Differentially Private Location Recommendations in Geosocial Networks

Jia-Dong Zhang
Dept. of Computer Science
City University of Hong Kong
Email: jzhang26-c@my.cityu.edu.hk

Gabriel Ghinita
Dept. of Computer Science
University of Massachusetts Boston
Email: gabriel.ghinita@umb.edu

Chi-Yin Chow
Dept. of Computer Science
City University of Hong Kong
Email: chiychow@cityu.edu.hk

***Abstract*—Location-tagged social media have an increasingly important role in shaping behavior of individuals. With the help of location recommendations, users are able to learn about events, products or places of interest that are relevant to their preferences. User locations and movement patterns are available from geosocial networks such as Foursquare, mass transit logs or traffic monitoring systems. However, disclosing movement data raises serious privacy concerns, as the history of visited locations can reveal sensitive details about an individual's health status, alternative lifestyle, etc. In this paper, we investigate mechanisms to sanitize location data used in recommendations with the help of differential privacy. We also identify the main factors that must be taken into account to improve accuracy. Extensive experimental results on real-world datasets show that a careful choice of differential privacy technique leads to satisfactory location recommendation results.**

I. Introduction

Latest-generation mobile devices allow users to participate in location-based social networks (LBSNs) and other applications with a significant component of geo-tagged media. One important area that is gaining increasing attention is that of location recommendations. There are many factors that determine a user's interest profile, such as age, gender, education level [1]. Apart from such factors, users' movement patterns become increasingly important in recommendations, as geographical relationship to events or to other users is an important decision factor.

Location recommendations can be performed at various levels of granularity. For instance, a location recommendation system may take into account exact coordinates of users as provided by a GPS system, or just coarser-grained areas, such as a city block, or a zipcode area. Another important aspect in achieving good-quality recommendations is the length of movement history. Some earlier work [2], [3], [4], [5] considers that the current user position is sufficient to determine a good recommendation. However, this may be an over-simplifying approach, as the user's previous trajectory and timeframe are also important.

Providing good-quality recommendations requires looking at large amounts of trajectory data from a diverse set of users. Such trajectory repositories are increasingly available, thanks to trajectory traces collected by various location-based applications, e.g., the check-in history of a LBSN, the logs of a mass-transit operator that uses magnetic access tokens, or a traffic monitoring system. However, making such datasets directly available to recommender systems may lead to serious privacy breaches. Previous research [6], [7], [8], [9] showed that disclosing trajectory data can reveal sensitive details about an individual's health status, political affiliations or alternative lifestyle.

Several existing techniques address location privacy threats. The cryptographic approach in [10] is appropriate when one needs to privately retrieve a specific item from a dataset, e.g., nearest-neighbor queries. However, location recommendations require a broader set of operations to determine movement patterns. Furthermore, cryptographic methods are very expensive computationally. Other techniques use ad-hoc definition of protection: in the case of K-anonymity [6], [7], [8], each published location must be indistinguishable among other $K-1$ locations. This type of transformation is fast and simple, but it has been shown recently [11], [12] that it does not provide protection in the presence of background knowledge.

Differential privacy [11], [12] provides formal protection guarantees against adversaries with background knowledge. Differential privacy allows only statistical queries on the data, and adds noise to each answer to achieve protection. An adversary cannot learn with significant probability whether a certain individual is included in the dataset or not. In the context of location data, this translates into withholding information about whether a particular trajectory can be found in the dataset. However, it is still possible to learn aggregate information about trajectories, which may be sufficient for tasks such as travel pattern mining.

We study location recommendations using datasets sanitized according to differential privacy. Our focus is on recommendations that take into account several consecutive observations across users' movement history, as this approach provides superior accuracy. We study two approaches: the first represents movement history as a point in the multi-dimensional Cartesian product space of individual snapshots, whereas the second indexes and sanitizes directly sequences of locations in a hierarchical structure.

We evaluate experimentally the impact of history length, privacy budget and data density on recommendation accuracy. Results show that it is possible to provide good-quality recommendations by carefully sanitizing datasets. To the best of our knowledge, this is the first study showing the feasibility of applying differential privacy in the context of location recommendations.

In summary, our contributions are: (i) We study for the first time the problem of location recommendations using trajectory datasets protected with differential privacy; (ii) We consider two alternative techniques for differentially-private location recommendations, which provide interesting trade-offs with respect to history length, data density and data skew; (iii) We perform an extensive experimental evaluation that shows the feasibility of applying differential privacy for location recommendations, and highlights the relative trade-offs between the two considered alternative approaches.

Section II introduces background information. Section III presents the studied mechanisms for private location recommendations. Section IV contains experimental evaluation results, followed by a review of related work in Section V. Section VI concludes with directions for future research.

II. Background

Our work brings together concepts and techniques from two distinct domains, namely location recommendations and differential privacy. We review background information about location recommendations in Section II-A, followed by a brief primer on differential privacy in Section II-B.

A. Location Recommendations

Location-based social networks (LBSNs) such as Foursquare, Gowalla and Facebook Places, attracted millions of users. LBSNs enable users to share their experiences in visiting specific locations of interest, e.g., restaurants, museums, etc. Upon visiting a location, a user performs a virtual *check-in* operation to let others know about his or her current location. Additional features may be provided based on check-ins, such as determining a user's social status and/or rewarding customers based on the count and frequency of their visits to a specific location. To further enhance user experience and social interaction [13], [14], such systems provide the functionality of *location recommendations*, whereby users receive suggestions about which places to visit, based on the (positive) experiences of other users when visiting those locations. This functionality is performed by recording and consulting user check-in histories. Based on the fact that human movement exhibits sequential patterns [15], the recent studies by [2], [3], [4], [5] proposed techniques for location recommendations based on the first-order Markov chain derived from trajectories. First-order Markov chains are used to represent "memoryless" processes, and model the sequential patterns from check-in location sequences as transition probabilities from one location to another.

Definition 1 (Location sequence): A location sequence of a user is a set of check-in locations of that user increasingly ordered by their check-in time. A location sequence is denoted by $s = \langle l_1, l_2, \ldots, l_n \rangle$. $s' = \langle l_i, l_{i+1} \ldots, l_{i+j} \rangle$ $(1 \leq i \leq i+j \leq n)$ is called a subsequence of s.

Definition 2 (First-order Markov chain): Given a user's location sequence $s = \langle l_1, l_2, \ldots, l_n \rangle$, the first-order Markov chain assumes the probability of the user visiting next location l_{n+1} depends only on last visited location l_n:

$$p(l_{n+1}|s) = p(l_{n+1}|l_n) = \frac{N(\langle l_n, l_{n+1} \rangle)}{N(\langle l_n \rangle)}, \qquad (1)$$

where $N(\langle \cdot \rangle)$ is the number of occurrences of subsequence $\langle \cdot \rangle$ in the collection of location sequences of all users.

To improve the quality of location recommendations, we generalize the first-order Markov chain to the higher order Markov chain in the context of location recommendations.

Definition 3 (m^{th}-order Markov chain): Given a user's visited location sequence $s = \langle l_1, l_2, \ldots, l_n \rangle$, the m^{th}-order Markov chain estimates the probability of the user visiting the next location l_{n+1} based on the latest m visited locations $l_n, \ldots, l_{n-m+1}$. Formally,

$$p(l_{n+1}|s) = p(l_{n+1}|l_n, \ldots, l_{n-m+1}) = \\ = \frac{N(\langle l_{n-m+1}, \ldots, l_{n+1} \rangle)}{N(\langle l_{n-m+1}, \ldots, l_n \rangle)}. \qquad (2)$$

To provide location recommendations, a recommender system will return the top-k locations l_{n+1} with the highest probability, i.e., $p(l_{n+1}|s)$ according to Eq. (2). In Eq. (2), the essential task is to compute the number of times that a certain subsequence occurs in the collection of location sequences of all users. As we will discuss next in Section II-B, providing an interface for privately answering count queries is the objective of differential privacy, which makes the latter an ideal candidate for privacy-preserving location recommendations.

B. Differential Privacy

Differential privacy was introduced [16] to address the limitation of syntactic models which are vulnerable against background knowledge attacks. It is a semantic model, which argues that one should minimize the risk of disclosure that arises from an individual's participation in a dataset. Differential privacy is formulated in the context of statistical databases, and allows only aggregate queries to be asked. In the context of spatial databases [12], this boils down to answering range count queries such as *"find the number of locations enclosed in a region"*. Although in theory there is no restriction on the range query shape, in practice rectangular queries are typical.

The interaction between a user and the database is expressed as a *transcript*. Formally,

Definition 4 (Transcript): Let $\mathcal{Q} = \{Q_1, \ldots, Q_q\}$ be a set of COUNT queries, and denote by $Q_i^T{}_{(1 \leq i \leq q)}$ the result of answering query Q_i on dataset T. A *transcript*

$$\mathcal{TR}_{\mathcal{Q}}^T = \{(Q_1, Q_1^T), \ldots, (Q_q, Q_q^T)\}$$

is the response to the query set $\mathcal{Q}$ on dataset T.

A transcript satisfies ϵ-differential privacy if the ϵ-indistinguishability condition [11] is fulfilled, where ϵ is a parameter that quantifies the amount of desired privacy:

Definition 5 (ϵ-indistinguishability): Consider a statistical database that produces the transcript $\mathcal{U}$ on the set of queries $\mathcal{Q} = \{Q_1, \ldots, Q_q\}$, and let $\epsilon > 0$ be an arbitrarily-small real constant. Transcript $\mathcal{U}$ satisfies ϵ-*indistinguishability* if for every pair of datasets T_1, T_2 such that $|T_1| = |T_2|$ and T_1, T_2 differ in only one record

$$\left| \ln \frac{Pr[\mathcal{TR}_{\mathcal{Q}}^{T_1} = \mathcal{U}]}{Pr[\mathcal{TR}_{\mathcal{Q}}^{T_2} = \mathcal{U}]} \right| \leq \epsilon$$

In other words, an attacker is not able to learn, with significant probability, whether the transcript was obtained by answering the query set $\mathcal{Q}$ on T_1, or on T_2. To achieve indistinguishability, differential privacy requires random noise to be added to each query result. The magnitude of the noise depends on the privacy parameter ϵ, and the *sensitivity* of the query set $\mathcal{Q}$, defined as follows:

Definition 6 (L_1-sensitivity [11]): Given any two datasets T_1, T_2 such that $|T_1| = |T_2|$ and T_1, T_2 differ in only one record, the L_1-*sensitivity* of query set $\mathcal{Q} = \{Q_1, \ldots, Q_q\}$ is measured as $S_{L_1}(\mathcal{Q}) = \max_{\forall T_1, T_2} \sum_{i=1}^{q} |Q_i^{T_1} - Q_i^{T_2}|$.

The following theorem gives a sufficient condition for a data release to satisfy ϵ-differential privacy [11]:

Theorem 1: Let $\mathcal{Q}$ be a set of queries answered by a statistical database, and denote by $S_{L_1}(\mathcal{Q})$ the L_1-sensitivity of $\mathcal{Q}$. Then, differential privacy with parameter ϵ can be achieved by adding to each query result random noise X, i.e., $Q_i^T \leftarrow Q_i^T + X$, where X is a random, i.i.d. variable drawn from a Laplace distribution with mean 0 and magnitude $\lambda \geq S_{L_1}(\mathcal{Q})/\epsilon$.

An essential operation in enforcing differential privacy is determining the sensitivity $S_{L_1}(\mathcal{Q})$. It is shown in [11] that $S_{L_1}(\mathcal{Q})$ is independent of the data set T, and can be determined based on the query set $\mathcal{Q}$ alone. However, for arbitrary query sets, it is shown in [17] that computing sensitivity is NP-hard. Nevertheless, sensitivity (or its approximation) for sets of $COUNT$ queries can be efficiently computed:

1) If all queries in $\mathcal{Q}$ have disjoint ranges, $S_{L_1}(\mathcal{Q}) = 2$.
2) If queries in $\mathcal{Q}$ have overlapping ranges, then a 2-approximation of $S_{L_1}(\mathcal{Q})$ is given by the maximum number of queries that overlap the same point in the data space.

III. Private Location Recommendations

Location data that may be generated from a variety of sources, such as LBSN user check-ins, logs of traffic monitoring systems, geo-tagged media, etc. Figure 1 illustrates the envisioned architecture, where trajectory data is gathered at a trusted repository and sanitized according to differential privacy. The adversary may access a set of recommendations from the system, with the purpose of exposing the check-in patterns of an individual. The presence of the sanitization layer prevents such attacks.

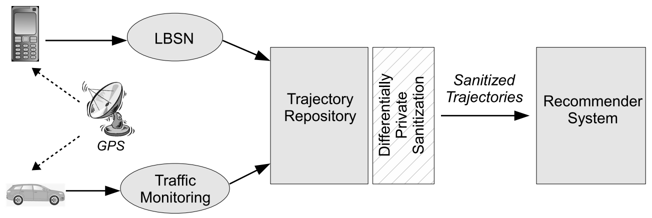


Figure 1. Architecture for Private Location Recommendation Systems

We consider two approaches for sanitization. First (Section III-A) we represent each sequence as a point in a multi-dimensional space, and apply private spatial decomposition [12] techniques for sanitization. Next (Section III-B) we investigate the use of *n-gram* sanitization techniques which can directly deal with sequential data [18]. The rationale for this duality is that the former technique is likely to perform well for short check-in histories, whereas the latter is more suitable for longer histories.

A. Cartesian Product Check-In History Representation

We represent each location sequence as a multi-dimensional point in the space given by the Cartesian product of the set of all check-in locations. Check-in locations are represented as discrete objects with unique labels, such as *"Starbucks on Broadway and 31st"*. Assume that there are in total eight possible check-in locations, $L = \{l_1, l_2, \ldots, l_8\}$, and three check-in sequences of users: $< l_1, l_4 >$, $< l_2, l_5 >$, $< l_6, l_4 >$ and $< l_7, l_3 >$. Figure 2 shows how the sequences are represented in space $L \times L$. In this case, for ease of illustration, we considered that each sequence has at most two locations. In general, the representation space is L^m, where m is the maximum sequence length.

The Cartesian product representation allows us to apply sanitization techniques specifically designed for *private spatial decompositions (PSD)*, introduced in [12]. PSD builds noisy spatial tree index structures on top of the dataset, and publishes the contents of these indexes. Either space-partitioning or data-partitioning indexes can be used. The indexes are built entirely based on the noisy *COUNT* queries received from the differentially private interface. The dataset is recursively split, and the *minimum bounding rectangles (MBRs)* of nodes at all levels of the tree are published together with their noisy counts.

One essential aspect is *privacy budget allocation*. Tree indexes have multiple levels, and the spatial extent of a node covers the span of extents of its siblings. Therefore, there exists overlap among published counts. According to the sensitivity properties discussed in Section II-B, overlap requires more noise to be added to each query answer. To minimize overlap, PSD decompositions allow only index structures that *do not* cause overlap within the same tree level, but only across different levels. It results that each point in the dataspace is covered by exactly h nodes, or equivalently, h queries, where h is the tree index height. With the overlap amount bounded by the tree height, the sensitivity of the set of queries employed in the PSD construction equals $2h$, according to Section II-B.

If ϵ_i is the budget at level i, then

$$\epsilon = \sum_{i=0}^{h} \epsilon_i,$$

However, not all levels must be granted an equal portion of privacy budget. Given a global privacy budget constraint ϵ, one challenge is how to choose $\epsilon_i, i = 0 \ldots h$ to minimize the accuracy error when answering queries. The allocation choice is dependent on the type of structure used. Furthermore, the accuracy is influenced by the tree height, as more levels will compete for the same budget amount. Therefore, it is important to decide the *split stopping condition*, i.e., at which height to stop extending the tree.

According to [12], given a random rectangular query Q over the dataspace domain, a closed-form, average-case scenario can be formulated for the number of index nodes $n(Q)$ that intersect query Q. Also, denote by $n_i(Q)$ the corresponding number of nodes at level i that maximally contribute their counts to the query result (i.e., their extents are completely enclosed by the query range Q). Therefore,

$$n(Q) = \sum_{i=0}^{h} n_i$$

For the quadtree index type [19], it is shown in [12] that

$$n_i \leq 8 \cdot 2^{h-i}$$

and by summation over the entire height of the tree

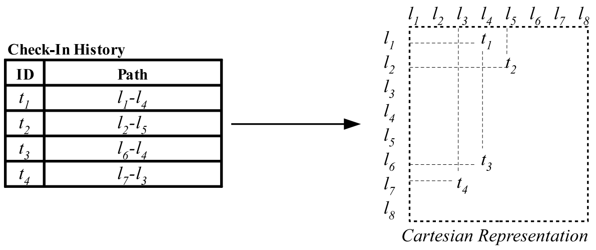
$$n(Q) \leq 8(2^{h+1} - 1) = O(4^{h/2})$$

Similarly, for kd-trees [19], another popular index structure, $n(Q) = O(2^{h/2})$. Solving a constraint optimization problem that minimizes the expected query answering error subject to the available maximum budget ϵ, the choice of *geometric* budget allocation over tree levels proves the best alternative. Namely the budget allocated to tree level i should be set to:

$$\epsilon_i = 2^{(h-i)/3} \epsilon \frac{2^{1/3} - 1}{2^{(h+1)/3} - 1}$$

Data-Independent Decompositions: Quad-trees.

For data-independent decompositions, the structure of the index does not depend on the dataset. For instance, in the case of quad-trees [19], each split is performed recursively on the middle coordinate of the current partition. Partitions are always square, and all the nodes in the tree at the same depth correspond to a dataspace partition of the same dimensions (and consequently, the same area or volume). Therefore, no information about the data is disclosed by the structure of the published tree structure, since it is


Check-In History

ID	Path
t_1	l_1-l_4
t_2	l_2-l_5
t_3	l_6-l_4
t_4	l_7-l_3

Figure 2. Cartesian product check-in history representation

completely independent of the data. Quad-trees result in partitions that are non-overlapping. If one perceives each index partition, say a leaf node, as one of the elements that are part of the query set $\mathcal{Q}$, then all elements at the same level at the tree are non-overlapping, leading to constant sensitivity with low value 2. This is clearly a desirable property of such an index.

In a d-dimensional space, quad-trees are constructed by recursively dividing the space into two sub-spaces using $(d-1)$-dimensional hyperplanes. In two dimensions, each step corresponds to selecting a line and partitioning the space based on this line until certain requirements are met. In three dimensions, instead of lines, planes are used to split the space. The heuristic method for selecting the splitting hyperplanes depends on the application area, as does the stopping condition.

Each split occurs on an axis-orthogonal hyperplane. The axis is chosen based on the depth of the tree node which is being split. Specifically, if the depth of node n is denoted $Depth(n)$, the splitting hyperplane is orthogonal to axis $Depth(n)\%d + 1$ and splits the space into two equal halves.

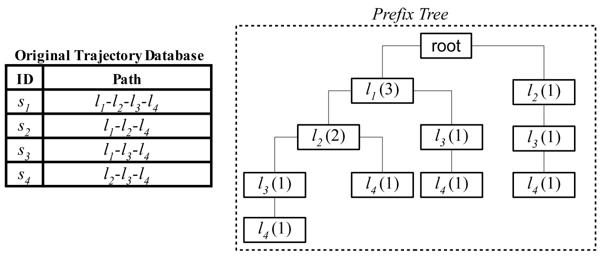
Let $\mathcal{Q}^h$ be a query for the entire quad-tree index on some dataset T. Since partition extents are built based on attribute domains only, $\mathcal{Q}^h$ is equivalent to a set of 2^h count queries, one for each leaf node's extent. Notice that regardless of h and the distribution of T, the query regions will be disjoint. Therefore, the sensitivity of $\mathcal{Q}^h$ is 2 at each level of the tree.

Data-Dependent Decompositions: kd-trees.

For data-dependent indexes, the procedure to obtain a differentially-private version of a spatial dataset is more complex, because not only the individual counters in each index node must be protected, but also the structure of the tree. Additional measures such as protecting the result of each index split decision are necessary.

In the case of kd-trees [19], the split algorithm recursively divides the space on the *median* value across one of the space dimensions. Instead of computing the actual median, a version of "noisy" median is determined using the exponential mechanism [20] of differential privacy. Specifically, each element in the dataset is assigned a merit score according to the distance it has from the actual median, and a randomized algorithm that picks an element according to these merit scores is used to select a noisy pseudo-median.

Another important change in comparison to the case



Original Trajectory Database

ID	Path
s_1	l_1-l_2-l_3-l_4
s_2	l_1-l_2-l_4
s_3	l_1-l_3-l_4
s_4	l_2-l_3-l_4

Figure 3. Trajectory Database and Resulting Prefix Tree

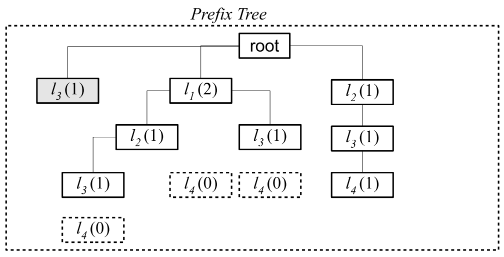


Figure 4. Differentially Private Prefix Tree

of data-independent indexes is that the budget must be split between the Laplace mechanism and the exponential mechanism. An inherent trade-off develops: on the one hand, the more budget is allocated to the exponential mechanism, the index structure will be more balanced, and accuracy improved. But on the other hand, there is less remaining budget to be used for releasing index node counts.

B. Location Sequence Indexing

The Cartesian product may serve well when location recommendation is done on a short check-in history, but when m increases, the dimensionality becomes too large for spatial indexing techniques to be effective. In this section, we turn our attention to techniques for trajectory sanitization, such as [18], [21]. These techniques are easy to adopt into our framework for differentially private location recommendations, due to the similarity of the data model used.

Differentially private publication of trajectories was first tackled in [18]. Location data are discretized, which matches well the setting of a public transportation network, where each snapshot in a user trajectory can be identified with a bus or train stop. This model also works well for histories of user check-ins. Differential privacy is applied directly on trajectories, i.e., the dataset T from the Definition 4 is a set of location sequences $T = \{s_1, s_2, \ldots\}$. Each sequence is an ordered set from a location universe $L = \{l_1, l_2, \ldots\}$. It is assumed that there is at most one trajectory per individual, and the goal is to release a sanitized database of trajectories, such that an adversary is not able to tell if a particular individual's trajectory is included in the release or not.

A *prefix tree* is proposed in [18] that has a virtual root node corresponding to a special "start-of-trajectory" marker (which is not an actual location in L). Starting from that node, a path in the tree is created for each individual trajectory. Trajectories that share an exact prefix will have identical beginning sub-paths in the tree.

Figure 3 shows a trajectory prefix tree, corresponding to the set of four trajectories in the table. The counters shown in parentheses represent the number of trajectories that share that particular prefix (i.e., the prefix starting from the virtual root and ending in the respective node). Note that, the prefix tree is constructed using a deterministic procedure from the dataset, so the structure must be sanitized before release.

There are two types of disclosure that need to be prevented. First, the actual counters in the prefix tree need to be perturbed. This can be done using the Laplace mechanism. Second, the actual structure of the prefix tree gives information about what particular trajectories exist and do not exist in the dataset. For instance, inspecting the tree in Figure 3, one can assert with certainty that there is no trajectory that starts at location l_3. Similarly, there is no trajectory that has as prefix l_1, l_4. To address this latter concern, it is necessary to alter the structure of the prefix tree and create *fake* nodes.

Figure 4 shows the tree after sanitization, where both counters and structure are perturbed. After sanitization, some nodes may have their count reduced to zero, so they are removed from publication, even though there are some trajectories in the original dataset represented by that path in the tree. Conversely, some nodes that do not correspond to any existing trajectory prefix are created. These fake nodes are necessary to comply with differential privacy.

In this example, the former leaf nodes l_4 (i.e., all on the left half of the initial subtree) have had their counters added with random negative noise (-1), so as a result their noisy counter is zero, and they are removed from the tree (their perimeters are shown with discontinued line in the diagram for illustration purposes, but they do not appear at all in the published version). In contrast, for every possible child of a node - not only those in the initial prefix tree, but for all *possible* trajectories, a virtual node is created. This node has initial counter zero, but following addition of noise, they could be promoted to actual nodes. This is the case of node l_3, shown in shaded texture, which has been added as a child of the root node. Note however, that fake nodes can be added in any part of the tree, not just under the root node.

The method from [18] has a limitation: in order for an edge in the prefix tree to have a count larger than 1, it is necessary for multiple trajectories to share the *exact* same prefix up to that edge. In practice, there may be sub-paths (i.e., location sequences) that are common, even if there is no trajectory that starts with that sequence. For instance, there are many people that first go to a specific restaurant and then to the cinema within the same mall complex, so this sequence appears a lot in the check-in history. However, their trajectories may have started at their workplaces earlier, which are heterogeneous. Therefore, all paths will be considered completely distinct, and their nodes appear with a count of 1. As a result, the noise added by differential privacy will be large compared to the actual counter value of 1, leading to large errors.

To address this issue, the work in [21] indexes partial trajectories, regardless of their start point. This allows common trajectory segments that occur frequently in the dataset to have their counts added up together before noise added by differential privacy. The method uses a concept that is common in text mining, namely that of *n-grams*, which are sequences of up to n_{max} locations that appear frequently (n_{max} is a system parameter). The idea behind [21] is to extend the concept of prefix tree to n-gram tree.

There are two important parameters in the n-gram method: the maximum length of considered n-grams, and the maximum allowable length of an entire trajectory. The former one is basically parameter n_{max}. The height of the released tree is important, as the budget ϵ gets redistributed among distinct tree levels. A larger tree height leads to less accurate information at each level.

In our context, the n-gram tree consists of sequences of check-ins, and the height depends on the length of the history, i.e., the length of the m^{th}-order Markov chain from Section II-A. Consequently, in our adaptation of the method, we set $n_{max} = m + 1$, corresponding to a history of m previously visited locations plus the next visited location (i.e., the recommended one). The maximum allowable length of a trajectory determines the overall sensitivity. In the worst case, a single trajectory could affect a number of counters in the n-gram tree equal to that trajectory's path length. The sensitivity is the maximum trajectory length, denoted by l_{max}. We follow the setting in [21] which specifies a guideline of $l_{max} = 20$ for trajectory datasets.

IV. Experimental Evaluation

We present the experimental setup, dataset description and performance metrics in Section IV-A. In Section IV-B we provide experimental results and a discussion of our findings.

A. Experimental Settings

Dataset. We use a publicly available large-scale real check-in history dataset corresponding to the users of the Gowalla location-based social network [15]. The characteristics of the data set are summarized in Table I.

Table I
Gowalla Dataset Statistics

Parameter	Value
Number of users	196,591
Number of locations	1,280,969
Number of check-ins	6,442,890
Average visited POIs per user	37.18
Average check-ins per location	3.11

We split the dataset into the training set and the testing set, each with 50% of the data. The training set is used to learn the recommendation models for each of the evaluated techniques described below. We perform partitioning based on the check-in times in the data, rather than using a random allocation of data points to partitions. The rationale behind this choice is that in practice, we can only utilize the past check-in data to predict future check-in events.

Evaluated Techniques. In our evaluation, we consider four different techniques for location recommendation. Their description and the labels used are as follows:

- m-**OMC** (m^{th}**-order Markov chain**): This benchmark does not provide any privacy, and performs recommendations based on revealed data. It implements the location recommendation technique based on inspecting past check-ins with a maximum look-back count of m. In other words, the previous sequence of m check-ins is used to determine the next visited location. The following methods also use a history of m check-ins, but they work on the sanitized datasets.
- **NGrams**: This method is the differentially private n-gram publication method from Section III-B. Due to its design, the method is expected to perform better for larger values of history lookup (i.e., larger m).
- **QuadTree**: This method implements data-independent partitioning using quad-trees (Section III-A). It is expected to perform well for relatively low values of m, and for uniform spatial distribution of check-ins.
- **KDTree**: This method implements the data-dependent spatial partitioning method using kd-trees (Section III-A). It is expected to perform well for relatively low values of m, and to handle well skewed spatial distribution of check-ins.

Performance Metrics. Given a target user, recommendation techniques compute a score for each candidate item (i.e., next location), and return as results the locations with the **top-**k highest scores. In our experiments, we choose $k = 10$.

To evaluate the quality of location recommendations, it is important to find out how many locations that are being recommended are actually visited by the target user in the testing data set. Also, it is important to know how many of the locations actually visited were recommended by the evaluated technique. The former aspect is captured by *precision* and the latter by *recall* [22], [23]. We define a *discovered* location as a location that is both recommended and actually visited by the target user. Formally,

- *Precision* defines the ratio of the number of discovered locations to the total number k of recommended locations, i.e.,

$$\textit{precision} = \frac{\textit{number of discovered locations}}{k}.$$

- *Recall* defines the ratio of the number of discovered locations to the total number of locations actually visited by the target user in the testing set, i.e.,

$$\textit{recall} = \frac{\textit{number of discovered locations}}{\textit{number of visited locations}}.$$

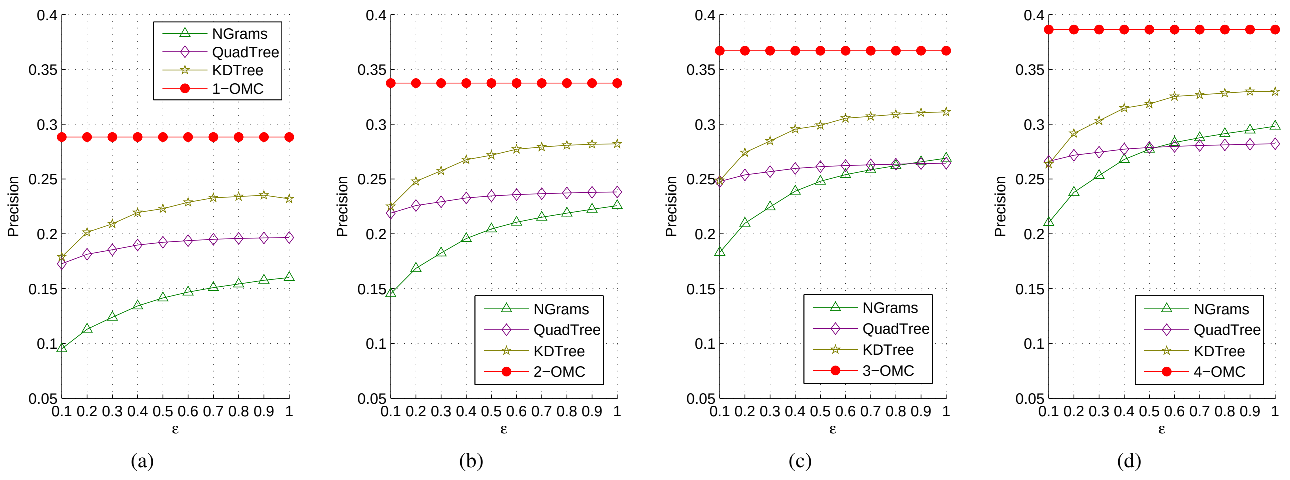


Figure 5. Precision, $5km \times 5km$ grid, geometric tree budget

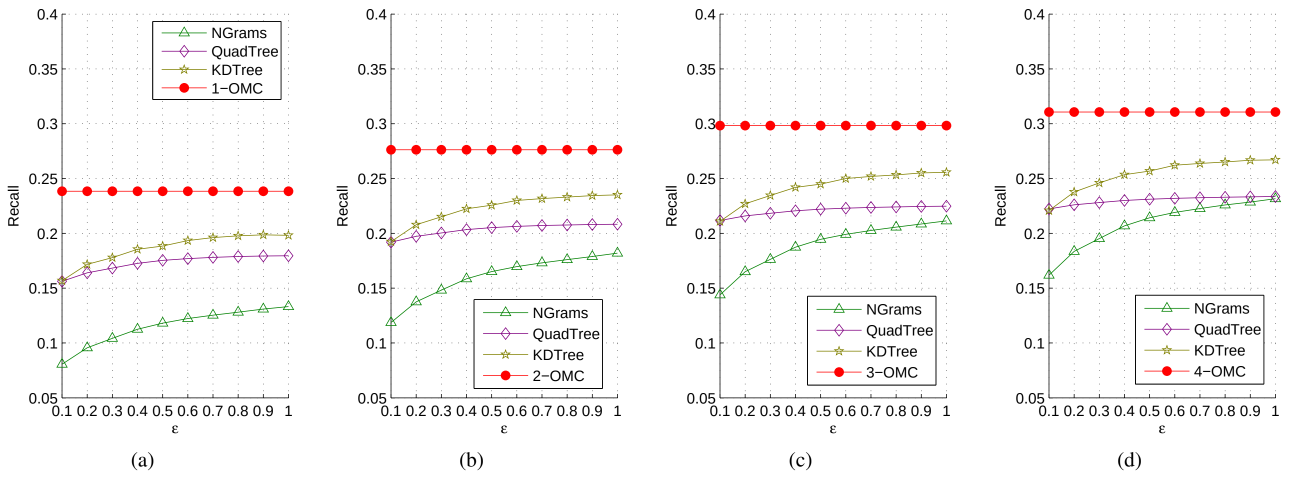


Figure 6. Recall, $5km \times 5km$ grid, geometric tree budget

Parameter Settings.

- *Grid Size*: The granularity of locations considered in the recommendation process may vary based on the application and the type of query. We superimpose a regular grid on top of the dataset of locations, and we consider two different grid sizes: 5×5 km and 10×10 km. For fairness, all considered methods are used in conjunction with the same grid size.
- ε: The privacy budget ϵ is an essential parameter for differential privacy. We consider a broad range of values which are typical in the literature on differential privacy, with values from 0.1 to 1.0 in increments of 0.1. A lower ϵ value signifies a tighter privacy requirement.
- l_{max} and n_{max}: These parameters are specific to the $NGrams$ technique. l_{max} is used to decrease the sensitivity of a trajectory dataset, and it specifies the maximum number of check-ins in any single individual history. n_{max} is the maximum length of n-grams extracted from the dataset. In our case, we set $l_{max} = 20$ which is the setting recommended by the proponents of the technique in [18]. The value of n_{max} is set to $m+1$, where m is the length of the Markov chain. Note that, since all the queries on the trajectory dataset will have at most length $m+1$ (the history of m check-ins plus the next visited location), it does not make sense to extract n-grams of longer length than $m+1$.

B. Experiment Results

Figure 5(a)-(d) shows the precision measurements for the $5km \times 5km$ grid setting, in increasing order of Markov chain order m. The geometric budget allocation is used for the PSD methods. As a first observation, note how increasing the check-in history length helps improving recommendation precision (all y-axes are set to the same range in order to better illustrate this effect). This motivates the need to use trajectory data in location recommendation, as opposed to

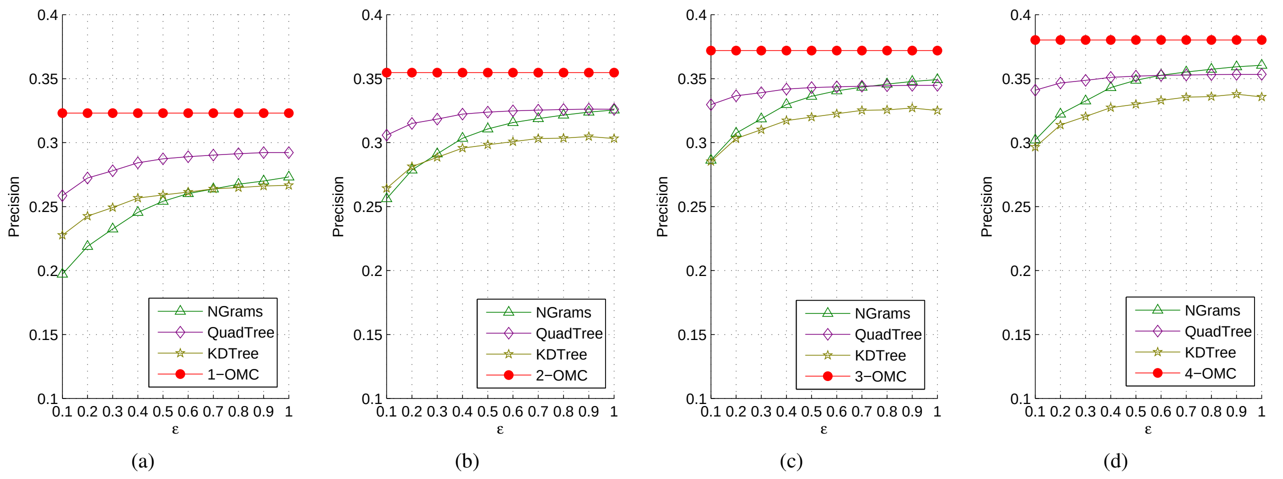


Figure 7. Precision, $10km \times 10km$ grid, geometric tree budget

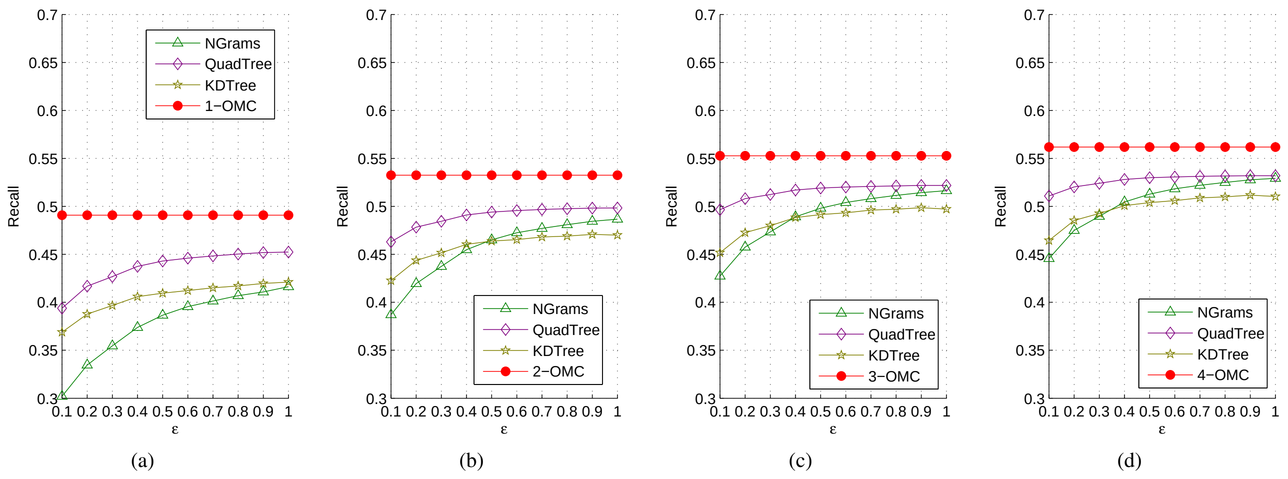


Figure 8. Recall, $10km \times 10km$ grid, geometric tree budget

a last-known location, and it also reinforces the need for effective solutions to the challenging differentially private sanitization of trajectories (which is a considerably more difficult problem that sanitization of individual snapshots).

As expected, a lower privacy budget results to less accuracy. The $KDTree$ technique performs the best, due to a relatively low density of the locations in the grid. Furthermore, a finer granularity also creates more skewness, which disadvantages the data-independent $QuadTree$ method. Note an important trend: as the length of the check-in history increases, the $NGrams$ method catches up with the partitioning methods in terms of precision.

For $m = 4$ and the higher end of the privacy budget range, the precision of $NGrams$ is somewhere in between the precision of the partitioning methods, with $KDTree$ being the best. In terms of absolute precision value, the performance of the best private method is not far below that of the non-private method: for the larger budget range, the precision of $KDTree$ is 5% lower than that of $4-OMC$.

Figure 6(a)-(d) shows the recall measurements for the same setting. The same trends can be observed for recall. The $NGrams$ method does not overtake the $QuadTree$ method under any setting for recall, but it does equal its performance for higher privacy budget when $m = 4$. We repeated the experiment with uniform budget allocation, and obtained very similar precision and recall (we omit the graphs for brevity).

In the next experiment, we use a coarser grid granularity, $10km \times 10km$. The effect of this change is to increase data density, and at the same time reduce the amount of skewness, as all locations that fall in the same grid cell are considered to be uniformly distributed within that cell's span. As shown in the previous measurements, the type of budget used for the tree methods does not significantly influence recommendation quality, so for the sake of brevity we only present the geometric budget allocation results (similar results were obtained for uniform allocation). Figures 7 and 8 (a)-(d) show the obtained precision and recall for various m settings. The same earlier trend is observed, whereby the increase in check-in history length improves absolute accuracy, and also favors the $NGrams$ method, whose performance improves in relative terms compared with $KDTree$ and $QuadTree$ as m increases. Note, however, that the relative performance of $KDTree$ and $QuadTree$ has inverted for the denser dataspace. $QuadTree$ outperforms now, due to the fact that its data-independent nature is favored by the increased density and decreased skewness of the dataset. The effect of the budget quota spent by $KDTree$ to create a balanced structure no longer offsets the lower accuracy when querying tree nodes, and for this grid granularity the resulting precision and recall are worse than competitors.

Our results show that differential privacy can lead to reasonably accurate solutions for location recommendations. As the history length grows, the precision and recall improve. The $NGrams$ method tends to perform better than $QuadTree$ and $KDTree$ when history size grows. Among space partitioning techniques, data density and skew are decisive factors for accuracy. When data density is low and skew is high, the accuracy is worse. Data-dependent techniques like $KDTree$ outperform data-independent techniques in such cases. Conversely, for denser datasets with less skew the $QuadTree$ is the method of choice.

V. Related work

Location Recommendations. With the rapid growth of location-based social networks like Foursquare, Gowalla, and Facebook Places, location recommendations became an essential functionality. There are three main categories of location recommendation techniques. (1) *GPS trajectory based*: Some studies provide location recommendations using GPS trajectory data [24], [25]. However, such data is relatively difficult to obtain. The other two categories focus on using the check-in data extracted from LBSNs. (2) *Topic-based*: the works of [26], [27] explicitly model profiles of locations and users as a mixture of topics, and then derive the likelihood of a user visiting a location based on their profiles. (3) *Collaborative filtering*: most current studies [28], [22], [29], [30], [23], [13] employ conventional filtering techniques which aim at looking for patterns of agreement among users' ratings for locations derived from check-in data. The intuition behind collaborative filtering is that if a user has agreed with her neighbors in the past, she will continue to do so for future locations. In particular, the works of [2], [3], [4], [5] extract sequential patterns from location sequences as a location-location transition probability matrix and generate location recommendations using the first-order Markov chain on the transition probability matrix. These research results have shown the potential of sequential patterns for personalized recommendations. However, none of these studies provide any privacy features.

Location Privacy has been extensively studied over the past decade. A significant amount of research focused on the problem of private location-based queries, where users send their coordinates to obtain points of interest in their proximity. Some of the earlier work attempted to protect locations of real users by generating fake locations. For instance, in [31] the querying user sends to the server $k-1$ fake locations to reduce the likelihood of identifying the actual user position.

A new direction of research started by [6] and continued by [32], [33], [8] relies on the concept of *Cloaking Regions (CRs)*. CR-based solutions implement the spatial k-anonymity (SKA) [33] paradigm. For each query, a trusted anonymizer service generates CRs that contain at least k *real* user locations. If the resulting CRs are *reciprocal* [33], SKA guarantees privacy for snapshots of user locations. However, supporting continuous queries [34] requires generating large-sized CRs. The problem with CR-based methods is that the underlying k-anonymity paradigm is vulnerable to background knowledge attacks. This is particularly a problem in the case of moving users, since trajectory information can be used to derive the identities behind reported locations. In our work, we employ *differential privacy* [11], a provably secure model for semantic privacy that is currently the de-facto standard for privacy-preserving data publication.

VI. Conclusion

We investigated the application of differential privacy techniques to the process of location recommendations. We considered two categories of techniques, based on indexing in multi-dimensional spaces and n-gram trees. Experimental results show that the two approaches offer reasonable accuracy, and provide interesting tradeoffs relative to the check-in history length, data density and data skew. In future work, we plan to customize sanitization techniques to take into account specific characteristics of location recommendation query workloads. This will result in custom budget allocation strategies that will further increase accuracy.

References

[1] G. Adomavicius and A. Tuzhilin, "Towards the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 6, pp. 734–749, 2005.

[2] Z. Chen, H. T. Shen, and X. Zhou, "Discovering popular routes from trajectories," in *Proc. of IEEE International Conference on Data Engineering*, 2011, pp. 900–911.

[3] A.-J. Cheng, Y.-Y. Chen, Y.-T. Huang, W. H. Hsu, and H.-Y. M. Liao, "Personalized travel recommendation by mining people attributes from community-contributed photos," in *Proc. of ACM Intl. Conf. on Multimedia*, 2011, pp. 83–92.

[4] T. Kurashima, T. Iwata, G. Irie, and K. Fujimura, "Travel route recommendation using geotags in photo sharing sites," in *Proc. of ACM Intl. Conf. on Information and Knowledge Management*, 2010, pp. 579–588.

[5] Y.-T. Zheng, Z.-J. Zha, and T.-S. Chua, "Mining travel patterns from geotagged photos," *ACM Trans. on Intelligent Systems and Technology*, vol. 3, no. 3, pp. 56:1–56:18, 2012.

[6] M. Gruteser and D. Grunwald, "Anonymous Usage of Location-Based Services Through Spatial and Temporal Cloaking," in *USENIX MobiSys*, 2003, pp. 31–42.

[7] B. Gedik and L. Liu, "Protecting Location Privacy with Personalized k-Anonymity: Architecture and Algorithms," *IEEE Trans. on Mobile Computing*, vol. 7(1), pp. 1–18, 2008.

[8] M. F. Mokbel, C.-Y. Chow, and W. G. Aref, "The New Casper: Query Processing for Location Services without Compromising Privacy," in *VLDB*, pp. 763–774.

[9] G. Ghinita, P. Kalnis, and S. Skiadopoulos, "PRIVE: Anonymous Location-based Queries in Distributed Mobile Systems," in *Proc. of Intl. Conf. on WWW*, 2007, pp. 371–380.

[10] G. Ghinita, P. Kalnis, A. Khoshgozaran, C. Shahabi, and K. L. Tan, "Private Queries in Location Based Services: Anonymizers are not Necessary," in *Proc. of Intl. Conference on Management of Data (ACM SIGMOD)*, 2008.

[11] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Proc. of Theory of Cryptography Conf. (TCC)*, 2006, pp. 265–284.

[12] G. Cormode, C. Procopiuc, E. Shen, D. Srivastava, and T. Yu, "Differentially private spatial decompositions," in *Proc. of Intl. Conf. on Data Engineering (ICDE)*, 2012, pp. 20–31.

[13] J.-D. Zhang and C.-Y. Chow, "iGSLR: Personalized geo-social location recommendation - a kernel density estimation approach," in *Proc. of ACM SIGSPATIAL Intl. Conf. on Geographic Information Systems (GIS)*, 2013, pp. 334–343.

[14] J.-D. Zhang, C.-Y. Chow, and Y. Li, "iGeoRec: A personalized and efficient geographical location recommendation framework," *IEEE Transactions on Services Computing*, 2014 (to appear).

[15] E. Cho, S. A. Myers, and J. Leskovec, "Friendship and mobility: User movement in location-based social networks," in *Proc. of ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*, 2011, pp. 1082–1090.

[16] C. Dwork, "Differential privacy," in *Proc. of Intl. Colloquium on Automata, Languages and Programming (ICALP)*, 2006, pp. 1–12.

[17] X. Xiao and Y. Tao, "Output perturbation with query relaxation," *Proc. of VLDB*, vol. 1, no. 1, pp. 857–869, 2008.

[18] R. Chen, B. C. Fung, B. C. Desai, and N. M. Sossou, "Differentially private transit data publication: a case study on the Montreal transportation system," in *Proc. of ACM SIGKDD Intl. Conf. on Knowledge discovery and data mining*, 2012, pp. 213–221.

[19] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, *Computational Geometry: Algorithms and Applications*, 2nd ed. Springer-Verlag, 2000.

[20] F. McSherry and K. Talwar, "Mechanism design via differential privacy," in *Proc. of IEEE Symp. on Foundations of Computer Science (FOCS)*, 2007, pp. 94–103.

[21] R. Chen, G. Acs, and C. Castelluccia, "Differentially private sequential data publication via variable-length n-grams," in *Proc. of ACM Conf. on Computer and Communications Security (CCS)*, 2012, pp. 638–649.

[22] C. Cheng, H. Yang, I. King, and M. R. Lyu, "Fused matrix factorization with geographical and social influence in location-based social networks," in *Proc. of 26th AAAI Conference on Artificial Intelligence*, 2012.

[23] M. Ye, P. Yin, W.-C. Lee, and D.-L. Lee, "Exploiting geographical influence for collaborative point-of-interest recommendation," in *Proc. of ACM SIGIR Conf. on Research and Development in Information Retrieval*, 2011, pp. 325–334.

[24] V. W. Zheng, Y. Zheng, X. Xie, and Q. Yang, "Towards mobile intelligence: Learning from gps history data for collaborative recommendation," *Artificial Intelligence*, vol. 184-185, pp. 17–37, 2012.

[25] Y. Zheng, L. Zhang, Z. Ma, X. Xie, and W.-Y. Ma, "Recommending friends and locations based on individual location history," *ACM Transactions on the Web*, vol. 5, no. 1, pp. 5:1–5:44, 2011.

[26] T. Kurashima, T. Iwata, T. Hoshide, N. Takaya, and K. Fujimura, "Geo topic model: Joint modeling of user's activity area and interests for location recommendation," in *Proc. of ACM Intl. Conf. on Web Search and Data Mining*, 2013, pp. 375–384.

[27] B. Liu, Y. Fu, Z. Yao, and H. Xiong, "Learning geographical preferences for point-of-interest recommendation," in *Proc. of ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*, 2013, pp. 1043–1051.

[28] J. Bao, Y. Zheng, and M. Mokbel, "Location-based and preference-aware recommendation using sparse geo-social networking data," in *Proc. of ACM SIGSPATIAL Intl. Conf. on Geographic Information Systems (GIS)*, 2012, pp. 199–208.

[29] H. Gao, J. Tang, and H. Liu, "gSCorr: Modeling geo-social correlations for new check-ins on location-based social networks," in *Proc. of ACM Intl. Conf. on Information and Knowledge Management*, 2012, pp. 1582–1586.

[30] J. J. Levandoski, M. Sarwat, A. Eldawy, and M. F. Mokbel, "Lars: A location-aware recommender system," in *Proc. of IEEE Intl. Conf. on Data Engineering*, 2012, pp. 450–461.

[31] H. Kido, Y. Yanagisawa, and T. Satoh, "An anonymous communication technique using dummies for location-based services," in *International Conference on Pervasive Services (ICPS)*, 2005, pp. 88–97.

[32] B. Gedik and L. Liu, "Location Privacy in Mobile Systems: A Personalized Anonymization Model." in *Proc. of ICDCS*, 2005, pp. 620–629.

[33] P. Kalnis, G. Ghinita, K. Mouratidis, and D. Papadias, "Preserving Location-based Identity Inference in Anonymous Spatial Queries," *IEEE TKDE*, vol. 19, no. 12, 2007.

[34] C.-Y. Chow and M. F. Mokbel, "Enabling Private Continuous Queries for Revealed User Locations," in *SSTD*, 2007, pp. 258–275.