FriendSeeker: Inferring Hidden Friendship in Mobile Social Networks with Sparse Check-in Data

Shan Chang*, Yuting Tao*, Hongzi Zhu†, and Bo Li‡
*School of Computer Science and Technology, Donghua University, Shanghai, China
†Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China
‡Department of Computer Science and Engineering, Hong Kong University of Science and Technology, HongKong, China

***Abstract*—Check-in data widely published in mobile social networks (MSNs) pose serious privacy threats to users. Existing inference attack methods show that pairwise social relationship could be estimated by analyzing check-in user records. However, the efficacy of such attacks heavily depends on the density of check-in data, which is often not present in practice. In this work, we propose a new inference attack scheme, which for the first time can effectively reveal hidden social friendship in both the real world and in cyberspace among users with only *sparse* check-in data. Our attack method enjoys two salient features. First, it requires no prior knowledge about social connections, instead it estimates users' social proximity by exploiting both physical presence and social proximities. Second, our attack scheme can automatically learn representative features based on the significance of various check-in records, rather than relying on heuristic features. We conduct extensive trace-driven simulations, and the results demonstrate that our inference attack method can improve the efficacy of the state-of- the-art learning-based schemes up to 40%. Moreover, our proposed attack method is also robust against common data obfuscation mechanisms.**

***Index Terms*—hidden friendship inference, sparse check-in data, spatial-temporal proximity feature, social proximity feature**

I. Introduction

We have have witnessed the blooming development of mobile social networks (MSNs) in past decade, such as Twitter, Facebook and WeChat, where people can easily make friends in the real world and in cyberspace, while sharing instant information at their fingertips. However, disseminating information on MSNs has raised serious privacy concerns. For instance, as reported in a recent study [1], the private profile of a user can be precisely inferred by analyzing eight of his/her contacts on an MSN platform. Moreover, many users are accustomed to publish their check-in records at point-of-interest (POI) locations ((*e.g.*, a nice restaurant, a fabulous shop, or a stunning scenery), which contain rich set of private spatial-temporal presence information about users. Nowadays, realizing the potential privacy leakage risks, more and more MSN users start to hide their friend lists [2]. However, little attention has been paid to protect check-in data, making it possible to infer the friendship among users by mining these check-in data. Such attacks, referred to as *friendship inference attacks*, can seriously violate the privacy of users, even impair the value of MSNs.

Launching a successful inference attack need to satisfy the following two requirements. First, the attack should accurately identify as many friends as possible, both in the real world and in cyberspace. It is intuitive to identify real-world friends by examining their check-in records as they tend to participate in common activities, *e.g.*, seeing a movie together or enjoying a party. In contrast, it is difficult to capture cyber friendship as they are usually strangers in the real world, and share no common physical activities and check-in records. Second, the attack must deal with a crucial problem - the sparsity of the check-in data. Though check-in data are public and generally accessible from an MSN, they are often sparse in terms of temporal and spatial distributions because a large portion of users are reluctant to update their states.

In general, existing friendship inference schemes can be roughly divided into two categories,*i.e.*, knowledge-based and learning-based. In knowledge-based schemes [3] [4] [5] [6] [7] [8] [9], for each pair of users, plenty of check-in records with the same places, referred to as co-locations, imply a potential real-world friendship between such pair of users. When check-in data are sparse, only a small number of co-locations can be discovered, making such attacks less effective. Noticeably, however, such schemes cannot capture cyber friendship. In learning-based schemes [10] [11] [12] [13], the correlation between the spatial-temporal proximity of a pair of friends and their friendship can be learnt with machine learning algorithms. On one hand, such schemes are capable of identifying both real-world and cyber friends. On the other, they are prone to find false positive real-world friends because nearby strangers (e.g., those living in the same neighborhood) often present similar spatial-temporal proximity. As a result, there is no effective scheme, to the best of of our knowledge, to tackle both real-world and cyber friendship inference with sparse check-in data.

In this paper, we propose a two-phase hidden friendship inference scheme, called *FriendSeeker*, which can effectively predict both real-world and cyber friendship using sparse check-in data. Based on our empirical study on two real-world MSN trace datasets, there are two observations: 1) real-world friends do share common physical POIs; 2) cyber friends have common social structure embedded in an MSN. Therefore, in the first phase, FriendSeeker aims to identify hidden pairwise real-world friends by utilizing effective spatial- temporal proximity features, extracted from the sparse check-in data. This allows us to establish an initial social graph with all identified pairwise real-world friends. In the second phase, new pairwise hidden cyber friends can be accurately inferred based on social proximity features, i.e., pairwise social graphs which indicate

how easy it is for two people to "being friends" through their social connections, extracted from the initial social graph. The second phase results in the update of the initial graph, which repeats until a stable social network with all predicted real-world and cyber friends are obtained.

There are two main challenges in designing FriendSeeker. First, it is difficult to accurately infer hidden real-world friends solely based on sparse check-in data. One intuitive method is to analyze the joint occurrences of a pair of users by casting check-in records of both users into a predefined voxel grid, referred to as *joint occurrence cuboid* (JOC), and train a machine learning model with JOC samples for friendship prediction. However, raw JOCs are sparse and noisy, and high-dimensional, which implies both high false negative and positive, as well as low inference efficiency, making them inferior features. To deal with this problem, we utilize an autoencoder to derive superior spatial-temporal proximity features, referred to as *presence proximity features*, thus suitable for friendship prediction. More specifically, we label a small number of raw JOC samples derived from check-in data with the ground truth of friendship, and use them to jointly train an autoencoder and a real-world friendship classifier in an end-to-end fashion. The temporal and spatial features which contribute more to correct classification (friend or not) will be learnt and retained in the encoder in a more compact way. After that, we apply the pre-trained encoder to extract presence proximity features from raw testing JOCs for real- world friendship inference.

Second, it is challenging to predict hidden friends on a given social network with sparse links. Conventional heuristic features, such as common neighbors (*e.g.*, more common neighbors implies higher probability of a pair of nodes being friends) and Katz centrality [14] [15] [16] (*e.g.*, more paths between a pair of users indicates higher social proximity between them) can be used for link prediction in social networks. These features are inaccurate for link prediction when the initial social network is sparse in terms of the number of links. To tackle this problem, we introduce a new social proximity feature, called *k-hop reachable subgraph*, which refers to the subgraph constituted by all paths within k hops between a pair of users. As long paths may not derive meaningful friendship, the intuition of this social proximity feature is to increase the number of paths between a pair of users for a better proximity presentation, and to reduce the number of long paths at the same time by selecting a proper value of k. Moreover, we embed k-hop reachable subgraphs into social proximity features vectors, and combine them with corresponding presence proximity features to train a hidden friendship classifier for new friendship prediction. Since the integrated two-factor features balance the importance of presence and social proximities, those misidentified social links (close-range strangers) will be pruned from the initial graph, while at the same time hidden friends will be revealed and added to the graph, leading to a social graph approaching to the truth.

We conduct extensive trace-driven simulations on two large-scale MSN trace datasets collected from Gowalla and Brightkite. The results indicate that FriendSeeker can quickly converge by reaching a stable social graph after only 5 or 6 iterations. FriendSeeker outperforms the state-of-the-art knowledge-based and learning-based methods by 46.9%, and 16%, respectively. The results also demonstrate that FriendSeeker can effectively handle sparse check-in data by discovering 29.6% friends with less than 25 check-in records. In addition, it can identify 68.13% friends sharing no common locations. We further apply two commonly used spatial and temporal obfuscation mechanisms, i.e., concealing partial check-in records and blurring check-in records of users, to defend against FriendSeeker. Experiment results show that our attack can still achieve superior performance.

The main contributions are summarized below: 1) we design an effective spatial-temporal proximity feature extraction method using the autoencoder structure based on sparse check-in data; 2) we propose a novel social proximity feature, called k-hop reachable subgraph, for social link prediction; 3) we devise a two-phase friendship inference attack method and its efficacy verified on real-world traces.

II. Problem Definition and Attack Model

A. Problem Definition

We define a trajectory of a user as a sequence of timestamped check-ins, each of which represents the user visits a Point Of Interest (POI, e.g., a shopping store, a drinking bar, etc.). We formally define the notion of POI, Check-in and Trajectory as follows.

Definition 1. *(POI) A POI p is an exact place, indicated by a triple (lng, lat, rad), where lng and lat are the longitude and latitude of the geographical center of p, and rad is the radius of p, implying geographical coverage.*

We denote $\mathbb{U}$, $\mathbb{P}$, and $\mathbb{T}$ as the set of users, the set of POIs and the time domain, respectively.

Definition 2. *(Check-in) A check-in c is a triple $(u, p, t) \in \mathbb{U} \times \mathbb{P} \times \mathbb{T}$, which implies a user u has visited p at time t.*

We emphasize that geographical coordinate-based check-ins can be easily converted into such POI-based check-ins with the knowledge of $\mathbb{P}$.

Definition 3. *(Trajectory) The trajectory of a user u_a is the set $\mathcal{T}_{u_a} = \{(u_a, p, t) \in \mathcal{T}\}$, where $\mathcal{T}$ is a collection of check-ins.*

Definition 4. *(Co-location) A co-location of two users u_a and u_b to a location p is defined as an event that u_a and u_b report two check-ins (u_a, p, t_a) and (u_b, p, t_b), respectively.*

Definition 5. *(Social Graph) A Social Graph is defined as an undirected graph $\mathbb{G} = (\mathbb{U}, \mathbb{E})$, where $\mathbb{U}$ is the set of vertices, each representing a user, and $\mathbb{E}$ is the set of edges between vertices. $e_{(a,b)} \in \mathbb{E}$ represents the friendship between users u_a and u_b.*

We explain that the above definition implicitly assumes the friendship is symmetric, i.e., if u_a is a friend of u_b, then u_b is also a friend of u_a. Although, the connection between users can be unidirectional in some social networks, i.e., one user is the follower and the other being the followee, we consider the

bi-directional relationship since it is common on most social networks, for example Facebook.

Definition 6. *(Induced Path) Given a graph $\mathbb{G}$ and a path $\mathcal{P}$ on it, then $\mathcal{P}$ is called an induced path if there are no additional edges in the subgraph induced by the vertex set of $\mathcal{P}$, i.e., each two adjacent vertices on $\mathcal{P}$ are connected by an edge in $\mathbb{G}$, and each two nonadjacent vertices on $\mathcal{P}$ are not connected by any edge in $\mathbb{G}$.*

Based on the above consensus, we define our problem as follows:

Definition 7. *(Friendship Inference Problem) Given a group of users $\mathbb{U}$ and their trajectories $\mathcal{T}_{\mathbb{U}}$, for each pair of users(u_a, u_b) in $\mathbb{U}$, determine whether they are friends or not, i.e.,$f(u_a, u_b) = 0/1$*

B. Attack Model

An attacker launching a friendship inference attack attempts to unveil friendship among anonymized users. The objective of the attacker is to identify as many anonymized friends as possible, utilizing check-in records. To be precise, in the proposed attack, the attacker predicts friendship through the following steps:

1) The attacker is able to access a small anonymized dataset of check-in trajectories with friendship labels, which acts as our training dataset. There are various ways to obtain such a dataset, for example, learning from those public available datasets from MSNs.

2) The attacker is able to obtain another targeted dataset of check-in trajectories (without friendship labels). It should be noticed that the users in the training dataset are not necessary to overlap with that of our targeted testing dataset.

3) The attacker carries out the strategy of friendship inference such that friends in the targeted dataset can be identified.

C. Empirical Data Analysis

Datasets. We use two real-world datasets from Gowalla and Brightkite. Each dataset contains both user check-in records and the social graph of users. We use the check-in records to infer friendship among users, and the provided social graph as the ground truth to evaluate the performance of our attack. Specifically, in Gowalla dataset, there are 36,001,957 check-in records obtained from 319,063 users during the period from March 21, 2009 to November 2, 2011, and there are 407,533 nodes and 2,209,170 edges in the corresponding social graph. In Brightkite dataset, we have 4,747,287 check-in records collected from 51,406 users during the period from April 18, 2008 to October 21, 2010, and we have 58,228 nodes and 214,078 edges in the corresponding social graph. Each check-in is organized in the following form:

<user-ID, time, latitude, longitude, location-ID>.

We exclude users who never check in or only check in once from our experiments. Table I shows the basic statistics of the selected datasets.

TABLE I: Statistics of two real-world MSN trace datasets.

Dataset	# POIs	# Users	# Check-ins	# Links
Brightkite	157,279	14,897	1,360,524	93,754
Gowalla	104,568	12,439	656,642	51,270

Statistics and Observations. There exist two types of friendship online: friends in reality (for example, families or schoolmates), and likeminded peoples who might be strangers in the real word, i.e., cyber friends. In the former case, online friends may exhibit similar mobilities, while in the latter case, they may show distinctive mobility patterns, however similar social tie structures, e.g., sharing many common friends. For example, in Gowalla, 27.71% friends have no common location however at least one common friend. We make a study on the percentages of friends and non-friends who have co-locations and common friends, respectively. Statistical results are listed in Table II. For example, we can learn from it that 13.01% friends share no co-locations however have co-friends in Gowalla. Thus, both the presence and social proximity among users provides us opportunities to identify online friendship.

We further count the number of co-locations and common friends owned by friends and non-friends on the both datasets. Fig. 1 shows the statistical results. The CDFs of the numbers of friends and non-friends sharing different numbers of common POIs and friends are illustrated in Fig. 1(a) and (b), respectively. We can conclude that there are indeed differences between CDFs belonging to friends and non-friends. For example, when a pair of users share more than *ten* co-locations, it is almost certainly that the two users are friends, and about 92% non-friend pairs have no common friends, while only 20% friends share no friends. However, directly using co-locations and common friends as indicators is not enough to distinguish between friends and non-friends, since a large proportion of pairs have no co-locations no matter friends or not, and non-friends may also share many friends. For example, it can be seen Fig. 1(a) that 97% non-friends and 71% friends have never share a common location.

TABLE II: The proportion of users to whether they have co-friends (C-F) and co-locations (C-L) of Gowalla (above), Brightkite (below).

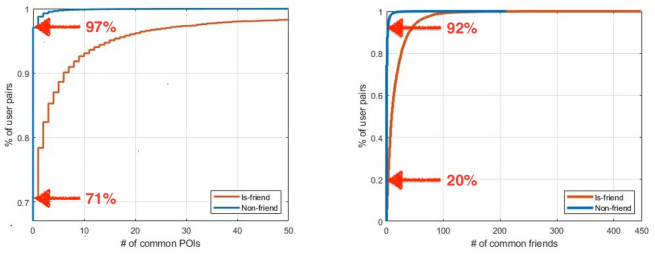
C-F \ C-L	Yes		No	
	Friends	Non-friends	Friends	Non-friends
Yes	52.49%	1.67%	13.01%	13.05%
No	27.71%	3.93%	6.79%	81.35%

C-F \ C-L	Yes		No	
	Friends	Non-friends	Friends	Non-friends
Yes	79.05%	1.08%	4.24%	10.83%
No	9.09%	3.93%	29.17%	55.76%

III. Design of FriendSeeker

A. Overview

In general, FriendSeeker consists of two phases: *real-world friends inference phase* and *iterative hidden friends inference*



(a) common POIs (b) common friends

Fig. 1: CDFs of the numbers of friends and non-friends sharing different numbers of common POIs and friends.

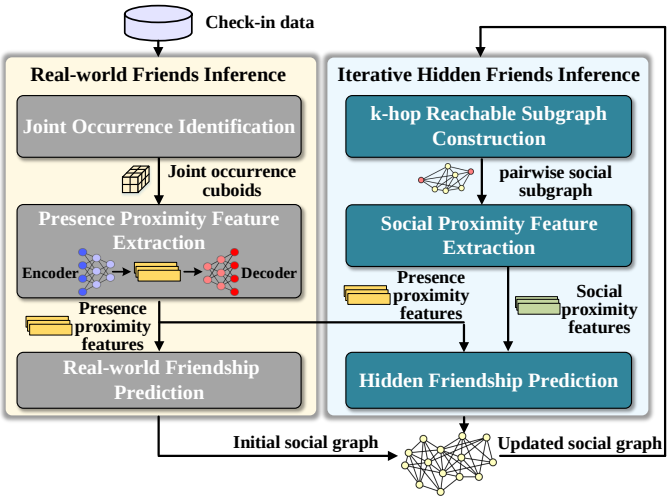


Fig. 2: Architecture of FriendSeeker, consisting of two phases, *i.e.*, Real-world Friends Inference and Iterative Hidden Friends Inference.

phase as shown in Fig. 2. In the real-world friends inference phase, pairwise check-in trajectories are first captured by *joint occurrence identification* (JOI) which outputs joint occurrence cuboids. Second, predictive power of occurrences in cuboids are modeled by latent variables in *presence proximity feature generation*, outputting compressed feature vectors, which are used to train a *real-world friendship prediction model*, outputting a social graph of physical friends. In the iterative hidden friends inference phase, the *k-hop reachable subgraph construction* is applied to each pair of people on the resulted social graph of physical friends, generating pairwise subgraphs. *Social proximity feature extraction* encodes those subgraph as feature vectors, which are combined with their corresponding presence proximity features to train a *hidden-friendship prediction model*. The prediction results lead to a more accurate social graph, containing cyber-space friends and avoiding mis-identified real-world friends.

B. Real-world Friends Inference

The objective of this stage is to learn compressed representations of pairwise spatial-temporal proximity features, and to fully exploit them for real-world friends inference, simultaneously.

1) Identifying Joint Occurrences: we first construct a spatial-temporal division defined formally as follows:

Definition 8. *(Spatial-Temporal Division) We divide a geographical region of interest (represented as ranges of latitude/longitude) into I grids, and a time interval concerned into J slots. The resulting cube is a spatial-temporal division (STD) of size $I \times J$. A cell, denoted as $c_i^{(j)}$ $(1 \leq i \leq I, 1 \leq j \leq J)$, refers the i-th grid, and the j-th time slot, respectively. An STD cell is the finest granularity we use for extracting presence proximity features.*

In FriendSeeker, time domain is partitioned into equal slots of length τ. One simple division of space is to uniformly partition the space into equal size grids, which is however inflexible and inefficient due to that the density of POIs varies greatly across geographic area. For example, in downtown area, the POI density is much higher compared to a countryside area. To satisfying different grids contain a similar number of POIs, we recursively divide a region of interest into four equal grids, until the number of POIs in each grid is smaller than an empirical threshold σ.

Definition 9. *(Joint Occurrence Cuboid: JOC) Let $\mathcal{T}_{u_a}$ and $\mathcal{T}_{u_b}$ be two trajectories of users u_a and u_b, respectively. Given an STD of size $I \times J \times M$, we cast $\mathcal{T}_{u_a}$ and $\mathcal{T}_{u_b}$ into it simultaneously, and each check-in falls in one certain cell $c_i^{(j)}$. For each cell, we calculate three indicators: the total number of check-ins of u_a and u_b, denoted by n_a and n_b and the number of POIs visited by both u_a and u_b, denoted by $n_{a,b}$. Let $o_{i,j}^{(u_a,u_b)} = (n_a, n_b, n_{a,b})$ belong to $c_i^{(j)}$. Then the corresponding joint occurrence cuboid of the user pair (u_a, u_b) is an $I \times J$ cuboid, where each cell is $o_{i,j}^{(u_a,u_b)}$, i.e., $O_{(a,b)} = \{o_{i,j}^{(u_a,u_b)} | 1 \leq i \leq I, 1 \leq j \leq J\}$.*

The JOC is designed to capture presence proximities among user pairs within each cell. Fig. 3 illustrates a JOC generated from two trajectories. The size of STD used to build JOCs can be adjusted through turning parameters σ and τ. For a large and diverse set of trajectories, large σ and τ can be used. The two parameters should be decreased for a small trajectory set, avoiding over-fitting. Appropriate values of σ and τ can be selected empirically according to experimental results on the specific dataset.

2) Presence Proximity Feature Extraction: Noticing that the JOCs are usually highly sparse, we feed the JOCs into an autoencoder, denoted as $\mathbb{A}$, to compress each of them into a dimensional vector representation. An autoencoder consists of two components: an encoder and a decoder. The objective of an encoder is to compress the JOCs, such that the compressed representations can reconstruct the original inputs with a significantly small error by the decoder [17]–[20].

Given an input $O_{(a,b)}$, autoencoder first encodes it to an encoding network with $\mathcal{R}$ hidden layers (i.e., the encoder) through $\mathcal{R}$ encoding processes, then the output of the encoder is passed through the decoder (usually the same network structure as encoder but in opposite orientation) to obtain an output $O'_{(a,b)}$, giving the best closest match to $O_{(a,b)}$. The

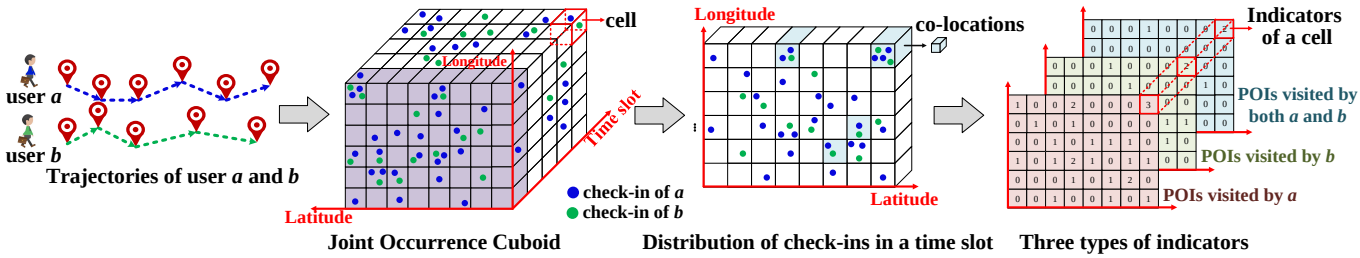


Fig. 3: An example of constructing the JOC relevant to users a and b.

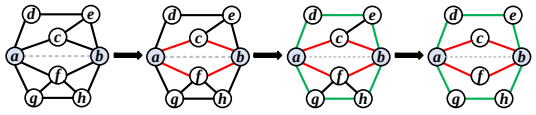


Fig. 4: An example of constructing a 3-hop reachable subgraph $\mathcal{G}^3_{(a,b)}$ between a and b.

output of the γ-th hidden layer of the autoencoder can be summarized as:

$$h^{(r)}_{(a,b)} = \varpi\left(W^{(r)}_{\mathbb{A}} h^{(r-1)}_{(a,b)} + b^{(r)}_{\mathbb{A}}\right), r = 1, ..., 2\mathcal{R}$$

where ϖ a non-linear activation function,$W^{(r)}_{\mathbb{A}}$ and $b^{(r)}_{\mathbb{A}}$are weight matrix and bias vector of theare weight matrix and bias vector of the γ-th hidden layer. Notice that, $h^{(0)}_{(a,b)}$ is the input $O_{(a,b)}$, the output of the encoder and decoder are $h^{(\mathcal{R})}_{(a,b)}$, and $h^{(2\mathcal{R})}_{(a,b)}$ (also denoted as $\hat{O}_{(a,b)}$), respectively. The final layer of the encoder has the smallest number, denoted as d, of hidden units, and $h^{(\mathcal{R})}_{(a,b)}$ is the d-dimensional compressed representation of $O_{(a,b)}$. The number of hidden units on the final layer of the decoder equals to the size of the input, such that $\hat{O}_{(a,b)}$ is the reconstructed input.

The encoder and decoder are trained together. The training objective is to learn the weight matrices $W^{(r)}_{\mathbb{A}}$ and bias vectors $b^{(r)}_{\mathbb{A}}$ by minimizing the reconstruction error $\mathcal{L}_{auto}$ as follows:

$$\mathcal{L}_{auto} = \sum_{u_a, u_b \in \mathbb{U}} \left\| \hat{O}_{(a,b)} - O_{(a,b)} \right\|^2_2.$$

$W^{(r)}_{\mathbb{A}}$ and $b^{(r)}_{\mathbb{A}}$ denote the latent variables, measuring the significance of each cell in the input JOC, i.e., $o^{(u_a,u_b)}_{i,j}$. The value of d also impacts the performance of our attack, and we empirically decide the appropriate value of it in our experiments, by turning the number of units on hidden layers.

3) Real-world friendship Prediction: the training of autoencoder is unsupervised, which means it only learns the input structure, and the learned model is blind to the nature of the supervised tasks. Thus, the compressed representations extracted by hidden layers only retain information of the original JOCs, but provide no guarantees to be useful in classification tasks. To circumvent this issue, we leverage friendship labels in the training dataset to extract reconstructive and discriminative representations. More specifically, we add a classification network, i.e., real-world prediction model, to supervise the autoencoder. The compressed representations generated by the encoder, i.e., $h^{(\mathcal{R})}_{(a,b)}$, are fed into a classifier $\mathbb{C}$, and the prediction results will be compared with the labels. The training objective is to learn the weight matrices and bias vectors ($W^{(.)}_{\mathbb{C}}$ and $b^{(.)}_{\mathbb{C}}$) by minimizing the the cross-entropy loss, shown as follows:

$$\mathcal{L}_{cla} = -\frac{1}{\mathcal{N}} \sum_{\mathcal{N}} (y \log \hat{y} + (1-y) \log(1-\hat{y}))$$

where y and $\hat{y}$ are the labels and the outputs of the classifier $\mathbb{C}$ on compressed representations, respectively, and $\mathcal{N}$ is the number of JOCs in the training dataset. Notice that the addition of the supervised loss to the autoencoder better directs $h^{(R)}_{(a,b)}$ towards effective for friendship prediction.

Aiming at obtaining more discriminative representations, we form a unified loss function for training the autoencoder, which combines the reconstruction and classification errors, and we introduce a weight α to balance retraining underlying structure, as well as providing accurate classification performance, shown as follows:

$$\mathcal{L} = \mathcal{L}_{auto} + \alpha \mathcal{L}_{cla}$$

To minimize $\mathcal{L}$ with respect to $W^{(.)}_{\mathbb{A}}$, $W^{(.)}_{\mathbb{C}}$, and $b^{(.)}_{\mathbb{A}}$, $b^{(.)}_{\mathbb{C}}$,we adopt the gradient descent method to derive the solution. The autoencoder and the classifier are trained simultaneously. The details of the proposed training algorithm are summarized in Algorithm 1. Once training is terminated, the encoder is taken out from the autoencoder network and will be used for extracting reconstructive and discriminative representations from input, and the classifier will be used for friendship prediction.

In this way, given a set of trajectories $\mathcal{T}_{\mathbb{U}}$, for each pair of users (u_a, u_b) in $\mathbb{U}$, we can obtain a compressed representation of the corresponding JOC, i.e., $h^{(\mathcal{R})}_{(a,b)}$ from the encoder, and a prediction $f(u_a, u_b)$ about their friendship from the classifier $\mathbb{C}$. Consequently, an initial social graph of $\mathbb{G}^{(0)} = (\mathbb{U}, \mathbb{E}^{(0)})$ can be derived. If users u_a and u_b are predicted as friends, there exists an edge between u_a and u_b, $e^{(0)}_{(a,b)} \in \mathbb{E}^{(0)}$.

C. Iterative Hidden Friends Inference

This stage aims at correctly identifying hidden friends whose presence proximities provide no evidence of being friends.

Algorithm 1 Autoencoder and Classifier Training

Input: a set of JOCs relevant to $\mathcal{N}$ user pairs, denoted by $\mathbb{O} = [O_1; ...; O_{\mathcal{N}}]$, and the corresponding friendship labels, i.e., $y_i \in \{0, 1\}$, learning rate β, the architectures of an autoencoder $\mathbb{A}$ and a classifier $\mathbb{C}$, where the numbers of hidden layers of $\mathbb{A}$ and $\mathbb{C}$ are $2\mathcal{R}$ and $\mathcal{H}$, respectively, and the dimension of layer $\mathcal{R}$ in $\mathbb{A}$ is d, and the balance weight α between $\mathbb{A}$ and $\mathbb{C}$.

Output: well trained $\mathbb{A}$ and $\mathbb{C}$, JOC embeddings, i.e., the output of layer $\mathcal{R}$ on $\mathbb{A}$, denoted by $h^{(\mathcal{R})}$.

/* Apply gradient descent to train the network: */
Randomly initialize the weight matrix $W_{\mathbb{A}}$ and bias vector $b_{\mathbb{A}}$ of $\mathbb{A}$ together with the weight matrix $W_{\mathbb{C}}$ and bias vector $b_{\mathbb{C}}$ of $\mathbb{C}$.
for epoch $= 1$ to m **do**
  for each batch in $\mathbb{O}$ **do**
    /* batch size is n. */
    $\mathcal{L}_{auto} \leftarrow -\frac{1}{n}\sum_{i=1}^{n}\left(\hat{O}_i - O_i\right)^2$;
    /* $\hat{O}_i$: the reconstructed JOCs, i.e., the output of $\mathbb{A}$. */
    $\mathcal{L}_{cla} \leftarrow -\frac{1}{n}\sum_{i=1}^{n}\left(y_i \log \hat{y}_i + (1 - y_i)\log\left(1 - \hat{y}_i\right)\right)$;
    /* $\hat{y}_i \in \{0, 1\}$: the prediction result of $\mathbb{C}$. */
    for $r = 2\mathcal{R}$ to 1 **do**
      $W_{\mathbb{A}}^{(r)} \leftarrow W_{\mathbb{A}}^{(r)} - \beta\frac{\partial}{\partial W_{\mathbb{A}}^{(r)}}\mathcal{L}_{auto}$;
      $b_{\mathbb{A}}^{(r)} \leftarrow b_{\mathbb{A}}^{(r)} - \beta\frac{\partial}{\partial b_{\mathbb{A}}^{(r)}}\mathcal{L}_{auto}$;
    end for
    for $r = \mathcal{H}$ to 1 **do**
      $W_{\mathbb{C}}^{(r)} \leftarrow W_{\mathbb{C}}^{(r)} - \beta\frac{\partial}{\partial W_{\mathbb{C}}^{(r)}}\mathcal{L}_{cla}$;
      $b_{\mathbb{C}}^{(r)} \leftarrow b_{\mathbb{C}}^{(r)} - \beta\frac{\partial}{\partial b_{\mathbb{C}}^{(r)}}\mathcal{L}_{cla}$;
    end for
    for $r = \mathcal{R}$ to 1 **do**
      $W_{\mathbb{A}}^{(r)} \leftarrow W_{\mathbb{A}}^{(r)} - \alpha \cdot \beta\frac{\partial}{\partial W_{\mathbb{A}}^{(r)}}\mathcal{L}_{cla}$;
      $b_{\mathbb{A}}^{(r)} \leftarrow b_{\mathbb{A}}^{(r)} - \alpha \cdot \beta\frac{\partial}{\partial b_{\mathbb{A}}^{(r)}}\mathcal{L}_{cla}$;
    end for
  end for
end for
return $W_{\mathbb{A}}$, $b_{\mathbb{A}}$, $W_{\mathbb{C}}$, $b_{\mathbb{C}}$.

1) k-hop Reachable Subgraph Construction: we aim to extract graph structure features from $\mathbb{G}^{(0)}$, which reflect social proximities (connections) among users, and utilize those features to re-estimate friendship. The basic idea is two users who are close enough but not yet connected on the initial social graph $\mathbb{G}^{(0)}$ may have a higher likelihood of being friends.

Thus, we adopt a local eyeshot on the network proximity, and extract a ***k-hop reachable*** subgraph for each user pair, which describes the pairwise "k-hop achievability" between users. For a given graph $\mathbb{G} = (U, E)$, we describe the procedure for extracting the k-hop reachable subgraph between u_a and u_b , denoted by $\mathcal{G}^k_{(a,b)}$, in the following steps:

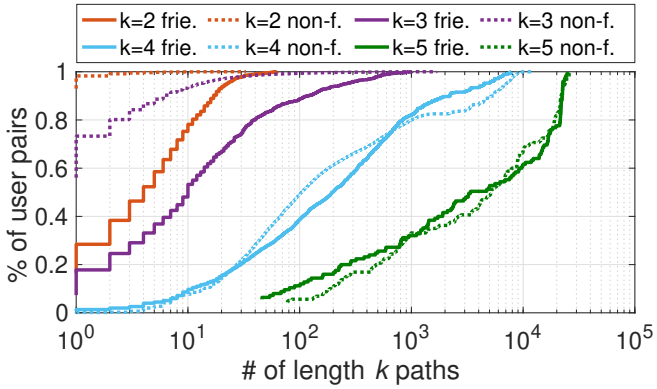


Fig. 5: The CDFs of the number of friends and non-friends with different numbers of k length paths

Step 1: set path length (i.e., the number of edges contained) l as 2, and initialize $\mathcal{G}^k_{(a,b)}$ as an empty graph;

Step 2: find all length l paths between (u_a, u_b) in $\mathbb{G}$, and include all paths found into $\mathcal{G}^k_{(a,b)}$, then update $\mathbb{G}$ by excluding all nodes and edges in $\mathcal{G}^k_{(a,b)}$ from it;

Step 3: increase path length l by one and repeat step 2 until path length exceeds k.

Theorem 1. Consider $\mathcal{G}^k_{(a,b)}$ is the k-reachable subgraph between user pair $(u_a.u_b)$, it satisfies:

1. All paths in $\mathcal{G}^k_{(a,b)}$ are induced paths of length l $(2 \leq l \leq k)$;

2. Paths with different lengths share no common edges.

Proof. **1.** If a non-induced path $\mathcal{P}$ belongs to $\mathcal{G}^k_{(a,b)}$, represented as a sequence of vertices $(..., u_i, u_{i+1}, ...)$, there must exist two nonadjacent vertices u_i and u_j $(\|j - i\| > 1)$ on $\mathcal{P}$ which are connected by an extra edge outside the path. It means that we can find a shorter path $\mathcal{P}'$ by deleting the subsequence of vertices $(u_{i+1}, ..., u_{j-1})$ from $\mathcal{P}$, and include $\mathcal{P}'$ in $\mathcal{G}^k_{(a,b)}$ before $\mathcal{P}$. Thus, all vertices in $\mathcal{P}'$ will be excluded from the following procedure. Therefore, $\mathcal{P}$ must not be in $\mathcal{G}^k_{(a,b)}$. **2.** If two paths $\mathcal{P}$ and $\mathcal{P}'$ are with different lengths, then they are included in $\mathcal{G}^k_{(a,b)}$ in different turns, thus the vertices and edges of them are mutually exclusive. □

For any edge e in $\mathcal{G}^k_{(a,b)}$, if there exist multiple paths on $\mathbb{G}$ whose lengths are smaller than k containing it, only the shortest path is included in $\mathcal{G}^k_{(a,b)}$. For example, in Fig. 4, paths $a-c-e-b$, $a-f-h-b$ and $a-f-g-h-b$ will be removed during constructing $\mathcal{G}^3_{(a,b)}$. Above properties bring two advantages. First, give priority to short paths, considering short paths are more informative. Second, any edge will not be reused in extracting structure features with different orders, since an edge will not occur on two paths of different lengths.

Selecting appropriate k is important, one may think a large k implying more topological information will lead to a better feature. Surprisingly, we find that 3 is the optimum value of

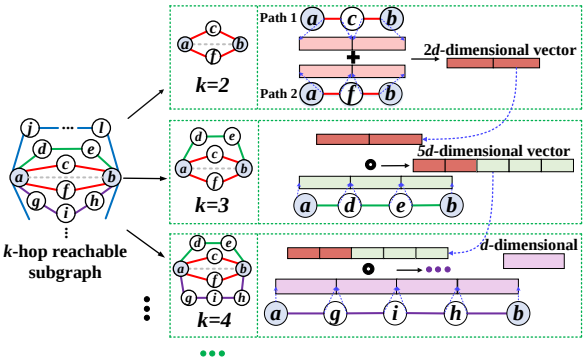


Fig. 6: An example of extracting social proximity feature vectors from k-hop reachable subgraphs.

k according to our analysis on real datasets. Fig. 5 depicts the CDFs of the number of friends and non-friends with different numbers of k length paths, it can be seen when k is larger than 3, the number of paths between friends and non-friends exhibits no obvious differences. The reason is that human societies are small-world-type networks characterized by short path-lengths [21]. It implies even strangers can be linked by a short chain of acquaintances. Thus, only those really short paths are meaningful of supporting friendship between user pairs.

2) Refining Social Graph with Social Proximity Features: Given the graph $\mathbb{G}^{(0)}$, we perform the following two procedures to train a classifier $\mathbb{C}'$ to decide whether two users are friends or not:

- *Social Proximity Feature Extraction:* first, for each pair of users u_a and u_b in $\mathbb{U}$, extract the k-reachable subgraph $\mathcal{G}^{(0),k}_{(a,b)}$ of u_a and u_b. Then, extract the social proximity features $s^{(0)}_{(a,b)}$ by utilizing both $\mathcal{G}^{(0),k}_{(a,b)}$, and the compressed spatial-temporal feature vector $h^{(\mathcal{R})}_{(i,j)}$ of each edge $e^{(0)}_{(i,j)}$ in $\mathcal{G}^{(0),k}_{(a,b)}$ obtained in stage 1. Specifically, we first add vectors of same length paths, and then concatenate resulted vectors of different lengths. We emphasize that we don't necessarily use learning-based methods such as graph embedding, to extract social proximity features, because in a k-hop reachable subgraph between u_a and u_b, the degree of any vertex is 2 (excepting u_a and u_b) and k is set as 3 in practice, which imply the subgraphs are simple and small. Fig. 6 illustrates the extraction of social proximity features from k-hop reachable subgraph.
- *Hidden Friendship Prediction:* first, concatenate the presence and social proximity features, i.e., $h^{(\mathcal{R})}_{(a,b)}$ and $s^{(0)}_{(a,b)}$, to obtain the composite feature vector $v^{(0)}_{(a,b)}$. Then, use each $v^{(0)}_{(a,b)}$ and the corresponding label to learn $\mathbb{C}'$. After finishing training $\mathbb{C}'$, according to the decision of $\mathbb{C}'$, derive a new social graph $\mathbb{G}^{(1)}$.

The above processes are repeated until the difference between $\mathbb{G}^{(i)}$ and $\mathbb{G}^{(i-1)}$ is small enough. Then $\mathbb{C}'$ is considered a well-trained model.

In procedure of testing, the same procedures are performed iteratively, except using $\mathbb{C}'$ directly without training, and thus the final social graph $\mathbb{G}^{(i)}$ is obtained when the stopping criterion is reached.

IV. Evaluations

A. Experimental Setup

We use the datasets described in Subsection II-C to conduct experiments. We use 70% and 30% data to train and to test.

Baseline models. We compare our method with the following methods which are proposed in related literature inferring social ties by mobility data.

- **Co-location based**. H. P. Hsieh *et al.* extract heuristic features about co-locations and build a co-location graph, to capture both direct and indirect social linkages among users [22].
- **Distance based**. We calculate the center location of a user based on the check-in frequencies of POIs [12], and use the Euclidean distance among users to identify social ties.
- **Walk2friend**. Walk2friend [10] applies random-walk-based graph embedding on a user-location bipartite graph to infer social ties.
- **User graph embedding**. Y. Yu *et al.* [11] adopt graph embedding method on a user graph (edges represent meeting frequencies among users) with additional POI category information.

Metric. We use the F-measure accuracy (F1-Score) as the evaluation metric, which is non-sensitive to class distribution and can avoid misleading accuracy measurement when the true class distribution is unbalanced.

B. Parameter Sensitivity

In our experiments, we adopt fully connected topology of autoencoder networks. In the structure of the encoder, we set consecutive layers with half the number of nodes as in the preceding layer, excluding the last layer (which is set according to the dimension of the spatial-temporal proximity feature d). On the other hand, in the structure of a decoder, we use the same network structure as encoder but in opposite orientation. According to the size of the joint occurrence Cuboid fed into the encoder, we adjust the number of layers in it. We use a simple KNN and SVM as the classifier $\mathbb{C}$ and $\mathbb{C}'$, respectively. We use RBF as the kernel function of the SVM. The learning rate of all networks is set as 0.005, and the weight α which balances the loss of the autoencoder and the classifier $\mathbb{C}$ is set as 1.

We emphasize that the purpose of this work is to investigate the feasibility of our friendship inference attack rather than to design better network structures. For this reason, we select the simplest autoencoder and supervised classifiers. We claim that the proposed approach is independent from the type of autoencoder and classifiers used and can be applied to other more complex variations of autoencoders.

We examine how the different choices of the three hyper-parameters., the number of POIs in a grid σ, length of time slot τ, and the dimension of spatial-temporal proximity feature

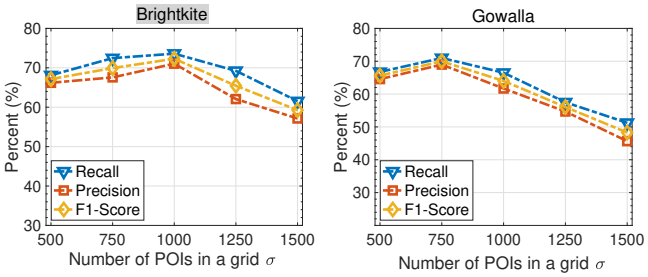


Fig. 7: Attack performances with respect to the maximum number of POIs in a grid.

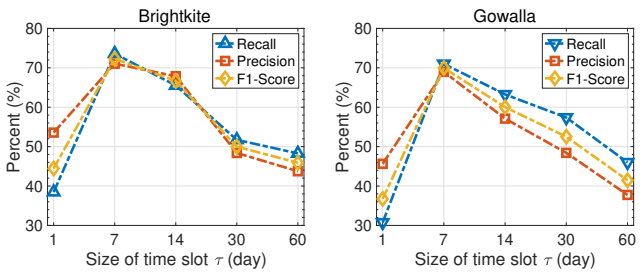


Fig. 8: Attack performances with respect to the size of time slot.

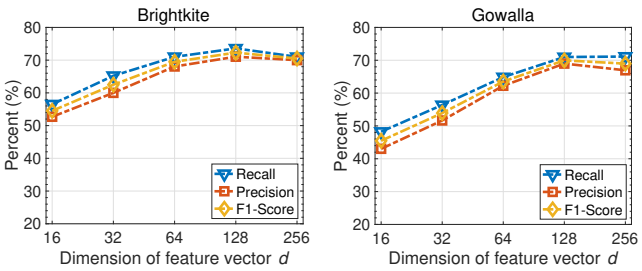


Fig. 9: Attack performances with respect to the dimension of presence proximity feature vectors.

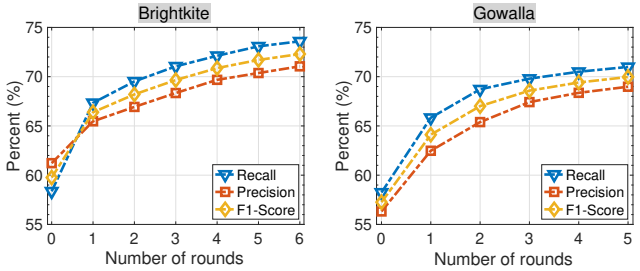


Fig. 10: Attack performances with respect to the number of iterations.

vector d, affect our attack performance. When testing each parameter, the two remaining ones are kept to their default settings, i.e., τ=7 (days) , d = 128, and σ=1000. The iteration of social graph refinement will be terminated if the number of edges changed in a new graph is less than 1% compared with the last graph.

First, we vary σ from 500 to 1,500 with an interval of 250. Fig. 7 plots the F1-Score, recall and precision as functions of σ for both datasets. It can be seen from Fig. 7 that, for the Brightkite dataset, the accuracy achieves a 4.56% gain when σ increases from 500 to 1,000 and then declines gradually with the growth of σ. Similar trend can also be seen for the Gawalla dataset as shown in Fig. 7 except that the maximum appears when σ is equal to 750. The reason is that POIs in the Gawalla dataset are more dispersed than in the Brightkite dataset, which implies that the geographical area of one grid in Gawalla is larger than in Brightkite. As a result, the best value of σ in Gawalla is smaller than Brightkite, which avoids distant check-ins falling into in a large grid.

Second, we vary τ from one day to 60 days with an interval of 7 days. Fig. 8 plots the F1-Score, recall and precision as functions of τ for both datasets. F1-Score reaches the peak in both datasets when τ is set as 7 days. It is very reasonable, since the activities of human being tend to show periodicity on a weekly basis.

In general, the parameters of σ and τ have an effect on the spatial-temporal matrix of user pairs, and further influence the eigenvector, i.e., spatial-temporal vector. Intuitively, the undersized or oversized values of σ and τ would result in poor performance of the attack because large values may inject much noise into spatial-temporal vectors, and small values might lose many useful features in vector. By comparing Fig. 7 and Fig. 8, it can be seen that the parameter τ has a greater impact on the F1-Score, recall and precision, than parameter σ does.

Third, we further examine the impact of the dimension of spatial-temporal proximity feature vector d to the attack performance. We double increase d from 16 to 256. Fig. 9 plots F1-Score, recall and precision as functions of d for both datasets. It can be seen that in Fig. 9. On one hand, it is easy to understand that the higher the dimension of spatial-temporal relation vectors is, the more information such vectors can contain, leading to better attack performance. On the other hand, higher dimension spatial-temporal proximity vectors also cause too much noise, which degrades the accuracy of an attack.

C. Social Relation Inference Results

According to above studies on parameters, we use the best value of each parameter, i.e., τ=7, d = 128, and σ =750 and 1000 in Gowalla and Brightkite, respectively. The iteration of social graph refinement will be terminated if the number of edges changed in a new graph is less than 1% compared with the last graph.

Fig. 10 depicts how the number of iterations affects the accuracy of our inference, in both datasets. It can be seen that iteration always improves the F1-Score, recall and precision. Moreover, we can see that the total number of iterations required to satisfy the termination criterion is 4 and 5, in Gowalla and Brightkite, respectively.

We compare our inference attack against all the baseline models, Fig. 11 shows that our attack outperforms all the baseline models significantly. For the best performing baseline model, i.e., embedding, we achieve a 5% performance gain

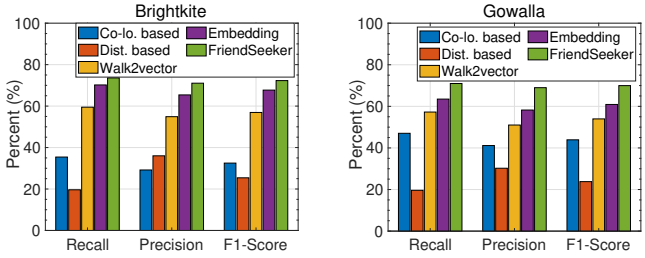


Fig. 11: Comparison of FriendSeeker against baseline models.

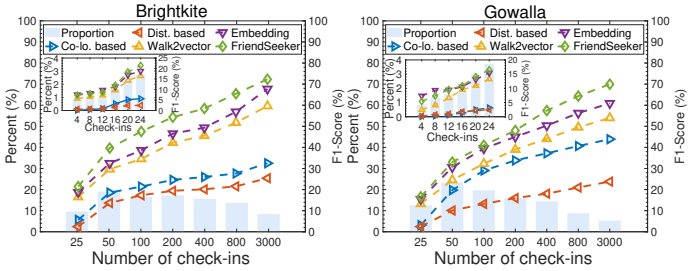


Fig. 12: The accuracy vs. different numbers of check-ins.

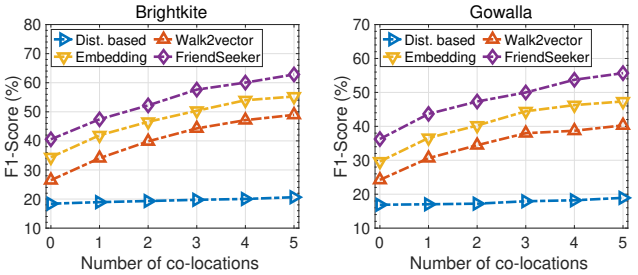


Fig. 13: The F1-Score vs. different numbers of co-locations.

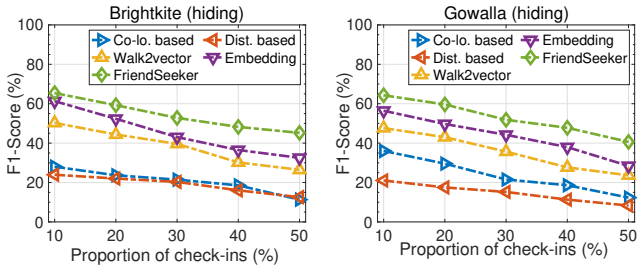


Fig. 14: The F1-Score vs. different proportions of hiding check-ins.

in Brightkite, and a 10% gain in Gowalla. This shows that our attack is more effective than the existing state-of-the-art attacks.

We compare our attack against baseline methods on pairs of users with less than five common locations. Fig. 12 plots F1-Score as a function of the number of common locations in both datasets. It can be seen that two learning-based baselines indeed achieve better accuracy compared with the knowledge-based method as reported in the original papers, but our attack still outperforms the best baseline model around 10% in both datasets. Notice that we cannot calculate F1-Score of co-location method, since the denominator in formula of F1-Score will be zero when the number of common locations is zero.

Fig. 13 illustrates the F1-Score of our attack against baseline methods as functions of the number of check-ins owned by a pair of users, and the probability distribution of the number of check ins. The purpose of this experiment is to explore whether our method is robust for users with different numbers of check-in records. Obviously, the more users check in, the more accurate the behavior pattern of users can be modeled. Although, all methods show poor performance if few check-ins are available, our method performs best regardless the numbers of check-ins.

D. Countermeasure Effects

1) Obfuscation Mechanisms: Adding perturbations on original data is the most common way to conceal location information. We apply the following two kinds of obfuscation mechanisms on both datasets, and launch our attack as well as baseline attacks to check if these attacks still work effectively.

- Hiding. In this mechanism, we randomly remove a certain proportion of check-in records. Consider that removing more than half data will significantly reduce the utility of mobility data, we vary the proportion of removed check-ins from 10% to 50% at in interval of 10%. To prevent check-ins of partial users from being removed totally (this could happen frequently, especially under a high hiding rate), before removing a check-in, we first check if this is the last check-in left over for its owner, if no, remove it, otherwise, skip this check-in.
- Blurring [23]. In this mechanism, we randomly select some check-ins, and replaces the locations with other locations. Again, we vary the proportion of replaced check-ins from 10% to 50% at in interval of 10%. Particularly, the blurring of locations can be divided into two cases: in-grid and cross-grids, which refer to that the location in a check-in data is randomly replaced with another location (POI) in the same grid, and in a different grid, respectively. To retain the utility of dataset as much as possible, in cross-grids blurring, first, we randomly select one of the four neighborhoods of the target grid, then randomly select another POI in the grid to replace the original one.

2) Attack Evaluation: We evaluate the impact of obfuscation mechanisms against FriendSeeker and all baseline methods. Fig. 14, 15 and 16 depict the F1-Score as a function of the proportion of perturbed check-ins. Experimental results show that, in all attacks, the inference accuracies decrease when increasing the proportion of perturbation. However, in all settings, FriendSeeker is robust to obfuscations compared with baseline methods. For example, in perturbed Brightkite, it can be seen that the F1-Score of FriendSeeker, embedding and walk2vector, decrease 20.9%, 28.78% and 23.8% respectively when concealing ratio of check-ins increases from 10% to 50%. Although, a high proportion of obfuscations will lead to a relatively low inference accuracy, for all settings, FriendSeeker

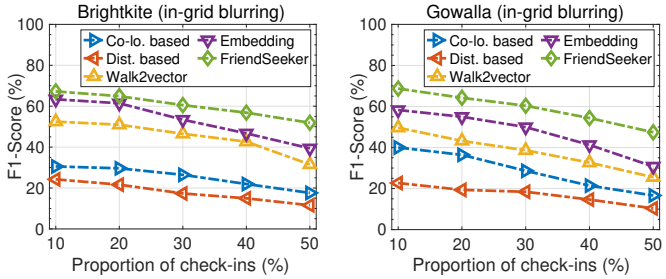


Fig. 15: The F1-Score vs. different proportions of in-grid blurring check-ins.

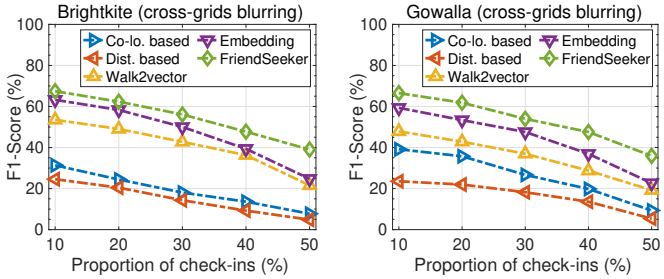


Fig. 16: The F1-Score vs. different proportions of cross-grid blurring check-ins.

always outperforms baseline methods. In addition, for all obfuscation mechanisms, the F1-Score of FriendSeeker remains around 40% even half check-ins are obfuscated.

On perspective of countermeasure efficacy, we can see the both obfuscation mechanisms can defend against two knowledge-based methods, i.e., co-location based and distance based, effectively. When half data are perturbed, the F1-Score of both methods decreases to around 10%. However, these obfuscation mechanisms show low performances on defending against the three learning-based attacks. It is worth mentioning that cross-grids blurring is more effective than hiding and in-grid blurring. This is because that, in cross-grids blurring, an POI will be replaced with another far distant POI, which however brings significance noise into original datasets.

V. Related Work

We divide the related literature into two categories based on their methodologies on feature extractions, *i.e.*, knowledge-based and learning-based methods.

A. Knowledge-based Methods

Li *et al.* predict friends by mining the geographical similarity among individuals, which considers the sequence of human activities and extracts geographical features hierarchically [3] . Xiao *et al.* calculate mobility similarity using semantic of locations [4]. Crandall *et al.* infer friendship based on geographical coincidences estimated by Bayesian model [24]. Wang *et al.* infer social ties by considering personal, global and temporal factors of mobilities [5]. Personal factor measures the significance of co-locations for each individual. They assume that if two people meet at locations which they seldom visit, they are more likely to be friends. Global factor concentrates on the popularity of locations measured by location entropy. The authors of reference [25] take meeting frequency, location popularity, meeting duration and time into consideration of relationship inference. Cheng *et al.* [26] also adopt co-occurrence as an indicator of friendship. Yang *et al.* construct a co-visitation matrix of user pair to predict social connection by machine learning [6]. This work also suggests that distance from co-location to user home is also an important factor, because people's activities are limited to a few kilometers around their homes. All above feature selection schemes are based on co-locations or co-visitations and do not work well for those friends without check-ins at same places. Furthermore, this type of methods cannot automatically learn the importance of different places and time. Although some of them quantify the impact of different locations and moments heuristically, e.g., using entropy, these methods fail to distinguish the predictive power of different locations/moments for different users.

B. Learning-based Methods

Two learning-based methods published recently apply graph-embedding to extract individual features. Backes *et al.* propose a feature learning technique to summarize users' mobility features automatically [10]. The basic idea is a user's mobility neighbors reflect his mobility profile in a large part. They adopt random walking on the user-location bipartite graph to obtain traces, which contain user's neighbors in the mobility context. Then, they learn the feature vector of each user from these traces by skip-gram model. If the similarity between two users' vectors is high, they are more likely to be friends. Yu *et al.* come up with a new definition of neighbors or context of a user based on the work of Backes. To put it another way, they are no longer random walk on the user-location bipartite but on the user mobility interaction graph i.e., a meeting graph which if two users appear at the same time, there is an edge between them on graph, and the weight of the edge is the meeting frequency of them [11]. Furthermore, they consider the attributes of the meeting location, but assign different weights to the meeting frequency of different locations only through prior knowledge. In such a way as to affect the result of random walks, and thus the representation vector of each user. Both methods generate mobility of individual through a series of random walking on a graph of users constructed based on meeting events. In the graph, there exists an edge between two users if they share the same check-in places, or share the same check-in places with the same user. Several link prediction methods utilize heuristic structures of social graph, e.g., common neighbors, shortest path etc., to infer friendship [27]–[29]. The premise of these methods is that a majority of the graph is available, and they utilize the graph to infer a small number of missing links or to predict future links. How to learn structure features of anonymized social networks without any previous knowledge about friendship is a critical challenge.

The social proximity among users provides an opportunity to identify cyber friends. Several link prediction methods

utilize heuristic structures of social graph, e.g., common neighbors, shortest path etc., to infer a small number of missing links or to predict future links on a given social graph (or the majority of the graph), such prior knowledge is however not available in practice.

VI. Conclusion

In this work, we have proposed a friendship inference attack, called FriendSeeker, based on meticulously estimated spatial-temporal and social proximities among individuals. FriendSeeker is able to discover not only real-world friends who demonstrate presence similarities, but also those hidden cyber friends who are always geographically far away from each other. Extensive evaluations on two real-world datasets show that FriendSeeker consistently outperforms four state-of-the-art baseline methods, particularly when the number of check-ins and co-locations are small. Experimental results also demonstrate that none of the commonly used data perturbation schemes can provide enough protection on friendship privacy under FriendSeeker. In future work, we plan to apply our FriendSeeker to more datasets, and design an obfuscation mechanism to effectively protect friendship from being unveiled by inference attacks.

VII. Acknowledgment

This work was supported in part by the National Natural Science Foundation of China (Grant No. 61972081), and the Natural Science Foundation of Shanghai (Grant No. 22ZR1400200), the RGC RIF grant under the contract R6021-20, and RGC GRF grants under the contracts 16209120, 16200221 and 16207922.

References

[1] James P Bagrow, Xipei Liu, and Lewis Mitchell. Information flow reveals prediction limits in online social activity. *Nature human behaviour*, 3(2):122–128, 2019.

[2] Ratan Dey, Zubin Jelveh, and Keith Ross. Facebook users have become much more private: A large-scale study. In *Proceedings of the IEEE International Conference on Pervasive Computing and Communications Workshops*, pages 346–352. IEEE, 2012.

[3] Quannan Li, Yu Zheng, Xing Xie, Yukun Chen, Wenyu Liu, and Wei-Ying Ma. Mining user similarity based on location history. In *Proceedings of the ACM International Conference on Advances in Geographic Information Systems*, pages 1–10. ACM, 2008.

[4] Xiangye Xiao, Yu Zheng, Qiong Luo, and Xing Xie. Finding similar users using category-based location history. In *Proceedings of the International Conference on Advances In Geographic Information Systems*, pages 442–445. ACM, 2010.

[5] Hongjian Wang, Zhenhui Li, and Wang-Chien Lee. Pgt: Measuring mobility relationship using personal, global and temporal factors. In *Proceedings of the IEEE International Conference on Data Mining*, pages 570–579. IEEE, 2014.

[6] Guolei Yang and Andreas Züfle. Spatio-temporal prediction of social connections. In *Proceedings of the International ACM Workshop on Managing and Mining Enriched Geo-Spatial Data*, pages 1–6. ACM, 2017.

[7] Cheng He, Chao Peng, Na Li, Xiang Chen, and Lanying Guo. Exploiting spatiotemporal features to infer friendship in location-based social networks. In *Proceedings of the Pacific Rim International Conference on Artificial Intelligence*, pages 395–403. Springer, 2018.

[8] Wenbin Tang, Honglei Zhuang, and Jie Tang. Learning to infer social ties in large networks. In *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 381–397. Springer, 2011.

[9] Xiangye Xiao, Yu Zheng, Qiong Luo, and Xing Xie. Inferring social ties between users with human location history. *Journal of Ambient Intelligence and Humanized Computing*, 5(1):3–19, 2014.

[10] Michael Backes, Mathias Humbert, Jun Pang, and Yang Zhang. walk2friends: Inferring social links from mobility profiles. In *Proceedings of the ACM Conference on Computer and Communications Security*, pages 1943–1957. ACM, 2017.

[11] Yanwei Yu, Hongjian Wang, and Zhenhui Li. Inferring mobility relationship via graph embedding. In *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, pages 1–21. ACM, 2018.

[12] Hsun-Ping Hsieh and Cheng-Te Li. Inferring social relationships from mobile sensor data. In *Proceedings of the International Conference on World Wide Web*, pages 293–294. ACM, 2014.

[13] Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining*, pages 1225–1234. ACM, 2016.

[14] Sogol Haghani and Mohammad Reza Keyvanpour. A systemic analysis of link prediction in social network. *Artificial Intelligence Review*, 52(3):1961–1995, 2019.

[15] Xiaoyi Li, Nan Du, Hui Li, Kang Li, Jing Gao, and Aidong Zhang. A deep learning approach to link prediction in dynamic networks. In *Proceedings of the SIAM International Conference on Data Mining*, pages 289–297. SIAM, 2014.

[16] Peng Wang, BaoWen Xu, YuRong Wu, and XiaoYu Zhou. Link prediction in social networks: the state-of-the-art. *Science China Information Sciences*, 58(1):1–38, 2015.

[17] Fang Du, Jiangshe Zhang, Nannan Ji, Junying Hu, and Chunxia Zhang. Discriminative representation learning with supervised auto-encoder. *Neural Processing Letters*, 49(2):507–520, 2019.

[18] Mohammad Mehdi Keikha, Maseud Rahgozar, and Masoud Asadpour. Deeplink: a novel link prediction framework based on deep learning. *Journal of Information Science*, 47(5):642 – 657, 2021.

[19] Lei Le, Andrew Patterson, and Martha White. Supervised autoencoders: Improving generalization performance with unsupervised regularizers. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 107–117, 2018.

[20] Abhinav Mehrotra and Mirco Musolesi. Using autoencoders to automatically extract mobility features for predicting depressive states. In *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, pages 1–20. ACM, 2018.

[21] Ashwin S Dharmadhikari. Six degrees of separation: use of social network analysis to better understand outbreaks of nosocomial transmission of extensively drug-resistant tuberculosis. *The Journal of Infectious Diseases*, 207(1):1–3, 2013.

[22] Hsun-Ping Hsieh, Rui Yan, and Cheng-Te Li. Where you go reveals who you know: Analyzing social ties from millions of footprints. In *Proceedings of the ACM International Conference on Information and Knowledge Management*, pages 1839–1842. ACM, 2015.

[23] Dapeng Zhao, Yuanyuan Jin, Kai Zhang, Xiaoling Wang, Patrick CK Hung, and Wendi Ji. Epla: efficient personal location anonymity. *GeoInformatica*, 22(1):29–47, 2018.

[24] David J Crandall, Lars Backstrom, Dan Cosley, Siddharth Suri, Daniel Huttenlocher, and Jon Kleinberg. Inferring social ties from geographic coincidences. In *Proceedings of the National Academy of Sciences*, volume 107, pages 22436–22441, 2010.

[25] Justin Cranshaw, Eran Toch, Jason Hong, Aniket Kittur, and Norman Sadeh. Bridging the gap between physical location and online social networks. In *Proceedings of the ACM International Conference on Ubiquitous Computing*, pages 119–128. ACM, 2010.

[26] Ran Cheng, Jun Pang, and Yang Zhang. Inferring friendship from check-in data of location-based social networks. In *Proceedings of the IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*, pages 1284–1291. ACM, 2015.

[27] Linyuan Lü, Ci-Hang Jin, and Tao Zhou. Similarity index based on local paths for link prediction of complex networks. *Physical Review E*, 80(4):046122, 2009.

[28] Dashun Wang, Dino Pedreschi, Chaoming Song, Fosca Giannotti, and Albert-Laszlo Barabasi. Human mobility, social ties, and link prediction. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining*, pages 1100–1108. ACM, 2011.

[29] Hsun-Ping Hsieh and Cheng-Te Li. Inferring online social ties from offline geographical activities. *ACM Transactions on Intelligent Systems and Technology*, 10(2):17, 2019.