Analyzing Sensitive Information Leakage in Trajectory Embedding Models

Jiaxin Ding*
jiaxinding@sjtu.edu.cn
Shanghai Jiao Tong University

Shichuan Xi*
Xi_Shichuan@sjtu.edu.cn
Shanghai Jiao Tong University

Kailong Wu
1473686097@sjtu.edu.cn
Shanghai Jiao Tong University

Pan Liu
wslp1999@sjtu.edu.cn
Shanghai Jiao Tong University

Xinbing Wang
xwang8@sjtu.edu.cn
Shanghai Jiao Tong University

Chenghu Zhou
zhouch@lreis.ac.cn
Institute of Geographical Science and Natural Resources Research, Chinese Academy of Sciences

ABSTRACT

With the proliferation of the mobile networks and location-based services, huge volume of user trajectories are collected to analyze the similarity among users and further unveil human mobility patterns for downstream tasks, such as point-of-interest recommendation and tourism planning. In recent works, trajectory embedding methods have been studied as efficient ways of trajectory similarity computation and effective inputs for downstream tasks, which embed trajectories into latent vector spaces equipped with the Euclidean distance to approximate the trajectory similarity and capture the characteristics of human mobility patterns. However, we demonstrate that such embedding, though hiding the locations, can leak the sensitive information of the trajectories, combined with auxiliary data. In this work, we propose trajectory embedding attack schemes to analyze the sensitive information leakage of the embedding vectors. In the experiment, we demonstrate that the passing areas, visited ROIs, and exact shapes of the trajectories are vulnerable under attacks on embedding vectors by the adversary with auxiliary information.

CCS CONCEPTS

• **Security and privacy** → **Privacy protections**; • **Computing methodologies** → **Learning latent representations**.

KEYWORDS

Spatio-temporal embedding, privacy, sensitive information leakage

ACM Reference Format:
Jiaxin Ding, Shichuan Xi, Kailong Wu, Pan Liu, Xinbing Wang, and Chenghu Zhou. 2022. Analyzing Sensitive Information Leakage in Trajectory Embedding Models. In *The 30th International Conference on Advances in Geographic Information Systems (SIGSPATIAL '22), November 1–4, 2022, Seattle, WA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3557915.3561021

*Both authors contributed equally to this research.

SIGSPATIAL '22, November 1–4, 2022, Seattle, WA, USA

ACM ISBN 978-1-4503-9529-8/22/11…$15.00
https://doi.org/10.1145/3557915.3561021

1 INTRODUCTION

With the proliferation of mobile networks and location-based services, huge volume of spatio-temporal data is generated and collected from various sources, such as the GPS records from devices with location service, user check-in data on social media, user call detail records by telecommunication companies, appearance of vehicles captured via roadside units, etc. A sequence of the spatio-temporal data of a user forms a trajectory, which reflects the mobility patterns and daily habits of the user. The large-scale trajectory datasets provide us unprecedented opportunities to mine trajectories and study human mobility patterns, which can be leveraged to traffic monitoring [10, 27], planning [26], tourism planning [25], pandemics spread prediction [24], etc.

Trajectory data mining and analyses are inherited with difficulties from the heterogeneity of trajectories: trajectories are typically collected with nonuniform sampling rates and different sampling lengths usually with data missing and sparsity, and organized in various structures, which makes it hard for downstream tasks to take advantage of most data mining and machine learning models requiring uniform-dimensional vectors as inputs. Traditional feature extraction requires heavy labor and is not generally applicable among different tasks. Moreover, similarity or distance computation between trajectories, which is the most fundamental problem of trajectory analyses and mining tasks, such as clustering[14], classification[13], anomaly detection[15], is time consuming. Such distance metrics include Hausdorff distance , Fréchet distance, Dynamic Time Warping (DTW), Edit distance on Real Edit distance with Real Penalty (ERP) [5], Edit Distance on Real sequence (EDR) [6], etc. Most of these metrics require quadratic computation complexity with respect to the sampling lengths of trajectories, which makes it a hurdle to process large-scale trajectory data.

In face of the above challenges, there has been a new research area of trajectory embedding, which embeds the trajectories into uniform low-dimensional latent vector spaces to capture the characteristics of trajectories, with various machine learning models, such as Seq2Seq [16], LSTM [30], and Transformer [7], inspired by the success of embedding models in natural language processing and graph representation learning. Such embedding vectors can be used as inputs to facilitate downstream tasks with general

data mining and machine learning models [7, 13, 34]. Besides, the similarity between trajectories can be thereby approximated with the Euclidean distance or cosine distance between the embedding vectors, reducing the quadratic complexity of trajectory distance computation to linear [16, 29, 30, 32]. Therefore, trajectory embedding models, inherently capturing human mobility from trajectories, have potentials to bridge the gap between trajectory data and general data mining along with machine learning tasks requiring uniform-dimensional vector inputs.

However, trajectory data is highly private and can reveal personal sensitive information, such as frequent locations [20], social ties [11], personal mobility motifs [8], etc, and privacy issues of location reports and trajectories of users have long been recognized as a serious concern [1, 9]. In the real world applications, there are usually demands for the trajectory embeddings to be shared among different downstream tasks, or even shared to the public, sometimes along with labels. A natural question to ask is whether trajectory embedding vectors leak sensitive information of users. Different from word tokens in word embedding and vertices in graph embedding, spatio-temporal points on a trajectory are inherently correlated. We would also like to ask: is such correlation preserved in the trajectory embedding as well? Will the adversary revert the sensitive locations on the trajectories with auxiliary information? If the answer of either question is yes, the trajectory embedding is not ready to be widely shared between different parties. These questions lead us to initiate a systematic analysis of the privacy of trajectory embedding models by conducting adversarial inference attacks by taking advantage of such correlation and auxiliary information. We assume that the adversary can have certain levels of auxiliary information, such as the embeddings of specific locations, or the embeddings of auxiliary trajectory datasets. Such assumptions are common and reasonable in the real world scenarios, since in different applications, the embedding models are allowed to run locally on the users' devices and request the users to upload their embeddings for better service, or users are allowed to make queries on the models, and therefore, the embedding vectors of locations and trajectories can be obtained, not to mention the possible embedding information leakage by adversary's hacking infrastructures with weak security guarantees. The attacks we consider can be summarized into two classes: the similarity-based attack and the learning-based attack. In the similarity-based attack, the adversary wants to find an area passed by the user by linking the trajectory embedding and the auxiliary information of the location embeddings. The objective of the learning-based attack, is to find the locations on trajectories that are frequently visited by users, and to revert the shape of the whole trajectory, by training machine learning models with the help of auxiliary trajectory datasets and the corresponding embeddings. An example of these attacks can be found in Figure 1.

To the best of our knowledge, our work is the first to study trajectory embedding attacks. This work is inspired partly by previous works of attacks on text embedding [19, 21], where the attacks are conducted to recover the text or sensitive information from word or sentence embeddings. However, the embedding attacks on trajectories are significantly different from those on sentences, since trajectories are inherent with spatio-temporal correlation and geometric properties in contrast to co-occurrence and semantic relations for natural language processing, and the way in which such information is kept in the embedding is different, which needs systematically studying. In this work, we first propose the potential risks of trajectory embedding, provide attack models based on the spatio-temporal characteristics of trajectories, and design new metrics to evaluate the performance of different attacks based on the trajectory geometric properties. All above, we provide a new framework to study trajectory embedding information leakage risks, which, on the other hand, can also provide interpretation for the information kept in the embedding. Our contribution can be summarized as follows:

- To the best of our knowledge, we are the first to analyze the sensitive information leakage in trajectory embedding models, which is also a thorough interpretation of the features preserved in the trajectory embeddings from the viewpoint of adversary.
- We propose two classes of attacks: similarity-based attack and learning-based attack. We also propose evaluation metrics to analyze the sensitive information leakage of users.
- We conduct extensive experiments on different trajectory embedding models under the attacks. The experiments demonstrate that the trajectory embedding models are vulnerable under attacks and can leak sensitive information of users.

2 RELATED WORK

Trajectory Embedding Models. Trajectory embedding models, mostly based on Recurrent Neural Networks (RNN), are proposed to capture features of trajectories and enhance performance of downstream tasks such as similarity computation, clustering and classification for different types of trajectories in real-world applications. With user check-in trajectory data, TULER [13] uses trajectory embedding to link trajectories to users. At2vec [34] utilizes a Seq2Seq framework to learn trajectory representation to calculate activity similarity of users. For GPS trajectory data, t2vec [16] adopts the encoder-decoder framework with cell sequences of sub-trajectories as input, and its objective is to maximize the probability of the decoder recovering the original cell sequences given the embedding vectors. NEUTRAJ [30] combines metric learning, and improves the RNN-based model by adding a spatial attention memory module. In Traj2SimVec [32], sub-trajectory distance and optimal matching are included in the loss function to embed trajectories with structural information. T3S [29] combines self-attention mechanism and LSTM networks into the embedding model, to preserve more structural and spatial information of trajectories. Toast [7]is a Transformer-based embedding model. It utilizes a traffic context aware skip-gram module for pre-training on road networks and a trajectory enhanced Transformer module for learning road segments' and trajectories' representations. Apart from works on trajectory embedding models, there are also some studies on location or POI embedding models. Geo-Teaser[35] adopts word2vec framework and learns POI embeddings with temporal information under different temporal states like weekdays and weekends. DeepMove[37] adopts the Skip-gram framework to learn place representations based on large scale human movement data. By considering the co-occurrence of origin-destination locations for extracting human mobility patterns, Yao et al. proposed a N-gram

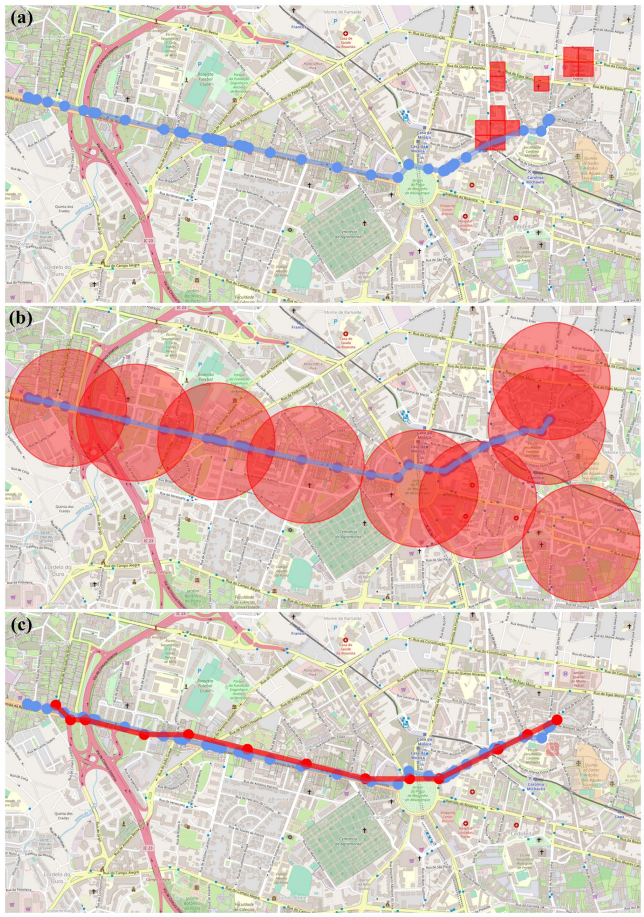

Figure 1: An example of attack results on t2vec trajectory embedding model. The blue curve in the figures denotes the original trajectory. The highlighted regions in (a), (b) are the recovered passing areas and visited ROIs. The red curve in (c) is the trajectory recovered from the embedding.

model to learn city zones' embeddings[31]. TALE[23] utilizes a tree structure in a hierarchical softmax model to extract temporal information from user trajectories to learn distributed location representations. CTLE[17] proposed a contextual location embedding model based on bidirectional Transformer to better encode a location's variable functionalities, and a temporal encoding module to incorporate temporal information into location embeddings. The POI embedding is less sensitive and can be viewed as a special trajectory embedding with only one location. In this work, we mainly focus on the trajectory embedding.

Embedding Attacks. Embedding attacks are mainly studied in the areas of natural language processing (NLP), and graph representation learning. Pan et al. construct two attacks against general-purpose language models, pattern reconstruction attack and keyword inference attack, to obtain sensitive information about the original text from embeddings [19]. Song et al. study the attack models for word embedding and sentence embedding in white box and black box scenarios, including embedding inversion, attribute inference attack and membership inference attack [21]. In the domain of graph learning, adversarial attacks against graph embedding models have been extensively studied in black box or white box scenarios [2–4], and data poisoning is also discussed [22, 33].

As far as we know, there are no systematic studies on trajectory embedding attacks and our work is the first study on information leakage in trajectory embedding models.

3 PRELIMINARIES

In this section, we present the definitions and introduce the attack methods in this work.

3.1 Definitions

Definition 1. (Trajectory). A trajectory $T^i (i = 1, 2, ...)$ is a sequence of points sampled from a continuous movement curve. Specifically, $T^i = [P_1^i, P_2^i, ..., P_l^i]$, where P_j^i denotes the location of the j_{th} point of T^i.

The locations on a trajectory are in the form of longitude and latitude pair for GPS trajectories. Usually to reduce the complexity, we can partition region into grid cells with acceptable resolution and map the coordinate of a location P_j^i to the cell C_j^i it belongs to, and thereafter we obtain a trajectory cell sequence of T^i, denoted $T_C^i = [C_1^i, C_2^i, ..., C_l^i]$.

Definition 2. (Trajectory Embedding). An embedding model Φ maps a trajectory T^i to an embedding vector $e^i = \Phi(T^i) \in \mathbb{R}^d$.

Definition 3. (Region of interest) A region of interest (ROI) R is a region that is frequently traversed by trajectories and has rich semantic information.

Through clustering algorithms such as DBSCAN, trajectory points can be clustered into multiple clusters, and ROIs can be defined according to these clusters without further information.

3.2 Threat Model

The threat model consists of three parts: embedding model Φ, sensitive trajectory dataset D_{sen} and auxiliary trajectory dataset D_{aux}. (1) The trajectory embedding model Φ generates the corresponding trajectory embeddings based on the input trajectories. For the adversary, the model Φ can be queried, that is, if the adversary sends trajectories to the model as input, the model will then output the corresponding embeddings. The attack model is black-box, that is, the adversary can only query the model without knowing the details of the model. (2) The sensitive trajectory dataset D_{sen} is a dataset containing all the sensitive trajectories. These sensitive trajectories are generated by some users and contain sensitive information, such as the passing areas and visited ROIs. The trajectory embeddings of all trajectories in D_{sen} constitute the sensitive embedding set E_{sen}. For the adversary, only E_{sen}, but not D_{sen}, can be obtained. The adversary employ attack methods to recover the sensitive information contained in D_{sen} by attacking E_{sen}. (3) The auxiliary trajectory dataset D_{aux} is the dataset obtained by the adversary and used to assist and realize the attack. With the auxiliary trajectory dataset, the adversary can extract popular areas (ROIs) in the spatial region, and the trajectory embeddings can be obtained by querying the embedding model. Then, the adversary uses these

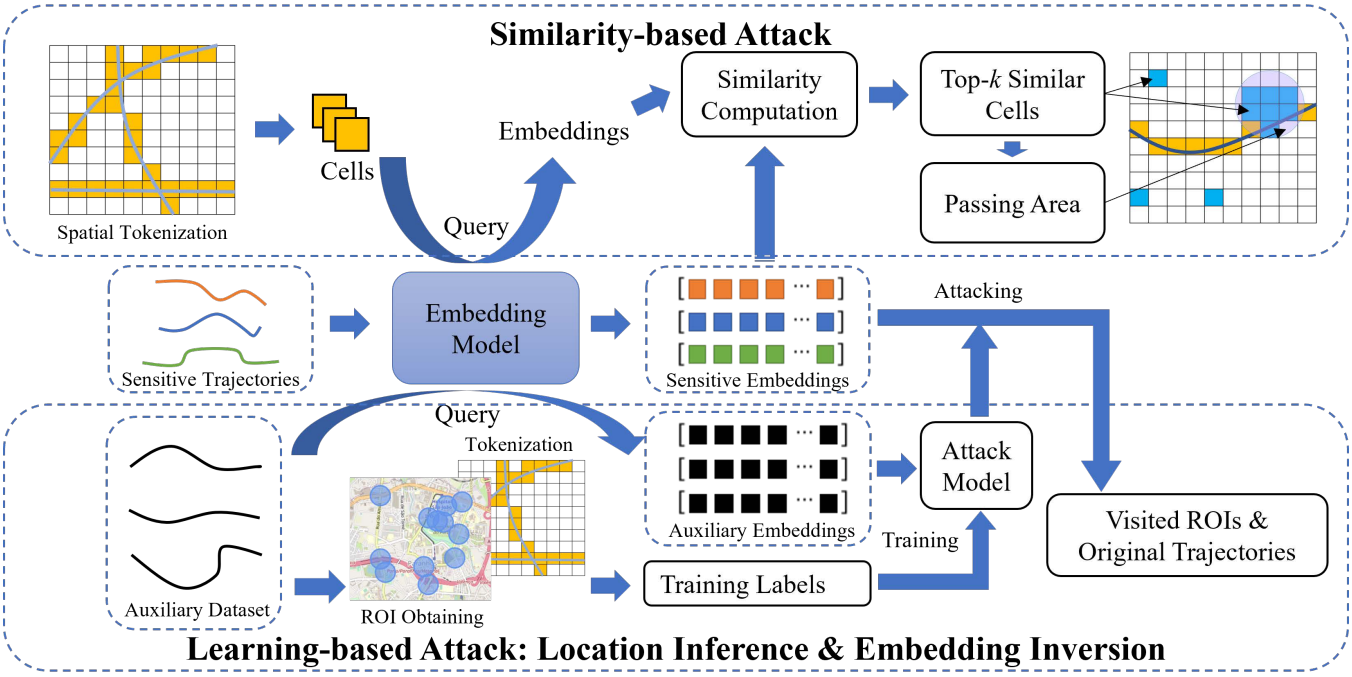

Figure 2: Overview of attack methods, including similarity-based attack and learning-based attack.

auxiliary trajectories and corresponding embeddings to train the inference models and the inversion models. In our work, we propose two types of attack: similarity-based attack and learning-based attack. An overview of the attacks is shown in Figure 2.

The fundamental assumption of the similarity-based attack, based on the spatial correlation of locations on a trajectory, is that a trajectory embedding should be more similar to the embedding of a trajectory's nearby area than that of a far-away area. Therefore, in the similarity-based attack, the adversary infers the *coarse-grained* area that a trajectory is likely to go through, with the auxiliary information of embeddings of pre-defined spatial regions, which are generally the embedding vectors of the centers of the regions, and the regions are not necessarily divided the same as the original embedding models. Here, we assume that the adversary can issue queries on the embedding model on the location level.

In the learning-based attack, including location inference attack and embedding inversion attack, the adversary's objective is to infer *fine-grained* sensitive information including exact locations and shapes of trajectories, with the machine learning models trained by the auxiliary information composed of an auxiliary trajectory dataset and the corresponding embeddings obtained from the embedding model. Here, we assume that the adversary is allowed to query the embedding model on the trajectory level. Thus, in this attack, the adversary can collect enough trajectory embeddings to train their attack models to predict even more sensitive information in the trajectory dataset.

All above, the three attack methods in our work are aimed at three different attack goals. (1) The goal of the similarity-based attack is to find an area through which the original trajectory passes. (2) The goal of location inference attack is to predict whether a trajectory passes some regions of interest (ROIs). (3) The goal of embedding inversion attack is to invert trajectory embedding $\Phi(T)$ to obtain the original trajectory T.

4 PROPOSED ATTACK MODELS

In this section, we present the proposed attack models. Before getting into the details of attack models, we first describe the way we represent the space of trajectories with grid cells and data-driven ROIs. In the attack models, since it is always hard, if not impossible, to infer the exact coordinates of trajectory locations, it is reasonable to define the success of an attack if the adversary can infer the location within an acceptable range, such as grid cell region, near the true location. Besides, in the learning-based attacks, it is more convenient that the input and output of locations for the trained adversarial models are tokenized into discrete labels, namely, regions the locations belong to, instead of the real-valued coordinates, analogous to the vocabulary in natural language processing tasks. Therefore, it is necessary to divide the space trajectories cover into regions for the attacks. In this work, we provide two ways to partition the trajectory location space. The first is uniform grid-cell partition, which is adopted in similarity-based attack and embedding inversion attack, where all cells form the cell set $\mathcal{V} = \{C_i\}_{i=1:|V|}$. The second is data-driven, clustering trajectory location points in the auxiliary dataset to obtain the potential ROIs, which is used in location inference attack.

4.1 Similarity-based Attack

The goal of the similarity-based attack is to find an area through which the original trajectory passes, by computing the similarities

between all areas' embeddings and the trajectory embedding, which corresponds to the scenario that the adversary can make location queries on the embedding models and obtain the embeddings of each area.

We show that with such limited auxiliary information, the adversary can still infer the coarse-grained location information of the trajectories. To implement such an attack, we break the whole process down into three steps: (1) constructing the embeddings for each grid cell; (2) searching for similar cells; (3) constructing dense passing areas. The implementation process of the similarity-based attack is shown in **Algorithm** 1.

Constructing Cell Embeddings. Without loss of generality, the adversary can divide the space into uniform grid cells with acceptable resolution. A cell vocabulary $\mathcal{V} = \{C_i\}_{i=1:|\mathcal{V}|}$ represents all the cells. For each cell C_i, the adversary obtains its embedding $\Phi(C_i)$ from the embedding model, by making location-level embedding queries in the assumption. After traversing all cells in $\mathcal{V}$, the adversary forms the embedding matrix $\mathcal{M}$, whose i_{th} row is equal to $\Phi(C_i)$, $\mathcal{M}[i] = \Phi(C_i)$ for $i = 1, 2, ..., |\mathcal{V}|$.

Searching for Similar Cells. When attacking an sensitive trajectory embedding e, the adversary searches for k most similar cell embeddings in $\mathcal{M}$. The similarity can be calculated with cosine similarity as follows:

$$s_i = \frac{e_i \cdot e}{||e_i|| \cdot ||e||} \quad \text{for } i = 1, 2, ..., |\mathcal{V}| \tag{1}$$

where $e_i = \mathcal{M}[i]$ is the i_{th} row of $\mathcal{M}$. Remark that other similarity metrics, such as Euclidean distance, Pearson correlation coefficient and Hamming distance, can also be used instead of cosine similarity. After calculating all the similarity values, the adversary further finds the k largest values to obtain the top k cells $\{C_{top_1}, C_{top_2}, ..., C_{top_k}\}$, most closely related to the original trajectory.

Constructing Dense Areas. The next step is to cluster the top k cells to obtain the areas that the original trajectory is most likely to pass through. The density clustering algorithm, DBSCAN [12], is adopted in this work. The k cells $\{C_{top_1}, C_{top_2}, ..., C_{top_k}\}$ are converted back to two-dimensional coordinates $\{P_{top_1}, P_{top_2}, ..., P_{top_k}\}$, where P_{top_i} is the center of C_{top_i}. After that, the adversary uses DBSCAN to divide these k points into density-based clusters and chooses the largest cluster G as the dense set to construct the dense area A_r and uses $P_{center} = \frac{1}{n}\sum_{P \in G} P$ as the center of this area.

Algorithm 1 Similarity-based Attack

Input: Embedding model Φ, vocabulary $\mathcal{V}$, sensitive trajectory embedding e, similar cell number k, DBSCAN distance threshold ϵ

1: **for** C_i in $\mathcal{V}$ **do**
2: $\mathcal{M}[i] = \Phi(C_i)$
3: get s_i based on Equation 1
4: **end for**
5: Searching for k largest values in $\{s_i\}$:
6: $\text{topk}(\{s_i\}) = \{s_{top_1}, s_{top_2}, ..., s_{top_k}\}$
7: get coordinates of the top k nearest cell centers $\{P_{top_1}, P_{top_2}, ..., P_{top_k}\}$
8: Coordinates clustering:
9: $G = \text{DBSCAN}_\epsilon(\{P_{top_1}, P_{top_2}, ..., P_{top_k}\})$

Output: Center of the dense area: $P_{center} = \frac{1}{n}\sum_{P \in G} P$

4.2 Learning-based Attack: Location Inference

When the adversary is allowed to query the embedding model with enough auxiliary trajectories, learning-based attacks are possible to conduct. It is reasonable to consider such an attack for three reasons: (1) In reality, the adversary is likely to have access to querying the embedding model, since the model is proposed to better handle downstream tasks. So the adversary can easily obtain the embedding set $\mathcal{E}_{aux}$ for auxiliary trajectory dataset $\mathcal{D}_{aux}$. (2) Usually it is not difficult for the adversary to obtain auxiliary trajectory data as prior knowledge. These data may be public trajectory datasets such as Porto [18] and Geolife [36], or they may be generated by the adversary himself, or they may even be artificial "fake" trajectories. (3) In most instances, $\mathcal{D}_{aux}$ has almost the same distribution as $\mathcal{D}_{sen}$, if both datasets are large enough to reflect the common mobility patterns in the whole area, so the label distribution of $\mathcal{D}_{aux}$ can be considered to be a good approximation for that of $\mathcal{D}_{sen}$ within limited error. Therefore, we propose two learning-based attacks for the adversary to achieve fine-grained attack results with more auxiliary data.

The first learning-based attack is the location inference attack, whose goal is to predict whether a trajectory passes regions of interest. In a spatial region, if some parts of it are frequently visited by vehicles and pedestrians, it means that the parts are probably with specific semantic information and certain functionality, such as commercial area, entertainment zones, important crossings on the road networks, tourist attractions, etc. Therefore, such regions are regions of interest obtained from the real data, if no further information is provided. If a trajectory passes through some of these ROIs, the vehicle or the pedestrian is likely to have visited these ROIs, where sensitive information of visiting such regions is easy to reveal. With such information, the adversary can further infer what the user may have done, leading to a potential privacy leakage risk, given more related knowledge over the ROIs. For example, if a trajectory starts or ends in a region where most visitors of the region would go to a hospital, there is a good chance that the person with the trajectory or the person's family has recently fallen ill. Such information, if obtained by an adversary, may further reveal the user's private information.

In the location inference attack, the adversary's goal is to infer the ROIs passed by the trajectory. To achieve this, the adversary first needs to determine the ROIs to attack. One way is to take data such as places of interest (POIs) from a publicly available geographic dataset and use them to construct ROIs. The other way is data-driven. The adversary can use the trajectory dataset he has to obtain the regions that may be frequently visited and adopt them as ROIs. In this work, we use the second method, considering that the actual trajectories may better reflect how frequently each region is visited and how popular it is.

Obtaining ROI Set and Training Labels. The first step in the location inference attack is to obtain the ROI set $\mathcal{R}$. The adversary can use the auxiliary trajectory dataset to construct the location set $\mathcal{L} = \{P_1, P_2, ...\}$. Using the density clustering algorithm DBSCAN, these locations can be clustered into multiple clusters $\{G_1, G_2, ...\}$. Each cluster corresponds to an ROI, and the average coordinates of

all points in the cluster are taken as the center of the cluster. Each cluster is then given a radius r to frame the area covered by the ROI. To obtain the training labels, the adversary only needs to consider which ROIs the trajectory has visited. The size of each label is $|\mathcal{R}|$, and each element represents an ROI. When an element is 1, it means that the corresponding ROI is visited. The implementation of obtaining the ROI set is shown in **Algorithm** 2.

Algorithm 2 Obtaining ROI Set

Input: auxiliary dataset $\mathcal{D}_{aux}$, DBSCAN distance threshold ϵ, DBSCAN min-samples k and ROI radius r

$\mathcal{L} = \{\}$
for $T = \{P_1, P_2, ...\}$ in $\mathcal{D}_{aux}$ **do**
  $\mathcal{L} \leftarrow \mathcal{L} \cup T$
end for
$\mathcal{G} = \{G_1, G_2, ...\} = \text{DBSCAN}(\mathcal{L}, \epsilon, k)$
$\mathcal{R} = \{\}$
for G in $\mathcal{G}$ **do**
  Construct ROI R:
    Get the ROI center: $P_{center} = \frac{1}{|G|} \sum_{P \in G} P$
    Set the ROI radius: $r_R = r$
    $\mathcal{R} \leftarrow \mathcal{R} + R$
end for

Output: ROI set $\mathcal{R}$

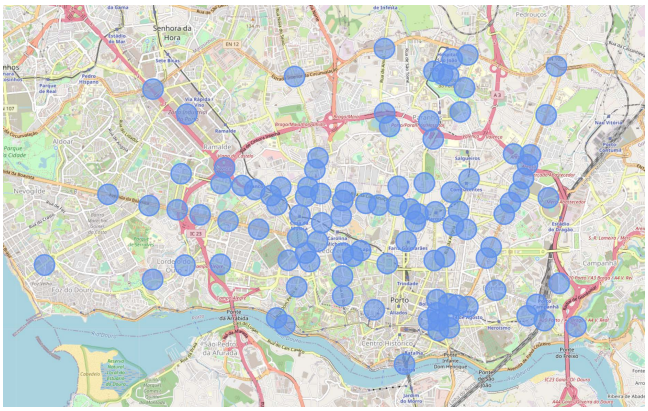

Figure 3: An example of an ROI set obtained from auxiliary trajectory clustering. Each blue circle represents an ROI. The adversary wants to find whether a trajectory passes trough such ROIs in the location inference attacks.

Location Inference. The attack model is trained to infer ROIs visited by the trajectory T given its embedding $\Phi(T)$, which is a multi-label classification (MLC) problem, where a binary label is assigned to each ROI to denote whether it is visited by T. The objective is to minimize the following cross-entropy loss function measuring the difference between the predicted visiting probability and the true probability:

$$\mathcal{L}_{MLC} = -\sum_{R \in \mathcal{R}} \mathcal{Y}_R \log(\hat{\mathcal{Y}}_R) + (1 - \mathcal{Y}_R) \log(1 - \hat{\mathcal{Y}}_R) \quad (2)$$

where $\mathcal{R}$ is the ROI set, $\hat{\mathcal{Y}}_R = P(\mathcal{Y}_R|\Phi(T))$ is the predicted probability that the ROI R is visited by the trajectory, conditioned on $\Phi(T)$, and $\mathcal{Y}_R$ is the truth whether the ROI R is visited or not, $\mathcal{Y}_R = 1$ if R is visited by T and 0 otherwise.

4.3 Learning-based Attack: Embedding Inversion

The second learning-based attack is embedding inversion attack, whose goal is to further invert trajectory embedding $\Phi(T)$ to obtain the original trajectory T. It is reasonable to consider such attack considering that the result of location inference attack demonstrates the feasibility of learning-based approaches to attack embedding models. As mentioned earlier, with access to querying the embedding model with auxiliary dataset $\mathcal{D}_{aux}$, the adversary is able to obtain enough auxiliary embeddings to train an inversion model.

The inversion attack is the most challenging in the three attacks. As we have introduced the spatial tokenization method, here, we focus on recovering the tokenized trajectory, i.e. the cell sequence, instead of the GPS coordinate sequence, mainly considering that discrete tokens are more conducive to the establishment and training of deep learning models. The shapes of trajectories do not change much if any sampling points in the middle of a line segment are not recovered considering that trajectory sampling points are highly spatially correlated. Therefore, in order to simplify the attack model, we assume that the trajectory sequences are of constant length here. From the geometry perspective, we would like to recover the exact shape of the trajectories, and therefore, the geometric distances between the recovered trajectory and the original are also used to evaluate the performance of attacks in our experiments.

Embedding Inversion. Recurrent neural network (RNN) is used for embedding inversion, considering that trajectory is a typical sequence data. The initial input of the RNN network is the embedding of a trajectory. Dealing with the input through multi-layers, the network outputs a cell label at each step. From the second step, the output of the previous step is taken as input to guide the prediction of subsequent cells. The conditional probability of the RNN network is as follows:

$$\mathbb{P}(T_C^i|\Phi(T^i)) = \mathbb{P}(O_1^i|\Phi(T^i)) \prod_{t=2}^{l} \mathbb{P}(O_t^i|O_{1:t-1}^i, \Phi(T^i)), \quad (3)$$

where h_t is the hidden state and O_t^i is the output cell for the i_{th} trajectory at step t, $\mathbb{P}(O_t^i|O_{1:t-1}^i, \Phi(T^i)) = \mathbb{P}(O_t^i|h_t)$ is the conditional probability that the output is O_t^i. The hidden state is then multiplied by a projection matrix W, to obtain the output cell label of the current step. The final conditional probability is:

$$\mathbb{P}(O_t^i = C_j|O_{1:t-1}^i, \Phi(T^i)) = \frac{exp(W_j h_t)}{\sum_{k \in |\mathcal{V}|} exp(W_k h_t)} \quad (4)$$

where W_j and W_k is the j_{th} and k_{th} row of matrix W.

The goal of training is defined as maximizing the probability of outputting the correct cell sequences:

$$\max \quad \prod_{i=1}^{n} \mathbb{P}(T_C^i|\Phi(T^i)) \quad (5)$$

Cross entropy loss is used to measure the difference between the target sequence and the predicted sequence, as follows:

$$L = -\sum_{i=1}^{n}\sum_{t=1}^{l} \log \mathbb{P}(O_t^i|h_t) \tag{6}$$

With the above training process, the adversary can train the inversion model with auxiliary trajectory dataset and the corresponding embeddings to revert the original embeddings of the sensitive trajectory dataset back to trajectories.

All above, we present the potential attacks from the adversary based on trajectory similarity and machine learning to infer the sensitive information of users, including passing areas, ROIs, and exact shape of trajectories. Thereafter, we conduct experiments to demonstrate the performance of such attacks.

5 EXPERIMENTS AND RESULTS

We implement the above attack methods on three embedding models and obtain sensitive information from trajectory embeddings. The attack results demonstrate the sensitive information leakage risk in trajectory embedding models.

5.1 Embedding Models and Datasets

Embedding Models. Three representatives of trajectory embedding models, t2vec [16], NEUTRAJ [30] and Toast [7], are selected as our attack targets. t2vec is a Seq2Seq deep representation learning method. During training, the model takes cell sequences of sub-trajectories as input of the encoder to get embedding vectors, with the objective of maximizing the probability for the decoder to recover the cell sequence of the original trajectory. NEUTRAJ is an RNN-based embedding model, which combines metric learning with deep representation of trajectory, and improves the RNN-based model by adding a spatial attention memory module. Toast is a Transformer-based embedding model. Toast utilizes Skip-gram framework to conduct pre-training on the road network and obtain the representations of road segments. After that, the trajectory is mapped to the road segments, and the trajectory representation model is trained under a Transformer-based framework.

The reasons for choosing these three models are two-fold. First, these models cover three typical representative approaches for processing sequence data, namely Seq2Seq, RNN, and Transformer. Second, they treat the trajectory data in different forms, namely, grid cell sequences, real-valued coordinates, and walks on road networks. For t2vec, its input is spatially tokenized trajectory grid cell sequences, while NEUTRAJ's input is real-valued GPS sequences. Different from the previous two models, Toast maps the trajectory to the road network and represents the trajectory using road segment ids. Therefore, taking these three models as our attack targets can cover different underlying methods for processing trajectory sequences, and also cover different trajectory processing methods. The attack performance on these three models can be generalized to other models falling into these categories of trajectory data processing and representation.

Datasets. Our experiments are based on an open-sourced dataset of taxi GPS trajectories in Porto [18]. For the learning-based attack methods, we randomly selected 180,000 trajectories as training data and 10,000 trajectories as validation data. Besides, 10,000 test trajectories are randomly selected for all the experiments.

Table 1: Similarity-based Attack Results.

Embedding Model	$k = 30,\ \epsilon = 0.6$ km SR@0.5	SR@1	SR@2	$\bar{d_c}(m)$
t2vec-128	77.54%	85.98%	90.95%	734.30
NEUTRAJ-128	4.48%	11.49%	38.99%	2581.98
t2vec-256	93.64%	96.53%	98.01%	259.63
NEUTRAJ-256	9.76%	19.03%	42.63%	2485.38
Embedding Model	$k = 20,\ \epsilon = 0.5$ km SR@0.5	SR@1	SR@2	$\bar{d_c}(m)$
t2vec-128	76.31%	84.36%	89.55%	859.45
NEUTRAJ-128	4.48%	11.40%	38.84%	2588.91
t2vec-256	93.65%	96.36%	97.76%	272.29
NEUTRAJ-256	9.60%	18.94%	42.59%	2494.26

5.2 Similarity-based Attack

Experiment Settings. We use cosine similarity as the similarity metric and apply DBSCAN to cluster the most similar locations. There are two metrics to evaluate the performance of the similarity-based attack: (1) The first metric is the attack success rate SR@r, which is the ratio that the original trajectory actually passes through the inferred dense area A_r, a circle centered at the inferred center P_{center} with radius r km. (2) The second metric is d_c, the minimum distance between the inferred area center P_{center} and the original trajectory.

Note that here we do not include attacking Toast model in this experiment, since in the similarity-based attack, the embedding of a single cell, i.e. a point level embedding is required, while Toast uses FastMapMatching[28] algorithm to map the trajectory into a sequence composed of road segment ids in the resolution of lines, when embedding the trajectory, and FastMapMatching are not suitable to match road segments for such a single point or a grid cell.

Experiment Results. An example of the attack is demonstrated in Figure 1(a). The original trajectory is plotted in blue and the cells in dense set G are plotted in red. It can be observed that these areas are close to the original trajectory. Table 1 illustrates the evaluation results. The results on different parameters of similar cell number k and distance range threshold ϵ for the clustering are shown. If k, ϵ are increased, more locations and larger areas will be contained in G, which increases the success rate of the attacks. Generally, the similarity-based attack on t2vec achieves better performance than that on NEUTRAJ, which means t2vec is more vulnerable under this attack. The average distance $\bar{d_c}$ between the area center and the trajectory on t2vec is less than 1 km, while the value is above 2km for NEUTRAJ. The success rate of passing area with the radius of 0.5km is already above 75% for t2vec, while the value is below 5% for NEUTRAJ and even when the radius is increased to 2km, the value is below 40%. The performance difference between attacks on 128-dimensional embeddings and 256-dimensional embeddings suggests that the embeddings of higher dimensions can capture

Table 2: Learning-based Attack Results.

Attack	Metric	t2vec-128	NEUTRAJ-128	Toast-128	t2vec-256	NEUTRAJ-256	Toast-256
Location Inference	*Precision*	81.82%	66.78%	89.17%	86.28%	70.56%	89.72%
	Recall	77.23%	62.96%	87.77%	82.16%	71.41%	87.00%
Embedding Inversion	*Precision*	92.98%	77.90%	91.40%	94.75%	78.64%	92.47%
	Recall	65.85%	46.81%	51.25%	68.13%	47.54%	52.30%
	Frechet (m)	839.56	961.76	1345.44	771.33	941.42	1265.77
	Hausdorff (m)	790.91	919.39	1179.57	723.25	903.49	1133.02
	F_SE	0.1973	0.2641	0.3600	0.19	0.2602	0.3461
	H_SE	0.1827	0.2476	0.3069	0.1752	0.2447	0.3024
	F_D	0.1678	0.2326	0.3124	0.1599	0.2296	0.2994
	H_D	0.1587	0.2209	0.2759	0.151	0.2186	0.2697

more information of the original trajectory and are more vulnerable to the similarity-based attack.

5.3 Learning-based Attack

Experiment Settings. We train a 3-layer MLP to implement location inference attack, and early stopping and dynamic learning rate are applied to avoid over-fitting and improve the training performance. For embedding inversion attack, we build a 3-layer GRU model with the hidden layer dimension to be 256 and set dropout rate to be 0.2. The number of layers is determined based on the model complexity, training speed and attack performance. As for the hidden layer dimension and dropout rate, we refer to the settings in relevant trajectory embedding models like t2vec [16] and adjust the parameters based on the characteristics of our attack models. Cross entropy loss is adopted to calculate the loss of each step, and the Adam optimizer with a learning rate of 0.0001 is selected to optimize the model.

Two types of metrics are used to evaluate the performance of learning-based attack. (1) The first metric, including *Precision* and *Recall*, are used to measure the accuracy of location inference and embedding inversion. For location inference attack, *Precision* indicates how much of the predicted ROIs are correct, while *Recall* represents how much of the ROIs the trajectory actually passes through have been correctly predicted. Note that in embedding inversion attack, we allow for the inversion error of a trajectory with its nearest neighbors and the *Precision* and *Recall* here both take into account the surrounding neighbors of the trajectory cell sequence. Equation 7 and 8 are the *Precision* and *Recall* for embedding inversion attack, where $\mathcal{N}_{T_C}$ and $\mathcal{N}_{\hat{T}_C}$ are the sets composed of 10-nearest neighbours of each cell on the original trajectory T_C and the recovered trajectory $\hat{T}_C$. (2) The second type of metrics take geometric distances into account to measure the shape difference between the original trajectory and the inversion one. The shape of the recovered trajectory $\hat{T}$ is obtained by connecting all the center points of the recovered trajectory cell sequence. Fréchet distance and Hausdorff distance, denoted as $Frechet$ and $Hausdorff$ in Table 2 , are used to measure the distance between $\hat{T}$ and T. We also notice that the absolute distance for a trajectory staying within a small region and for a trajectory travelling far-away should be different. Therefore, we normalize the distances with the distance between the start and the end of a trajectory, *SE*, and the diameter of the minimum covering circle for a trajectory, *D*, which are indicators for the range of the trajectory activity. *F_SE* is the ratio between Fréchet distance and Start-End distance *SE* as shown in Equation 9. Similarly, we define *H_SE*, *F_D* and *H_D*, *F* and *H* for Fréchet and Hausdorff distance, *_SE* and *_D* for the distance normalized by *SE* and *D*.

$$Precision = \frac{\left|\hat{T}_C \cap \mathcal{N}_{T_C}\right|}{\left|\hat{T}_C\right|} \tag{7}$$

$$Recall = \frac{\left|T_C \cap \mathcal{N}_{\hat{T}_C}\right|}{|T_C|} \tag{8}$$

$$F\_SE = \frac{Frechet(T, \hat{T})}{SE(T)} \tag{9}$$

Remark that for the first metrics of ratios, the bigger the value is, the better the attack performance, while on the contrary, for the second metrics of distances, the smaller value means better attack performance. With the above metrics, we evalute the performance of the embedding models under attacks.

Experiment Results. An example of learning-based attack is shown in Figure 1(b) and Figure 1(c). In Figure 1(b), the red cells are the inferred ROIs. In Figure 1(c), the red trajectory is the recovered trajectory. These two figures indicate that the ROIs or trajectory obtained by learning-based attack is mostly consistent with the original trajectory. All learning-based attacks' evaluation results are illustrated in Table 2. The *Precision* and *Recall* of learning-based attack achieve better performance on t2vec and Toast than on NEUTRAJ, in general.

Among all the three embedding models, the location inference attack performance on Toast is the best, followed by that on t2vec, while the performance on NEUTRAJ is the worst. The *Recall* for NEUTRAJ is much lower than that of t2vec and Toast, indicating that the adversary can obtain the least sensitive information about ROIs from NEUTRAJ embeddings. Meanwhile, both *Precision* and *Recall* of location inference attack on Toast reach more than 85%, suggesting that Toast is the most vulnerable to such attack.

A similar trend is observed in the embedding inversion attack. The performance of this attack on NEUTRAJ is also the worst. But

slightly different from the location inference attack, the performance of the inversion attack on t2vec is a little better than that on Toast considering *Precision* and *Recall*. In terms of shape distance for embedding inversion, however, the attack performance on Toast is the worst, probably because Toast is better at retaining semantic information about trajectories than shape features. In general, all the average distances between the recovered trajectories and the original trajectories are less than or close to 1km. In addition, it can be seen from the metrics *F_SE*, *H_SE*, *F_D* and *H_D* that the distance between the original trajectory and the recovered trajectory is small compared with the length and coverage area of the original trajectory.

All these results demonstrate that learning-based attack can be effectively applied to attack trajectory embedding model and achieve meaningful attack performance, regardless of whether the embedding model is based on Seq2Seq, RNN or Transformer, and whether the trajectory data is processed into cell grids, real values, or road segments.

5.4 Analyzing Embedding Models

In both similarity-based and learning-based attacks, the attack performance on t2vec is better than that on NEUTRAJ, which means that t2vec is more vulnerable. Since Toast is not included in the similarity-based attack, we just focus on comparing t2vec and NEUTRAJ. We also notice that in the learning-based attack, the attack performance on NEUTRAJ achieves almost similar results with that of t2vec on *Precision* and shape distance, and a lot worse on *Recall*. It indicates that NEUTRAJ can be better at preserving the shape of a trajectory than keeping the exact locations. Our explanation is that NEUTRAJ, providing embedding for all the coordinates of the points on the trajectory instead of only embedding the cell tokens in the other models, could suffer from the complexity of such embedding function where there could be of high variance and unstable on embedding nearby points, and therefore, the attack performance on a single point or passing region is not as good as that of a long trajectory.

We further conduct experiments to prove this. First, we calculate two kinds of similarity: the similarity between the embedding of i_{th} cell on the trajectory and the embedding of the whole trajectory, denoted as S^i_{on}, and the average similarity between the embeddings of all off-trajectory cells and that of the trajectory, denoted as $\bar{S}_{off}$. The ratio of S^i_{on} over $\bar{S}_{off}$ is shown in Figure 4(a). The value is designed by the assumption that the embedding of a cell on the trajectory should be more similar to the embedding of the trajectory than the embeddings of cells off the trajectory. However, in Figure 4(a), it demonstrates that for NEUTRAJ, S^i_{on} is almost equal to $\bar{S}_{off}$ for all the on-trajectory cells, while the ratios are larger in t2vec. Second, we compress the trajectories with Douglas Peucker algorithm to reduce the location points while preserving the shapes of trajectories, and compare the embeddings of the compressed trajectories with those of the original by checking the ratio that the top 100 most similar trajectories found by the embeddings of the original trajectories are within top 100 most similar found by the embeddings of the compressed trajectories. The result is shown in Figure 4(b). With the increase of the compression ratio, more locations are deleted, but the embedding similarity between the compressed trajectories and the original on NEUTRAJ is higher than that on t2vec. This result validates that NEUTRAJ is better at preserving the shape of a trajectory than the exact locations. The above experiments also explain the reason why t2vec is more vulnerable under attacks.

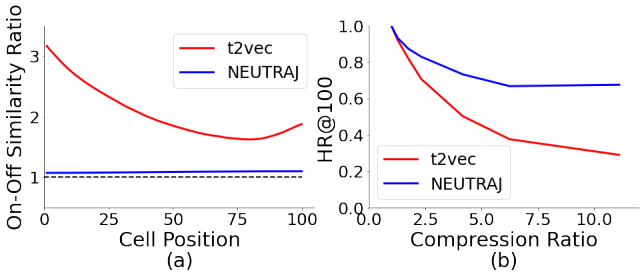

Figure 4: (a) Ratio of S^i_{on} over $\bar{S}_{off}$. (b) Similarity of trajectory embeddings under different compression ratio.

6 CONCLUSION

To the best of our knowledge, we conducted the first study on trajectory embedding attack. At present, most trajectory embedding models are RNN-based or Transformer-based, and lack of protection for trajectory sensitive information in the training process, which makes trajectory embedding vulnerable under attacks and limit the potential of widely-adoption of trajectory embeddings between different mining and learning tasks or different parties owning the data or models. We propose two kinds of attack methods: similarity-based attack and learning-based attack, which can also be viewed as a thorough interpretation of the features preserved in the trajectory embedding models. Empirically, both of them can unveil users' sensitive information in different trajectory embedding models including the mainstream approaches of processing and embedding trajectory data, and demonstrate the vulnerability of such embedding models, which we should take special care of. In future work, we will further study the potential attacks and protection of trajectory embedding to achieve a best trade-off between accuracy and privacy.

ACKNOWLEDGMENT

We would like to acknowledge supports from NSF China under Grant (No. 42050105, 62202299, 62020106005, 62041205, 61960206002, 61829201, 62041205), Shanghai Sailing Program 20YF1421300, Natural Science Foundation of Shanghai No. 22ZR1429100.

REFERENCES

[1] Miguel E Andrés, Nicolás E Bordenabe, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. 2013. Geo-indistinguishability: Differential privacy for location-based systems. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. 901–914.

[2] Aleksandar Bojchevski and Stephan Günnemann. 2019. Adversarial attacks on node embeddings via graph poisoning. In *International Conference on Machine Learning*. PMLR, 695–704.

[3] Heng Chang, Yu Rong, Tingyang Xu, Wenbing Huang, Honglei Zhang, Peng Cui, Wenwu Zhu, and Junzhou Huang. 2020. A restricted black-box adversarial framework towards attacking graph embedding models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 3389–3396.

[4] Jinyin Chen, Yangyang Wu, Xuanheng Xu, Yixian Chen, Haibin Zheng, and Qi Xuan. 2018. Fast gradient attack on network embedding. *arXiv preprint arXiv:1809.02797* (2018).

[5] Lei Chen and Raymond Ng. 2004. On the marriage of lp-norms and edit distance. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*. 792–803.

[6] Lei Chen, M Tamer Özsu, and Vincent Oria. 2005. Robust and fast similarity search for moving object trajectories. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*. 491–502.

[7] Yile Chen, Xiucheng Li, Gao Cong, Zhifeng Bao, Cheng Long, Yiding Liu, Arun Kumar Chandran, and Richard Ellison. 2021. Robust Road Network Representation Learning: When Traffic Patterns Meet Traveling Semantics. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 211–220.

[8] Yves-Alexandre De Montjoye, César A Hidalgo, Michel Verleysen, and Vincent D Blondel. 2013. Unique in the crowd: The privacy bounds of human mobility. *Scientific reports* 3, 1 (2013), 1–5.

[9] Jiaxin Ding, Chien-Chun Ni, and Jie Gao. 2017. Fighting statistical Re-Identification in human trajectory publication. In *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. 1–4.

[10] Jiaxin Ding, Chien-Chun Ni, Mengyu Zhou, and Jie Gao. 2017. Minhash hierarchy for privacy preserving trajectory sensing and query. In *2017 16th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*. IEEE, 17–28.

[11] Nathan Eagle, Alex Pentland, and David Lazer. 2009. Inferring friendship network structure by using mobile phone data. *Proceedings of the national academy of sciences* 106, 36 (2009), 15274–15278.

[12] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. 1996. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining* (Portland, Oregon) *(KDD'96)*. AAAI Press, 226–231.

[13] Qiang Gao, Fan Zhou, Kunpeng Zhang, Goce Trajcevski, Xucheng Luo, and Fengli Zhang. 2017. Identifying Human Mobility via Trajectory Embeddings. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence* (Melbourne, Australia) *(IJCAI'17)*. AAAI Press, 1689–1695.

[14] Chih-Chieh Hung, Wen-Chih Peng, and Wang-Chien Lee. 2015. Clustering and aggregating clues of trajectories for mining trajectory patterns and routes. *The VLDB Journal* 24, 2 (2015), 169–192.

[15] Xiangjie Kong, Ximeng Song, Feng Xia, Haochen Guo, Jinzhong Wang, and Amr Tolba. 2018. LoTAD: Long-term traffic anomaly detection based on crowdsourced bus trajectory data. *World Wide Web* 21, 3 (2018), 825–847.

[16] Xiucheng Li, Kaiqi Zhao, Gao Cong, Christian S Jensen, and Wei Wei. 2018. Deep representation learning for trajectory similarity computation. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*. IEEE, 617–628.

[17] Yan Lin, Huaiyu Wan, Shengnan Guo, and Youfang Lin. 2021. Pre-training context and time aware location embeddings from spatial-temporal trajectories for user next location prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 4241–4248.

[18] Luís Moreira-Matias, João Gama, Michel Ferreira, João Mendes-Moreira, and Luis Damas. 2016. Time-evolving OD matrix estimation using high-speed GPS data streams. *Expert systems with Applications* 44 (2016), 275–288.

[19] Xudong Pan, Mi Zhang, Shouling Ji, and Min Yang. 2020. Privacy Risks of General-Purpose Language Models. In *2020 IEEE Symposium on Security and Privacy (SP)*. 1314–1331. https://doi.org/10.1109/SP40000.2020.00095

[20] Chaoming Song, Zehui Qu, Nicholas Blumm, and Albert-László Barabási. 2010. Limits of predictability in human mobility. *Science* 327, 5968 (2010), 1018–1021.

[21] Congzheng Song and Ananth Raghunathan. 2020. Information Leakage in Embedding Models *(CCS '20)*. Association for Computing Machinery, New York, NY, USA, 377–390. https://doi.org/10.1145/3372297.3417270

[22] Mingjie Sun, Jian Tang, Huichen Li, Bo Li, Chaowei Xiao, Yao Chen, and Dawn Song. 2018. Data poisoning attack against unsupervised node embedding methods. *arXiv preprint arXiv:1810.12881* (2018).

[23] Huaiyu Wan, Fuchen Li, Shengnan Guo, Zhong Cao, and Youfang Lin. 2019. Learning time-aware distributed representations of locations from spatio-temporal trajectories. In *International Conference on Database Systems for Advanced Applications*. Springer, 268–272.

[24] Haotian Wang, Abhirup Ghosh, Jiaxin Ding, Rik Sarkar, and Jie Gao. 2021. Heterogeneous interventions reduce the spread of COVID-19 in simulations on real mobility data. *Scientific reports* 11, 1 (2021), 1–12.

[25] Sheng Wang, Mingzhao Li, Yipeng Zhang, Zhifeng Bao, David Alexander Tedjopurnomo, and Xiaolin Qin. 2018. Trip planning by an integrated search paradigm. In *Proceedings of the 2018 International Conference on Management of Data*. 1673–1676.

[26] Yong Wang, Guoliang Li, and Nan Tang. 2019. Querying shortest paths on time dependent road networks. *Proceedings of the VLDB Endowment* 12, 11 (2019), 1249–1261.

[27] Dong Xie, Feifei Li, and Jeff M Phillips. 2017. Distributed trajectory similarity search. *Proceedings of the VLDB Endowment* 10, 11 (2017), 1478–1489.

[28] Can Yang and Gyozo Gidofalvi. 2018. Fast map matching, an algorithm integrating hidden Markov model with precomputation. *International Journal of Geographical Information Science* 32, 3 (2018), 547–570.

[29] Peilun Yang, Hanchen Wang, Ying Zhang, Lu Qin, Wenjie Zhang, and Xuemin Lin. 2021. T3S: Effective Representation Learning for Trajectory Similarity Computation. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 2183–2188.

[30] Di Yao, Gao Cong, Chao Zhang, and Jingping Bi. 2019. Computing Trajectory Similarity in Linear Time: A Generic Seed-Guided Neural Metric Learning Approach. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. 1358–1369. https://doi.org/10.1109/ICDE.2019.00123

[31] Zijun Yao, Yanjie Fu, Bin Liu, Wangsu Hu, and Hui Xiong. 2018. Representing urban functions through zone embedding with human mobility patterns. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-18)*.

[32] Hanyuan Zhang, Xingyu Zhang, Qize Jiang, Baihua Zheng, Zhenbang Sun, and Weiwei Sun. 2020. Trajectory similarity learning with auxiliary supervision and optimal matching.(2020). In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, Yokohama, Japan*. 11–17.

[33] Hengtong Zhang, Tianhang Zheng, Jing Gao, Chenglin Miao, Lu Su, Yaliang Li, and Kui Ren. 2019. Data poisoning attack against knowledge graph embedding. *arXiv preprint arXiv:1904.12052* (2019).

[34] Yifan Zhang, An Liu, Guanfeng Liu, Zhixu Li, and Qing Li. 2019. Deep representation learning of activity trajectory similarity computation. In *2019 IEEE International Conference on Web Services (ICWS)*. IEEE, 312–319.

[35] Shenglin Zhao, Tong Zhao, Irwin King, and Michael R Lyu. 2017. Geo-teaser: Geo-temporal sequential embedding rank for point-of-interest recommendation. In *Proceedings of the 26th international conference on world wide web companion*. 153–162.

[36] Yu Zheng, Xing Xie, Wei-Ying Ma, et al. 2010. Geolife: A collaborative social networking service among user, location and trajectory. *IEEE Data Eng. Bull.* 33, 2 (2010), 32–39.

[37] Yang Zhou and Yan Huang. 2018. Deepmove: Learning place representations through large scale movement data. In *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 2403–2412.