Feel Free to Check-in: Privacy Alert against Hidden Location Inference Attacks in GeoSNs

Zheng Huo[1] ⋆, Xiaofeng Meng[1], Rui Zhang[2]

[1]School of Information, Renmin University of China, Beijing, China
{huozheng,xfmeng}@ruc.edu.cn
[2]Department of Computing and Information Systems, University of Melbourne, Australia
rui@csse.unimelb.edu.au

Abstract. Check-in services, one of the most popular services in Geo-Social Networks (GeoSNs) may cause users' personal location privacy leakage. Although users may avoid checking in places which they regard as sensitive, adversaries can still infer where a user has been through linkage of multiple background information. In this paper, we propose a new location privacy attack in GeoSNs, called *hidden location inference attack*, in which adversaries infer users' location based on users' check-in history as well as check-in history of her friends and similar users. Then we develop three inference models (*baseline inference model*, *CF-based inference model* and *HMM-based inference model*) to capture the hidden location privacy leakage probability. Moreover, we design a privacy alert framework to warn users the most probable leaked locations. At last, we conduct a comprehensive performance evaluation using two real-world datasets collected from Gowalla and Brightkite. Experiment results show the accuracy of our proposed inference models and the effectiveness of the privacy alert framework.

Keywords: Privacy-preserving, location privacy, location-based social network, inference attack

1 Introduction

Geo-Social Network (GeoSN) is a kind of social network services where one or more individuals of similar interests or commonalities, connect with each other based on their geographical locations, such as, Foursquare, Gowalla and Brightkite. One of the most popular services in GeoSNs is **check-in service**, from which a user can report publicly which POI (Point of Interest) she has visited. Moreover, users can share their experiences at these places, such as giving commentary on the service or taste of food in a restaurant. The popularity of check-in service is not only because of its fun and online connectivity to friends, but also because of the material benefits. For example, free drinks or coupons are given to users who frequently check in a coffee shop. Therefore,

⋆ This research was partially supported by the grants from the Natural Science Foundation of China (No. 61070055, 91024032, 91124001); the National 863 High-tech Program (No. 2012AA010701, 2013AA013204); the Fundamental Research Funds for the Central Universities, and the Research Funds of Renmin University (No. 11XNL010).

users usually have a strong incentive to check in as more POIs as they can if there are no privacy concerns.

Generally speaking, users engaged in GeoSNs are in two categories: *real name users* and *pseudonym users*. Real name users use their true names as identifiers, people might easily associate their usernames in GeoSNs with individuals in reality. Pseudonym users prefer to use pseudonyms as their identifiers, which can avoid being directly associated with themselves in reality. For both kinds of users, publication of locations from check-in services raise severe privacy threats. Location is considered as private when it is itself sensitive or it may lead to disclosure of sensitive information. For example, by knowing a person is at a church during a religious festival, adversaries may infer the user's religious belief with a high probability. Even if the user is using her real name in GeoSNs, she may not be willing to expose her religious belief. On the other hand, location itself may act as a quasi-identifier. When location information is linked with other external knowledge, it may compromise user's anonymity and hence allow adversaries to associate the user's identity to sensitive information. People may argue that, users should be aware of location privacy leakage and avoid checking in sensitive places, or places that may expose their identities. This is undesirable for the following two reasons: (i). Users are eager to check in when they go to somewhere because of mental joy and material benefits. If too many POIs are considered as sensitive and could not be checked in, it may cause a negative impact on user experience; (ii). Although users may avoid checking in sensitive places (e.g., churches, hospitals or bars), they may not be aware of their *hidden location privacy leakage*, which is caused by linkage of users' check-in time, historical check-in behavior, as well as check-in behavior of other users, etc. That is to say, if a user does not check in a POI which she regards as sensitive, adversaries can still infer the probability of the user's visit to the POI. Leakage of hidden location privacy raises even more serious privacy threats to GeoSN users, since most of these POIs are where users intend to hide. A detailed example of hidden location privacy leakage is shown in Figure 1.

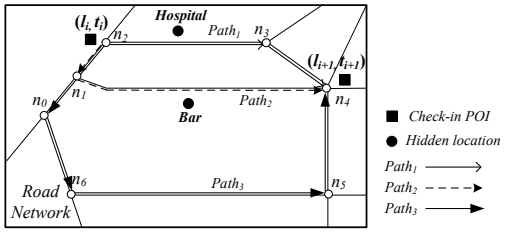


Fig. 1. An example of hidden location privacy leakage

As shown in Figure 1, a GeoSN user u_k checks in POI l_i at time t_i. When she reaches l_{i+1} at time t_{i+1}, she sends a check-in request at l_{i+1}. There might be privacy threat in this situation. Since the road network is publicly accessible, adversaries know that u_k might have taken one of the paths among $Path_1(n_2n_3n_4)$, $Path_2(n_1n_4)$ and $Path_3(n_1n_0n_6n_5n_4)$ to get to l_{i+1}. Since $Path_3$ is a long way, u_k might not prefer to travel on it. Suppose there is an AIDS hospital on $Path_1$ and a bar on $Path_2$. Although u_k doesn't check in the bar or the hospital for fear of privacy leakage, adversaries may

infer how likely she has visited one of them through linkage of multiple background information, summarized as follows.

- Geographical information, e.g., road networks, travel distances and check-in time, etc. If the time spent between l_i and l_{i+1} is less than or equal to the shortest time that one can reach l_{i+1} from l_i, then u_k definitely does not visit any other POIs during her movement.
- Historical information, e.g., u'_ks historical check-ins and her friends' historical check-ins, etc. If adversaries know the majority of users may check in the bar on $Path_2$ when they move from l_i to l_{i+1}, adversaries may infer that u_k probably visited the bar during her movement from l_i to l_{i+1}.
- Social information, e.g., social relationships, user closeness and similarity, etc. Suppose u_k checks in l_i and l_{i+1} at t_i and t_{i+1} respectively, a friend of u_k, say u_j checks in l_i around t_i and l_{i+1} shortly before t_{i+1}. Apparently, u_k was hanging out with u_j. If u_j checks in the bar on $Path_2$, adversaries may infer u_k also visited the bar although she didn't check in.

In this paper, we study the problem of **hidden location inference attack** in GeoSNs. The key challenge of our proposal is how to accurately and efficiently evaluate the hidden location leakage probability (HLPL-probability) and rank hidden locations based on the probability, as well as designing a privacy alert framework for GeoSN users. Contributions of this paper are summarized as follows.

- We propose a new privacy attack model, called *hidden location inference attack* in GeoSNs, from which users' most probable visited locations can be inferred through linkage of multiple background information.
- We propose three inference models to derive the HLPL-probability, they are baseline inference model, CF-based inference model and HMM-based inference model. The basic idea is that a user's visit behavior may be inferred by historical check-ins, as well as check-ins of her close friends and similar users. Then we employ Bayes' law, collaborative filtering and hidden Markov model respectively to derive the HLPL-probability.
- We design a novel privacy alert framework for GeoSN users to detect hidden location privacy leakage. We then implement our proposed inference models in this framework under road network constraints.
- Finally, we experimentally evaluate the inference models on two real-world datasets. Results show accuracy of the inference models and the effectiveness of the privacy alert framework.

The rest of the paper is organized as follows. Section 2 summarizes related work. Section 3 formally defines concepts that we study in this paper. In section 4, we illustrate our proposed inference models. Section 5 proposes the privacy alert framework. Experimental results are presented in Section 6. Finally, Section 7 concludes the paper.

2 Related Work

We conduct a brief review of existing studies on GeoSN data analysis, privacy-preserving and location prediction in GeoSNs.

Studies on GeoSN data analysis mainly focus on analyzing GeoSN users' behavioral patterns, social relationships, how social relationships and user mobilities affect each other, etc. Noulas et al. analyze user check-in dynamics, demonstrating how it reveals meaningful spatio-temporal patterns in Foursquare [6]. Studies in [8] mainly focus on how users' social relationships are affected by their distances and how users' social ties stretch across space. The above two papers study either spatial property or social property in GeoSNs separately. Cho et al. study both GeoSN user mobilities and social relationships jointly in [2]. It is claimed that more than 50% of GeoSN users' movements are affected by their friends. This fully support our research in this paper. Meanwhile, new privacy threats arise in the context of GeoSNs. In [3], authors notice that contents published by GeoSN users may lead to users' location privacy leakage or absence privacy leakage. The former is caused by uncontrolled exposure of users' geographical locations, while the latter concerns on the absence of a user from a geographic position. Authors propose a spatial generalization and publication delay method to solve these problems. Predicting users' social ties and locations is a new research topic in GeoSNs. Backstrom et al. predict home addresses of Facebook users based on addresses of users' friends [1]. It is claimed that this method dominates an IP-based one. More recently, [9] proposes to predict a trip's destination against data sparsity problems, then they develop a method which can select a minimum number of locations that a user has to hide in order to avoid privacy leak. The most related work to ours is proposed by Sadilek et al. in [7], which proposes to predict both social ties and locations of a user. In location prediction, a dynamic Bayesian network model is utilized to predict un-observed locations discretely.

Our proposal is different from [7] in the following two aspects: (i). We infer users' HLPL-probability between users' two observed check-ins, and rank hidden locations based on the probability. In our proposal, hidden locations are real-world semantic places where users might visit. However, situations in [7] are quite different, the inferred locations might be any geographic location. (ii). We propose a privacy alert framework under road network constraints, the most probable leaked hidden locations and corresponding HLPL-probability are pushed to users, which is obviously different from [7].

3 Preliminaries

A check-in activity consists of three factors: user, POI and check-in time. A GeoSN user may check in several POIs in her everyday life trajectories. Checks-in activities ordered by time constitute of a check-in sequence of the user.

Definition 1. ***(Check-in Sequence)*** *A user's check-in sequence* $\mathbb{S}$ *is a set of POIs ordered by check-in time* $\mathbb{S}=\{u_k, (l_1, t_1), \ldots, (l_i, t_i), \ldots, (l_n, t_n)\}$*, where* u_k *is the identifier of the user,* l_i *and* t_i *represent where* u_k *checks in and when she checks in respectively.*

In real-world, a user may repeatedly check in several POIs where she frequently visits. Therefore, a POI may appear more than once in one's check-in sequence, with different timestamps. A similar definition of check-in sequence is *visit sequence*, which consists of a set of doublets of POIs where the user has visited and corresponding timestamps. Note that, *visit sequence* is a superset of check-in sequence, since people may visit a number of POIs where they do not check in.

Definition 2. ***(Hidden Location)*** *Given POIs l_i and l_{i+1} which are checked in by user u_k at time t_i and t_{i+1} respectively. A hidden location l_m is a POI that might be visited by u_k when she moves from l_i to l_{i+1}, but not checked in by u_k at time t_m $(t_i < t_m < t_{i+1})$.*

Hidden location inference attack is a kind of location privacy attacks, from which adversaries can infer users' most probable visited hidden locations. One of the most serious consequences of hidden location privacy leakage is that users cannot control the privacy leakage, sometimes they do not even aware of their hidden location privacy leakage. Therefore, a privacy alert mechanism is required to warn users when their hidden location privacy is threatened.

4 Inference Models

The GeoSN service providers collect user generated data, including check-in sequences, social relationships and user profile information, etc. We make a widely accepted assumption that GeoSN service providers are un-trusted, which means service providers may analyze user data itself or may share it with the third party. Either of them may cause users' hidden location privacy leakage. Take user u_k as an example. Suppose u_k has checked in POI l_i and l_{i+1} at t_i and t_{i+1} respectively, l_i and l_{i+1} are called *observed locations*. The time interval between two check-ins can be represented as $\Delta t = t_{i+1} - t_i$. Given a hidden location l_m between two observed locations, we propose three inference models to infer the HLPL-probability, explained in the following subsections.

4.1 Baseline Inference Model

Users' check-in behaviors follow certain patterns or change periodically between two POIs, this can be obtained by aggregating users' history check-in behaviors [6]. Thus, adversaries can use majority users' behavioral patterns to "guess" how likely one would visit a POI. Given a hidden location l_m and the time interval between two check-ins Δt, the HLPL-probability can be denoted as a posterior probability $P(V_k^{i,m,i+1}|\Delta t)$, where $V_k^{i,m,i+1}$ denotes u_k's *sub-visit sequence* that contains l_i, l_m and l_{i+1} sequentially. Since u_k has already checked in l_i and l_{i+1}, the key point of $P(V_k^{i,m,i+1}|\Delta t)$ is how likely u_k would have visited l_m during her movement from l_i to l_{i+1}. According to Bayes' Law, $P(V_k^{i,m,i+1}|\Delta t)$ can be calculated as follows.

$$P(V_k^{i,m,i+1}|\Delta t) = \frac{P(\Delta t|V_k^{i,m,i+1})P(V_k^{i,m,i+1})}{P(\Delta t)} \tag{1}$$

Given a definite Δt for u_k, $P(\Delta t)$ is a constant, equation (1) can be simplified to $P(V_k^{i,m,i+1}|\Delta t) \approx P(V_k^{i,m,i+1}) \times P(\Delta t|V_k^{i,m,i+1})$. The baseline inference model aggregates users' historical check-in behaviors to draw an inference. Thus, $P(V_k^{i,m,i+1})$ can be calculated by equation (2).

$$P(V_k^{i,m,i+1}) = \frac{\sum_s C_s^{i,m,i+1}}{\sum_s C_s^{i,i+1}} \tag{2}$$

Equation (2) measures the fraction of sub-check-in sequences which check in l_m from all sub-check-in sequences that move from l_i to l_{i+1}. $C_s^{i,i+1}$ represents a sub-check-in sequence s that consists of l_i and l_{i+1} sequentially. $C_s^{i,m,i+1}$=1 if and only if one checks in l_i, l_m and l_{i+1} sequentially in s, otherwise, $C_s^{i,m,i+1}$=0.

In equation (1), Δt= t_{i+1} - t_i which represents the time interval between u_k's two observed check-ins also has spatial reach-ability meaning. If u_k cannot reach l_{i+1} from l_i within Δt, u_k definitely does not have time to visit any other locations, $P(V_k^{i,m,i+1}|\Delta t)$ must be zero. Note that, the time interval of two check-ins may not be exactly the same. Thus, we make use of an upper bound of Δt. Formally, we have $P(\Delta t|V_k^{i,m,i+1}) \leq P(\Delta t_s \leq \Delta t|V_k^{i,m,i+1})$, which can be calculated by equation (3).

$$P(\Delta t|V_k^{i,m,i+1}) \leq P(\Delta t_s \leq \Delta t|V_k^{i,m,i+1}) = \frac{\sum_s C_s^{i,m,i+1} \times P(\Delta t_s \leq \Delta t)}{\sum_s C_s^{i,m,i+1}} \tag{3}$$

where Δt_s represents the time interval between checking in l_i and l_{i+1} of sub-check-in sequence s. We then multiply $P(V_k^{i,m,i+1})$ and $P(\Delta t|V_k^{i,m,i+1})$ to get u_k's HLPL-probability of hidden location l_m.

• ***Weighted by Friend Closeness***

In practice, friends tend to have similar behaviors because they are friends and might share lots of common interests or always hang out together, thus leading to similar visit behaviors. Besides, friends in GeoSNs tend to have friendships in reality [4]. Previous studies show that friendships in GeoSNs have more influences over one's behaviors [8]. Thus, we refine the baseline inference model by introducing a parameter called *friend closeness*. Given two users u_j and u_k, u_j is one of u_k's friends, we adopt a formula in [10] to compute closeness of u_k and u_j, also shown as follows.

$$\omega_c(u_k, u_j) = \alpha \frac{|F_k \cap F_j|}{|F_k \cup F_j|} + (1-\alpha)\frac{|L_k \cap L_j|}{|L_k \cup L_j|} \tag{4}$$

where α is a turning parameter ranging within [0,1]. F_k and F_j represent friend sets of u_k and u_j respectively, L_k and L_j represent POI sets where u_k and u_j have checked in respectively. Friends with more common friends means they have close social ties; friends who show more similar check-in behaviors should have much similar tastes. We have an observation that, friends with close social ties and similar tastes may have higher probability to hang out together. Therefore, we give it a higher weight in the inference equation, thus, $P(V_k^{i,m,i+1})$ can be calculated by equation (5).

$$P(V_k^{i,m,i+1}) = \frac{\sum_s (1+\omega_c(u_k, u_j)) C_s^{i,m,i+1}}{\sum_s (1+\omega_c(u_k, u_j)) C_s^{i,i+1}} \tag{5}$$

where, $C_s^{i,m,i+1}$ and $C_s^{i,i+1}$ have no difference from equation (2) and $P(\Delta t|V_k^{i,m,i+1})$ is calculated by equation (3) as previously. By inclusion of friend closeness, the HLPL-probability is more affected by one's close friends than other normal users. It should be noted that, the baseline inference model can deal with one hidden location each time, if there are multiple hidden locations among two observed check-ins, it should be executed several times.

4.2 CF-based Inference Model

Collaborative Filtering (CF) is a method of making predictions about interests of a user by collecting preferences or taste information of many other users, a user's *rating on an object* can be inferred through ratings of her similar users. In GeoSNs, similar users who might share a lot of common interests tend to have similar visit behaviors. Accordingly, adversaries can infer the HLPL-probability using CF. In order to calculate user similarity, we introduce a concept called *visit probability sequence* to evaluate users' visit probabilities to a set of POIs.

Definition 3. ***(Visit Probability Sequence)*** *Given a sub-check-in sequence* $s=\{l_1, l_2, \ldots, l_n\}$. $u'_k s$ *visit probability sequence of* s *is a set of probabilities* $PV_{uk}=\{PV^1_{uk}, PV^2_{uk}, \ldots, PV^n_{uk}\}$, *where* $PV^i_{u_k} \in [0,1]$ *denotes* $u'_k s$ *visit probability to POI* l_i.

User similarity between u_k and u_j can be calculated by the cosine similarity of their visit probability sequences, as shown in equation (6).

$$sim(u_k, u_j) = \frac{\sum_i PV^i_{u_k} PV^i_{u_j}}{\sqrt{\sum_i PV^{i\ 2}_{u_k}} \sqrt{\sum_i PV^{i\ 2}_{u_j}}} \tag{6}$$

In equation (6), u_k and u_j do not need to have friendships in GeoSNs. $PV^i_{u_k} \in [0,1]$ and $PV^i_{u_j} \in [0,1]$ denote u_k and u'_js visit probability to POI l_i respectively. A special case of visit probability sequence is the check-in sequence, in which $PV^i_{uk}=1$ if u_k checks in l_i and $PV^i_{uk}=0$ if u_k does not. For each user u_k, we utilize two matrices to calculate u'_ks visit probability to hidden locations, as shown in Figure 2. Matrix S is a user-user matrix, each value s_{kj} in S represents the similarity between u_k and u_j, calculated by equation (6). Matrix U is a location-user matrix, each value u_{kn} represents u'_ks visit probability to l_n. Users in matrix U are the top-n most similar users of u_k and locations in matrix U are u'_ks hidden locations between two observed check-ins. Matrix U is initialized by u'_ks check-in sequence. We then calculate user similarity through their check-in sequences, and put user similarities into matrix S for initialization. In matrix U, u'_ks visit probabilities to hidden locations are the missing values. Equations (7) and (8) illustrate how to infer the missing values using classical CF method.

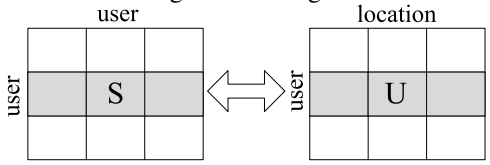


Fig. 2. Matrices in CF-based inference model

$$r_{u_k,l_n} = k \times \sum_{u_j \in S_k} sim(u_k, u_j) \times r_{u_j,l_n} \tag{7}$$

$$k = \frac{1}{\sum_{u_j \in S_k} |sim(u_k, u_j)|} \tag{8}$$

where r_{u_k,l_n} represents u'_ks visit probability to hidden location l_n, the higher the value is, more probable that u_k has visited l_n. S_k denotes u'_ks similar user set, r_{u_j,l_n} is

u'_js visit probability to location l_n, where, u_j is one of u'_ks similar users. In equation (7), the similarity between u_k and u_j, $sim(u_k, u_j)$ is used as a weight, more similar u_k and u_j are, more weight r_{uj,l_n} will carry in predicting r_{uk,l_n}. When we say u_j is a similar user of u_k, we mean $sim(u_k, u_j) > 0$. After initialization of matrices S and U, we begin to calculate u'_ks ratings on hidden locations in matrix U. We get u'_ks ratings on hidden locations in matrix U through repeatedly calculating the following steps, until the matrices converge.

- Calculate users' visit probabilities to hidden locations through equation (7) and (8), then update matrix U.
- Calculate user similarities using equation (6), then update matrix S.

We then compute the posterior probability to derive the HLPL-probability according to equation (1). Given Δt, the HLPL-probability $P(V_k^{i,m,i+1}|\Delta t)$ can be calculated by equation (9).

$$P(V_k^{i,m,i+1}|\Delta t) = r_{u_k,l_m} \times \frac{\sum_{u_j \in S_k} C_{u_j}^{i,m,i+1} P(\Delta t_{u_j} \leq \Delta t)}{\sum_{u_j \in S_k} C_{u_j}^{i,i+1}} \tag{9}$$

In equation (9), S_k denotes u'_k similar user set, u_j is one of u'_ks similar users. Δt_{uj} is the time interval between u_j checks in l_i and l_{i+1}. Once CF-based inference model is executed, the missing values in matrix U is completed, thus, u'_ks HLPL-probability for each hidden location is derived.

4.3 HMM-based Inference Model

Hidden Markov Model (HMM) is a statistical Markov model in which the system being modeled is assumed to be a Markov process with hidden states. If one can make predictions for the future of the process based solely on its present state, it is a Markov process. In the previous section, we hold the opinion that users' visit probability to a POI can be inferred though historical information, but this does not mean users' next location can be predicted based on all historical check-in sequences. Actually, given the moving speed and road network conditions, u'_ks current location can be inferred solely based on the previous state (where u_k checks in, as well as where u'_ks friends and similar users check in, etc.). Therefore, we utilize Hidden Markov Model (HMM) to represent users' transitions among hidden locations.

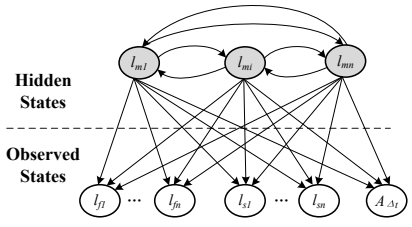


Fig. 3. HMM-based Inference Model

Given two POIs l_i and l_{i+1} which u_k has checked in, we build a HMM-based inference model for each user, as shown in Figure 3. Nodes in white are observed nodes, while nodes in gray are hidden nodes which correspond to hidden locations that u_k might visit during two observed check-ins. Here we just list three hidden nodes as an example. The observed nodes l_{f1} to l_{fn} are where u'_k s top-n closest friends check in within time Δt (time interval between two observed check-ins). The observed nodes l_{s1} to l_{sn} represent where u'_ks top-n similar users check in within time Δt, but without users who appear in the top-n closest friends. Node $A_{\Delta t}$ is the attribute of Δt, e.g., weekends or weekdays, morning, noon or evening of a day. All the observed nodes and hidden nodes are discrete.

We train the HMM model using the real-world datasets through Expectation - Maximization (EM) algorithm. In the expectation step, we complete the dataset using the current parameters $\theta^{(t)}$ (always start with random values) to calculate the expected values of the log likelihood function, denoted as equation (10).

$$Q(\theta|\theta^{(t)}) = E_{H|O,\theta^{(t)}}[logP(O,H|\theta)] \tag{10}$$

where O is a set of observed nodes, while H is a set of hidden nodes. In the Maximization step, we use the *completed* dataset to find a new maximum likelihood estimate for a vector of unknown parameter θ, the target function is denoted as equation (11).

$$\theta^{(t+1)} = \arg_\theta maxlog(Q(\theta|\theta^{(t)})) \tag{11}$$

where $\theta^{(t+1)}$ is the final parameters of the HMM model. The algorithm repeats the two steps until it converges. After we get a set of parameters of the HMM model, transition probabilities among hidden locations are derived, which happens to be the HLPL-probability in our settings. Then we infer the hidden location sequence that is most likely to have generated the observed locations. We use Viterbi algorithm to efficiently infer the hidden location sequence. The target function can be denoted as equation (12).

$$H^* = \arg maxlog(P(H|O)) \tag{12}$$

Equation (12) represents the conditional probability of the hidden node H, given observations O. We apply dynamic programming in this problem, thus achieving a time complexity of $O(|O| \times |H|^2)$, where $|O|$ is the number of observations and $|H|$ is the number of hidden nodes.

5 Hidden location privacy alert framework

5.1 System Model

Our hidden location privacy inference framework is based on the *client-inference server-GeoSN server* model as depicted in Figure 4.

There are two main components in the inference server, namely, *hidden location finder* and *probability estimator*. Hidden location finder finds out all the hidden locations between l_i and l_{i+1} for each user, and calculates the shortest road network distance between l_i and l_{i+1}. Probability estimator is in charge of estimating users' HLPL-probabilities using the inference models we previously defined and rank hidden locations based on the probability. Each client u_k, sends her check-in request $C_k(l_{i+1}, t_{i+1})$

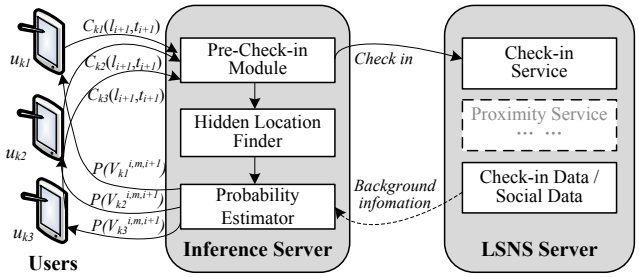


Fig. 4. System Architecture

at location l_{i+1} along with check-in time t_{i+1} to the *pre-check-in module*. The received requests are sent to the *hidden location finder*, which derives u'_ks latest check-in location l_i and check-in time t_i. Hidden locations between two check-ins are found out by *hidden location finder*, and passed to *probability estimator*. The background information for inference is acquired from *GeoSN server*. At last, the most probable leaked hidden locations with corresponding HLPL-probability are pushed to users, as a privacy alert. Users could decide whether to check in or not according to their privacy preferences.

5.2 Algorithms

The inference models and the privacy alert framework are implemented in the road network space rather than free space. In our settings, a road network is represented as an undirected graph $G(V,E)$; each road segment is represented by edges in G, intersections of the routes are represented by vertices in V. GeoSN users' moving speed is bounded by the average maximum moving speed of road segments.

Given a user u_k, we prune two situations that u_k may not visit any hidden location: (i). If $\Delta t \leq \frac{Dist(l_i,l_{i+1})}{v_{max}}$, u_k must definitely not visit any hidden location, since the time is not enough for u_k to visit any location when she moves from l_i to l_{i+1}. Where $Dist(l_i,l_{i+1})$ is the shortest road network distance between l_i and l_{i+1}, v_{max} is the average maximum moving speed of routes $l_i \rightarrow l_{i+1}$. (ii). If $\frac{Dist(l_i,l_{i+1})}{v_{max}} < \Delta t < Min(\Delta t_j)$, u_k may probably not visit hidden locations, since the time spent between l_i and l_{i+1} is less than any sub-check-in sequence that begins with l_i and ends with l_{i+1}. Where $Min(\Delta t_j)$ is the minimum time cost between two observed check-ins for any user. In the last case, if $\Delta t \geq Min(\Delta t_j)$, we need to find out all the hidden locations between l_i and l_{i+1}, which is accomplished by algorithm **FindHiddenLocation**, then we utilize three inference models to calculate the HLPL-probability (due to space limitation, the details will not be explained again in this subsection).

Finding out all the un-checked-in POIs between l_i and l_{i+1} as hidden locations is unrealistic and useless, since POIs located on a far or unpopular path may rarely be visited by GeoSN users. Therefore, we only take two kinds of POIs into consideration: (i). POIs on the shortest path between l_i and l_{i+1}. (ii). POIs on the popular paths between l_i and l_{i+1}. The details are shown in Algorithm 1.

In Algorithm 1, we utilize the A^* algorithm to find the shortest path between l_i and l_{i+1}, since the shortest path is always a choice to users when they go to somewhere (line

Algorithm 1: FindHiddenLocation

Input : POIs l_i and l_{i+1}; road network $G(V,E)$; a set of history check-in sequences $\overline{\mathbb{S}}$;
Output: A set of hidden locations L_m; length of the shortest path $Dist(l_i, l_{i+1})$;

1 $Path_{sh} \leftarrow A^*(G, l_i, l_{i+1})$;
2 $L_m \leftarrow$ POIs on $Path_{sh}$;
3 $Dist(l_i,l_{i+1}) \leftarrow$ distance between l_i and l_{i+1} on $Path_{sh}$;
4 Tag each check-in sequence in $\overline{\mathbb{S}}$ as *unscanned*;
5 **while** *exists a check-in sequence in* $\overline{\mathbb{S}}$ *unscanned* **do**
6 Scan the next check-in sequence $\mathbb{S}$;
7 Tag $\mathbb{S}$ as scanned;
8 **if** $count(l_i \xrightarrow{\Delta t_1<\delta t} l_m \xrightarrow{\Delta t_2<\delta t} l_{i+1}) > \delta_p$ **then**
9 $L_m \leftarrow L_m \bigcup l_m$
10 **Return** L_m and $Dist(l_i, l_{i+1})$;

1-3). Without loss of generality, we scan each check-in sequence in $\overline{\mathbb{S}}$ and identify POIs that frequently appear on a popular path between l_i and l_{i+1}, then put them into L_m. If the support of a check-in pattern $l_i \xrightarrow{\Delta t_1<\delta t} l_m \xrightarrow{\Delta t_2<\delta t} l_{i+1}$ is larger than the support threshold δ_p, l_m can be regarded as a hidden location (line 4-9). We have a key observation that the temporally consecutive check-ins of u_k could signal correlations between two POIs, but as the time interval increases, that two check-ins are not strictly consecutive. In order to solve this problem, we make use of a time threshold δ_t, which is used to estimate whether a check-in pattern is valid or not. Δt_1 and Δt_2 represent the corresponding check-in time intervals. At last, hidden location set L_m and the shortest path distance between l_i and l_{i+1} are returned.

6 Experiments

The design of the experiments aims to achieve the following goals: (i) Analyze properties of the experiment datasets; (ii) Learn the performance of each inference model; (iii) Learn the effectiveness of the privacy alert framework.

6.1 Experimental Setup

We run our experiments on two real-world GeoSN datasets, made available by Cho et al. [2]. The datasets collect users' social relationships and check-in behaviors from Feb. 2009 to Oct. 2010 in Gowalla and Apr. 2008 to Oct. 2010 in Brightkite. We also obtain the road network data of California, which contains 21,693 edges and 104,407 POIs. We pre-process both datasets, check-ins in California and related users are left. As a result, the Gowalla dataset contains 15,116 users and 675,809 check-ins, while the Brightkite dataset contains 9,435 users and 541,169 check-ins. Detailed information about both datasets are shown in Table 1.

Besides, we randomly select 22,495 pairs of consecutive check-ins, distributions of check-in time interval and distance between two consecutive check-ins are shown

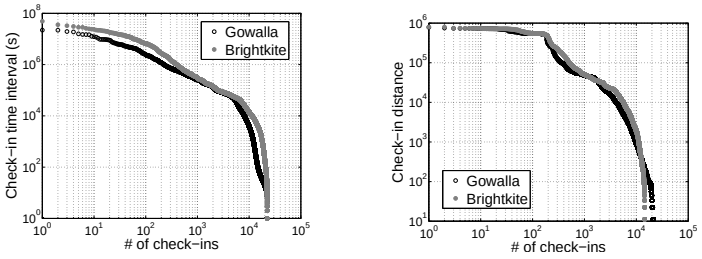
Total check-ins		**Total users**		**Area size** (km^2)		**Avg. check-in interval**(h)	
Gowalla	Brightkite	Gowalla	Brightkite	Gowalla	Brightkite	Gowalla	Brightkite
675,809	541,169	15,116	9,435	443,556	443,556	40.31	56.81
Density($user/km^2$)		**Check-in / user**		**Check-in / POI**		**Avg. check-in distance**(km)	
Gowalla	Brightkite	Gowalla	Brightkite	Gowalla	Brightkite	Gowalla	Brightkite
0.03	0.02	44.71	57.36	6.29	6.10	19.02	15.39

Table 1. Detailed information about Gowalla and Brightkite

in Figure 5. It can be seen that, about 60% consecutive check-ins occur within a time interval of $10^4 s$ in Gowalla and 33% in Brightkite, and about 55% consecutive check-ins occur within a distance of $1km$ for both datasets.

6.2 Performance Metrics

Since real hidden locations are invisible in check-in datasets, we make an assumption of our dataset that a user visits a POI if and only if she checks in the POI. Then two hidden location datasets are generated in our experiments. Given a user u_k, *hidden location set I* is generated by marking off a part of POIs that u_k has checked in; *hidden location set II* is generated by adding POIs which are geographically located between l_i and l_{i+1} that u_k does not check in. We then evaluate four performance metrics: (i) The ratio of recovered hidden locations to the number of hidden locations in hidden location set I, represented as *true positive rate*; (ii) Average HLPL-probability of hidden location set I, represented as $AVG@I$; (iii) The ratio of recovered hidden locations to the number of hidden location set II, represented as *false positive rate*; (iv) Average HLPL-probability of hidden location set II, represented as $AVG@II$. The *true negative rate* and *false negative rate* can be derived from the above metrics.



(a) Check-in interval (b) Check-in distance

Fig. 5. Datasets attribute

6.3 Study on Accuracy

We evaluate the performance of following inference models: baseline inference model, denoted by BI; baseline inference model weighted by friend closeness, denoted by WFI; CF-based inference, denoted by CFI; HMM-based inference, denoted by $HMMI$.

We measure *AVG@I*, *AVG@II*, *true positive rate* and *false positive rate* on both datasets. For each inference model, we randomly select 3,000 users and choose l_i and l_{i+1} in each user's check-in sequence. The selection of l_i and l_{i+1} should satisfy a constraint that time interval Δt is bounded by 5 times of average check-in time interval (*since more than 96.6% consecutive check-ins have a time interval within 5 times of average check-in time interval for Gowalla and 97.1% for Brightkite*). For each user, we randomly mark off 5, 10, 15 and 20 visited POIs between l_i and l_{i+1} as *hidden location set I*, and add 5, 10, 15 and 20 un-checked-in POIs that geographically locate between l_i and l_{i+1} as *hidden location set II*. In the HMM-based inference model, we use Matlab toolbox for HMM [5] to train the HMM model. We utilize historical check-ins of one's top-5 close friends and top-5 similar users as training set.

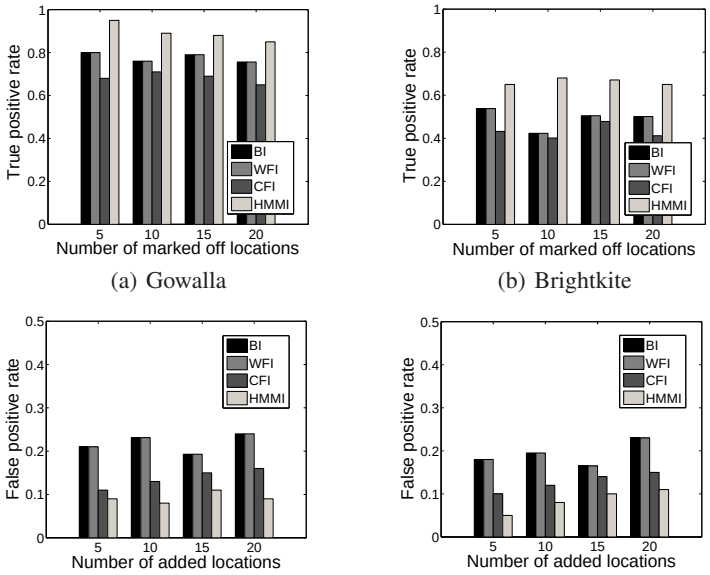
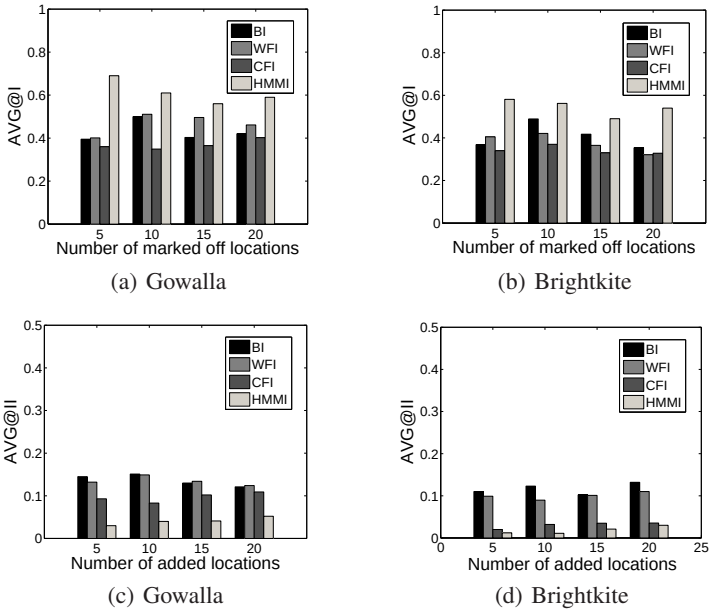


Fig. 6. Performance estimation on true / false positive rate.

The experimental results in Figure 6 and Figure 7 indicate that, performance on all the metrics are not seriously affected by the increasing number of the marked off / added locations, showing that our inference models work with large number of hidden locations. In Figure 6, the higher the *true positive rate* is, the better the inference model performs, and the *false positive rate* represents just the opposite. It can be seen that the HMM-based inference model always exhibits the best performance in terms of *true positive rate* and *false positive rate*, under all values of marked off / added locations, showing the strength of combining locations of one's friends and similar users to infer one's hidden location. Besides, the learning technique which captures the transition probability between one's visited locations also makes the HMM-based inference model outstanding. *BI* and WFI have the same *true positive rate* and *false positive rate*, since WFI is a special case of *BI*, it can recover hidden locations which *BI* can also do. It should be noted that, although the *BI* and WFI performs not bad on true positive rate,

but the performance differences among users are quite huge, about 16.1% users have zero true positives on Gowalla, and about 18% on Brightkite. This is because some users' check-in sequences are un-popular, making it hard to infer wether the user has visited the hidden locations through historical check-in sequences. The HMM-based inference model is much better, only less than 5% users have zero true positives on both datasets. Another drawback of *BI* and WFI inference models is that, they have high false positive rate, especially in areas with dense check-in activities. It should be noted that, CF-based inference model have both low *true positive rate* and low *false positive rate*, which is also caused by the data sparsity problem. Since there are few similar users that visit a hidden location in a given time span, thus resulting in the poor performance on true positive rate.



(a) Gowalla (b) Brightkite

(c) Gowalla (d) Brightkite

Fig. 7. Performance estimation on average HLPL-probability.

In Figure 7, the higher the $AVG@I$ value is, the better the inference model performs, while $AVG@II$ value just represent the opposite. It can be seen that, the $AVG@I$ value dominate the $AVG@II$ value on all numbers of marked off / added locations, showing that our inference models can recover the hidden locations and derive the HLPL-probabilities effectively. WFI and *BI* have the same true positive rate and false positive rate, the difference is that, WFI performs slightly better on both metrics than that of *BI*, since both datasets are sparse (as shown in Table 1), the co-appearance of two users does not always happen, making the WFI model performs not as well as expected. The HMM-based inference model performs the best on $AVG@I$ and $AVG@II$ among all inference models, the $AVG@I$ value of $HMMI$ reaches almost 60% and the $AVG@II$ value is even below 5% on all values of marked off / added locations. We do not test the precision of hidden location visit sequences generated by $HMMI$, since it is not our concern in this paper.

6.4 Study on efficiency

We study the efficiency of the privacy alert framework, showing that the average response time from sending a check-in request to the inference server to get a privacy alert is in minutes level. Although the response time does not satisfy the real-time applications well, the framework is still available, since the response time is much shorter than users' average stay time at a place. How to improve the efficiency of the privacy alert framework will be considered in the future work. One possible solution is to separate the whole procedure into the *training phase* and the *inference phase* [9]. The training phase can be performed offline, which may sharply reduce the response time.

7 Conclusions and Future Work

Check-in service in GeoSNs raises serious privacy concerns. In this paper, we propose a new privacy attack called hidden location inference attack. To accurately derive the HLPL-probability, we propose three inference models. At last, we design a privacy alert framework to warn users the most probable leaked hidden locations/visit sequence. We evaluate the inference models on two real-world datasets, experiment results indicate that the HMM-based inference model has the best performance among all.

Our future work are in two-fold. First, we will investigate how to improve the performance of the inference models against data sparsity problem. Second, we plan to study how to protect users' hidden location privacy against the attack models. One of the possible solutions could be adding fake check-in POIs when a check-in activity happens. While the challenging problems are how to balance the privacy and utility of users' check-in, as well as considering users' privacy preferences, etc.

References

1. L. Backstrom, E. Sun, and C. Marlow. Find me if you can: improving geographical prediction with social and spatial proximity. In *WWW'10*, pages 61–70, 2010.
2. E. Cho, S. A. Myers, and J. Leskovec. Friendship and mobility: user movement in location-based social networks. In *KDD'11*, pages 1082–1090, 2011.
3. D. Freni, C. R. Vicente, S. Mascetti, C. Bettini, and C. S. Jensen. Preserving location and absence privacy in geo-social networks. In *CIKM'10*, pages 309–318, 2010.
4. A. Gruzd, B. Wellman, and Y. Takhteyev. Imagining twitter as an imagined community. 2011.
5. MIT. Hidden markov model (hmm) toolbox for matlab. http://www.cs.ubc.ca/ murphyk/-Software/HMM/hmm.html.
6. A. Noulas, S. Scellato, C. Mascolo, and M. Pontil. An empirical study of geographic user activity patterns in foursquare. In *ICWSM'11*, 2011.
7. A. Sadilek, H. A. Kautz, and J. P. Bigham. Finding your friends and following them to where you are. In *WSDM'12*, pages 723–732, 2012.
8. S. Scellato, A. Noulas, R. Lambiotte, and C. Mascolo. Socio-spatial properties of online location-based social networks. In *ICWSM'11*, 2011.
9. A. Y. Xue, R. Zhang, Y. Zheng, X. Xie, J. Huang, and Z. Xu. Destination prediction by sub-trajectory synthesis and privacy protection against such prediction. In *ICDE'13*, 2013.
10. M. Ye, P. Yin, W.-C. Lee, and D. L. Lee. Exploiting geographical influence for collaborative point-of-interest recommendation. In *SIGIR'11*, pages 325–334, 2011.