I. Jordan, and M. J. Wainwright, "Local privacy and statistical minimax rates," in *Proc. IEEE 54th Annu. Symp. Found. Comput. Sci.*, Oct. 2013, pp. 429–438.

[21] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Theory Cryptogr. Conf.*, 2006, pp. 265–284.

[22] F. D. McSherry, "Privacy integrated queries: An extensible platform for privacy-preserving data analysis," in *Proc. 35th SIGMOD Int. Conf. Manage. Data (SIGMOD)*, 2009, pp. 19–30.

[23] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proc. CCS*, 2016, pp. 308–318.

[24] S. L. Warner, "Randomized response: A survey technique for eliminating evasive answer bias," *J. Amer. Stat. Assoc.*, vol. 60, no. 309, pp. 63–69, Mar. 1965.

[25] T. Wang, J. Blocki, N. Li, and S. Jha, "Locally differentially private protocols for frequency estimation," in *Proc. USENIX Secur. Symp.*, 2017, pp. 729–745.

[26] J. Acharya, Z. Sun, and H. Zhang, "Hadamard response: Estimating distributions privately, efficiently, and with little communication," in *Proc. PMLR*, 2019, pp. 1120–1129.

[27] S. Kim, H. Shin, C. Baek, S. Kim, and J. Shin, "Learning new words from keystroke data with local differential privacy," *IEEE Trans. Knowl. Data Eng.*, vol. 32, no. 3, pp. 479–491, Mar. 2020.

[28] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "BPR: Bayesian personalized ranking from implicit feedback," 2012, *arXiv:1205.2618*. [Online]. Available: http://arxiv.org/abs/1205.2618

[29] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets," in *Proc. 8th IEEE Int. Conf. Data Mining*, Dec. 2008, pp. 263–272.

[30] N. Wang, X. Xiao, Y. Yang, J. Zhao, S. C. Hui, H. Shin, J. Shin, and G. Yu, "Collecting and analyzing multidimensional data with local differential privacy," in *Proc. IEEE 35th Int. Conf. Data Eng. (ICDE)*, Apr. 2019, pp. 638–649.

[31] J. C. Duchi, M. I. Jordan, and M. J. Wainwright, "Privacy aware learning," *J. ACM*, vol. 61, no. 6, pp. 1–57, Dec. 2014.

[32] M. Li, T. Zhang, Y. Chen, and A. J. Smola, "Efficient mini-batch training for stochastic optimization," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2014, pp. 661–670.

[33] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *J. Mach. Learn. Res.*, vol. 12, pp. 2121–2159, Feb. 2011.

[34] G. Hinton, N. Srivastava, and K. Swersky, "Neural networks for machine learning lecture 6A overview of mini-batch gradient descent," *Cited*, vol. 14, no. 8, 2012.

[35] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: http://arxiv.org/abs/1412.6980

[36] H. Yin, X. Zhou, B. Cui, H. Wang, K. Zheng, and Q. V. H. Nguyen, "Adapting to user interest drift for POI recommendation," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 10, pp. 2566–2581, Oct. 2016.

[37] T. Guo, J. Luo, K. Dong, and M. Yang, "Locally differentially private item-based collaborative filtering," *Inf. Sci.*, vol. 502, pp. 220–246, 2019.

[38] N. Wang, X. Xiao, Y. Yang, T. D. Hoang, H. Shin, J. Shin, and G. Yu, "PrivTrie: Effective frequent term discovery under local differential privacy," in *Proc. IEEE 34th Int. Conf. Data Eng. (ICDE)*, Apr. 2018, pp. 821–832.

[39] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme, "Factorizing personalized Markov chains for next-basket recommendation," in *Proc. 19th Int. Conf. World Wide Web (WWW)*, 2010, pp. 811–820.

[40] S. Feng, X. Li, Y. Zeng, G. Cong, and Y. M. Chee, "Personalized ranking metric embedding for next new poi recommendation," in *Proc. IJCAI*, 2015, pp. 2069–2075.

[41] H. A. Rahmani, M. Aliannejadi, S. Ahmadian, M. Baratchi, M. Afsharchi, and F. Crestani, "LGLMF: Local geographical based logistic matrix factorization model for poi recommendation," in *Proc. Asia Inf. Retr. Symp.*, 2019, pp. 66–78.

[42] G. Fanti, V. Pihur, and Ú. Erlingsson, "Building a RAPPOR with the unknown: Privacy-preserving learning of associations and data dictionaries," *Proc. Privacy Enhancing Technol.*, vol. 2016, no. 3, pp. 41–61, Jul. 2016.

[43] R. Chen, G. Acs, and C. Castelluccia, "Differentially private sequential data publication via variable-length n-grams," in *Proc. ACM Conf. Comput. Commun. Secur. (CCS)*, 2012, pp. 638–649.

[44] M. Asada, M. Yoshikawa, and Y. Cao, "'When and where do you want to hide?'-recommendation of location privacy preferences with local differential privacy" in *Proc. IFIP Annu. Conf. Data Appl. Secur. Privacy*. Springer, 2019, pp. 164–176.

[45] C. Chen, J. Zhou, B. Wu, W. Fang, L. Wang, Y. Qi, and X. Zheng, "Practical privacy preserving POI recommendation," *ACM Trans. Intell. Syst. Technol.*, vol. 11, no. 5, pp. 1–20, Sep. 2020.

JONG SEON KIM received the B.S. degree in computer science from Korea University, in 2016, where he is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering. His research interests include location privacy and recommendation systems.

JONG WOOK KIM (Member, IEEE) received the Ph.D. degree from the Computer Science Department, Arizona State University, in 2009. He was a Software Engineer with the Query Optimization Group, Teradata, from 2010 to 2013. He is currently an Assistant Professor of computer science with Sangmyung University. His current research interests include data privacy, distributed databases, and query optimization.

YON DOHN CHUNG (Member, IEEE) received the B.S. degree in computer science from Korea University, and the M.S. and Ph.D. degrees in computer science from the Korea Advanced Institute of Science and Technology. In 2006, he joined as a Faculty Member with the Department of Computer Science and Engineering, Korea University. His research interests include data privacy, spatial databases, and array databases.

• • •