Synthesizing Privacy-Preserving Location Traces Including Co-locations

Synthesizing Privacy-Preserving Location Traces Including Co-locations

Jun Narita[1], Yayoi Suganuma[1], Masakatsu Nishigaki[1], Takao Murakami[2], and Tetsushi Ohki[1]

[1] Shizuoka University, Hamamatsu, Shizuoka, Japan
{narita,suganuma}@sec.inf.shizuoka.ac.jp
{nisigaki,ohki}@inf.shizuoka.ac.jp
[2] AIST, Koto-ku, Tokyo, Japan
takao-murakami@aist.go.jp

Abstract. Location traces are useful for various types of geo-data analysis tasks, and synthesizing location traces is a promising approach to geo-data analysis while protecting user privacy. However, existing location synthesizers do not consider friendship information of users. In particular, a co-location between friends is an important factor for synthesizing more realistic location traces.
In this paper, we propose a novel location synthesizer that generates synthetic traces including co-locations between friends. Our synthesizer models the information about the co-locations by two parameters: friendship probability and co-location count matrix. The friendship probability represents a probability that two users will be a friend, whereas the co-location count matrix comprises a co-location count for each time instant and each location. Our synthesizer also provides DP (Differential Privacy) for training data. We evaluate our synthesizer using the Foursquare dataset. Our experimental results show that our synthesizer preserves the information about co-locations and other statistical information (e.g., population distribution, transition matrix) while providing DP with a reasonable privacy budget (e.g., smaller than 1).

Keywords: synthetic trace · co-location · differential privacy · Laplace mechanism · wavelet transform · Viterbi algorithm

1 Introduction

With the widespread use of mobile phones, LBS (Location-Based Services), which utilize a user's location information for some services (e.g., predicting traffic congestion, user-tailored route suggestion), are becoming increasingly popular. LBS providers can also provide a large amount of location traces (time-series location trails) to a third party (data analyst) to perform various geo-data analyses such as finding popular POIs (Point-of-Interests) in the surrounding area [1] and modeling human mobility patterns [2]. However, the disclosure of location traces raises serious privacy concerns. For example, location traces may include sensitive locations such as frequently visited hospitals and users' home.

To address the privacy issue, privacy-preserving location synthesizers (e.g., [3–5]) have been widely studied. The location synthesizer trains a generative model from real trace data, and then generates synthetic traces based on the trained generative model. Ideal synthetic traces should preserve various statistical features (e.g., population distribution [1], transition matrix [6]) of real trace data while being able to protect user privacy. Synthetic traces can also be used as a synthetic dataset for research purposes [7] and competitions [8].

However, the existing location synthesizers do not take into account friendship information between users. In particular, friends tend to be in the same place at the same time [9], which is also known as a *co-location* [10, 11]. Thus, a co-location between friends is an important factor for synthesizing more realistic location traces. For example, a synthetic trace dataset including co-locations between friends may be useful for studying the effect of friend recommendation [9].

In this paper, we propose a novel location synthesizer that generates synthetic traces including co-locations between friends. To preserve co-locations information, our proposed method trains two parameters from real trace data: *friendship probability* and *co-location count matrix*. The friendship probability models a probability that two users will become friends. The co-location count matrix comprises a co-location count for each time instant and each location, and models a location that is likely to be visited by friends at a certain time period (e.g., bars at night). Both the friendship probability and the co-location count matrix provide strong privacy guarantee: *DP (Differential Privacy)* [12], a gold standard for data privacy. Our contributions are as follows:

– We propose a novel location synthesizer that generates location traces including co-locations between friends. Our proposed method models the information about co-locations between friends by two parameters: *friendship probability* and *co-location count matrix*. Both the friendship probability and the co-location count matrix provide DP for training data.
– We show that synthetic traces generated by the proposed method provide high utility and privacy using the Foursquare dataset [9]. Specifically, we show that our synthetic traces preserve the information about co-locations and other statistical features (e.g., population distribution, transition matrix) while providing DP with a reasonable privacy budget ε (e.g., $\varepsilon \leq 1$).

2 Related Work

2.1 Co-locations

In this paper, we define a co-location as an *event that two users are in the same place at the same time*. In particular, we focus on a co-location between friends and use it to generate synthetic traces. Olteanu et al. [10, 11] showed that co-location information improves the accuracy of location estimation attacks. They also studied the users' benefits of sharing co-locations and the impact of co-locations on location privacy [13]. Yang et al. [9] showed a correlation between co-location information and friendships on Twitter.

However, no studies have utilized co-location information for generating synthetic traces, to the best of our knowledge.

2.2 Location Synthesizers

Location synthesizers have been widely studied for over a decade (see [3, 5] for detailed surveys). Bindschaeder and Shokri [3] proposed generating synthetic traces by a synthetic location generation model that considers semantic features of locations (e.g., most people stay a night at their home locations, which are geographically different but semantically the same). He et al. [4] proposed generating synthetic traces that satisfy small ε-differential privacy by training a transition probability matrix common to all users. Murakami et al. [14] proposed generating synthetic traces with high utility by clustering a transition matrix for each user using tensor factorization.

However, there are no studies in which a co-location between friends is considered for generating synthetic traces, to our knowledge. As shown in [9], there is a positive correlation between co-locations and friendships; i.e., friends tend to be in the same place at the same time. Therefore, location synthesizers considering such co-location information are important to synthesize more realistic traces. To our knowledge, we are the first to address this issue.

3 Problem Formalization

3.1 Notations

Let $\mathbb{N}$, $\mathbb{Z}_{\geq 0}$, $\mathbb{R}$, and $\mathbb{R}_{\geq 0}$ be the set of natural numbers, non-negative integers, real numbers, and non-negative real numbers, respectively. For a finite set $\mathcal{Z}$, let $\mathcal{Z}^*$ be the set of all finite sequences of elements of $\mathcal{Z}$.

Let $\mathcal{U}$ be a finite set of users in training data, and $n \in \mathbb{N}$ be the number of users; i.e., $n = |\mathcal{U}|$. Let $u_i \in \mathcal{U}$ be the i-th user. We discretize locations by dividing an area of interest into some regions or extracting some POIs. Let $\mathcal{X}$ be a finite set of locations, and x_i be the i-th location. We also discretize time by, for example, rounding down minutes to a multiple of 30. Let $\mathcal{T}$ be a finite set of time instants, and $t_i \in \mathcal{T}$ be the i-th time instant.

We also show the basic notations used in this paper in Table 1.

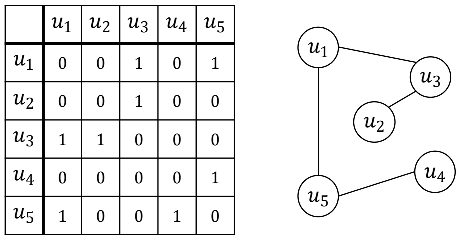
3.2 Friendship Data

In this paper, we use *friendship data* and *real trace data* as training data to generate synthetic location traces including co-locations.

The friendship data contains friendship information between any pair of two users, and therefore can be represented as an adjacency matrix of size $n \times n$. If two users are friends, then the corresponding element will be assigned value "1"; Otherwise, "0". The diagonal elements are "0" because there is no friend relationship between the user and him/herself. Fig. 1 shows an example of the

Table 1. Basic notations in this paper.

Symbol	Description
$\mathcal{U}$	Finite set of users
n	Number of users ($n = \|\mathcal{U}\|$)
$\mathcal{X}$	Finite set of locations
$\mathcal{T}$	Finite set of time instants
$\mathcal{E}$	Finite set of events ($\mathcal{E} = \mathcal{X} \times \mathcal{T}$)
$\mathcal{S}$	Finite set of training traces ($\mathcal{S} \subseteq \mathcal{U} \times \mathcal{E}^*$)
u_i	i-th user ($u_i \in \mathcal{U}$)
x_i	i-th location ($x_i \in \mathcal{X}$)
t_i	i-th time instant ($t_i \in \mathcal{T}$)
$\mathbf{A}$	Adjacency matrix ($\mathbf{A} \in \{0,1\}^{n \times n}$)
$\mathbf{a}_i$	i-th row of the adjacency matrix $\mathbf{A}$ ($\mathbf{a}_i \in \{0,1\}^n$)
s_i	i-th training trace ($s_i \in \mathcal{S}$)



	u_1	u_2	u_3	u_4	u_5
u_1	0	0	1	0	1
u_2	0	0	1	0	0
u_3	1	1	0	0	0
u_4	0	0	0	0	1
u_5	1	0	0	1	0

Fig. 1. An example of friendship data ($n = 5$).

friendship data. In this example, user u_1 is a friend with u_3 and u_5. u_2 is a friend with u_3.

Let $\mathbf{A} \in \{0,1\}^{n \times n}$ be an adjacency matrix, and $\mathbf{a}_i \in \{0,1\}^n$ be the i-th row of $\mathbf{A}$. In Fig. 1, $\mathbf{a}_1 = (0,0,1,0,1)$, $\mathbf{a}_2 = (0,0,1,0,0)$, $\cdots$, $\mathbf{a}_5 = (1,0,0,1,0)$. The friendship data can also be represented as a graph (as shown in Fig. 1), where a node represents a user and an edge represents a friendship between two users.

3.3 Trace Data

We define an event by a pair of a location and time instant, as in [5]. The trace data contains a location trace for each user, and a location trace contains events. Fig. 2 shows an example of the trace data. Co-location events are marked in red. In this example, users u_2 and u_3 have a co-location event at location x_1 and time instant t_1. u_1 and u_2 have a co-location event at x_2 and t_2.

Let $\mathcal{E} = \mathcal{X} \times \mathcal{T}$ be a finite set of events. Let $\mathcal{S} \subseteq \mathcal{U} \times \mathcal{E}^*$ be a finite set of training traces, and $s_i \in \mathcal{S}$ be the i-th training trace. Fig. 2 can be rewritten as $\mathcal{S} = \{s_1, s_2, s_3, s_4\}$ and $s_1 = (u_1, (x_3, t_1), (x_2, t_2), (x_3, t_3), (x_3, t_4))$, $\cdots$, $s_4 = (u_4, (x_2, t_1), (x_3, t_2), (x_2, t_3), (x_3, t_4))$.

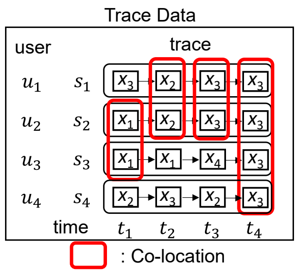


Fig. 2. An example of trace data ($n = 4$).

3.4 Threat Model and Differential Privacy

Threat Model. In this paper, we consider privacy preservation against an attacker with any background knowledge other than the training data (friendship data and trace data) used for generating synthetic traces. The attacker obtains the synthetic traces, and attempts to violate user privacy in the training data (e.g., membership inference attacks [15, 16]) based on the synthetic traces and his/her background knowledge.

To strongly protect user privacy in the training data from the attacker with any background knowledge, we use *differential privacy (DP)* [12,17] as a privacy metric. DP provides user privacy against attackers with any background knowledge and is known as a gold standard for data privacy. Below we explain DP for friendship data and trace data in detail.

DP for Friendship Data. In graphs, there are two types of DP: *edge DP* and *node DP* [18]. Edge DP hides the existence of one edge (friendship). In contrast, node DP hides the existence of one node (user) and all edges connected to the node, and hence guarantees much stronger privacy than edge DP. Therefore, we use node DP to protect user privacy in the friendship data.

Formally, node DP considers two *neighboring* adjacency matrices $\mathbf{A}$ and $\mathbf{A}'$ that differ in one user and his/her all friendship information. For example, consider a graph obtained by removing u_3 and all edges connected to u_3 in Fig. 1. Let $\mathbf{A}' \in \{0,1\}^{4\times 4}$ be its adjacency matrix. In this case, $\mathbf{a}'_1 = (0,0,0,1)$, $\mathbf{a}'_2 = (0,0,0,0)$, $\mathbf{a}'_3 = (0,0,0,1)$, and $\mathbf{a}'_4 = (1,0,1,0)$. $\mathbf{A} \in \{0,1\}^{5\times 5}$ in Fig. 1 and $\mathbf{A}'$ in this example are neighboring adjacency matrices.

Then node DP [18] is defined as follows.

Definition 1 (ε_1-node DP). *Let $\varepsilon_1 \in \mathbb{R}_{\geq 0}$. A randomized algorithm $\mathcal{M}_1$ provides ε_1-*node DP *if for any two neighboring adjacency matrices $\mathbf{A}$ and $\mathbf{A}'$ that differ in one node and its adjacent edges and any $Z \subseteq \text{Range}(\mathcal{M}_1)$,*

$$\Pr[\mathcal{M}_1(\mathbf{A}) \in Z] \leq e^{\varepsilon_1} \Pr[\mathcal{M}_1(\mathbf{A}') \in Z]. \quad (1)$$

Intuitively, node DP guarantees that an adversary who obtains the output of $\mathcal{M}_1$ cannot determine whether a particular node is included or not with a certain degree of confidence. This property is independent of the adversary's background knowledge. ε_1 is called the *privacy budget*. When the privacy budget ε_1 is small (e.g., $\varepsilon_1 \leq 1$ [19]), privacy of each node (user) is strongly protected.

DP for Trace Data. We can also consider two types of DP for trace data: *event-level DP* and *user-level DP* [20]. Event-level DP protects one event in a trace, whereas user-level DP protects the entire history (i.e., whole trace) of one user. Thus, user-level DP guarantees much stronger privacy than event-level DP. We use user-level DP to protect user privacy in the trace data.

Formally, we consider two *neighboring* sets $\mathcal{S}$ and $\mathcal{S}'$ of training traces such that $\mathcal{S}'$ is obtained by adding or removing a training trace of one user in $\mathcal{S}$. For example, consider a set $\mathcal{S}'$ of training traces obtained by removing s_3 in Fig. 2; i.e., $\mathcal{S}' = \{s_1, s_2, s_4\}$. $\mathcal{S}$ in Fig. 2 and $\mathcal{S}'$ in this example are neighboring.

Then user-level DP can be defined as follows.

Definition 2 (ε_2-user-level DP). *Let $\varepsilon_2 \in \mathbb{R}_{\geq 0}$. A randomized algorithm $\mathcal{M}_2$ provides* ε_2-user-level DP *if for any two neighboring sets $\mathcal{S}$ and $\mathcal{S}'$ of training traces that differ in one trace and any $Z \subseteq \mathrm{Range}(\mathcal{M}_2)$,*

$$\Pr[\mathcal{M}_2(\mathcal{S}) \in Z] \leq e^{\varepsilon_2} \Pr[\mathcal{M}_2(\mathcal{S}') \in Z]. \quad (2)$$

User-level DP guarantees that an adversary who obtains the output of $\mathcal{M}_2$ cannot determine whether a particular user is included or not in trace data. Again, privacy of each user is strongly protected when the privacy budget ε_2 is small (e.g., $\varepsilon_2 \leq 1$). In addition, the privacy budget for each user in the training data can be calculated as the sum of ε_1 and ε_1 by the composition theorem [21]. Therefore, user privacy in the training data is strongly protected by using ε_1-node DP with small ε_1 for friendship data and ε_2-user-level DP with small ε_2 for trace data.

4 Proposed Method

4.1 Overview

Fig. 3 shows the overview of generating synthetic traces using our proposed method. The proposed method uses a location dataset that includes both trace data and friendship data (e.g., Foursquare dataset in [9]) as training data. To distinguish between trace data in the training data and synthetic trace data, we refer to the former trace data as *real trace data*.

In a nutshell, the proposed method generates co-locations between friends using two parameters: *friendship probability* $p' \in [0,1]$ and *co-location count matrix* $\mathbf{Q}' \in \mathbb{Z}_{\geq 0}^{|\mathcal{T}| \times |\mathcal{X}|}$. The friendship probability p' represents the probability that two users are friends. The co-location count matrix $\mathbf{Q}'$ includes a co-location count for each time instant and each location. For example, if friends tend to meet at a bar from 6PM to 8PM, then the corresponding elements in $\mathbf{Q}'$ have large co-location counts.

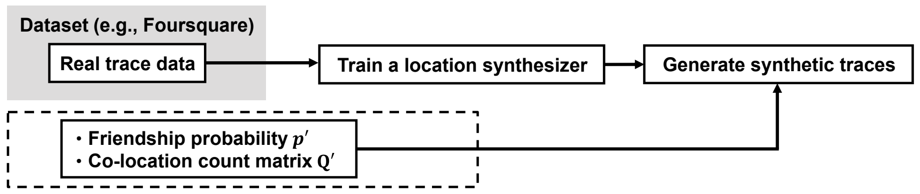

Dataset (e.g., Foursquare)
Real trace data
Train a location synthesizer
Generate synthetic traces
・Friendship probability p'
・Co-location count matrix $\mathbf{Q}'$

Fig. 3. Overview of generating synthetic traces using the proposed method. p' and $\mathbf{Q}'$ in the dotted box are parameters in the proposed method to generate co-locations.

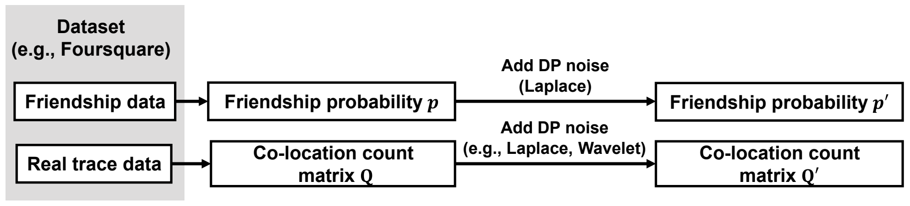

Dataset
(e.g., Foursquare)
Friendship data
Real trace data
Friendship probability p
Co-location count matrix $\mathbf{Q}$
Add DP noise (Laplace)
Add DP noise (e.g., Laplace, Wavelet)
Friendship probability p'
Co-location count matrix $\mathbf{Q}'$

Fig. 4. Overview of training parameters p' and $\mathbf{Q}'$ in the proposed method. p' and $\mathbf{Q}'$ provide node DP and user-level DP, respectively.

The proposed method randomly generates friendship information in synthetic traces using p'. Then it generates co-locations between friends at a specific location and time instant using $\mathbf{Q}'$. The generated co-locations preserve the information about $\mathbf{Q}'$; e.g., friends tend to meet at a bar from 6PM to 8PM. After generating co-locations, the proposed method generates the remaining events in synthetic traces using existing location synthesizers that model human movement patterns as a *transition matrix*. Such location synthesizers include the location synthesizers in [3–5, 22]. In our experiments, we use the synthetic data generator in [22] because it is easy to implement and provides user-level DP with a small privacy budget. We train the existing location synthesizer using real trace data.

Fig. 4 shows the overview of training p' and $\mathbf{Q}'$ in the proposed method. From friendship data, we calculate the friendship probability $p \in [0, 1]$. Then we generate p' by adding the Laplace noise [12] to p. The noisy friendship probability p' provides *node DP*.

From real trace data, we calculate the co-location count matrix $\mathbf{Q}$ by simply counting co-locations between friends. Then we generate a co-location count matrix $\mathbf{Q}'$ that provides *user-level DP* by adding noise to $\mathbf{Q}$. The simplest way to provide DP for $\mathbf{Q}'$ is to add the Laplace noise to each element in $\mathbf{Q}$. However, the total amount of the Laplace noise can be very large because the number of elements in $\mathbf{Q}$ is large ($|\mathcal{T}||\mathcal{X}|$ elements in total). Therefore, we use a DP mechanism based on the Wavelet transform called *Privelet* [23]. Privelet (for one-dimensional nominal data) applies a nominal Wavelet transform to a one-dimensional frequency matrix, and adds independent Laplace noise to each wavelet coefficient (each node in a tree structure). Privelet can add noise effectively when a category or tree structure of locations is known. For example, the Foursquare dataset [9] has a category (e.g., "travel & transport", "shop &

service") and sub-category (e.g., "train station", "subway", "electronics store", "hobby shop") of POIs [24], and each POI is associated with a category and sub-category. In this case, Privelet can be used to provide DP for $\mathbf{Q}'$ with a much smaller amount of noise for each POI category and each time instant (e.g., "travel & transport" from 7AM to 9AM). In our experiments, we compare Privelet with the Laplace mechanism, and show that the former significantly reduces the noise.

Features. The main feature of our proposed method is that it can generate synthetic traces including *co-locations*. The synthetic traces preserve the information about p' (i.e., how likely two users will be a friend) and $\mathbf{Q}'$ (i.e., how likely a co-location event will happen at a certain location for each time instant), both of which are trained from training data. In addition, the synthetic traces strongly protect user privacy: node DP for friendship data and user-level DP for real trace data. The privacy budgets are also reasonable (e.g., smaller than 1), as shown in our experiments.

4.2 Friendship Probability p'

Calculation of p'. Below we explain how to calculate the friendship probability $p' \in [0, 1]$ in our method.

First, we calculate the friendship probability p by simply calculating the proportion of edges in friendship data; i.e., the proportion of "1"s in the non-diagonal elements of adjacency matrix $\mathbf{A}$. In the example of Fig. 1, we can calculate p as $p = 8/(5 \times 4) = 0.4$. For $b \in \mathbb{R}_{\geq 0}$, let $\mathrm{Lap}(b)$ be a random variable that represents the Laplace noise with mean 0 and scale b. Then we calculate p' by adding the Laplace noise $\mathrm{Lap}(\frac{2}{n\varepsilon_1})$; i.e., $p' = p + \mathrm{Lap}(\frac{2}{n\varepsilon_1})$.

DP of p'. Let $\mathcal{M}_1^{\mathrm{Lap}}$ be a randomized algorithm that takes an adjacency matrix $\mathbf{A}$ as input and outputs p'. Then we have the following privacy guarantee.

Proposition 1. *$\mathcal{M}_1^{Lap}$ provides ε_1-node DP.*

Proof. Let $f : \{0, 1\}^{n \times n} \to [0, 1]$ be a function that takes $\mathbf{A}$ as input and outputs p. Let Δf be the global sensitivity [12] of f given by:

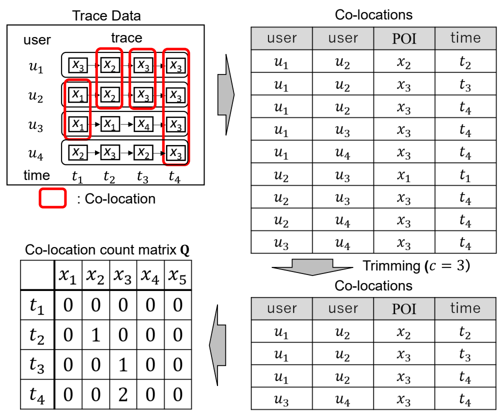
$$\Delta f = \max_{\mathbf{A} \sim \mathbf{A}'} |f(\mathbf{A}) - f(\mathbf{A}')|, \quad (3)$$

where $\mathbf{A} \sim \mathbf{A}'$ represents that $\mathbf{A}$ and $\mathbf{A}'$ are neighboring. Let $d \in \mathbb{Z}_{\geq 0}$ be the number of "1"s in $\mathbf{A}$. Assume that $\mathbf{A}'$ is obtained by removing one node in $\mathbf{A}$. In this case, Δf takes the maximum value when the removed node has edges with all the other nodes. Thus, when $n \geq 3$, we have:

$$|f(\mathbf{A}) - f(\mathbf{A}')| \leq \frac{d}{n(n-1)} - \frac{d-2(n-1)}{(n-1)(n-2)} < \frac{d}{n(n-1)} - \frac{d-2(n-1)}{n(n-1)} = \frac{2}{n}. \quad (4)$$

Note that $f(\mathbf{A}) \leq 1$ when $n = 2$ and $f(\mathbf{A}) = 0$ when $n \leq 1$. Thus, (4) also holds when $n < 3$. When $\mathbf{A}'$ is obtained by adding one node in $\mathbf{A}$, the right-hand side of (4) becomes $\frac{2}{n+1} < \frac{2}{n}$. Therefore, $\Delta f \leq \frac{2}{n}$.

Since adding the Laplace noise $\mathrm{Lap}(\Delta f/\varepsilon_1)$ to p provides ε_1-DP [12] and $\Delta f \leq \frac{2}{n}$, $\mathcal{M}_1^{\mathrm{Lap}}$ that adds $\mathrm{Lap}(\frac{2}{n\varepsilon_1})$ to p provides ε_1-node DP. □


Trace Data
user
trace
u_1 x_3 x_2 x_3 x_3
u_2 x_1 x_2 x_3 x_3
u_3 x_1 x_1 x_4 x_3
u_4 x_2 x_3 x_2 x_3
time t_1 t_2 t_3 t_4
: Co-location
Trimming ($c = 3$)

Co-locations

user	user	POI	time
u_1	u_2	x_2	t_2
u_1	u_2	x_3	t_3
u_1	u_2	x_3	t_4
u_1	u_3	x_3	t_4
u_1	u_4	x_3	t_4
u_2	u_3	x_1	t_1
u_2	u_3	x_3	t_4
u_2	u_4	x_3	t_4
u_3	u_4	x_3	t_4

Co-locations

user	user	POI	time
u_1	u_2	x_2	t_2
u_1	u_2	x_3	t_3
u_1	u_2	x_3	t_4
u_3	u_4	x_3	t_4

Co-location count matrix **Q**

	x_1	x_2	x_3	x_4	x_5
t_1	0	0	0	0	0
t_2	0	1	0	0	0
t_3	0	0	1	0	0
t_4	0	0	2	0	0

Fig. 5. Overview of calculating the co-location count matrix **Q**.

4.3 Co-location Count Matrix $\mathbf{Q}'$

Calculation of $\mathbf{Q}'$. Next we explain how to calculate the co-location count matrix $\mathbf{Q}' \in \mathbb{Z}_{\geq 0}^{|\mathcal{T}|\times|\mathcal{X}|}$ in the proposed method.

From the real trace data, we calculate the co-location count matrix $\mathbf{Q} \in [0,1]^{|\mathcal{T}|\times|\mathcal{X}|}$, which includes the number of co-locations for each time instant and each location. Here we introduce an upper limit $c \in \mathbb{Z}_{\geq 0}$ on the number of co-locations per user – when the number of co-locations reaches c, the user's co-locations are not read anymore. This is called *trimming* [25] and is used to upper-bound the global sensitivity in DP. Fig. 5 shows an overview of calculating $\mathbf{Q}$ when $c = 3$. In this example, user u_1's co-locations are not read after reading three co-locations of u_1.

Then we calculate $\mathbf{Q}'$ by adding noise to $\mathbf{Q}$. In this paper, we use the Laplace mechanism or Privelet (for nominal data) [23] to add noise. The Laplace mechanism simply adds $\mathrm{Lap}(c/\varepsilon_2)$ to each element of $\mathbf{Q}$. Privelet applies the Wavelet transform to a tree structure of locations, and adds the Laplace noise to a wavelet coefficient for each node in the tree. See [23] for more details.

DP of $\mathbf{Q}'$. Let $\mathcal{M}_2^{\mathrm{Lap}}$ be a randomized algorithm that takes a set $\mathcal{S}$ of training traces as input and outputs $\mathbf{Q}'$ by adding $\mathrm{Lap}(c/\varepsilon_2)$ to each element of $\mathbf{Q}$. Then we have the following privacy guarantee.

Proposition 2. *$\mathcal{M}_2^{Lap}$ provides ε_2-user-level DP.*

Proof. Since we read at most c co-locations per user from $\mathcal{S}$, adding or removing a training trace of one user in $\mathcal{S}$ changes each element of $\mathbf{Q}$ by at most c. Thus, the global sensitivity of each element of $\mathbf{Q}$ is at most c. Since $\mathcal{M}_2^{\mathrm{Lap}}$ adds $\mathrm{Lap}(c/\varepsilon_2)$ to each element of $\mathbf{Q}$, it provides ε_2-user-level DP. □

Let $\mathcal{M}_2^{\mathrm{Wavelet}}$ be Privelet. Then $\mathcal{M}_2^{\mathrm{Wavelet}}$ with the Laplace noise based on the global sensitivity c also provides ε_2-user-level DP. See [23] for the proof.

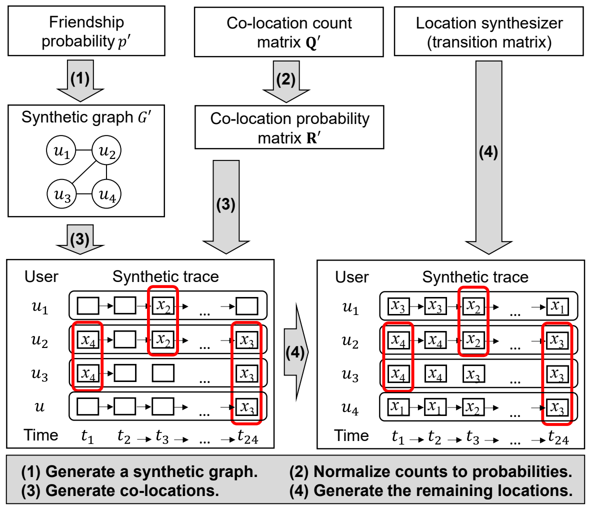

Friendship probability p'
Co-location count matrix $\mathbf{Q}'$
Location synthesizer (transition matrix)
(1)
(2)
Co-location probability matrix $\mathbf{R}'$
Synthetic graph G'
(3)
(4)
User
Synthetic trace
Time
(1) Generate a synthetic graph.
(2) Normalize counts to probabilities.
(3) Generate co-locations.
(4) Generate the remaining locations.

Fig. 6. Overview of generating synthetic traces in our proposed method.

4.4 Generating Synthetic Traces

We generate a synthetic trace using the friendship probability p', co-location count matrix $\mathbf{Q}'$, and a location synthesizer that trains a transition matrix from real trace data (e.g., [3–5, 22]).

Fig. 6 shows the overview of generating synthetic traces in our proposed method. Specifically, we generate a synthetic trace for each of n users as follows.

1. Generate a synthetic graph G' with n nodes based on the Erdös-Rényi model, which randomly generates each edge with probability p'.
2. Calculate the category co-location probability matrix $\mathbf{R}' \in [0,1]^{|\mathcal{T}|\times|\mathcal{X}|}$ by normalizing each row of the co-location count matrix $\mathbf{Q}'$ so that the sum of the rows is 1. Note that an element in $\mathbf{Q}'$ may have a negative value. Thus, we calculate each row of $\mathbf{R}'$ by adding the absolute value of the minimum value to all elements and dividing them by their sum.
3. Generate $\theta \in \mathbb{N}$ co-location events between friends (who have an edge in G'). Specifically, we iterate the following three steps until θ co-location events are obtained: (i) Randomly select a pair of friends from G'; (ii) Randomly select a time instant from $\mathcal{T}$; (iii) Randomly generate a co-location at the selected time instant by using the corresponding row of $\mathbf{R}'$. Note that if either of the two users in step (i) has already had a co-location at a time instant selected in step (ii), we use it as a co-location in step (iii) for consistency with the previously generated co-location.
4. Generate the remaining locations in n synthetic traces using the transition matrix of a location synthesizer (e.g., [3–5, 22]). Specifically, we use the Viterbi algorithm to complement the remaining locations.

Note that the number θ of co-location events is a parameter in our proposed method. We set θ to various values in our experiments. It is also possible to

Table 2. POI categories and sub-categories.

POI category	POI sub-category
travel & transport	train station, airport, platform, subway, airport terminal
shop & service	electronics store, hobby shop, record shop, mall
arts & entertainment	arcade
professional & other places	tech startup, convention center

calculate the frequency of co-location events (with DP noise) from real trace data, and set θ based on the co-location frequency.

Although we use the Erdös-Rényi model to generate a graph G' for simplicity, there exist more complicated and realistic graph models (e.g., Barabási–Albert model [26]) that have power-law degree distributions. An interesting avenue of future work is to incorporate such models into our proposed method.

5 Experimental Evaluation

5.1 Datasets

In our experiments, we used the Foursquare dataset in [9]. This dataset contains 22,809,624 check-ins from 114,324 users and 3,820,891 POIs. The dataset also includes the users' friendship data on SNS and a category and sub-category of POIs [24]. We used the check-in data in Tokyo (916,136 check-ins, 8,357 checked-in users, and 83,647 POIs). We set the length of a time instant to be one hour, and extracted two temporally-continuous location events from the dataset ($|\mathcal{T}| = 24$).

Since the number of check-ins for each POI is highly biased, the matrix $\mathbf{Q}$ becomes extremely sparse when we use all POIs. Therefore, we used check-in data for 100 POIs whose numbers of check-ins are the largest ($n = 8357$, $|\mathcal{X}| = 100$). The number m of POI categories was $m = 4$. The number of POI sub-categories was 12. Table 2 shows the POI categories and sub-categories.

5.2 Utility Metrics

Co-locations. To quantitatively show how our proposed method preserves the information about co-locations, we evaluated the utility of the friendship probability p' and the co-location count matrix $\mathbf{Q}'$.

Specifically, we evaluated the absolute error $|p - p'|$ between p and p' as a utility metric for p'. For utility of $\mathbf{Q}'$, co-location counts for each POI category and each time instant (e.g., "travel & transport" from 7AM to 9AM) are especially important. Thus, we did the following. Let $\mathbf{Q}^* \in \mathbb{Z}_{\geq 0}^{|\mathcal{T}|\times|\mathcal{X}|}$ be a co-location count matrix before adding noise when we do not perform trimming. $\mathbf{Q}^*$ is identical to $\mathbf{Q}$ when $c = \infty$. We calculated a *per-category* co-location count matrix $\overline{\mathbf{Q}}^* \in \mathbb{Z}_{\geq 0}^{|\mathcal{T}|\times m}$ ($|\mathcal{T}| = 24$, $m = 4$), which comprises of counts for each time instant and each POI category, by summing up counts in $\mathbf{Q}^*$ for each POI category. Similarly, we calculated a per-category co-location count matrix

$\overline{\mathbf{Q}}' \in \mathbb{Z}_{\geq 0}^{|\mathcal{T}|\times m}$ by summing up counts in $\mathbf{Q}'$ for each POI category. Then we evaluated the MAE (Mean Absolute Error) and MSE (Mean Square Error) between $\overline{\mathbf{Q}}^*$ and $\overline{\mathbf{Q}}'$. The MAE is given by: $\frac{1}{|\mathcal{T}|m}\sum_{i=1}^{|\mathcal{T}|}\sum_{j=1}^{m}|\overline{\mathbf{Q}}^*_{ij} - \overline{\mathbf{Q}}'_{ij}|$, where $\overline{\mathbf{Q}}^*_{ij}$ and $\overline{\mathbf{Q}}'_{ij}$ are the (i,j)-th elements of $\overline{\mathbf{Q}}^*$ and $\overline{\mathbf{Q}}'$, respectively. The MSE is given by: $\frac{1}{|\mathcal{T}|m}\sum_{i=1}^{|\mathcal{T}|}\sum_{j=1}^{m}(\overline{\mathbf{Q}}^*_{ij} - \overline{\mathbf{Q}}'_{ij})^2$. Note that the difference between $\overline{\mathbf{Q}}^*$ and $\overline{\mathbf{Q}}'$ can be caused by two factors: trimming and adding DP noise.

Our proposed method randomly generates co-locations in synthetic traces based on p' and $\mathbf{Q}'$. Thus, when the absolute error of p' is small, our synthetic traces preserve the information about how likely two users will be a friend. When the MAE and MSE of $\overline{\mathbf{Q}}'$ are small, our synthetic traces preserve the information about how likely a co-location event between friends will happen at a certain POI category for each time instant (e.g., "travel & transport" from 7 to 9AM).

Other statistical features. We also evaluated how our synthetic traces preserve statistical features (other than co-locations) about real trace data. Specifically, we calculated two basic statistical features for geo-data analysis: *population distribution* and *transition probability matrix*. The population distribution ($|\mathcal{X}|$-dimensional probability vector) is a key feature to find popular POIs [1], whereas the transition probability matrix ($|\mathcal{X}| \times |\mathcal{X}|$ matrix) is a key feature to model user movement patterns [6]. For both of them, we evaluated the MAE and MSE between real trace data and synthetic traces.

5.3 Location Synthesizers

We evaluated three location synthesizers for comparison. The first synthesizer is a simple one that independently and randomly generates a location at each time instant from the uniform distribution. We denote this method by Uniform.

The second synthesizer is the synthetic data generator in [3]. This synthesizer can be applied to any kind of data, and it was applied to location traces in [5]. This synthesizer can be applied to location traces as follows. First, we train a transition probability matrix ($|\mathcal{X}| \times |\mathcal{X}|$ martrix) common to all users from real trace data, and add the Laplace noise $\mathrm{Lap}(c/\varepsilon_2)$ to each element to provide ε_2-user-level DP. Then we randomly generate the first location based on a stationary distribution calculated from the transition matrix, and then generate the remaining locations using the transition matrix. Since this method is based on the transition probability matrix, we denote this method by TPM.

The third synthesizer is our proposed method. We denote it by Proposal. In Proposal, we trained p' from friendship data by adding the Laplace noise, and $\mathbf{Q}'$ from real trace data using the Laplace mechanism or Privelet. Then we generated θ co-locations using p' and $\mathbf{Q}'$. Finally, we generated the remaining locations using TPM.

In each location synthesizer, we set the length of a time instant to be one hour, and generated a trace with the length of one day for each of n users. For each synthesizer, we generated synthetic traces five times, and averaged the utility metrics over the five runs to stabilize the performance.

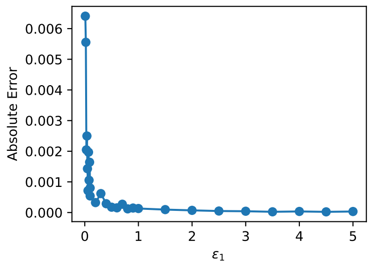


Fig. 7. Absolute error of p' versus ε_1.

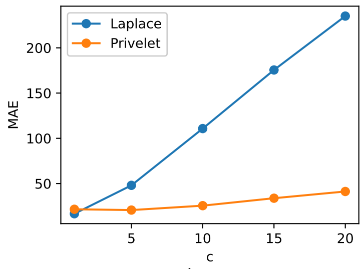


Fig. 8. MAE of $\overline{\mathbf{Q}}'$ versus c ($\varepsilon_2 = 1$).

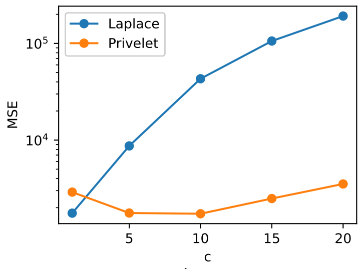


Fig. 9. MSE of $\overline{\mathbf{Q}}'$ versus c ($\varepsilon_2 = 1$).

5.4 Experimental Results

Friendship probability p'. Fig. 7 shows the absolute error of p' when we changed the privacy budget ε_1 from 0.01 to 5.

Fig. 7 shows that the absolute error rapidly decreases as ε_1 increases from 0.01 to 0.5. It also shows that the absolute error is very small and almost unchanged after $\varepsilon_1 = 0.5$, which means that we can accurately estimate the friendship probability p' with a small privacy budget $\varepsilon_1 = 0.5$ in node DP for friendship data.

Per-category co-location count matrix $\overline{\mathbf{Q}}'$. Fig. 8 and Fig. 9 show the MAE/MSE of $\overline{\mathbf{Q}}'$. Here we set the privacy budget ε_2 in user-level DP for real trace data to $\varepsilon_2 = 1$, and the upper limit c on the number of locations per user in trimming to $c = 1$, 5, 10, 15, or 20.

Fig. 8 and Fig. 9 show that Privelet achieves much smaller MAE and MSE than the Laplace mechanism, which means that Privelet significantly reduces the amount of noise for each POI category and each time instant (e.g., bars at night). These figures also show that when $c = 5$ and 10, Privelet achieves the smallest MAE and MSE, respectively. This indicates that there is a trade-off between the effect of trimming (which is large when c is small) and the Laplace noise (which is large when c is large).

Fig. 10 and Fig. 11 show the relationship between ε_2 and MAE/MSE, where $c = 5$. We observe that the MAE and MSE rapidly decreases as ε_2 increases from 0.1 to 1, and that they remain almost unchanged after $\varepsilon_2 = 1$.

In Fig. 11, when ε_2 is 2.5 or more, the MSE of Privelet is larger than that of Laplace. One reason for this is that Privelet algorithm adds noise to each node

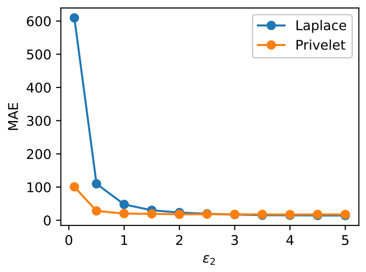


Fig. 10. MAE of $\overline{\mathbf{Q}}'$ versus ε_2 $(c = 5)$.

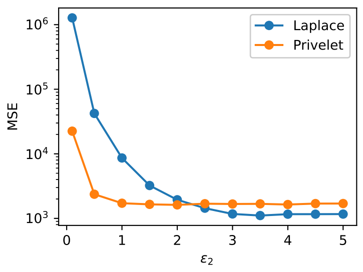


Fig. 11. MSE of $\overline{\mathbf{Q}}'$ versus ε_2 $(c = 5)$.

of a tree structure and the number of targets for noise addition (i.e., the number of nodes) in Privelet is larger than the number of elements in $\mathbf{Q}$.

Other statistical features. Finally, we evaluated the relationship between the number θ of generated co-location events and the MAE/MSE of the population distribution and the transition matrix. Fig. 12 and Fig. 13 show the MAE/MSE of the population distribution. Fig. 14 and Fig. 15 show the MAE/MSE of the transition matrix. Here we set θ to $\theta = 1$, 5, 10, 50, 100, 500, 1000, or 5000.

Fig. 12, Fig. 13, Fig. 14, and Fig. 15 show that when the number θ of generated co-location events is smaller than 1000, the proposed method (Proposal) achieves much smaller MAE and MSE than the uniform synthesizer (Uniform), and almost the same MAE and MSE as the synthetic data generator in [22] (TPM). Note that TPM does not generate co-location events between friends, unlike Proposal. In other words, Proposal can generate co-location events between friends based on p' and $\mathbf{Q}'$ while keeping high utility in terms of other statistical features such as the population distribution and transition matrix.

However, when the number θ of co-location events increases from 1000, the MAE and MSE in Proposal become larger. This is because there are too many co-locations in the synthetic traces and the population distribution and transition matrix are not preserved well even if we complement the remaining locations using the Viterbi algorithm. Therefore, we should determine an appropriate value of θ in advance, either manually or automatically. One way to automatically set θ is to calculate the frequency of co-location events from real trace data while providing DP for the real trace data, and then set θ based on the co-location frequency. Exploring such automatic setting of θ is left for future work.

6 Conclusion

In this paper, we proposed a location synthesizer for synthesizing location traces including co-location events, which are important for synthetic traces to be more useful and realistic. Our proposed method generates synthetic traces using the friendship probability and the co-location count matrix, while providing node DP for friendship data and user-level DP for real trace data. We showed that our location synthesizer can generate synthetic traces that preserve the information

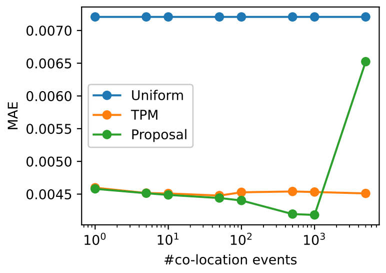


Fig. 12. MAE of the population distribution versus the number of co-location events ($\varepsilon_1 = \varepsilon_2 = 1$, $c = 5$).

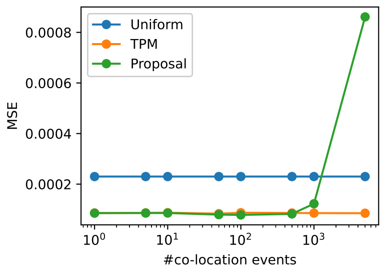


Fig. 13. MSE of the population distribution versus the number of co-location events ($\varepsilon_1 = \varepsilon_2 = 1$, $c = 5$).

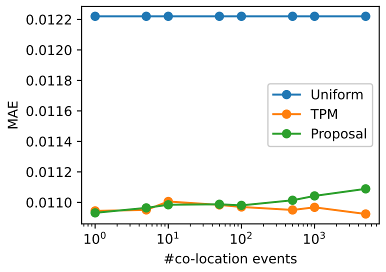


Fig. 14. MAE of the transition matrix versus the number of co-location events ($\varepsilon_1 = \varepsilon_2 = 1$, $c = 5$).

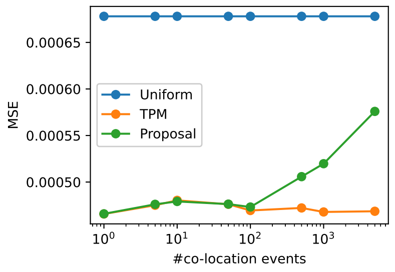


Fig. 15. MSE of the transition matrix versus the number of co-location events ($\varepsilon_1 = \varepsilon_2 = 1$, $c = 5$).

about the friendship probability and co-location count matrix, as well as other statistical features such as the population distribution and transition matrix. We also showed that our location synthesizer provides node DP and user-level DP with reasonable privacy budgets (e.g., smaller than 1).

For future work, we plan to evaluate the utility of various location synthesizers [3–5] when they are used to complement locations other than co-locations using the Viterbi algorithm in our proposed method. Another line of future work is to automatically determine an appropriate value of θ (number of generated co-location events) while providing DP for the real trace data. It would also be interesting to incorporate the friendship level (numerical value rather than 0/1) between users into our algorithm, and to assess the accuracy of privacy attacks such as membership inference as a function of ε.

References

1. Y. Zheng, L. Zhang, X. Xie, et al. Mining interesting locations and travel sequences from gps trajectories. In *WWW'09*, pages 791–800, 2009.
2. M. Lichman and P. Smyth. Modeling human location data with mixtures of kernel densities. In *KDD'14*, pages 35–44, 2014.

3. V. Bindschaedler and R. Shokri. Synthesizing plausible privacy-preserving location traces. In *IEEE S&P'16*, pages 546–563. IEEE, 2016.
4. X. He, G. Cormode, A. Machanavajjhala, et al. Dpt: differentially private trajectory synthesis using hierarchical reference systems. *PVLDB*, 8(11):1154–1165, 2015.
5. T. Murakami, K. Hamada, Y. Kawamoto, et al. Privacy-preserving multiple tensor factorization for synthesizing large-scale location traces. *PoPETs*, 2021(2):5–26, 2021.
6. L. Song, D. Kotz, R. Jain, et al. Evaluating next-cell predictors with extensive wi-fi mobility data. *IEEE T-MC*, 5(12):1633–1649, 2006.
7. T. Iwata and H. Shimizu. Neural collective graphical models for estimating spatio-temporal population flow from aggregated data. In *AAAI'19*, volume 33, pages 3935–3942, 2019.
8. PWS Cup 2019. https://www.iwsec.org/pws/2019/cup19_e.html, 2019.
9. D. Yang, B. Qu, J. Yang, et al. Revisiting user mobility and social relationships in lbsns: a hypergraph embedding approach. In *WWW'19*, pages 2147–2157, 2019.
10. A.M. Olteanu, K. Huguenin, R. Shokri, et al. Quantifying interdependent privacy risks with location data. *IEEE T-MC*, 16(3):829–842, 2016.
11. A.M. Olteanu, K. Huguenin, R. Shokri, et al. Quantifying the effect of co-location information on location privacy. In *PETS'14*, pages 184–203. Springer, 2014.
12. C. Dwork and A. Roth. *The Algorithmic Foundations of Differential Privacy*. Now Publishers, 2014.
13. A.M. Olteanu, M. Humbert, K. Huguenin, et al. The (co-)location sharing game. *PoPETs*, 2019(2):5–25, 2019.
14. T. Murakami and H. Watanabe. Localization attacks using matrix and tensor factorization. *IEEE T-IFS*, 11(8):1647–1660, 2016.
15. R. Shokri, M. Stronati, C. Song, et al. Membership inference attacks against machine learning models. In *S&P'17*, pages 3–18, 2017.
16. A. Pyrgelis, C. Troncoso, and E. De Cristofaro. Knock knock, who's there? membership inference on aggregate location data. In *NDSS*, 2018.
17. C. Dwork, F. McSherry, K. Nissim, et al. Calibrating noise to sensitivity in private data analysis. In *TCC'06*, pages 265–284. Springer, 2006.
18. R. Sofya and S. Adam. *Differentially Private Analysis of Graphs*, pages 543–547. Springer, 2016.
19. L. Ninghui, L. Min, and S. Dong. *Differential Privacy: From Theory to Practice*. Morgan & Claypool Publishers, 2016.
20. C. Dwork, M. Naor, T. Pitassi, et al. Differential privacy under continual observation. In *STOC'10*, pages 715–724, 2010.
21. B.C.M Fang, K. Wang, R. Chen, et al. Privacy-preserving data publishing: A survey of recent developments. *ACM Computing Surveys (Csur)*, 42(4):1–53, 2010.
22. V. Bindschaedler, R. Shokri, and CA. Gunter. Plausible deniability for privacy-preserving data synthesis. *VLDB Endowment*, 10(5), 2017.
23. X. Xiao, G. Wang, and J. Gehrke. Differential privacy via wavelet transforms. *IEEE T-KDE*, 23(8):1200–1214, 2010.
24. FOURSQUARE DEVELOPERS. Venue categories — build with foursquare. https://developer.foursquare.com/docs/build-with-foursquare/categories/, 2020. Accessed:2020/10/25.
25. Z. Liu, Y.X. Wang, and A. Smola. Fast differentially private matrix factorization. In *RecSys'15*, pages 171–178, 2015.
26. A.L. Barabási. *Network Science*. Cambridge University Press, 2016.