Entity Identification Based on Human Mobility Data

Huimin Ren

PhD Dissertation in Data Science

Worcester Polytechnic Institute, Worcester, MA

Dec. 2022

Committee Members:
Dr. Yanhua Li, Associate Professor, WPI. **Advisor**
Dr. Xiangnan Kong, Associate Professor, WPI.
Dr. Kyumin Lee, Associate Professor, WPI.
Dr. Yu Zheng, IEEE Fellow, JD Technology. **External Member**

Abstract

With the rapid development of the urbanization, the government has not only benefited from it, but also needs to deal with great challenges of urban governance, such as transportation issues, unhealthy economy development, public safety etc. To address the challenges, the government requires an accurate understanding of urban entities. For example, it is necessary to know whose work may involve criminal and where is the dangerous location. With the increasing adoption of smart devices and GPS modules, more and more human mobility data is generated in cities, such as trajectory data and check-in data, which makes it possible to infer the meaning of urban entities. In this dissertation, we propose and develop several novel machine learning techniques to identify entities in the urban environment based on human mobility data. In particular, we try to solve the entity identification problems in the following aspects.

1) Map-constrained Trajectory Recovery. First of all, most of the spatio-temporal data mining problems, including urban entity identification, rely on high-sampling-rate trajectories since fine-grained trajectories can provide more information. In this paper, we revisit the fundamental research topic, trajectory interpolation, to enhance the trajectory data and support the urban entity identification more effectively. We propose a Map-constrained Trajectory Recovery framework, MTrajRec, to recover the fine-grained points in trajectories and map match them on the road network in an end-to-end manner. Extensive experiments based on large-scale real-world trajectory data confirm the effectiveness and efficiency of our approach.

2) Human Mobility Signature Identification. Identifying human mobility signature is a significant task in the urban entity identification, which is helpful in many real-world applications, e.g., identifying drivers in ride-hailing service and criminal identification. In this topic, we propose the human mobility signature identification solution to identify human agents from their mobility data. We make the first attempt to match identities of human agents only from the observed location trajectory data by proposing a novel and efficient framework named Spatio-temporal Siamese Networks (ST-SiameseNet). Experimental results on a real-world taxi trajectory dataset show that

our proposed ST-SiamesNet can achieve an F_1 score of 0.8508, which significantly outperforms the state-of-the-art techniques.

3) Company Real Workplace Recognization. Local business development can help create jobs and increase tax revenue. Company real workplace identification is important for the government to manage companies. In this work, we recognize company's real workplace by implementing a novel two-step data mining method from employees' check-in data. Experimental results show that our proposed method significantly outperforms six baselines. And A real-world application system has been deployed in Nantong, China since September 2021, demonstrating the effectiveness of our solution.

4) Illegal Chemical Facility Detection. Detecting illegal chemical facilities is related to public safety. In this paper, we develop a system for illegal chemical facility detection based on hazardous chemical truck trajectories. We first generate candidate locations by clustering stay points extracted from trajectories and filtering out known locations. Then, we rank those locations in suspicion order by modeling whether it has loading/unloading events. ICFinder+ is evaluated over the real-world dataset from Nantong in China, and the deployed system identified 20 illegal chemical facilities in 3 months.

5) Learn to Stay. Stay points of trajectories are key to understanding location and human mobility. Most of the previous work focus on GPS trajectories which are limited since not all of the vehicles are equipped with GPS modules. In this work, we design a two-stage method named SAInf to detect stay points via surveillance camera records which can capture the moving patterns for all the vehicles. SAInf first detects which surveillance camera record pairs have stay events within them, and then uses a layer-by-layer stay area identification algorithm to infer the exact location of an object stay. Experimental results show that SAInf outperforms other baselines by 58%.

Acknowledgments

I would like to give my warmest thanks to my adviser, Prof. Yanhua Li, for his endless help and guidance. I sincerely thank him for encouraging me to get through all the challenges in my research. Especially, I'm grateful that he supported me to have a long-term internship during my Ph.D. study.

I would like to thank my committee members, Prof. Xiangnan Kong, Prof.Kyumin Lee, and Prof.Yu Zheng for carefully reading my publications and this manuscript and improving them with their numerous suggestions. I also thank all members of the DSRG group for listening to my preliminary and final talks and enriching them with bright ideas and critical comments.

Many special thanks go to my collaborators, Prof. Randy C. Paffenroth, Prof. Xun Zhou, Dr. Jun Luo, Dr. Menghai Pan, Dr. Chong Zhou, Dr. Matthew Weiss, and Yun Yue, Dr. Sijie Ruan, Dr. Tianfu He, Dr. Chuishi Meng, Dr. Ye Yuan, Prof. Ruiyuan Li, Zhipeng Ma, Boyang Han, Zheng Zhu, Shuncheng Liu and Zhi Xu, for their novel research ideas, critical comments, and productive collaborations. They provided a lot of exciting insights and incisive comments during our collaborations. I am also looking forward to future collaborations.

I also owe my gratitude to all my mentors and colleagues during my internship at JD technology. In particular, Prof. Yu Zheng and Dr.Jie Bao help me with both their excellent statistical expertise and amazing personality. They show me how to do research and most importantly, why to do research. I would also like to thank Dr. Sijie Ruan, Dr. Tianfu He, Huajun He, Yue Hu, and all the other colleagues for their brilliant ideas on and off work. The two-year internship in JD technology definitely would be one of the most memorable, invaluable journeys in my life.

I am endlessly grateful to my friends Yun Yue, Xuechun Li, Jingying Guo, Yu Zhu, and Wensi Li for their always supports during my Ph.D. studies. Those talks at late nights

are truly invaluable to me.

Last but not least, I would like to thank my family, Fu Ren and Yueping Yu, and Donghai Zhang for their love and support all the way. This dissertation is devoted to them.

Contents

List of Figures

List of Tables

List of Publications during my Ph.D. Studies at WPI

1. **Huimin Ren**, Sijie Ruan, Yanhua Li, Jie Bao, Chuishi Meng, Ruiyuan Li and Yu Zheng. MTrajRec: Map-Constrained Trajectory Recovery via Seq2Seq Multi-task Learning. The 27th SIGKDD conference on Knowledge Discovery and Data Mining, Singapore, August 14 - 18, 2021.

2. Zheng Zhu, **Huimin Ren**, Sijie Ruan, Boyang Han, Jie Bao, Ruiyuan Li, Yanhua Li and Yu Zheng. ICFinder: A Ubiquitous Approach to Detecting Illegal Hazardous Chemical Facilities with Truck Trajectories (Short Paper). In Proceedings of the 29th ACM International Conference on Advances in Geographical Information Systems, 2021.

3. **Huimin Ren**, Menghai Pan, Yanhua Li, Xun Zhou and Jun Luo. ST-SiameseNet: Spatio-Temporal Siamese Networks for Human Mobility Signature Identification. The 26th SIGKDD conference on Knowledge Discovery and Data Mining, San Diego, CA, August 23 - 27, 2020.

4. **Huimin Ren**, Yun Yue, Chong Zhou, Randy Paffenroth, Yanhua Li, and Matthew Weiss. Robust Variational Autoencoders: Generating Noise-Free Images from Corrupted Images. AdvML'20: 2nd Workshop on Adversarial Learning Methods for Machine Learning and Data Mining, San Diego, CA, August 24, 2020 (Held in conjunction with ACM SIGKDD 2020).

5. **Huimin Ren**, Sijie Ruan, Yanhua Li, Ye Yuan Jie Bao, Tianfu He, Shuncheng Liu, Yan Zhang, Chuishi Meng, Yu Zheng. A Novel Approach for Company Real Workplace Identification via E-commercial Data. (under submission).

6. **Huimin Ren**, Zheng Zhu, Sijie Ruan, Boyang Han, Jie Bao, Yanhua Li and Yu Zheng. Illegal Hazardous Chemical Facilities Detection with Truck Trajectories (under submission).

7. Zhipeng Ma, **Huimin Ren**, Chuishi Meng, Sijie Ruan, Je Bao, Tianrui Li, Yu Zheng. SAInf: A Framework for Stay Area Inference with Sparse Trajectories (under submission).

Chapter 1

Introduction

1.1 Map-constrained Trajectory Recovery

With the increasing adoption of GPS modules, there are a wide range of urban applications based on trajectory data analysis, such as vehicle navigation, travel time estimation, and driver behavior analysis. The effectiveness of urban applications relies greatly on the high sampling rates of trajectories precisely matched to the map. However, a large number of trajectories are collected under a low sampling rate in real-world practice, due to certain communication loss and energy constraints. To enhance the trajectory data and support the urban applications more effectively, many trajectory recovery methods are proposed to infer the trajectories in free space. In addition, the recovered trajectory still needs to be mapped to the road network, before it can be used in the applications. However, the two-stage pipeline, which first infers high-sampling-rate trajectories and then performs the map matching, is inaccurate and inefficient. In this paper, we propose a Map-constrained Trajectory Recovery framework, MTrajRec, to recover the fine-grained points in trajectories and map match them on the road network in an end-to-end manner. MTrajRec implements a multi-task sequence-to-sequence learning architecture to predict road segment and moving ratio simultaneously. *Constraint mask, attention mechanism*, and *attribute module* are proposed to overcome the limits of coarse grid representation and improve the performance.

1.2 Human Mobility Signature Identification

Given the historical movement trajectories of a set of individual human agents (e.g., pedestrians, taxi drivers) and a set of new trajectories claimed to be generated by a specific agent, the Human Mobility Signature Identification (HuMID) problem aims at validating if the incoming trajectories were indeed generated by the claimed agent. This problem is important in many real-world applications such as driver verification in ride-sharing services, risk analysis for auto insurance companies, and criminal identification. Prior work on identifying human mobility behaviors requires additional data from other sources besides the trajectories, e.g., sensor readings in the vehicle for driving behavior identification. However, these data might not be universally available and is costly to obtain. To deal with this challenge, in this work, we make the first attempt to match identities of human agents only from the observed location trajectory data by proposing a novel and efficient framework named Spatio-temporal Siamese Networks (ST-SiameseNet). For each human agent, we extract a set of profile and online features from his/her trajectories. We train ST-SiameseNet to predict the mobility signature similarity between each pair of agents, where each agent is represented by his/her trajectories and the extracted features.

1.3 Company Real Workplace Identification

Local business development is vital in city growth, which helps to create more jobs and increase tax revenue. Thus, many governments provide financial support and make strategic policies to attract companies. However, due to traffic congestion and labor shortage, companies may not settle their workplaces to the registered locations, leading to the government's financial loss and management inconvenience. Therefore, it is necessary to discover the real workplaces of the companies. However, existing solutions, such as active on-field screening and recruit site crawling, are inaccurate and incomplete. In this paper, we propose a novel two-step data mining method named LocRecognizer to discover the real workplaces of companies from human mobility data. The main idea is that the places where company-related users gather may be the workplaces of the companies. LocRecognizer adopts a two-stage clustering method to generate location candidates and further detect correlated workplaces by using a multi-learner deep learning model.

1.4 Illegal Chemical Facility Identification

Chemical materials are useful but sometimes hazardous, which requires strict regulation from the government. However, due to the potential economic benefits, many illegal hazardous chemical facilities are running underground, which poses a significant public safety threat. However, the traditional solutions, e.g., on-field screening and the anonymous tip-offs, involve a lot of human efforts. In this paper, we propose a ubiquitous approach called ICFinder to detecting illegal chemical facilities with chemical transportation trajectories. We first generate candidate locations by clustering stay points extracted from trajectories, and filter out known locations. Then, we rank those locations in suspicion order by modeling whether it has the loading/unloading events.

1.5 Learn2Stay

Points are not equally important in a trajectory. Among all points, stay points are key to understanding location and human mobility. Over the past decade, researchers have conducted extensive work based on stay point, which has had a positive impact. However, with the popularity of privacy awareness, it is becoming increasingly difficult for city managers to collect GPS trajectory data, even though they really need it. And traditional stay point detection algorithms are highly dependent on GPS trajectories. These becomes an obstacle to understanding the urban knowledge. Fortunately, surveillance cameras that are widely deployed in urban space provide us an opportunity to perceive stay points from trajectory. Unfortunately, the uncertainty introduced by surveillance camera record makes it difficult to determine whether stay event occurs and where stay in. And the robustness of surveillance cameras will enhance this uncertainty.

1.6 Outline of Dissertation

The dissertation is organized as follows:

Section 2 elaborates how to recover low-sampling-rate trajectories to high-sampling-rate ones and map match them onto the road network simultaneously.

Section 3 develops a siamese-based network for human mobility signature identification only based on the observed location trajectory data.

Section 4 proposes a novel two-step data mining method to discover the real workplaces of companies from human mobility data.

Section 5 implements a ubiquitous approach to detect illegal chemical facilities with chemical transportation trajectories.

Section 6 designs a ubiquitous approach to detect stay area with traffic camera trajectories.

Chapter 2

Map-constrained Trajectory Recovery

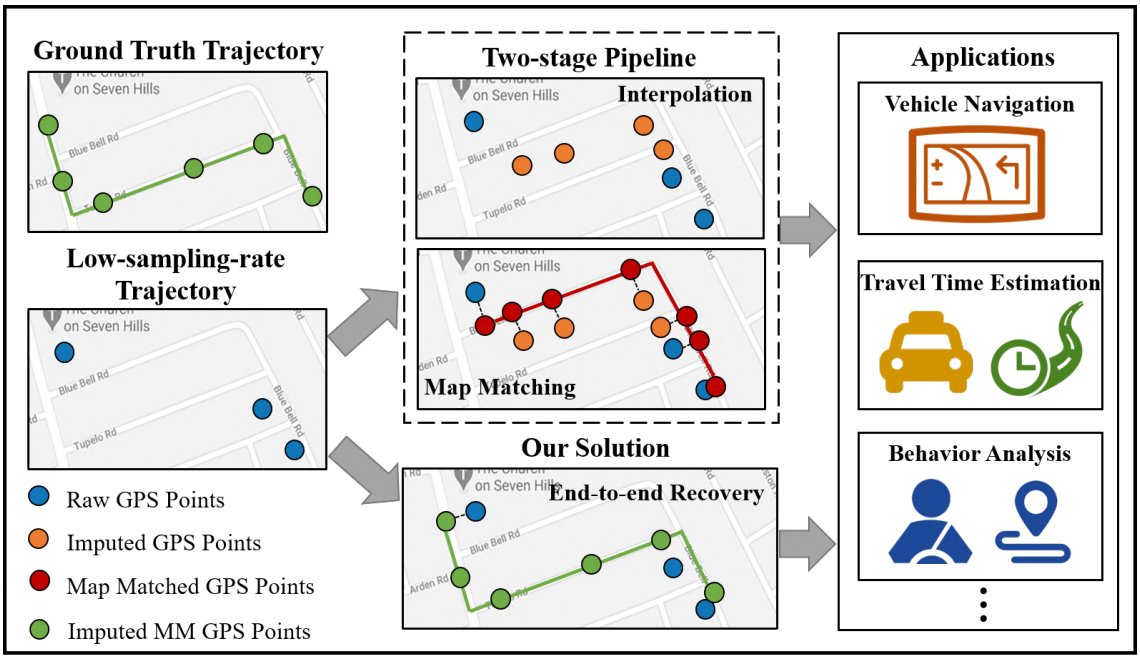


Figure 2.1: Intuition of Trajectory Recovery Problem

2.1 Introduction

Nowadays, GPS modules have been widely used throughout all kinds of mobile devices, and the generated trajectory data have empowered many applications, such as vehicle navigation [44], travel time estimation [27], driver behavior analysis [22]. The effectiveness of these applications relies on the sampling rate of trajectories, since low-sampling-rate trajectories lose detailed information for moving objects and increase the uncertainty between two consecutive sampled locations. However, in reality, there is a large quantity of

low-sampling-rate GPS trajectory data. For instance, taxis usually report GPS locations every $2 \sim 6$ minutes to reduce the energy consumption of communication [130].

In order to utilize these low-sampling-rate trajectories more effectively, many inference methods have been proposed to recover low-sampling-rate trajectories. One straight forward solution is to assume that vehicles are moving with the uniform speeds [38]. However, the dynamic behavior mobility pattern cannot be captured in this way. To tackle this challenge, many deep learning based models have been proposed [113, 119, 118]. For example, Wang et al. [113] recovered a high-sampling-rate trajectory from a low-sampling-rate one with DHTR, which predicts the coarse-grained grid of high-sampling-rate point by integrating a sequence-to-sequence model with a calibration component of Kalman Filter. However, after the trajectory recovery process, there is still a map matching [80] task to be done, before the trajectory data can be used by the applications. The map matching task aligns GPS points in trajectories with the road network, and is a fundamental pre-processing step. It not only empowers the road-based applications, e.g., vehicle navigation [44], travel time estimation [27], but also enriches trajectories with more semantic meanings to benefit driver-based applications, e.g., behavior analysis [22].

Traditional methods solve the map-constrained trajectory recovery problem through a two-stage pipeline, which first recovers low-sampling-rate trajectories and then implements a map matching algorithm to project trajectories onto the road network. Although we can perform map matching based on the recovered trajectory as shown in Fig. 2.1, the inference error can be accumulated. In addition, the two-stage pipeline is also inefficient because map matching algorithms are time-consuming [80, 130, 71]. Based on these observations, a natural question arises: can we recover a high-sampling-rate trajectory based on a low-sampling-rate one and perform map matching on it simultaneously? The end-to-end solution is expected to reduce the inference error as shown in the bottom part of Fig. 2.1, and improve the efficiency.

Luckily, with the renaissance of neural networks, the deep learning technique provides a promising computational framework to solve complicated tasks, which gives us an opportunity to tackle the challenges in an end-to-end fashion. To our best knowledge, this work is the first attempt to recover low-sampling-rate trajectories and map match them onto the road network simultaneously. Specifically, the map-constrained trajectory recovery problem is challenging due to the following reasons:

1. *Map constraints.* Previous work [113, 119, 118] focus on trajectory recovery in the free space. It is difficult for a deep learning model to generate road network constrained

coordinates.

2. *Coarse grid representations.* Converting the numerical coordinates to discrete units is a common preprocessing strategy for deep-learning-based trajectory modeling [113, 119, 92, 83, 28], since it can reduce the computational complexity of unconstrained numerical coordinates. However, using discrete units is likely to introduce noises or inaccurate information into the model, which incurs challenges in the fine-grained trajectory recovery problem.

3. *Diverse complex factors.* The recovery accuracy is influenced by traffic conditions, as vehicles are not moving at constant speeds in the real world. The traffic conditions are determined by many complex factors, such as the spatial context, temporal dependencies, and weather conditions [110]. In such scenario, only using the low-sampling-rate trajectory data is not be sufficient to recover the missing points accurately.

To tackle these challenges, in this paper, we propose a novel Map-constrained Trajectory Recovery model, i.e., MTrajRec, which is based on the sequence-to-sequence (Seq2Seq) multi-task learning. MTrajRec recoveries the trajectories by interpolating the missing points on the road network. First, to guarantee the recovered trajectories constrained on the road network, we introduce a multi-task learning into the classic Seq2Seq generation framework by predicting road segment IDs and moving ratios simultaneously. In order to deal with coarse grid representation, we design a constraint mask layer to extract the fine-grained information. Finally, since the traffic is affected by complex factors, we employ an attribute module to capture the external influences. Overall, our main contributions can be summarized as follows:

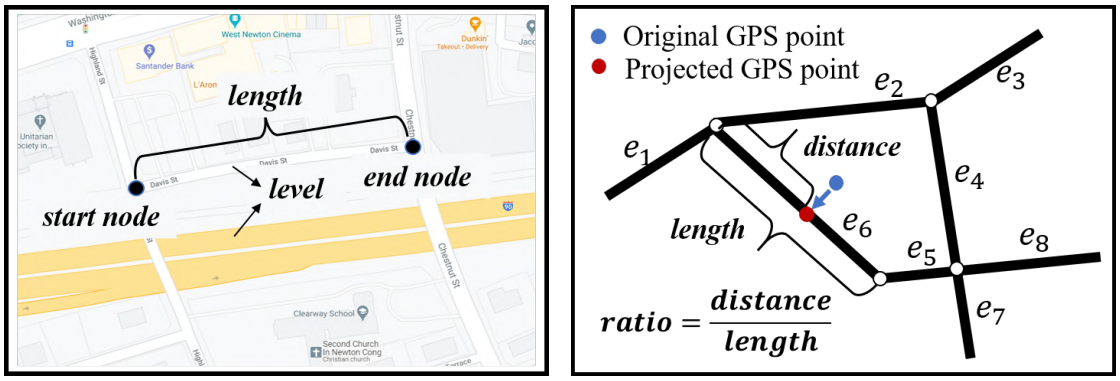
- We present the first attempt to solve the map-constrained trajectory recovery problem via Seq2Seq multi-task learning.
- We devise a novel MTrajRec model, which can recover the trajectory and map match it onto the road network simultaneously. We utilize *constraint mask, attention mechanism* and *attribute module* to improve the performance.
- We conduct substantial experiments using a real-world taxi trajectory dataset to evaluate the effectiveness and efficiency of our proposed MTrajRec.

2.2 Overview

2.2.1 Preliminaries

Definition 1. ***Trajectory.*** *A trajectory τ can be defined as a sequence of GPS positions with timestamps, i.e, $\tau = \langle p_1, p_2, \cdots, p_n \rangle$, where $p_i = \langle lat, lng, t \rangle, \forall i, 1 \leq i \leq n$, which captures the latitude and longitude of the GPS position at timestamp t.*

Definition 2. ***Road Network.*** *A road network is a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, v_2, \cdots, v_K\}$ is a set of nodes representing intersections of road segments, and $\mathcal{E} = \{e_1, e_2, \cdots, e_L\}$ refers a set of edges representing the road segments which connect nodes v in $\mathcal{V}$. For each $e \in \mathcal{E}$, it contains three properties: 1) start and end nodes, indicating the start and end GPS position of a road segment; 2) length, which refers to the distance of a road segment in meter; 3) level, which indicates the type of road, such as highway, street, etc., with different colors shown in Fig. 2.2(a).*



(a) Road Segments Properties (b) Moving Ratio r

Figure 2.2: Examples of Preliminary Concepts

Definition 3. ***Map Matching.*** *Due to GPS device measurement errors, the GPS data is not precise. Map matching is a procedure to convert a sequence of raw latitude/longitude coordinates, so that the raw GPS points will be projected onto the road network.*

Definition 4. ***Map-matched Trajectory Point.*** *A map-matched trajectory point is denoted as $a = \langle e, r, t \rangle$, where e is the road segment ID, r is the moving ratio, which represents the ratio of moving distance over the length of the road segment, and t is the timestamp. Fig.2.2(b) gives a detailed explanation of r. With the road segment ID e and the moving ratio r, we can uniquely represent a location on the road network. The convert function*

is formulated as follows:

$$\begin{aligned} p.lat &= a.e.start.lat + a.r * (a.e.end.lat - a.e.start.lat) \\ p.lng &= a.e.start.lng + a.r * (a.e.end.lng - a.e.start.lng) \\ p.t &= a.t \end{aligned} \tag{2.1}$$

Definition 5. ***Sampling Rate.*** *A sampling rate ϵ is the time difference between two consecutive sampled points for a trajectory, which usually depends on device settings.*

Definition 6. ***Map-matched ϵ-Sampling Rate Trajectory.*** *A map-matched trajectory $\tilde{\tau}$ with ϵ-sampling rate is a sequence of map-matched trajectory points, i.e, $\tilde{\tau} = \langle a_1, a_2, \cdots, a_m \rangle$, where $a_j = \langle e, r, t \rangle, \forall j, 1 \leq j \leq m$ and $a_{j+1}.t - a_j.t = \epsilon$. For simplicity, we name $\tilde{\tau}$ as ϵ-MM trajectory.*

Note that although an ϵ-MM trajectory $\tilde{\tau}$ is uniformly sampled, points in a trajectory τ may not be uniformly distributed in timestamps, which is more challenging than uniformly distributed. Besides, the ϵ-MM trajectory $\tilde{\tau}$ is map matched on the road network, while the trajectory τ is not perfectly constrained on the road network due to GPS noises. Usually, points in trajectory τ are collected with a low sampling rate.

2.2.2 Problem Definition

Given a low-sampling-rate trajectory $\tau = \langle p_1, p_2, \cdots, p_n \rangle$ and a target sampling rate ϵ, we aim to recover the real map-matched ϵ-sampling-rate trajectory $\tilde{\tau} = \langle a_1, a_2, \cdots, a_m \rangle$. This is to say, for each low-sampling-rate trajectory, we will infer its missing points and map match it onto the road network simultaneously.

2.3 Methodology

To solve the trajectory recovery problem, we are inspired by the classic Seq2Seq model [103], since our trajectory recovery problem is similar to the machine translation problem, where low-sampling-rate trajectories can be treated as English sentences and ϵ-MM trajectories can be treated as the same sentences in French. Thus, Seq2Seq structure could be potentially used to solve the trajectory recovery problem. However, the challenges of our problem mentioned in Sec. 2.1 prevent the Seq2Seq model from solving it, since the Seq2Seq model can generate locations step by step but cannot guarantee the generated

trajectories are constrained on the road network in an end-to-end manner. Hence, we propose a novel model – MTrajRec, which can recover missing points and map match them onto the road network simultaneously.

In this section, we first introduce the basic structure of a multi-task Seq2Seq model to tackle challenge (1) *map constraint* (see Sec. 2.3.1), and then present three enhancement components of our MTrajRec: a constraint mask layer, which deals with challenge (2) *coarse point representation* (see Sec. 2.3.2), an attention mechanism, which learns global correlations (see Sec. 2.3.3), and an attribute module to solve challenge (3) *diverse complex factors* (see Sec. 2.3.4). In the following, we elaborate them in details.

2.3.1 Multi-task Seq2Seq Structure

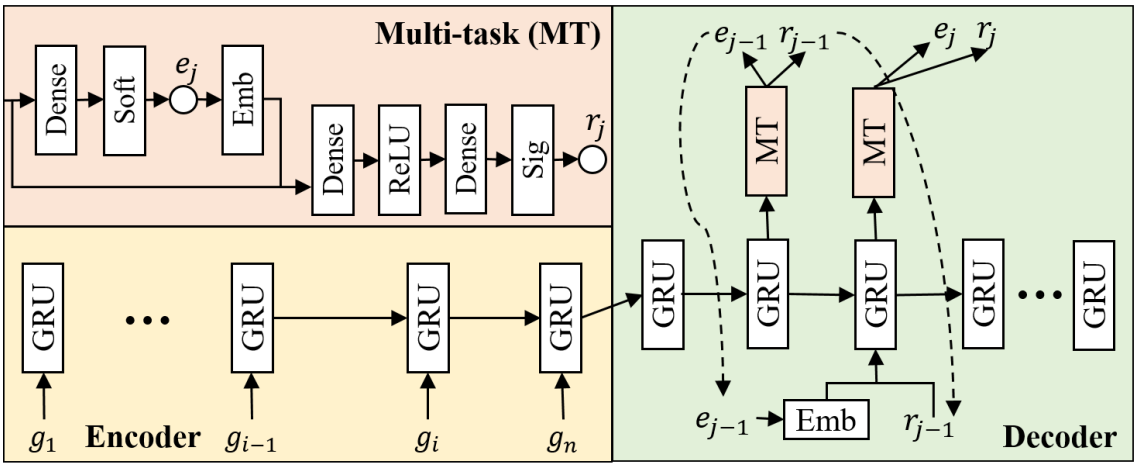


Figure 2.3: Structure of Basic MTrajRec

As illustrated in Fig. 2.3, MTrajRec is composed of an encoder and a decoder. The encoder learns sequential dependencies for low-sampling-rate trajectories, while the decoder predicts road segment ID e and moving ratio r iteratively, using the previous output as the input vector.

Encoder. The low-sampling-rate trajectory $\tau = \langle p_1, p_2, ..., p_n \rangle$ is encoded into a single vector to capture the spatial and temporal dependencies of τ. We refer the single vector as a context vector. Instead of directly using coordinates, we convert the GPS locations into discrete units. As [138] mentioned, it is more reliable and easy to model ID sequence than the original numerical sequence. Each numerical coordinate p_i can be converted to a unit, which is described as $g_i = \langle x_i, y_i, tid_i \rangle, \forall i, 1 \leq i \leq n$, where x_i and y_i represent i-th grid cell and $tid_i = \lfloor \frac{t_i - t_0}{\epsilon} \rfloor$ is the index of the points in the target ϵ-MM trajectory. Here,

ϵ is the target sampling rate and t_i is the timestamp of i-th point in the low-sampling-rate trajectory τ. The reason that we extract tid is to help the model learn how many points should be interpolated between two consecutive points. The low-sampling-rate trajectory τ is updated to $\tau' = \langle g_1, g_2, ..., g_n \rangle$. As can be seen in Fig. 2.4(a), trajectory $\tau = \langle p_1, p_2, p_3 \rangle$ is updated to $\tau' = \langle g_1, g_2, g_3 \rangle$ with $tid = \langle 1, 3, 7 \rangle$. Note that tid is not continuous because the low-sampling-rate trajectory is not uniformly sampled.

Since the trajectory data have sequential dependencies, we adopt the Gated Recurrent Unit (GRU) [16] as the encoder to obtain the context vector of low-sampling-rate trajectory. Such context vector will be used as the initial hidden state for the decoder. GRU is a variant of Long Short-term Memory networks (LSTM), which is capable of learning long-term dependencies for sequential data without performance decay. GRU sequentially updates a hidden-state by introducing an update gate $\mathbf{z}$ and a reset gate $\mathbf{r}$ to control the flow of information through the time steps. At each time step $i \in \{1, 2, \cdots, n\}$, the hidden-state vector $\mathbf{s}_i$ is:

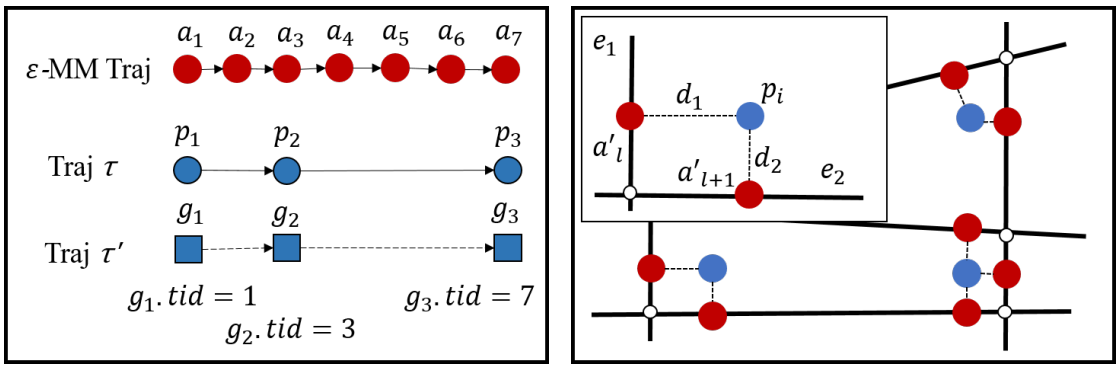
$$\begin{aligned} \mathbf{z}_i &= \sigma(\mathbf{W_z} \cdot [\mathbf{s}_{i-1}, \mathbf{g}_i] + \mathbf{b_z}) \\ \mathbf{r}_i &= \sigma(\mathbf{W_r} \cdot [\mathbf{s}_{i-1}, \mathbf{g}_i] + \mathbf{b_r}) \\ \tilde{\mathbf{s}}_i &= \tanh(\mathbf{W_s} \cdot [\mathbf{r}_i * \mathbf{s}_{i-1}, \mathbf{g}_i] + \mathbf{b_s}) \\ \mathbf{s}_i &= (1 - \mathbf{z}_i) * \mathbf{s}_{i-1} + \mathbf{z}_i * \tilde{\mathbf{s}}_i, \end{aligned} \tag{2.2}$$

where $\mathbf{W_x}$ represents the weights for the respective gate(x) neurons and $\mathbf{b_x}$ is the bias for the respective gate(x). To simplify, for getting the context vector of low-sampling-rate trajectory, the encoder derives the hidden state $\mathbf{s}_i$ as:

$$\mathbf{s}_i = GRU(\mathbf{s}_{i-1}, \mathbf{g}_{i-1}), \tag{2.3}$$

where the last state $\mathbf{s}_n$ will be considered as the context vector as well as the initial hidden state for the decoder.

Decoder. The decoder is used to recover the low-sampling-rate trajectory to the ϵ-MM trajectory $\tilde{\tau} = \langle a_1, a_2, \cdots, a_m \rangle$. As we mentioned before, the standard Seq2Seq model can predict the numerical coordinates but cannot guarantee the trajectories being map matched onto the road network. To tackle this challenge, instead of directly predicting the coordinate, we propose to predict road segments ID e and moving ratio r to make sure the predicted location must be constrained on the road network. We leverage multi-task learning [12], which solves those two tasks at the same time by sharing parameters



(a) Computation of tid (b) Constraint Distance

Figure 2.4: Illustration of concepts

across different tasks, since those two tasks are correlated.The input of the decoder is the concatenation of the embedding of the road segment ID e vector and moving ratio r. This is because road segment ID e is a categorical value which cannot directly be fed into the neural network. Comparing with the one-hot encoding [29], the embedding method can effectively reduce the input dimension and learn the semantic meanings for road segment ID e.

The top left in Fig. 2.3 demonstrates detailed information of the multi-task block, which predicts road segment ID e_j and moving ratio r_j simultaneously. After applying the GRU layer to get the output vector, we first use a dense layer with soft-max function to predict the road segment ID e_j. Then, the embedding of predicted e_j and the output vector from GRU layer are concatenated to go through the fully connected layers with a sigmoid function to predict moving ratio r_j. Since the moving ratio r_j highly depends on the related road segment ID e_j, we design a "series connection" mechanism to predict r_j. Note that although r_j depends on e_j, e_j and r_j have an influence on e_{j+1} and r_{j+1}. Thus, both road segment ID e_j and moving ratio r_j will be used as the input of the decoder.

To this end, for generating a missing point, our decoder derives the hidden state $\mathbf{h}_j$ as:

$$\mathbf{h}_j = GRU(\mathbf{h}_{j-1}, e_{j-1}, r_{j-1}). \tag{2.4}$$

Once we obtain the the hidden state $\mathbf{h}_j$ from the decoder, we further apply multi-task block to predict road segment ID e_j with a softmax function and to infer moving ratio r_j with a sigmoid function.

The structure mentioned above can guarantee the recovered trajectories constraint on the road network. However, challenges of inaccurate discrete units and diverse complex

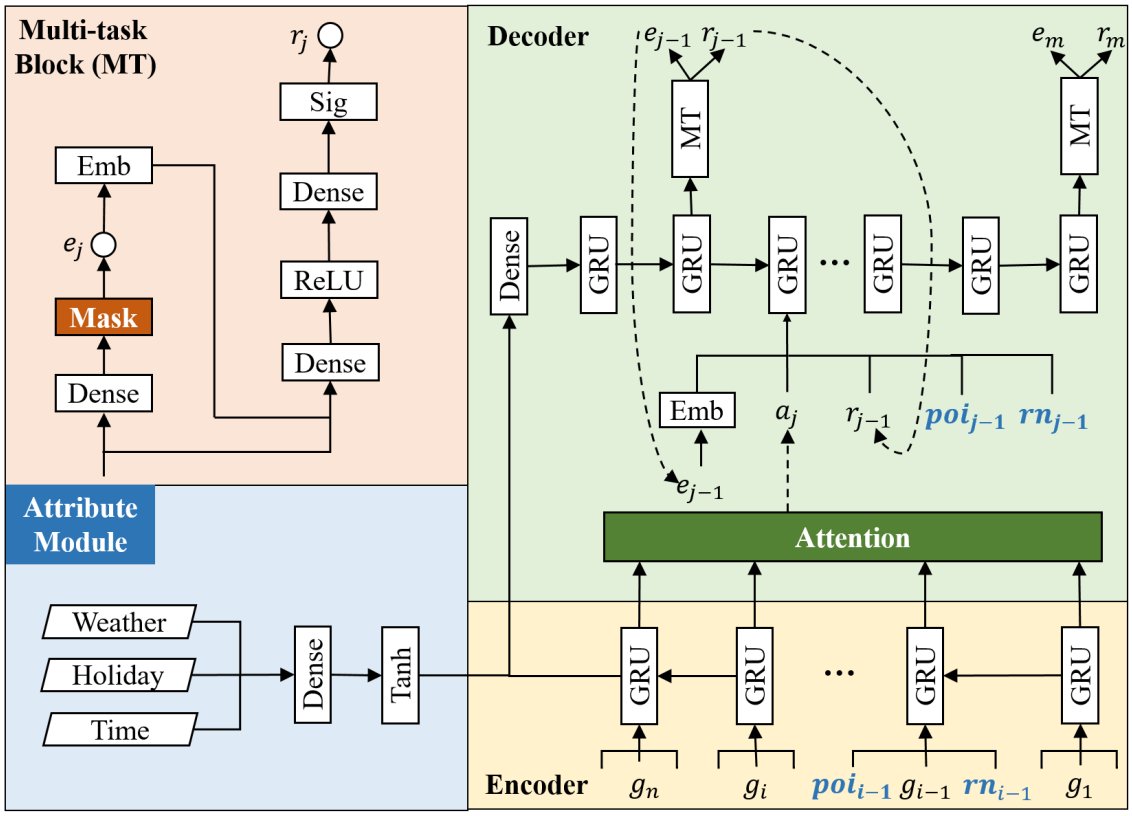


Figure 2.5: Structure of MTrajRec

factors affect the effectiveness of the trajectory recovery. In the following sections, we introduce three components to improve the performance: constraint mask layer, attention mechanism and attribute module. The overall structure of MTrajRec is illustrated in Fig. 2.5.

2.3.2 Constraint Mask Layer

As we mentioned in challenge (2), most of the previous studies [102, 28] convert the numerical coordinates into discrete units to mining trajectories since it can reduce training complexity. However, mapping the GPS coordinates into grid cells makes the precise information lost, which brings difficulties to the fine-grained MM trajectory recovery. Thus, there is a trade-off between complexity and accuracy by using discrete units or not. To tackle this challenge, we devise a *constraint mask layer* [130, 80]. With the discrete units in the encoder and the constraint mask layer in the multi-task block, we can benefit both from less complexity and higher accuracy.

In our model, we first define a *distance weight function* f [130], which represents the

influence of point p_i to candidate map matched point a'_ℓ based on the Euclidean distance between them, where p_i is the i-th sample from low-sampling-rate trajectory τ and a'_ℓ is the map-matched point on road segment ID e_ℓ. Note that such distance is caused by the sensing errors of GPS and theoretically p_i can be projected onto any road segment in the road network. As shown in Fig. 2.4(b), the blue points indicate the original GPS points in low-sampling-rate trajectory τ, while the red points are generated by projecting the original points to road segments. Let $d_{i,\ell}$ be the Euclidean distance of raw GPS point p_i and candidate point a'_ℓ, then $f(d_{i,\ell})$ denotes the impact of distance $d_{i,\ell}$ on the original point p_i. Following [130], we use an exponential function to capture the influence of distance, i.e., $f(d) = e^{-\frac{d^2}{\beta^2}}$, where β is a parameter with respect to the road network ($\beta = 15$ in our work). In practice, we only consider matching the blue points that are not far away from road segments [80]. We set 0 probability of any distance from a road segment that is more than 50 meters away from p_i. Note that we only calculate the distance weight for points existing in the low-sampling-rate trajectories, so that we set 1 probability of all road segments for missing values. Finally we combine such function with softmax in multi-task block as the *constraint mask layer* which is formally defined as:

$$\mathbf{c}_{j,\ell} = \begin{cases} f(d_{j,\ell}), & \text{if } j \in \{g_i.tid | i \in [1,n]\} \text{ and } d_{j,\ell} < 50 \\ 0, & \text{if } j \in \{g_i.tid | i \in [1,n]\} \text{ and } d_{i,\ell} \geq 50 \\ 1, & \text{otherwise} \end{cases}, \tag{2.5}$$
$$P(\hat{e}_j | \mathbf{h}_j) = \frac{\exp(\mathbf{h}_j^\top \cdot \mathbf{w}_c) \odot \mathbf{c}_j}{\sum_{c' \in C} \exp(\mathbf{h}_j^\top \cdot \mathbf{w}_{c'}) \odot \mathbf{c}_j},$$

where $\mathbf{w}_c$ is the c-th column vector from a trainable parameter matrix $\mathbf{W^C}$. Note that we use $argmax$ to get the final prediction of road segment ID $\hat{e}_j$. With the constraint mask layer, points in low-sampling-rate trajectory τ would be projected onto limited road segments instead of the whole road segments space.

2.3.3 Attention Mechanism

As above mentioned, we only consider local region constraint, while global correlations of low-sampling-rate trajectories are not modeled. Inspired by the attention mechanism that is widely used in natural language translation [107, 7], we introduce the attention mechanism [7] into the decoder.

The goal of attention mechanism is to compute the *similarity* between query vector

(i.e., current hidden state in the decoder) and key vectors (i.e., outputs from the encoder) to generate a context vector $\mathbf{a}$. Thus, the hidden state $\mathbf{h}_j$ in the decoder is updated to:

$$\mathbf{h}_j = GRU(\mathbf{h}_{j-1}, e_{j-1}, r_{j-1}, \mathbf{a}_j), \tag{2.6}$$

where the context vector $\mathbf{a}_j$ is computed by a weighted sum of all the output vectors $\mathbf{s}$ from the encoder, where $\mathbf{a}_j$ is formulated as:

$$\begin{aligned} \mathbf{a}_j &= \sum_{i=1}^{n} \alpha_{j,i} \mathbf{s}_i, \\ \alpha_{j,i} &= \frac{\exp(u_{j,i})}{\sum_{i'=1}^{n} \exp(u_{j,i'})}, \\ u_{j,i} &= \mathbf{v}^\top \cdot \tanh(\mathbf{W_h} \mathbf{h}_j + \mathbf{W_s} \mathbf{s}_i), \end{aligned} \tag{2.7}$$

where $\mathbf{v}$, $\mathbf{W_h}$ and $\mathbf{W_s}$ are the learnable parameters, and $\mathbf{h}_j$ denotes the current mobility status from the decoder.

2.3.4 Attribute Module

The recovery accuracy is also influenced by traffic conditions since vehicles are not moving at a constant speed in the real world. The traffic speed is affected by spatial context, temporal dependencies and external factors, such as weather conditions, time, POI distribution and etc [64]. To tackle our challenge (3), diverse complex factors, we incorporate such information to devise an attribute module in our model. The attribute module includes two types of factors:

- The environmental context features $\mathbf{f_e}$. As shown in Fig. 2.5, we incorporate the attributes of weather conditions (sunny, rainy, cloudy, etc), holiday (whether today is a holiday or not), and time (hour of the day) to extract the effect of the environmental context features. We attempt to implement the environmental context features as an independent block. Note that these factors are categorical values which cannot be directly fed into the neural network. We use one-hot encoding to represent them since the dimension of each factor is not large. After concatenating the one-hot encoding results, a fully connected network (FCN) is implemented to learn the embedding of all the features, which is fused with the last hidden state of the encoder as the initial hidden layer of the decoder, i.e. $\mathbf{h}_0 = FCN(\mathbf{s}_n, Emb(\mathbf{f_e}))$. The intuition is that the environmental context features do not change significantly with trajectories mobility. Therefore, we

combine the embedding vector with the context vector of the encoder and input them at the beginning of the decoder.

- The spatial context features $\mathbf{f_s}$. Different from the stable environmental context features, the spatial context features, such as POI and road network, change rapidly as vehicles move. Thus, we input such features for each time step in the encoder and the decoder respectively. We use the POIs density of different categories as POIs features, and extract the properties of road network as network features, i.e., number of intersections and level of road segments. Thus, the hidden state $\mathbf{s}_i$ of the encoder and the hidden state $\mathbf{h}_j$ are renewed as,

$$\begin{aligned} \mathbf{s}_i &= GRU(\mathbf{s}_{i-1}, \mathbf{g}_{i-1}, \mathbf{f}_{\mathbf{s}j-1}), \\ \mathbf{h}_j &= GRU(\mathbf{h}_{j-1}, e_{j-1}, r_{j-1}, \mathbf{a}_j, \mathbf{f}_{\mathbf{s}j-1}). \end{aligned} \tag{2.8}$$

2.3.5 Algorithm Training

We finally elaborate the training procedure of our model which is trained end to end. Recall that during the training phrase, we predict road segment ID and moving ratio simultaneously. For the road segment ID prediction, we adopt the cross entropy as the loss function:

$$\begin{aligned} \mathcal{L}_1(\theta) = -\sum_{(\tau,\tilde{\tau})\in\mathcal{D}_{sub}} \sum_{j=1}^{|\tilde{\tau}|} \sum_{\ell=1}^{L} \mathbf{1}\{a_j.e = e_\ell\} \log(P_\theta(\hat{a}_j.e = e_\ell | \mathbf{d}_{1:j-1})), \\ \text{s.t. } \mathbf{d}_{j-1} = (\tau, \tilde{\tau}_{1:j-1}, \mathbf{f}_{\mathbf{s}j-1}, \mathbf{f}_\mathbf{e}), \end{aligned} \tag{2.9}$$

where τ is the low-sampling-rate trajectory, $\tilde{\tau}$ is the target ϵ-MM trajectory, $|\tilde{\tau}|$ is the length of the ϵ-MM trajectory, L is the size of road segments, $a_j.e$ is the ground truth of road segment ID, $\hat{a}_j$ is the prediction, P_θ represents the neural network for predicting road segments, $\mathbf{f_s}$ and $\mathbf{f_e}$ are external factors vectors. $\mathcal{D}_{sub}$ means the dataset consisting of low-sampling-rate trajectories and ϵ-MM trajectories, which is the subset of the training set $\mathcal{D}$.

We also implement the mean squared error as the loss function for the moving ratio

prediction:

$$\begin{aligned} \mathcal{L}_2(\theta) = - \sum_{(\tau', \tilde{\tau}) \in \mathcal{D}_{sub}} \sum_{j=1}^{|\tilde{\tau}|} (a_j.r - R_\theta(\mathbf{d}_{j-1}))^2, \\ \text{s.t. } \mathbf{d}_{j-1} = (\tau', \tilde{\tau}_{1:j-1}, \mathbf{f_s}_{j-1}, \mathbf{f_e}), \end{aligned} \tag{2.10}$$

where $a_j.r$ is the ground truth of real moving ration and R_θ represents the neural network for predicting moving ratio and other parameters are the same as above.

Overall, the final model optimization function is weighted combined with these two loss functions and formulated as:

$$\mathcal{L}_t = \mathcal{L}_1(\theta) + \lambda \mathcal{L}_2(\theta) \tag{2.11}$$

where λ is a tunable parameter that linearly balances the trade-off between two tasks in our work.

Algorithm 1 illustrates the training process of the MTrajRec model. During the training process, we apply the gradient descent approach to update parameters $\boldsymbol{\theta}$, with learning rate η and a predefined $epochs_{max}$. We first extract attributes including external factors $\mathbf{f_e}$ and spatial context features $\mathbf{f_c}$. Then, we select one pair of trajectories τ and $\tilde{\tau}$. Lastly, we update MTrajRec parameters θ by using Eq 2.11, with η as the step size (Line 5).

Algorithm 1 MTrajRec Training

Require: Trajectories $\mathcal{T}$, ϵ-MM Trajectories $\tilde{\mathcal{T}}$, features $\mathbf{f_e}, \mathbf{f_c}$, initialized parameters $\boldsymbol{\theta}$, learning rate lr, max iteration $epochs_{max}$.
Ensure: A well trained MTrajRec with parameters $\boldsymbol{\theta}$.
1: Extract attributes.
2: Sample a pair of trajectories τ and $\tilde{\tau}$.
3: **while** epoch $< epochs_{max}$ **do**
4: Calculate gradient $\nabla \mathcal{L}(\boldsymbol{\theta})$ using Eq 2.11.
5: Update $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla \mathcal{L}(\boldsymbol{\theta})$.
6: **end while**

2.4 Experimental Evaluation

2.4.1 Experimental Settings

2.4.1.1 Data Description

We validate the effectiveness of our model on a real-world taxi trajectory dataset and a road network from OpenStreetMap [1]. The dataset contains trajectories of $122,390$ drivers and 6.20 million GPS records in Jinan, Shandong over a period of 1 month, in September 2017. All the trajectories are completed sampled every 15 seconds. It covers a rectangular area from (36.6456, 116.9854) to (36.6858, 117.0692) which is around 7.47 km long and 4.47 km wide. There are $2,571$ road segments in the area.

We split the dataset into training set, validation set and test set with a splitting ratio of 7: 2: 1. Since the dataset is completely sampled, we generate low-sampling-rate trajectories by randomly sampling points from high-sampling-rate trajectories with a keep ratio $kr\%$. According to [130], taxis usually report their GPS positions with a low sampling rate to save communication and energy cost. More than 60% trajectory data is sampled every $2 \sim 6$ mins. Thus, we further vary the keep ratio of $kr\%$ in the set $\{6.25\%, 12.5\%, 25\%\}$ to evaluate the robustness of our proposed model. Since the original trajectories are sampled every 15 seconds, the generated low-sampling-rate trajectories with $kr\% = 6.25\%, 12.5\%, 25\%$ can be considered as the average time interval of such trajectories is 4 mins, 2 mins and 1 min respectively. A smaller keep ratio indicates to a larger number of missing points in the low sampling rate trajectories. For the high-sampling-rate trajectories, we utilize HMM algorithm on the original trajectories [80] to get ground truth, since the map matching accuracy can reach as high as 99% with a sampling interval around $10 \sim 15$ seconds [80].

2.4.1.2 Evaluation Metrics

The purpose of our defined problem is to recover low-sampling-rate trajectories from free space to high-sampling-rate trajectories constrained onto the road network. Thus, we adopt both accuracy of road segments recovery and distance error of location inference to show the performance of our model and baseline methods as follows:

MAE & RMSE. We adopt two distance measurements to evaluate the location recovery performance. MAE is the mean absolute error and $RMSE$ is the root mean squared error

[1] `http://www.openstreetmap.org/`

between ground truth values and predicted values. However, as shown in Fig. 2.6, if we directly use earth distance to compute the error, the prediction of pre_2 is better than pre_1, while pre_2 cannot reach the ground truth through the dash line in the real world. Thus, we should calculate the distance error based on road network. We update earth distance to road network constrained distance to evaluate the distance error. The smaller values of MAE and $RMSE$ are, the better performance the model represents. MAE and $RMSE$ are formulated as:

$$\begin{aligned} MAE &= \frac{1}{m}\sum_{j=1}^{m}|dis(a_j,\hat{a}_j)|, \\ RMSE &= \sqrt{\frac{1}{m}\sum_{j=1}^{m}(dis(a_j,\hat{a}_j))^2}, \\ \text{s.t. } dis(a_j,\hat{a}_j) &= min(rn\_dis(a_j,\hat{a}_j), rn\_dis(\hat{a}_j,a_j)), \end{aligned} \tag{2.12}$$

where a_j is the ground truth location, $\hat{a}_j$ is the predicted map matched trajectory point and $rn\_dis(a_j,\hat{a}_j)$ is the distance of shortest path between prediction and ground truth. Since the road network is a directed graph, the road network based distance from a_j to $\hat{a}_j$ is not equal to the distance from $\hat{a}_j$ to a_j, thus we use the minimal distance as the final error.

Recall & Precision. We use recall and precision to evaluate the performance of route recovery by comparing the recovered road segments $\mathcal{E}_R$ to the ground truth $\mathcal{E}_G$. Following previous work [20, 53], $Recall$ is defined as $recall = \frac{|\mathcal{E}_R \cap \mathcal{E}_G|}{|\mathcal{E}_G|}$, and $Precision$ is denoted as $precision = \frac{|\mathcal{E}_R \cap \mathcal{E}_G|}{|\mathcal{E}_R|}$. The larger values of $Recall$ and $Precision$ indicate that the methods predict road segments more accurately.

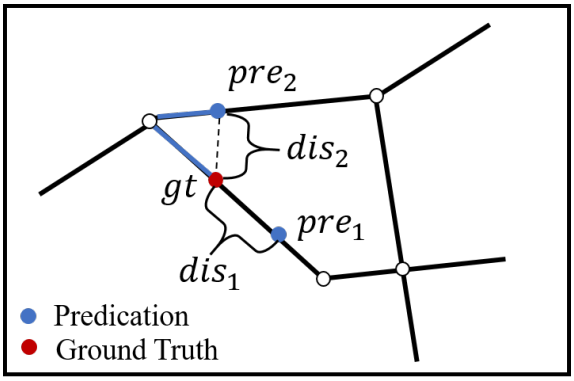


Figure 2.6: Examples of Distance Error

2.4.1.3 Baseline Algorithms

We compare our MTrajRec with several representative baselines. To our best knowledge, there is no existing solution that can recover low-sampling-rate trajectories to high-sampling-rate trajectories and map match onto the road network simultaneously. Therefore, we design the following two-stage pipelines for comparison:

- **Linear** [38] + **HMM** [80]: It recovers the locations by assuming trajectories are moving straightly and uniformly and then matches trajectories onto the road network. Here we implement HMM as the map matching algorithm due to its high accuracy in high-sampling-rate trajectories map matching [80].
- **DHTR** [113] + **HMM** [80]: It devises a subseq2seq model with Kalman Filter to recover trajectories in free space, which is the state-of-the-art method in the field of trajectory recovery. After getting the recovered high-sampling-rate trajectories, we introduce HMM to match them onto the road network.
- **DeepMove** [28] + **Rule**: It incorporates multiple factors with recurrent neural network to predict human mobility for next-step trajectory prediction. We adapt it to this task by consecutively predicting each missing road segment and then use the centric location as the final prediction.

2.4.1.4 Variants

To evaluate each component of our proposed model, we also compare it with different variants of MTrajRec:

- **MTrajRec-noCons**: We remove constraint mask layer in multi-task block to evaluate the relevance of this part.
- **MTrajRec-noAttn**: We remove the attention mechanism to detect the importance of attention mechanism.
- **MTrajRec-noAtts**: We remove the attribute module from our model to reveal the significance of this component.

2.4.1.5 Implementations

We train the deep neural network with machine learning library Pytorch, version 1.7.1. Our experiments run on a GPU server with 64 GB memory and Tesla V100 GPU.

Table 2.1: Overall performance comparison. The best result for each evaluation metric is in bold. A smaller keep ratio indicates a larger number of missing points in low-sampling-rate trajectories. Note that MAE and $RMSE$ are in km.

Methods	**6.25%**				**12.5%**				**25%**			
	Recall	Precision	MAE	RMSE	Recall	Precision	MAE	RMSE	Recall	Precision	MAE	RMSE
Linear+HMM	0.4019	0.3388	0.662	1.107	0.5122	0.4592	0.535	1.030	0.6291	0.6065	0.398	0.820
DHTR+HMM	0.5110	0.3601	0.637	1.099	0.5984	0.4379	0.485	0.958	0.6581	0.4916	0.368	0.844
DeepMove+Rule	0.6409	0.7754	0.339	0.696	0.6572	0.7924	0.311	0.666	0.6762	0.8085	0.279	0.628
MTrajRec-NoCons	0.6472	0.7835	0.303	0.648	0.6748	0.7998	0.260	0.596	0.7001	0.8066	0.227	0.575
MTrajRec-NoAttn	0.6837	0.7938	0.290	0.631	0.7105	0.8066	0.245	0.594	0.7308	0.8144	0.209	0.566
MTrajRec-NoAtts	0.6925	0.7981	0.278	0.628	0.7121	0.8073	0.238	0.603	0.7437	0.8116	0.199	0.567
MTrajRec	**0.6972**	**0.8018**	**0.270**	**0.610**	**0.7235**	**0.8137**	**0.229**	**0.590**	**0.7498**	**0.8198**	**0.194**	**0.564**

2.4.2 Results

2.4.2.1 Overall Performance

We compare our MTrajRec with the baseline models in terms of $Recall$, $Precision$, MAE and $RMSE$. The performance of different approaches with different keep ratios for trajectory recovery is presented in Table 2.1. We have the following observations:

1. The first two pipelines interpolate missing values for low-sampling-rate trajectories in free space and then map them onto the road network. Comparing these two models, we can see that DHTR + HMM is better than Linear + HMM, especially when the keep ratio is small. This is because DHTR is designed with advanced sequential neural networks, which is able to utilize the spatio-temporal information in low-sampling-rate trajectories, while linear interpolation cannot model complex mobility regularity.

2. DHTR + HMM and DeepMove + Rule are deep learning based method, but the performance of DeepMove + Rule is better than the former one. As the keep ratio decreases, i.e., the number of missing points increases as well as the uncertainty between two consecutive points grows, the performance of DHTR + HMM is worse than DeepMove + Rule. This is because DeepMove is directly trained to predict road segments, which can better capture the constraint of the road network compared with DHTR.

3. It is clear that our approach attains the best performance across all of the evaluation metrics with different keep ratios. Compared with other baselines, the improvement of ratio is the most significant at the lowest keep ratio. Specifically, $Recall$ and $Precision$ of MTrajRec outperform the best baseline by 8.7% and 3.4% when the keep ratio is 6.25%. MAE and $RMSE$ are reduced by 20.4% and 14.1% respectively. This proves the effectiveness of our MTrajRec in trajectory recovery. The fundamental reasons for such improvement lie in two aspects. First, we devise a multi-task Seq2Seq model to predict road segments and moving ratio simultaneously which is able to reduce error compared with two-stage pipelines. Second, we introduce several components such as the constraint mask layer, the attention mechanism and the attributes module to extract more information, which will be elaborated in next section.

2.4.2.2 Running Efficiency

To evaluate the efficiency of MTrajRec, we compare the running time of our algorithm with other baseline models. As can be seen in Fig. 2.7, MTrajRec and DeepMove+Linear

run much faster than other baseline models. This is because both of MTrajRec and DeepMove+Linear only need to make a forward computation of neural network to recover trajectories, which only requires $O(n)$ computation complexity. On the contrary, other two approaches rely on heavy computations of dynamic programming to do map matching task, with a computation complexity of $O(n^2)$. With the keep ratio increases, the running time of MTrajRec and DeepMove+Linear decreases, since less missing points need to be predicted. However, the running time of other two baseline methods increases since more and more points need to be matched by HMM. Although MTrajRec is slightly slower than DeepMove+Linear, the accuracy is much higher. Thus, such difference can be ignored.

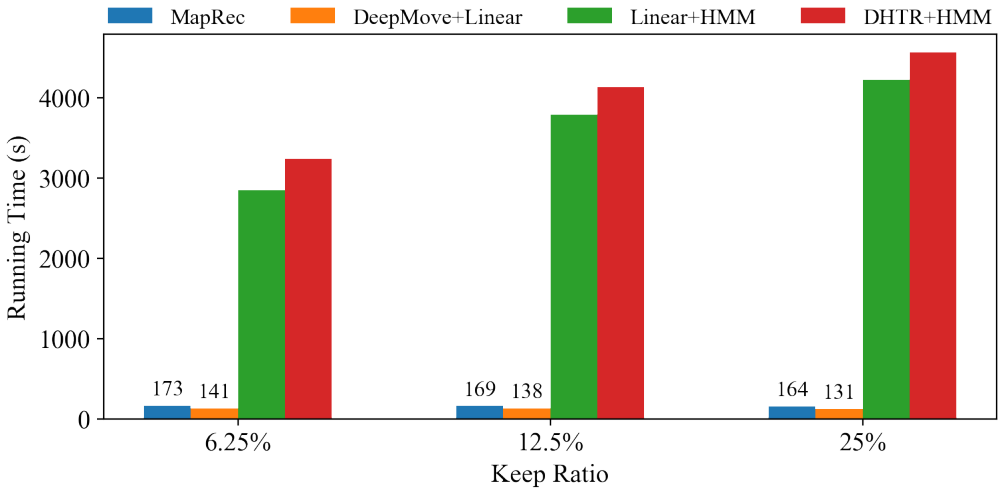


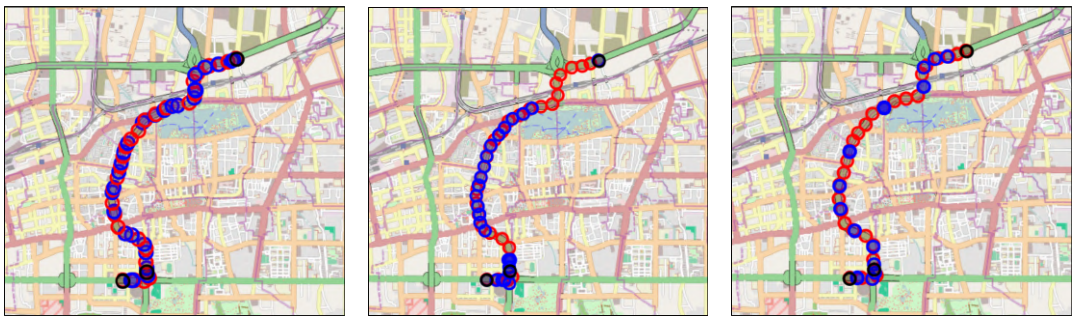
Figure 2.7: Running Efficiency

2.4.2.3 Importance of the Constraint Mask

We introduce a constraint mask layer into MTrajRec to overcome the limits of converting coordinates to grid cells. To evaluate the importance of constraint mask layer, we compare *MTrajRec-noCons* to MTrajRec. As can be seen from Table. 2.1, both MTrajRec and MTrajRec-noCons get a better performance as the keep ratio increases. But when the constraint mask layer is removed, the performance declines significantly. Especially, it causes MAE to grow 25% and $Recall$ to shrink 8% with 6.25% keep ratio. This is because the constraint masks introduce prior knowledge for the existing points, which enhances the missing information by utilizing grid cells.

2.4.2.4 Importance of the Attention Mechanism

In this section, we remove the attention mechanism from MTrajRec to test its contribution. Similar as above, the performance of both MTrajRec and MTrajRec-noAttn increase as the keep ratio increases because the missing points are reduced, making the recovery task easier. As illustrated in Table. 2.1, the results of MTrajRec-noAttn falls obviously comparing with MTrajRec. Particularly, MAE increases 7% and $Precision$ decreases 2% after removing the attention mechanism at the keep ratio 6.25%. A possible reason is that attention mechanism can effectively strengthen the spatial constraints for the missing locations.



(a) MTrajRec (b) DHTR+HMM (c) DeepMove+Rule

Figure 2.8: Screenshots of Trajectory Recovery. Black points represent the low-sampling-rate trajectory, points in red are ground truth coordinates of $15s$-MM trajectory and the blue points stand for predicted locations of $15s$-MM trajectory.

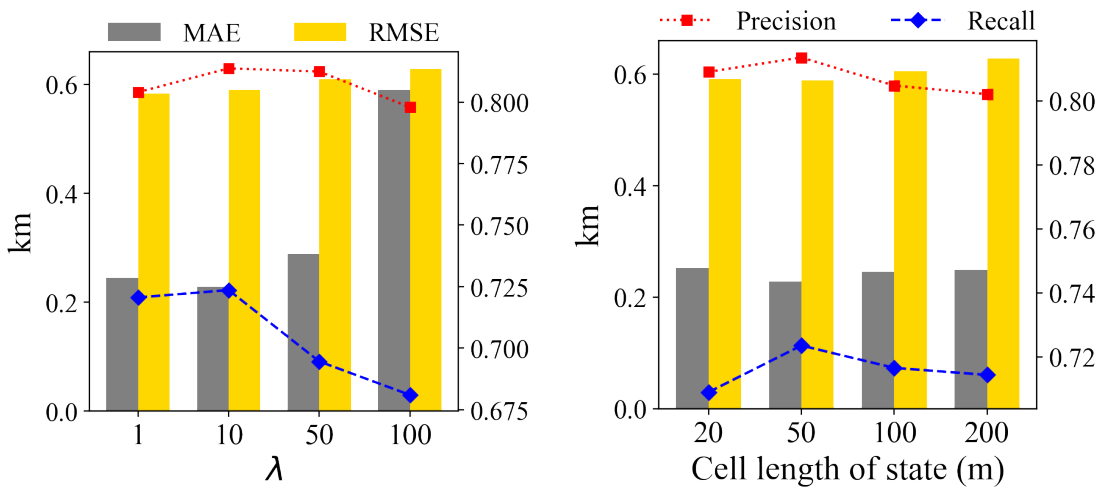
2.4.2.5 Importance of the Attribute Module

We also evaluate the relevance of the attribute module by removing all the external factors. Table. 2.1 shows that the results of MTrajRec-noAtts drops slightly when taking out the attribute module. The contribution of attribute module is not as significant as the constraint mask and the attention mechanism, i.e., the MAE of MTrajRec only drops 3% and the $Precision$ of MTrajRec only increases 0.7% when adding this component. Probably because the previous two provide enough constraints in the model.

2.4.2.6 Parameter Tuning

Apart from evaluating the components of our proposed model MTrajRec, there are two important parameters to tune in our model.

Weight of multi-task λ. To show the effectiveness of the multi-task learning component, we first evaluate our model under different combinations λ in range of 1, 10, 50 and 100.



(a) Varying of λ

(b) Varying of cell length

Figure 2.9: Parameters Tuning

As shown in Fig. 2.9(a), the gray and yellow bars show results of MAE and $RMSE$ and the red and blue lines present $Precision$ and $Recall$ respectively. We get the best performance when $\lambda = 10$, while the performance gets worse as the λ increases more or decreases. It indicates that our MTrajRec benefits from a good balance of this two tasks.

Cell length. As mentioned in Section 2.3, we map numerical coordinates into discrete units to extract spatial dependencies and reduce the computation complexity. When a smaller cell length is used, we can obtain more accurate spatial information but the number of grid cells increases leading to complex modeling problem and vice versa. Thus, there is a trade-off between accuracy and complexity with the setting of cell length. We vary the cell length to 20, 50, 100 and 200 meters to tune the parameter. Fig. 2.9(b) shows the varying results of different cell lengths. As can be seen, the performance achieves the best when the cell length is 50 meters. As the cell length increases to 100 and 200 meters, the $Precision$ and $Recall$ increase and MAE and $RMSE$ decrease, probably some of the spatial information is lost when mapping to grid cells. However, the performance also gets worse when the cell length decreases to 20 meters since the large number of grid cells increase the complexity of training.

2.4.3 Qualitative Analysis

Fig. 2.8 presents screenshots of the visualized recovery results of our MTrajRec compared with two baseline models (DHTR + HMM, DeepMove + Rule) on the same low-sampling-rate trajectory data at a keep ratio 6.25%. Black points represent the low-

sampling-rate trajectory. Points in red are ground truth coordinates of $15s$-MM trajectory and the blue points stand for predicted locations of $15s$-MM trajectory. It is clear that our MTrajRec finds the right route and the recovered positions are more reliable and adaptive than the other two baselines. Fig. 2.8(b) shows the recovery results of DHTR+HMM, which recovers the low-sampling-rate trajectory in free space and then implements a map matching algorithm to match the trajectory onto the road network. Although it predicts the right path, the locations are not correctly recovered especially no correct points are predicted in the top right area. A possible reason is that there are a large number of missing values to be predicted, while DHTR cannot recover locations with such low sampling rate. The keep ratio is only 6.25% which is about 4 times lower than the smallest keep ratio in DHTR [113]. Fig. 2.8(c) illustrates the recovery results of DeepMove + Rule, which employs a deep learning model to match the low-sampling-rate trajectory onto the road network and then uses the centric location as the final prediction. With the constraint of road network, DeepMove + Rule finds the right path as the left two methods. However, the recovered points move discontinuously and several blue points in the middle area fail to move forward. This is because DeepMove + Rule can only predict the possible road segments but it does not have the ability to infer moving speed of the vehicle.

2.5 Related Work

Trajectory Data Mining. Trajectory data mining [139] discovers various knowledge from massive trajectory data, namely a few, traffic condition prediction [65], travel time estimation [91, 37] and driver behavior learning [22, 92]. In particular, Hong et al. [37] leveraged heterogeneous information graph to solve estimated time of arrival task based on road network and high sampling rate vehicle trajectories. Dong et al. [22] proposed a deep-learning framework to analyze driving behavior based on trajectory data. Their empirical studies revealed that the performance of any approaches can become poor when the sampling rate is lower than 10 seconds. Such work relies on high-sampling-rate road network constrained trajectories, which can benefit from our work.

Trajectory Recovery. Recovering low-sampling-rate trajectory is an important problem to get more information and reduce uncertainty [113, 119, 118]. In particular, Wang et al. [113] recovered a high-sampling-rate trajectory from a irregular low-sampling-rate trajectory by integrating the subseq2seq with a calibration component of Kalman Filter. Xia et al. [119] proposed an attentional neural network model to densify individual

trajectory by recovering unobserved locations based on historical trajectories. Xi et al. [118] proposed a Bi-directional Spatial and Temporal Dependence and users' Dynamic Preferences model to identify missing POI check-in. Apart from these recovery methods, some works related to next-step or short-term location prediction can also be adopted for recovery [79, 28, 62]. However, such frameworks are implemented on free space, as opposed to our problem which aims to recover trajectories onto road network. Besides, most of the existing works model the sequence of location IDs rather than the numerical coordinate information.

Sequence-to-sequence Models. The sequence-to-sequence model (Seq2Seq) [103] is an architecture for domain translation, which has been widely used in trajectory data mining, namely but a few, trajectory generation, trajectory similarity learning, anomaly detection, etc [113, 83, 59, 69]. Park et al. [83] generated the future trajectory sequence of surrounding vehicles via Seq2Seq model. Li et al. [59] proposed a Seq2Seq framework to learn representations of trajectories to support trajectory similarity computation. However, to our best knowledge, we are the first one to employ multi-task learning into Seq2Seq model to recover map constrained trajectories.

2.6 Conclusion

In this paper, we propose a novel end-to-end deep learning model MTrajRec for recovering low-sampling-rate trajectories to high-sampling-rate map-matched trajectories. We introduce multi-task learning into Seq2Seq model to ensure the generated trajectories are map matched onto the road network. The *constraint mask, attention mechanism and attribute module* are implemented to improve the performance. The experimental results illustrate that MTrajRec outperforms state-of-the-art works by reducing recover error around 20.4% with the keep ratio at 6.25% on a real world dataset. As future work, we plan to extend the proposed model by incorporating more user preference, e.g., user identification information.

Chapter 3

Human Mobility Signature Identification

3.1 Introduction

Given the historical movement trajectories of a set of individual human agents (e.g., pedestrians, taxi drivers) and a set of new trajectories claimed to be generated by a specific agent, the Human Mobility Signature Identification (HuMID) problem aims at validating if the incoming trajectory was indeed generated by the claimed agent. The HuMID problem has many real-world applications. Fig. 3.1 shows a few such examples. One of the major applications is automatic driver identification for taxi and rider-sharing services. According to the New York City Taxi and Limousine Commission (TLC) released statistics, there were on average $850,000$ trips taken by taxis and ride-sharing services per day in New York City in 2018 [4]. Meanwhile, the safety concerns have been raised by people recently. For example, some unauthorised drivers are reported to have taken the place of authorised drivers, and behave offensively towards passengers. Companies like Uber have taken actions to ensure the safety of passengers by enabling on-trip reporting from the APP [101, 78]. HuMID can help identify the above illegal driver substitutions as early as possible and help improve the safety of the passengers. Another example is insurer identification in the auto insurance industry. Insurance companies need to make sure that a vehicle was driven by the insured driver rather than others when the insurer filed a claim. All of these examples are downstream applications of and can benefit from solving HuMID problems.

Many prior works pay attention to the driving behavior identification problem, an in-

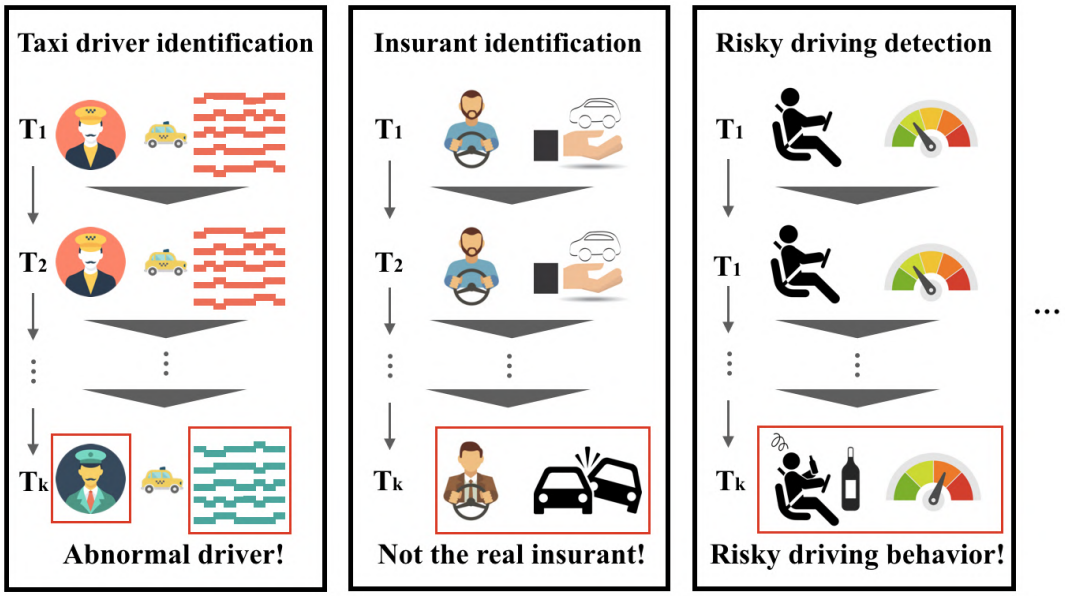


Figure 3.1: Applications of HuMID problem

stance of the HuMID problem. Hallac et al. [32] identified driver using automobile sensor data from a single turn. They monitored 12 sensors installed inside and outside the vehicle and implemented a hand-crafted rule-based classifier, which classifies up to 5 drivers. Chowdhury et al. [18] extracted 137 statistical features from smartphone sensors and used a random forest classifier to classify trajectories into small groups of 4 to 5 drivers. Kieu et al. [50] presented a multi-task learning model which captured geographic features and driving behavior features of trajectories in 3D images as input to perform trajectory clustering and driver identification. Oh and Iyengar [81] used inverse reinforcement learning in sequential anomaly detection problem. They estimated reward function for each driver and evaluated 10 individuals from GeoLife-GPS dataset and aggregated normal behaviors of taxi drivers from Taxi Service Trajectory dataset.

Nonetheless, there exist significant limitations when implementing these methods in real-world applications. First, some of these works require additional data rather than the GPS records by installing sensors on the vehicles, for example, 12 sensors in Hallac et al.'s work [32]. However, few vehicles is equipped with these additional sensors, and it will be costly to install sensors to the vehicles. Second, most existing works can only deal with a small group of drivers because they employ classification or clustering approaches. For example, Chowdhury et al. [18] employed random forest to classify the trajectories of 4 to 5 drivers, and Oh et al. [81] estimated 10 reward functions for 10 drivers by using

inverse reinforcement learning. In real-world cases, the pool of drivers is large. Thus, these methods are hard to be implemented. Third, a group of previous works require that all the drivers be known in advance [32, 18, 81]. Those methods are unsuitable for applications where only a subset of the drivers is known at training.

To address these limitations, in this paper we propose a Spatio-temporal Siamese networks (ST-SiameseNet) framework to identify the behavior of a large group of human agents (e.g., drivers) by using only their movement trajectory data. Since GPS devices are widely equipped on vehicles and smart phones nowadays, the data ST-SiameseNet requires can be easily collected. Also, ST-SiameseNet can deal with large groups of human agents in a single model and be used on new agents who are previously-unseen from training pool. To be more specific, we first extract different transit modes of the agents from the trajectory data. For example, there are **two** transit modes in taxi driving, i.e., the seeking trajectory where the vehicle has no passengers on-board, and the driving trajectory where the vehicle has passengers on-board. Besides, we extract different profile features and online features from the historical trajectories of each agent to augment the performance of ST-SiameseNet. Then, we input the trajectories together with the profile features to ST-SiameseNet pair-wisely and train the ST-SiameseNet to identify the similarity of each pair of inputs. Experiments on a real-world taxi trajectory dataset show that ST-SiameseNet outperforms all baselines in identification performances. *Our main contributions* are summarized as follows:

- We formulate the Human Mobility Signature Identification (HuMID) problem as a predictive analysis problem and, for the first time, employ the idea of the Siamese network to identify agents by their "mobility signatures" from solely their trajectory data.
- We design a novel ST-SiameseNet framework that can handle multimodal trajectory data. We also utilize both profile features and online features extracted from the agents' trajectory data to train ST-SiameseNet.
- We conduct substantial experiments using a real-world taxi trajectory dataset to evaluate the performance of our proposed ST-SiameseNet.

The remainder of the paper is organized as follows. Section 2 presents an overview of the key ideas of our research problem. Section 3 provides detailed methodology of our proposed model. We discuss the experimental results on different datasets and the related work in sections 4 and 5. Section 6 concludes the paper.

3.2 Overview

3.2.1 Problem Definition

In this section, we introduce some important definitions and formally define the problem.

Definition 7. ***Human-generated spatio-temporal trajectory*** tr. *With the wide use of GPS sets, people can generate massive spatio-temporal data while they are using the devices equipped with GPS sets, e.g., the GPS records of vehicles, smartphones, smart watches, etc. Each GPS point* p *consists of a location in latitude* lat *and longitude* lng*, and a time stamp* t*, i.e.* $p = \langle lat, lng, t \rangle$*. A trajectory* tr *is a sequence of GPS points with a label of the agent* a *who generated the data, denoted as* $tr = \{a, \langle p_1, p_2, ..., p_n \rangle\}$*, where the set of trajectories is* $\mathcal{T}$*.*

Definition 8. ***Transit mode.*** *Transit modes are defined as a set of categories of trajectories, where each category is generated under a different mobility pattern. For example, taxi driving trajectories can be categorized into two modes, i.e., with and without any passenger on-board. Private car trajectories can be grouped into commute and recreational driving trajectories, etc. In this paper, we use taxi driving as the application. The seeking trajectory* $\mathcal{T}_s$ *is the sequence of GPS records while the vehicle is without any passengers on-board, and the driver is seeking for passengers to serve. The driving trajectory* $\mathcal{T}_d$ *is the sequence of GPS records while the vehicle is with passengers on-board, and the driver is taking the passengers to the destination.*

Definition 9. ***Profile feature*** $\mathbf{f}_p$. *Each agent has unique personal (or profile) characteristics which can be extracted from his/ her trajectory data, such as frequent start/end locations, average trip time duration, and preferred geographic area, working as different dimensions* $f_{p,i}$ *of the profile features, where* i *is the* i*-th dimension of these features. The profile features of each agent can be extracted in different time period. Here we denote time period as* T*, where* T *can be one hour, one day or one week, etc.*

Definition 10. ***Online feature*** $\mathbf{f}_o$. *Online features represent agents' mobility patterns resulting from the agent's personal judgment, experience and skills, such as speed, acceleration, turning left, turning right of each grid cell, working as different dimensions* $f_{o,i}$ *of the online features, where* i *is the* i*-th dimension of these features. For each trajectory, we build the online features.*

Problem definition. Given a set of historical trajectories $\mathcal{T}$ collected from a group of agents A in time periods $T_0, T_1, ..., T_t$, we aim to develop a framework to verify if the

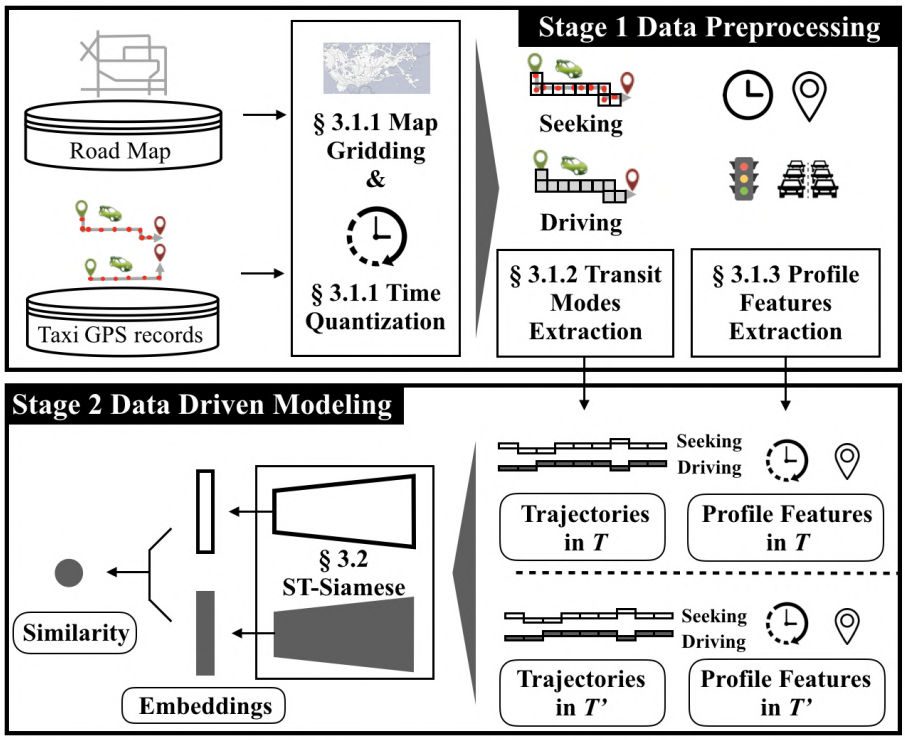


Figure 3.2: Solution framework

incoming trajectories $\mathcal{T}^{t+1}$ which are claimed being collected from an agent a's vehicle in T_{t+1} are indeed matching the agent a's behavior.

3.2.2 Solution Framework

Our proposed solution framework is outlined in Fig. 3.2, which takes two sources of urban data as inputs and contains two stages: (1) extracting trajectories and profile features in section 3.3.1, (2) identifying driving behavior in section 3.3.2. c

3.3 Methodology

3.3.1 Data Preprocessing

In this stage, we employ the GPS trajectory data and the road map data to extract the seeking and driving trajectories and online features, together with the profile features of each human agent in each time period.

3.3.1.1 Map Gridding and Time Quantization

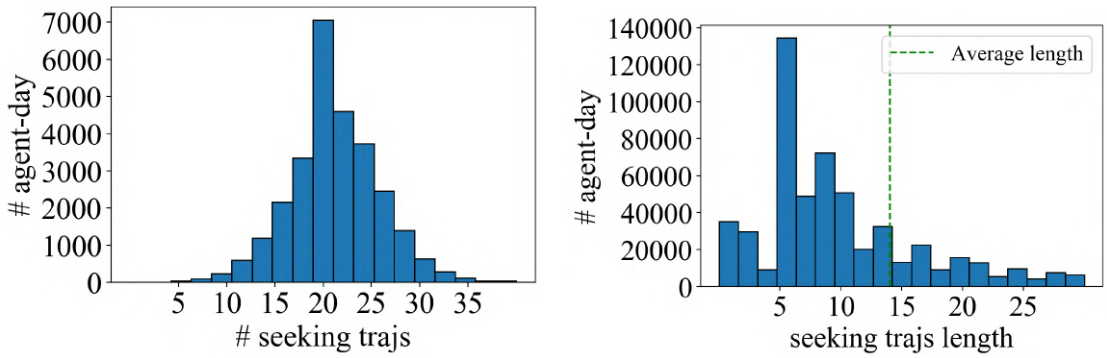
We use a standard quantization trick to reduce the size of the location space. Specifically, we divide the study area into equally-sized grid cells with a given side-length s in latitude and longitude. Our method has two advantages: (i) we have the flexibility to adjust the side-length to achieve different granularity, and (ii) it is easy to implement and highly scalable in practice [61, 60, 82]. Fig. 3.7(b) shows the actual grid in Shenzhen, China with a side-length $l = 0.01^\circ$ in latitude and longitude. Eliminating cells in the ocean, those unreachable from the city, and other irrelevant cells gives a total of 1,934 valid cells. We denote each grid cell as g_i, with $1 \leq g_i \leq 1,934$, and the complete grid cell set as $\mathcal{G} = \{g_i\}$. We divide each day into five-minute intervals for a total of 288 intervals per day, denoted as $\mathcal{I} = \{\tilde{t}_j\}$, with $1 \leq j \leq 288$. A spatio-temporal region r is a pair of a grid cell s and a time interval $\tilde{t}$. Each GPS record $p = \langle lat, lng, t \rangle$ and be represented as an aggregated state $s = \langle g, \tilde{t} \rangle$, where the location $(lat, lng) \in g$, the time stamp $t \in \tilde{t}$. A trajectory of agent a then can be mapped to sequences of spatio-temporal regions, $tr = \{a, \langle s_1, s_2, ..., s_n \rangle\}$.

3.3.1.2 Transit Modes Extraction

Different transit modes can show different patterns of driving behavior. In the taxi driving scenario, seeking and driving trajectories reflect different characteristics for each human agent taxi driver. Thus, we split the trajectories into seeking $\mathcal{T}_s$ and driving trajectories $\mathcal{T}_d$ based on the status of the vehicle whether there are passengers on board. Fig. 3.3(b) and 3.3(a) illustrate the distribution of the number of driving trajectories and the length of each driving trajectory for each agent in each day, respectively. Here, $T = 1$ day. The distributions suggest that most agents have 20 seeking trajectories every day, and the average length of each seeking trajectory is around 14.03 km. The ratio between driving and seeking trips per day is approximately 1:1.

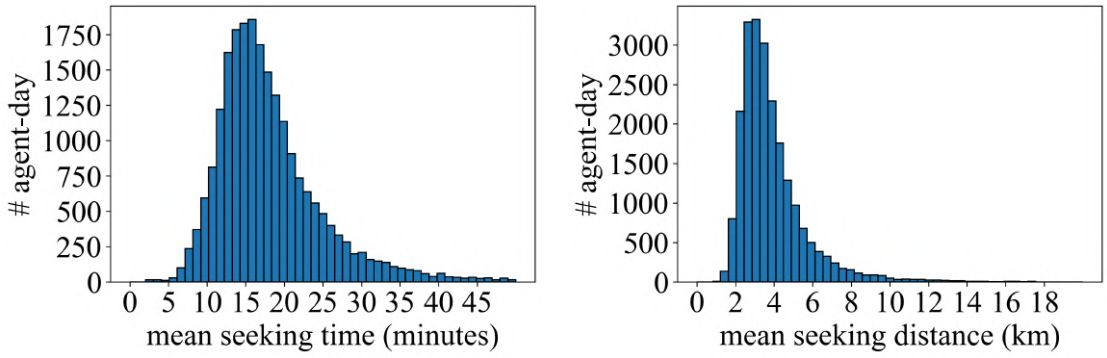
3.3.1.3 Features Extraction

Each agent has unique personal (or profile) characteristics, such as the location with the longest stay (possibly home location), daily working schedule (time duration), preferred geographic area, etc. These characteristics can be the statistical values extracted from their trajectory data. In this work, to augment the performance of driving behavior identification, we extracted the following 11 profile features for each agent in each time period



(a) Number of seeking trajectories (b) Length of seeking trajectories

Figure 3.3: Features of transition modes extraction



(a) Mean seeking time (b) Mean seeking distance

Figure 3.4: Profile features analysis

of analysis. Moreover, we extract one online feature, i.e., speed, over time for each trajectory.

$f_{p,1}$ & $f_{p,2}$: The coordinates (in longitude and latitude direction) of the longest-staying grid. Each agent can have his/her own preference on **where** to take a break during work, thus we extract $f_{p,1}$ & $f_{p,2}$*: longest-staying grid* to represent the place where an agent takes a break. The longest staying grid is the grid where the GPS records remain unchanged for the longest time.

$f_{p,3}$ & $f_{p,4}$: Break start & end time. Similarly, each agent can have his/her own preference on **when** to take a break during work, thus we extract $f_{p,3}$ & $f_{p,4}$*: Break start & end time* to capture the schedule when an agent takes a break.

$f_{p,5}$ & $f_{p,6}$: The coordinates of the most frequently visited grid. Each agent has his/her own favorite region to go, which can help identify the agent. Thus, we extract $f_{p,5}$ & $f_{p,6}$*: most frequently visited grid* to capture the region that an agent visits the most frequently in *T*.

$f_{p,7}$ & $f_{p,8}$: Average seeking trip distance & time. Each agent has his/her own efficiency on looking for passengers. The experienced agents can find passengers quickly after serving a trip, while the new agents may take longer time and distance to find a new passenger. Thus, we extract $f_{p,7}$ & $f_{p,8}$*: Average seeking trip time & distance* to capture their efficiency on finding new passengers. The distribution of these two features are shown in Fig. 3.4(a) and 3.4(b), respectively, where the x-axis the average seeking distance (in km) and the average seeking time (in min) for an agent in a day, and the y-axis is the number of driver-day's. Here $T = 1$ day. From the figures, we can see that averagely agents spend 15 minutes to seek passengers within 3 km from his/her current location.

$f_{p,9}$ & $f_{p,10}$: Average driving trip time & distance. Each of the agent can have his/her own preference on the length of trips that he/ she serves. For example, some agents prefer to serve long trips, because they think they can earn much more at a time, and they may look for passengers near the airport or train station where the passengers have higher probabilities of asking for long trips. Some other agents prefer to look for short trips because they think they can earn money more efficiently by serving short trips. Thus, we calculate the average driving trip time and distance to capture each agent's unique preference on the length of driving trips.

$f_{p,11}$: Number of trips served. Each agent has his/ her own strategy on looking for passengers. The experienced agents may serve more trips in *T* than the new agents. Thus, we count the number of trips served of each agent in *T* to capture each agent's level of experience. This feature is just the number of driving trajectories in *T*.

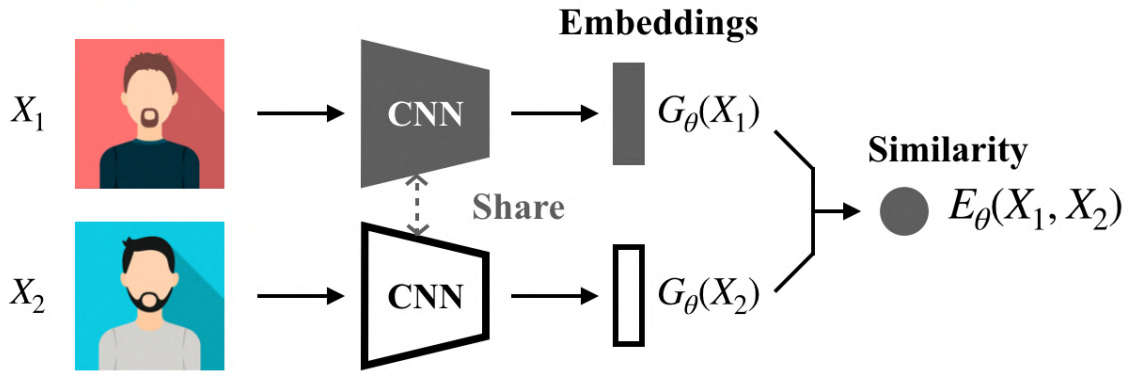


Figure 3.5: Siamese Network

$f_{o,1}$: *Speed.* Given a trajectory $tr = \{a, \langle(g_1, \tilde{t}_1)...(g_n, \tilde{t}_n)\rangle\}$, we also extract an online feature by calculating the speed information, denoted as v, for each data point in each trajectory to extract more information about driving behavior. The updated trajectory would be $\tau = \langle(g_1, \tilde{t}_1, v_1)...(g_n, \tilde{t}_n, v_n)\rangle$, where the set of trajectories is $\tilde{\mathcal{T}}$.

3.3.2 Data Driven Modeling

The increasingly pervasiveness of GPS sensors has accumulated large scale driving behavior data, which makes it possible to identify human mobility signature from trajectories. However, two challenges arise in achieving this goal. First, the pool of agents is large but the number of trajectories per agent is limited and a large number of new agents rise up every day, thus the data is sparse and maybe only subset of the data can be seen during training. Second, as a type of sequential data, trajectories has temporal dependencies which needs to be learned. We outline how we tackle these two main challenges next.

3.3.2.1 Siamese networks

To address the first challenge, we employ the siamese networks[17, 105]. Siamese networks train a metric to measure the similarity (or dissimilarity) from data, where the number of categories is very large or even not known during training, and where the size of training samples for a single category is very small. The key idea of the siamese networks is to find a function that maps the input patterns X into a lower-dimensional target space E_θ to approximate the "semantic" distance in the input space, where similar inputs are closer and dissimilar inputs are separated by a margin. Learning the dissimilarity metric

is done by training a network, which consists of two identical sub-networks with shared weights. Fig. 3.5 shows an illustration of this structure. In particular, the end-to-end dissimilarity metric learning is replicated twice (one for each input) and the representations $G_\theta(X_1)$, $G_\theta(X_2)$ are used to predict whether the two inputs belong to the same category. A commonly used optimization function for siamese networks training[17] is :

$$\begin{aligned} \min_{\theta} & -((1-Y)L_s(E_\theta(X_1,X_2)^i) + Y L_d(E_\theta(X_1,X_2)^i)) \\ & \text{s.t. } E_\theta(X_1,X_2) = \|G_\theta(X_1) - G_\theta(X_2)\|, \end{aligned} \tag{3.1}$$

where θ is the weights of the neural network, $Y = 0$ if the inputs X_1 and X_2 belong to the same category and $Y = 1$ otherwise, $E_\theta(X_1, X_2)^i$ is the i-th sample, which consists of a pair of inputs and a label (similar or dissimilar), L_s is the partial loss function for a similar pair, L_d is the partial loss function for an dissimilar pair.

3.3.2.2 Long short-term memory (LSTM) networks

The second challenge is that trajectory data has temporal dependencies. Therefore, we employ LSTM networks[35], which are capable of learning long-term dependencies for sequential data $(x_1, x_2, ..., x_T)$. LSTM sequentially updates a hidden-state representation by introducing a memory state C_t and input gate i_t, output gate o_t and forget gate f_t to control the flow of information through the time steps. At each time step $t \in \{1, 2, ..., T\}$, the hidden-state vector h_t as:

$$\begin{aligned} i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\ f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \\ o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\ \tilde{C}_t &= \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \\ C_t &= \sigma(f_t * C_{t-1} + i_t * \tilde{C}_t) \\ h_t &= \tanh(C_t) * o_t, \end{aligned} \tag{3.2}$$

where W_x represents the weights for the respective gate(x) neurons and b_x is the bias for the respective gate(x). Since the trajectory data is a sequence of $(g, \tilde{t}, v)$, we employ LSTM to learn the embeddings of trajectories in the framework of siamese networks.

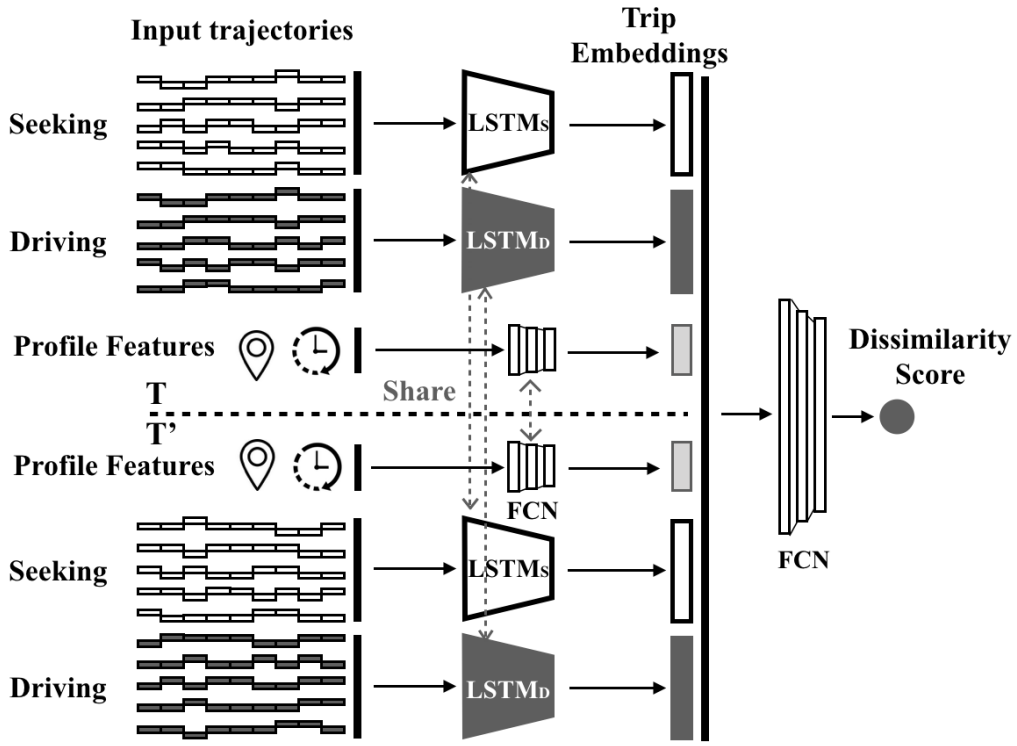


Figure 3.6: ST-SiameseNet framework

3.3.2.3 ST-SiameseNet

Given those two challenges, we present a spatio-temporal representation learning method to verify human mobility signature identity by implementing siamese networks with LSTM, named as ST-SiameseNet. The purpose of this paper is to learn the representation of trajectory with limited data and to use representation of trajectory data to compare or match new samples from previously-unseen categories (e.g. trajectories from agents not seen during training). To represent the feature of trajectory and learn the dissimilarity of driving behavior, we incorporate LSTM and fully-connected networks (FCN) into the middle layer of ST-SiameseNet, which is briefly depicted in Fig. 3.6.

We first extract profile features for each agent and the online feature for each data point in the trajectory. And then randomly select a pair of seeking trajectories $\tilde{\mathcal{T}}_{s,1}$ and $\tilde{\mathcal{T}}_{s,2}$, a pair of driving trajectories $\tilde{\mathcal{T}}_{d,1}$ and $\tilde{\mathcal{T}}_{d,2}$ (all the trajectories contain the online feature) and a pair of profile features $\mathbf{f}_{p,1}$ and $\mathbf{f}_{p,2}$ in each time period. In the original siamese networks, there are two sub-networks with identical weights. To a further step, we introduce six sub-networks where each two identical sub-networks share the set of weights since we

have three types of inputs, i.e., $\tilde{\mathcal{T}}_{s,1}$ and $\tilde{\mathcal{T}}_{s,2}$ would share the weights of $LSTM_S$, while $\tilde{\mathcal{T}}_{d,1}$ and $\tilde{\mathcal{T}}_{d,2}$ use the same $LSTM_D$ to learn the representation. Since each agent has a vectorized profile features in each time period, here we implement fully-connected layers as a *profile-learner* to project the profile features. ST-SiameseNet learns the driving behavior from seeking, driving trajectories and profile features respectively and aggregates the embedding layers with a sequence of fully-connected layers, i.e. *dissimilarity-learner* as the dissimilarity metric. Differing from previous works [17] that use the L_1 norm to approximate the "semantic" distance, we utilize neural networks (as a more powerful function) to learn the dissimilarity.

The learning process minimizes the binary cross entropy loss that drives the dissimilarity metric to be small for pairs of trajectories from the same agent, and large for those from different agents. To achieve this property, we pose the following ST-SiameseNet optimization problem:

$$\begin{aligned} \min_{\theta} & -(y \log(D_\theta(X_1, X_2)) + (1-y)\log(1 - D_\theta(X_1, X_2))), \\ & \text{s.t. } X_1 = (\tilde{\mathcal{T}}_{s,1}, \tilde{\mathcal{T}}_{d,1}, \mathbf{f}_{p,1}), X_2 = (\tilde{\mathcal{T}}_{s,2}, \tilde{\mathcal{T}}_{d,2}, \mathbf{f}_{p,2}), \end{aligned} \tag{3.3}$$

where $y = 0$ if the trajectories belong to the same agent and $y = 1$ if the trajectories come from two different agents, $D_\theta(X_1, X_2)$ is the prediction probability of how likely the trajectories are from two different agents.

Algorithm 2 shows the training process of the ST-SiameseNet model. During the training process, we apply the gradient descent approach to update parameters $\boldsymbol{\theta}$, with learning rate α and a predefined i_{max}, (i.e. the total number of iterations). We first extract profile features $\mathbf{f}_p$ from trajectories $\mathcal{T}$ for each agent. And then compute the online feature $f_{o,1}$, i.e. speed information in each grid cell and update trajectories with the online feature from $\mathcal{T}$ to $\tilde{\mathcal{T}}$. Moreover, we split trajectories into seeking trajectories $\tilde{\mathcal{T}}_s$ and driving trajectories $\tilde{\mathcal{T}}_d$ for each agent. Since ST-SiameseNet pair-wisely trains the data, we randomly select a pair of trajectories, either from the same agent or from different agents in two time periods, with equal probability in each iteration. Next, we update ST-SiameseNet parameters θ by using Eq 3.3, with α as the step size (Line 7).

3.4 Experimental Evaluation

In this section, we demonstrate the effectiveness of our proposed method by utilizing GPS records collected 10 workdays from 2197 taxis in Shenzhen, China in July 2016.

Algorithm 2 ST-SiameseNet Training

Require: Trajectories $\mathcal{T}$, initialized parameters $\boldsymbol{\theta}$, learning rate α, max iteration i_{max}.
Ensure: A well trained ST-SiameseNet with parameters $\boldsymbol{\theta}$.

```
Extract profile feature expectation vector f_p.
Calculate the online feature f_{o,1} and update trajectories from T to T̃.
Split seeking T̃_s and driving T̃_d trajectories.
Sample a pair of trajectories with the online feature T̃_{s,i}, T̃_{d,i} and a pair of profile features f_{p,i}.
while iter < i_max do
    Calculate gradient ∇g(θ) using Eq 3.3.
    Update θ ← θ + α∇g(θ).
end while
```

We compare our model with other baseline methods, analyze the generalization of our approach and evaluate the importance of transit modes and profile features for each agent. To support the reproducibility of the results in this paper, we have released our code at Github [1].

3.4.1 Data Description

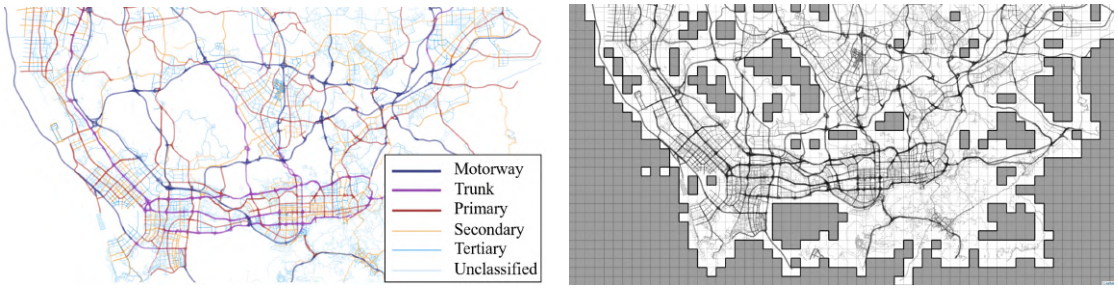
The purpose of our framework is to recognize human mobility signatures with GPS records. In this paper, we **use taxi driving scenario as an example** to demonstrate our techniques. However, the proposed solution can be easily generalized to other types of agents and trajectories. Our analytical framework takes two urban data sources as input, including (1) taxi trajectory data and (2) road map data. For consistency, both datasets are collected in Shenzhen, China in July 2016.

Taxi trajectory data contains GPS records collected from taxis in Shenzhen, China during July 2016. There were in total $17,877$ taxis equipped with GPS sets, where each GPS set generates a GPS point every 40 seconds on average. Overall, a total of $51,485,760$ GPS records are collected on each day, and each record contains five key data fields, including taxi ID, time stamp, passenger indicator, latitude and longitude. The passenger indicator field is a binary value, indicating if a passenger is aboard or not.

Road map data of Shenzhen covers the area defined between $22.44°$ to $22.87°$ in latitude and $113.75°$ to $114.63°$ in longitude. The data is from OpenStreetMap [3] and has $21,000$ roads of six levels.

The road map data includes $21,000$ road segments with six levels as shown in Fig. 3.7(a).

[1] `https://github.com/huiminren/ST-SiameseNet`



(a) Shenzhen road map (b) Map gridding

Figure 3.7: Shenzhen map data

In this paper, we utilize GPS records from 2197 taxis in Shenzhen, China from July 4^{th} to July 15^{th} (10 workdays) 2016. To keep enough information in each trajectory, we filter out the trajectory which length is less than 10 steps, where each step is a tuple of a grid and a time slot. After filtering out those grids that taxis cannot reach, there are 1934 valid grids as shown in Fig. 3.7(b). For each driver, we randomly select 5 seeking and 5 driving trajectories in each work day as partial inputs of our ST-Siamese. 11 profile features and 1 online feature are extracted from the GPS records data.

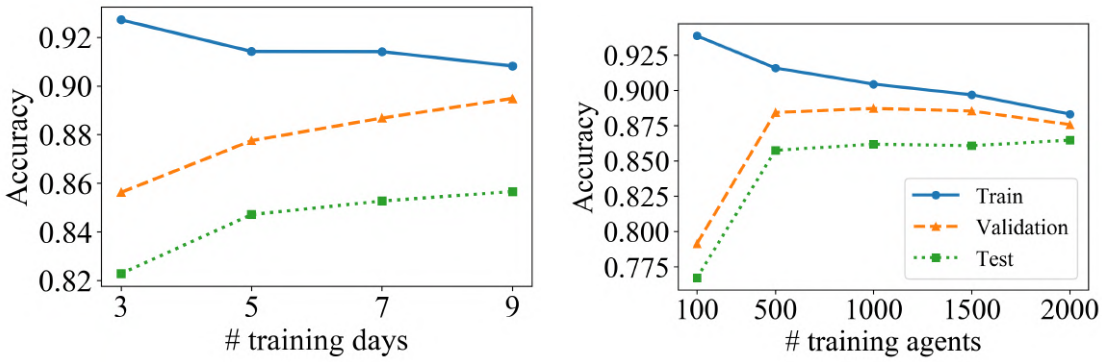
3.4.2 Evaluation Metrics

To evaluate the performance of our proposed model and baseline methods, we measure accuracy, precision, recall and F_1 score against the ground truth among labels. In our implementation, the dissimilarity score threshold is set to 0.5. If it is less than 0.5, we consider that the trajectories belong to the same agent. Otherwise they are from different agents. Note the threshold can be tuned on different datasets. In particular, precision is intuitively the ability of the classifier not to confuse different agents. Recall shows the ability of the classifier not to miss pairs of different drivers. The F_1 score is a weighted average of the precision and recall.

3.4.3 Baseline Algorithms

We compare the performances of our method against the following baseline algorithms.

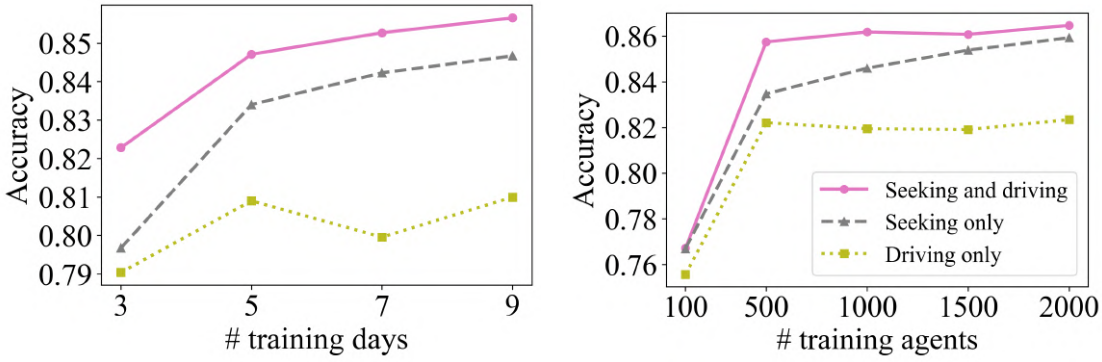
1. **Support Vector Machine (SVM).** Taigman et al. [105] tested the similarity between faces using a linear SVM. Here we utilize the set of profile features as input, described in section 3.3.1. We conduct absolute difference between profile feature



(a) Models accuracy across days

(b) Models accuracy across agents

Figure 3.8: Accuracy across days and agents



(a) Transit modes across days

(b) Transit modes across agents

Figure 3.9: Model comparison among transit modes

vectors of two agents and then employ SVM to classify whether these two agents are same.

2. **Fully-connected Neural Network (FNN).** Fully-connected neural network is a basic classification or regression model in deep learning. Here we concatenate all the trajectories in two time periods from two agents together as the inputs of the neural network. In addition, we compare the model accuracy with and without features.

3. **Naive Siamese Network.** Chopra et al. [17] used a Siamese architecture for face verification. Here we train a network which consists of two identical fully connected networks that share the same set of weights. We concatenate all the trajectories in two time periods from each agent and evaluate the baseline with and without features.

4. **ST-SiameseNet-L_1.** To evaluate the advantages of using FCN than a predefined function in learning dissimilarity, we replace FCN with the L_1-norm distance to approximate the "semantic" distance.

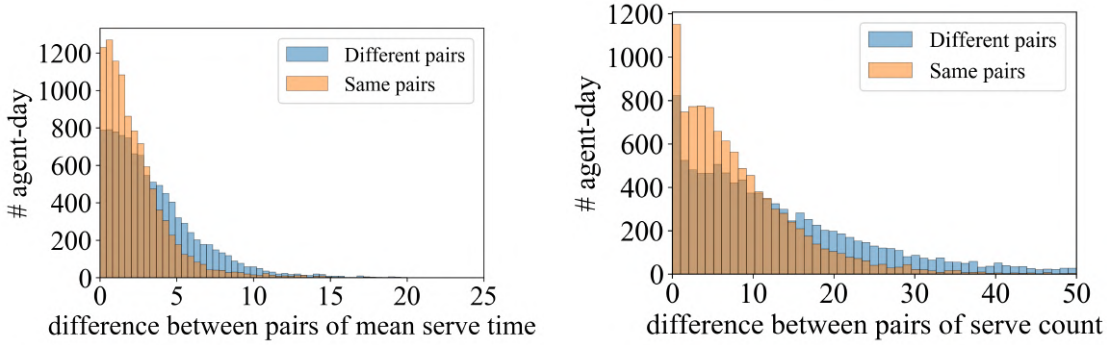
3.4.4 Results

3.4.4.1 Comparison results

We compare our ST-SiameseNet with the baseline models in terms of precision, recall and F_1 score. All the models train with trajectories from 500 agents in 5 days, validate with trajectories from the same agents as training set but in another 5 days and test with trajectories from 197 new agents in the latter 5 days. Similar to the training dataset, we uniformly sample two sets of trajectories from the same agent or different agents in two time periods during validation and testing. Table 3.1 shows the evaluation metrics from all the methods. It is clear that our approach achieves the best performance. SVM outperforms other baseline models using profile features but is worse than our model. This is because SVM is not able to model sequential inputs and the aggregation will lose information of driving behavior. With profile and basic features added, both FNN and Siamese FNN work better, indicating that features can provide useful information in both models. However, all of the deep learning and machine learning models perform worse than our model, since ST-SiameseNet has a more effective ability to capture the information of sequential inputs by using LSTM. In addition, ST-SiameseNet-L_1 performs worse than ST-SiameseNet with FCN to learn the dissimilarity, showing that L_1 norm has limited

Table 3.1: Average results on real-world dataset and comparison with baselines

Methods	Precision	Recall	F_1 score
ST-SiameseNet	**0.8710**	**0.8317**	**0.8508**
SVM	0.8100	0.7661	0.7874
FNN (with features)	0.6112	0.6298	0.6195
FNN (without features)	0.5266	0.5470	0.5365
Naive Siamese (with features)	0.6137	0.6707	0.6407
Naive Siamese (without features)	0.5580	0.5657	0.5617
ST-SiameseNet-L_1	0.8052	0.7775	0.7910



(a) Mean serving time

(b) Number of serves per day

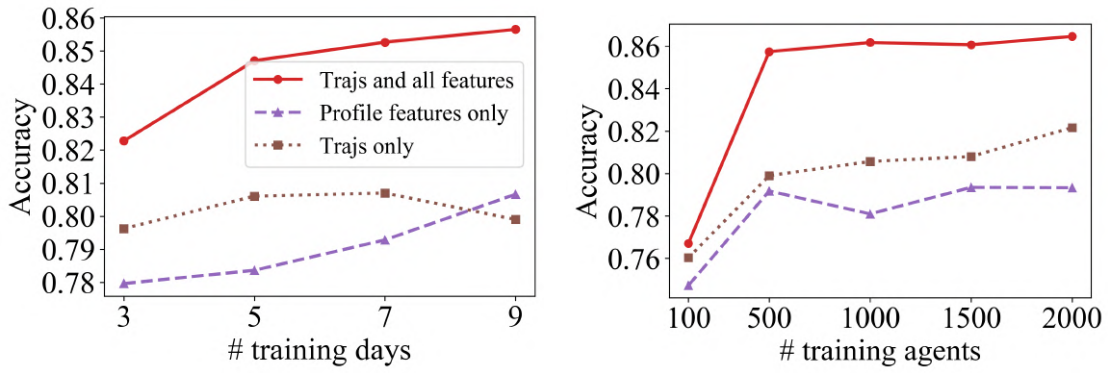
Figure 3.10: Analysis of profile features

ability to learn the dissimilarity between two identities. In particular, the F_1 score of ST-SiameseNet is over 0.85, which is significantly higher than all baselines.

3.4.4.2 Model generalization

We evaluate different design choices of our model on classification accuracy. Similar to the previous experiments, we use the trajectories of the first 500 agents in 10 consecutive days as training and validation sets and vary the split ratio.

Impact of different number of days. First, we vary the number of days in the training set of the 500 agents to $N_{day} = 3, 5, 7$ and 9 respectively. We train trajectories of 500 agents from Day 1 to Day N_{day}, validate trajectories of the same 500 agents from Day $(N_{day} + 1)$ to Day 10, test trajectories of the new 197 agents from Day $(N_{day} + 1)$ to Day 10. Fig. 3.8(a) depicts the training, validation and test accuracy across different days. As more days are added to the training dataset, the training accuracy decreases slightly, while validation and test accuracy gradually increase, indicating that larger datasets can



(a) Features across days

(b) Features across agents

Figure 3.11: Model comparison among features

help with over-fitting problem. In addition, when the number of days extending from 3 to 5, both the validation and test accuracy have a dramatically increase, while the validation and test accuracy have a small increase after adding more days, indicating that trajectories of 500 agents from 5 days contain enough information to learn the similarity of agents.

Impact of different number of agents. we also vary the training dataset size by using a subset of the agents. Subsets of sizes $N_{agents} = 100$, 500, 1000, 1500 and 2000 agents in 5 days are used. We train trajectories of N_{agents} agents from Day 1 to Day 5, validate trajectories of the same N_{agents} agents from Day 6 to Day 10, test trajectories of new 197 agents from Day 6 to Day 10. Fig. 3.8(b) shows the training, validation and test accuracy across different number of agents. With more agents added to the training dataset, the neural networks have better generalizability and can have a better performance when seeing new data. There is an enormous increase of validation and test accuracy when the number of training agents growing from 100 to 500, indicating that the neural networks benefits from the increase of diversity of agents. However, similar to the impact of different number of days, the validation and test accuracy are flattening when adding more agents to the training pool, showing that trajectories of 500 agents from 5 days are sufficient to learn the mobility signatures of agents.

3.4.4.3 Importance of transit modes

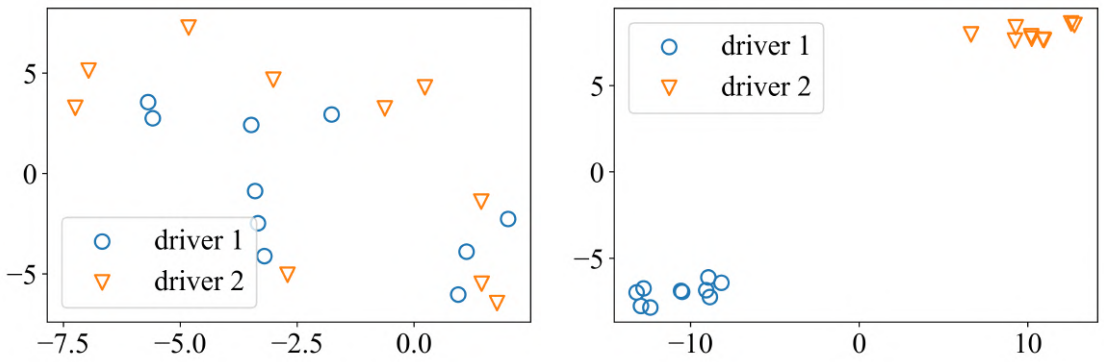
Transit modes can show different driving habits among different agents. Seeking and driving trajectories are two typical transit modes, defined based on the situation of the vehicle whether any passenger on-board. Different agents would have different strategies to seek passengers [82]. In this section, we would test if each of the transit mode can contribute

to identify the drivers' behavior. As can be seen from Fig. 3.9(a) and 3.9(b), both seeking and driving trajectories have the ability of discriminating the same and different agents. All the accuracy of seeking trajectories are higher than driving trajectories, which is consistent with human intuition that different agents have different strategies when seeking passengers, while agents do not have the choice of destination when passengers on board. With both seeking and driving trajectories included, ST-SiameseNet performs the best by integrating the information of both seeking and driving trajectories. Fig. 3.9(a) and 3.9(b) also show the same trend as Fig. 3.8(a) and 3.8(b) that models of seeking trajectories only and driving trajectories only benefit from larger dataset.

3.4.4.4 Importance of features

In this section, we evaluate the importance of features by comparing our model with and without features. Each agent has unique personal characteristics which can be extracted from his/her trajectory data over a time period [82]. In addition, Speed information of each trajectory are able to capture agents' driving behavior [22]. Fig.3.10(a) and 3.10(b) describe the distribution of two profile features, mean serving time and number of service trips. The orange bar depicts the absolute difference of profile features between same agents, while the blue bar otherwise. There is an obvious difference between same agents and different agents, indicating the profile features are able to identify the similarity of driving behavior.

From Fig. 3.11(a) and 3.11(b), we can see that our ST-SiameseNet with trajectories and features works the best comparing with the model with profile features only as well as with trajectories only in different number of training days and different number of training agents. In particular, the model with profile features only gets the worst performance, indicating that the aggregation may lose information of driving behavior. Besides, if we only use trajectories as inputs i.e. $tr = \langle s_1, s_2 ... s_n \rangle$ ($s = \langle g, \tilde{t} \rangle$), all the test accuracy across days and agents are also lower than our ST-SiameseNet with both trajectories and features included, probably because some statistical information, such as mean seeking distance, mean seeking time, number of serves, cannot be captured by LSTM. In addition, Fig. 3.11(a) and 3.11(b) shows the same trend as Fig. 3.8(a) and 3.8(b) that the neural networks benefit from larger datasets. Overall, with raw trajectory data and extracted features working together, we can learn the driving patterns from trajectory data more effectively and verify the agent more accurately. Note that we only extract 11 profile features and 1 online feature, which requires little human work of feature engineering



(a) Raw trajs projection (b) Embedded trajs projection

Figure 3.12: Projection comparison between raw and embedded trajectories

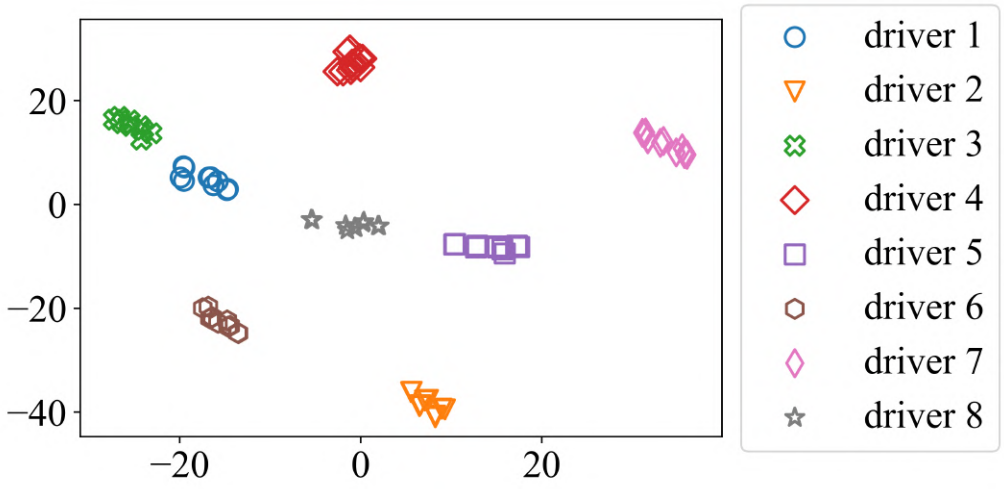
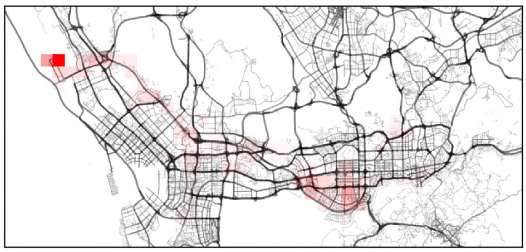


Figure 3.13: Multi-agents embeddings projection

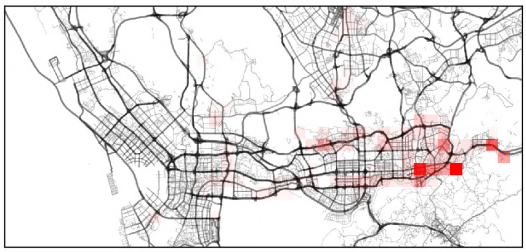
comparing with [18], which extracted 137 statistical human-defined features.

3.4.5 Embeddings Visualization

The main goal of this paper is to identify agents by learning driving similarity. To further access the accomplishment of this target, we provide some visualizations using the learned embeddings of trajectories for several agents. The term of embedding is showed in Fig. 3.6. We utilize t-SNE [75] to perform the representation learning and show the embeddings in a 2-dimensional space. For each agent, we use the first half embeddings from our model as the input of t-SNE since our model train two agents pair-wisely each time. The expected result is that trajectories from different agents are far from each other,

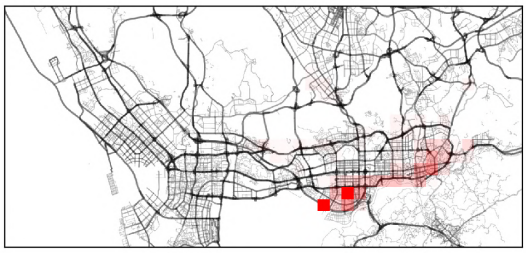


(a) Driver 1

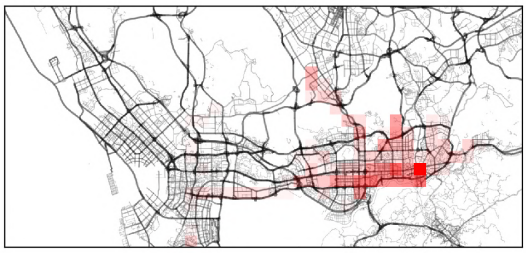


(b) Driver 2

Figure 3.14: Case 1 of identifying different drivers



(a) Driver 3



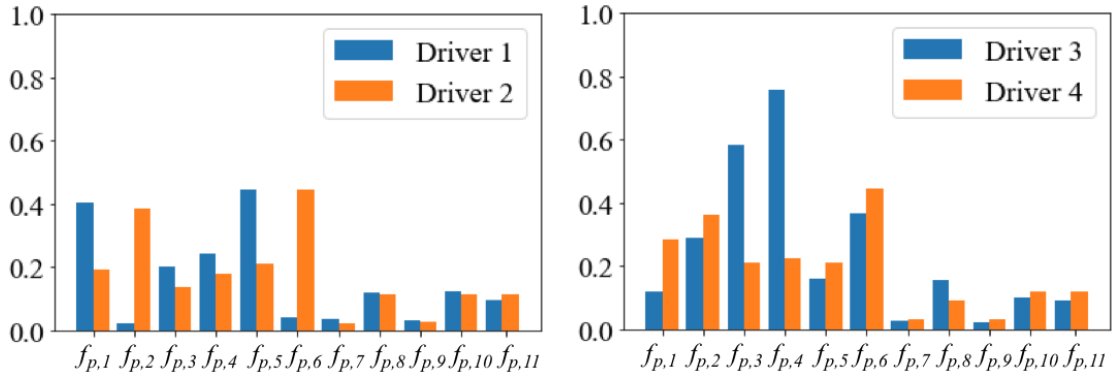
(b) Driver 4

Figure 3.15: Case 2 of identifying different drivers

while trajectories from the same agent are sufficiently close. For each agent, we select 10 subsets of trajectories from 5 days, i.e there are 10 points for each agent. First, we randomly select two agents to compare the projection of raw trajectories and embedded trajectories. Figure 3.12(a) and 3.12(b) illustrate the result of the comparison. We can see that the separation of embedded trajectories are extremely better than the raw trajectories. In particular, the raw trajectories projections of two agents are mixed together, while we can easily separate two agents from their embedded projections. Furthermore, we visualize the embedding results of multiple agents. As shown in Fig. 3.13, the separation of each agent is obvious, which is consistent with our expectation.

3.4.6 Case Studies

To further understand how ST-SiameseNet identifies agents' behaviors, we investigate individual agents' cases to show what factors ST-SiameseNet considers when identifying the agents. Four case studies are presented.



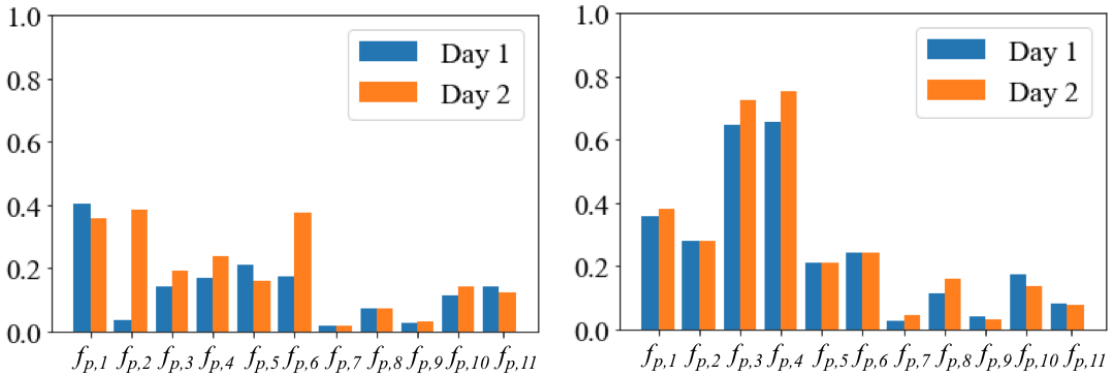
(a) Driver 1 & driver 2 (b) Driver 3 & driver 4

Figure 3.16: Profile feature comparison

3.4.6.1 Cases of identifying different drivers.

First, we show an example of two randomly selected human agent taxi driver, driver 1 and driver 2. We extract their trajectories and profile features on July 4^{th}, 2016, then, our proposed ST-SiameseNet consumes the trajectories and features and produces a dissimilarity score of **0.99**, which means ST-SiameseNet identifies that this pair of inputs is from two different agents. To figure out what factors that ST-SiameseNet consider to identify them, we visualize the heat map of their visitation frequency on that day to each grid of the city in Fig.3.14(a)&3.14(b). The darker red color in the grid indicates higher visitation frequency. Fig.3.14(a)&3.14(b) illustrate that driver 1 and driver 2 have significantly different active regions. Driver 1 likes working in the west part of the city, especially near the airport, while driver 2 prefers to work in the east part near the downtown area. The difference in the active regions of driver 1 and driver 2 helps ST-SiameseNet identify them. Fig.3.16(a) shows the comparison of the profile features of driver 1 and driver 2, which illustrates that they have significantly different feature values on $f_{p,2}$*: the longest staying grid id in latitude direction* and $f_{p,6}$*: the most frequently visited grid id in latitude direction*. This is consistent with the finding from the heat map, i.e., the difference in their active regions.

ST-SiameseNet can also identify different drivers even if their active regions are similar to each other. We select another case of identifying different drivers from our data randomly. Fig.3.15(a)&3.15(b) show the heat map of the visitation frequency of driver 3 and driver 4, which indicate that driver 3 has similar active region as driver 4. They both like working near the downtown area. And our proposed ST-SiameseNet successfully identifies them with a dissimilarity score of **0.88**, which means they are significantly



(a) Case 3 profile features (b) Case 4 profile features

Figure 3.17: Profile feature comparison for case 3 and 4

(a) Day 1 (b) Day 2

Figure 3.18: Case 3: Abnormal driving behavior

different. Fig.3.15(a)&3.15(b) show that driver 3 and driver 4 have similar active region near the downtown area, and the difference on $f_{p,2}$*: the longest staying grid id in longitude direction* and $f_{p,6}$*: the most frequently visited grid id in longitude direction* is small as shown in Fig.3.16(b). However, they have significantly different profile feature values on $f_{p,3}$ *&* $f_{p,4}$*: Break start & end time.*, and this information has been discovered by ST-SiameseNet, thus driver 3 and driver 4 can be identified by ST-SiameseNet.

3.4.6.2 Case of identifying abnormal behavior of one driver ID

Apart from the case of identifying different drivers, ST-SiameseNet can deal with the case of identifying abnormal driving behavior of "one" driver. Our dataset contains only the licence plate of each vehicle. Therefore, abnormal behaviors of the same vehicle can might suggest a change of driver. Here, we study a driver's behavior in 2 days from July 5^{th} to July 6^{th} 2016. Let's call this driver "John". Our ST-SiameseNet produces a dissimilarity score of **0.84** for the trajectories in these 2 days, which indicates that John's

(a) Day 1 (b) Day 2

Figure 3.19: Case 4: Normal driving behavior

behavior changes significantly from day 1 to day 2. To figure out how John's behavior changed significantly, we plot the heat map of his visitation frequency in Fig.3.18, from which, we find that his active region changes from the west part of the city in day 1 to the east part in day 2. Also, Fig.3.17(a) shows the profile features in these 2 days. Most of the profile features change significantly, e.g., $f_{p,2}$*: the longest staying grid id in longitude direction*, $f_{p,6}$*: the most frequently visited grid id in longitude direction*, $f_{p,3}$ & $f_{p,4}$*: Break start & end time.* We also study a few more days after day2, the behaviors are similar to that in day2, thus, this abnormal behavior after day 1 appears to be the result of a new driver operating the vehicle after day 1.

3.4.6.3 Case of identifying normal behavior of one driver ID

For the normal behavior of a driver, ST-SiameseNet can identify it correctly. Here, we study a driver's behavior in 2 days from July 5^{th} to July 6^{th} 2016. Let's call this driver "Mike". Our ST-SiameseNet produces a dissimilarity score of **0.03**, which indicates that Mike's behavior remains consistent from day 1 to day 2. The heat map of his visitation frequency in Fig.3.19 illustrates his active region does not change. Also, his profile features in these 2 days as shown in Fig.3.17(b) remain stable.

3.5 Related Work

Human mobility signature identification has been extensively studied in recent years due to the emergence of the ride-sharing business model and urban intelligence[60, 21, 122, 134]. However, to the best of our knowledge, *we make the first attempt to employ siamese network to verify human mobility signature identification.* Related work are summarized below.

Urban computing. Urban computing is a general research area which integrates

urban sensing, data management and data analytic together [73, 67, 132, 49, 74]. In particular, a group of work focus on taxi operation management, such as dispatching [99, 41] and passenger seeking [131, 123, 124]. They aim at finding an optimal actionable solution to improve the performance/revenue of individual taxi drivers or the entire fleet. Rong et al. [94] solved the passenger seeking problem by giving direction recommendations to drivers. However, all of these works focus on finding "what" are the best driving strategies (as an optimization problem), rather than considering the benefits of passengers. By contrast, our work focuses on driver identification, which can enhance the safety of passengers.

Driver behavior learning. Most existing literature on human driving behavior rely on human-defined driving style feature set. These handcrafted vehicle movement features derived from sensor data or constructed from real-world GPS data [70, 32, 25, 18, 134]. They used supervised classification, unsupervised clustering or reinforcement learning to solve problems as such driver identification, sequential anomaly detection,etc [70, 22, 134, 50, 81]. Ezzini et al. [25] addressed the driver identification problem using real driving datasets consisting of measurements taken from in-vehicle sensors, such as driver camera, smartphone are placed within the car and the driver is connected to electrodes and skin conductance response. However, such existing work require expensive sensor installed in the vehicle or excessively rely on human-defined features. Dong et al. [22] proposed a deep-learning framework to driving behavior analysis based on GPS data. They used CNN and RNN respectively to predict driver identity among 50 and 1000 drivers for a given trajectory. Such frameworks are generally require all the categories be known in advance as well as the training examples be available for all the categories, as opposed to our objective which only a subset of the categories is known at the time of training.

Siamese network. The siamese network [11] is an architecture for similarity learning of inputs, which has been widely used in multiple applications, namely but a few, vision area, unsupervised acoustic modelling, natural language processing [17, 36, 39, 104, 45]. Chopra et al.[17] learned complex similarity metrics of face verification by introducing convolutional networks to siamese networks. Hoffer and Ailon [36] proposed a variant of siamese networks, triplet networks to learn an image similarity. Hu et al.[39] applied siamese networks with convolutional layers to match two sentences. However, to our best knowledge, we are the first one to employ siamese network to human-generated spatio-temporal data.

3.6 Conclusion

In this paper, we propose the Spatio-temporal Siamese Networks (ST-SiameseNet) to solve the Human Mobility Signature Identification (HuMID) problem. The HuMID problem aims at validating if an income set of trajectories belong to a certain agent based on historical trajectory data. ST-SiameseNet can deal with large group of agents in a single model. Also, we extract several effective profile features from the trajectories to augment the performance of the ST-SiameseNet. The experimental results illustrate that ST-SiameseNet outperforms state-of-the-art works and achieves an F_1 score of 0.8508 on a real-world taxi trajectory dataset. Our proposed ST-SiameseNet framework can be applied to many other real-world cases with human-generated spatio-temporal data other than the ride-sharing and taxi case. In the future, we will continue studying the driver identification problem with multiple inputs rather than pairwise inputs.

Chapter 4

Real Workplace Detection

4.1 Introduction

Local business development is playing an increasingly important role in city growth, since it helps to create more jobs and increase tax revenue. Moreover, city government's capacity and economic performance, such as GDP, labor market, and logistics, are also the fundamental factors for a company's registration, expansion and relocation decisions. As a result, many governments provide financial aids and make strategic policies to attract companies. For instance, Texas county approves tax breaks for Tesla, if it builds a $1.1 billion car plant near Austin [1]. To ensure the effectiveness of the financial policies and management, the government requires the companies to keep updating their actual operating addresses. However, the address information is not always accurately maintained. For the extreme example, some entrepreneurs may exploit policy loopholes in a non-compliant manner, which they register the office in a city with preferential tax policies, while actually operate in other cities to take advantage of the labor, market or logistic benefits there. These phenomena may violate the original purpose of the policy, bring more inconvenience in government management, and even break the law [2, 1]. Therefore, it is vital to discover real workplaces of the companies to regulate their operation and ensure the healthy development of the local market.

To discover real workplaces of companies, most of the governments rely on the active on-field screening over daily regulation. However, this traditional approach is time-consuming and achieve limited coverage. Moreover, researchers [13] have attempted to

[1] https://www.cnbc.com/2020/07/14/texas-county-approves-tax-breaks-for-a-new-tesla-fa
html

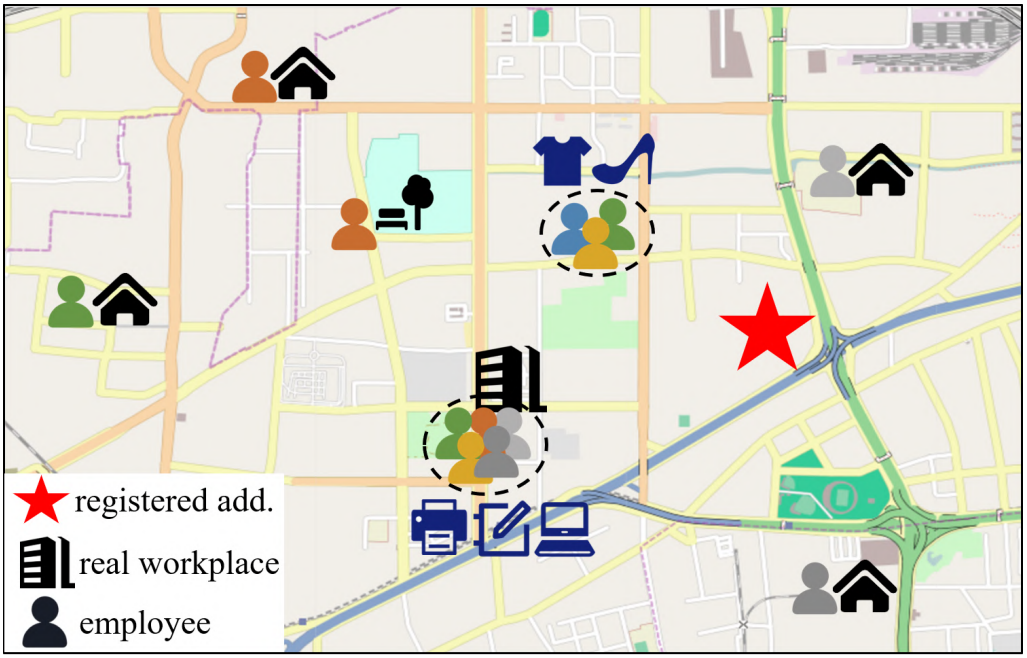


Figure 4.1: This figure shows our intuition. Different colors indicates different employees in a company. Icons in dark blue indicate online purchase items. Employees may stay in the same place in their daily life, which might indicate the real workplace of the company.

crawl company addresses from recruit sites as the real workplaces. However, it is difficult to handle the situation where companies intentionally hide their workplace and detailed address information.

To overcome the drawbacks of existing works, we can use a data mining method to infer the real workplaces. Intuitively, employees often gather in the workplace of the company. Fortunately, e-commerce apps can provide the information of: 1) company-employee relations; and 2) employees' position records. In practice, 1) the company-employee relations can be identified when users issue invoices with the company name for office supplies reimbursement; 2) the employees' position records including check-ins and shipping address are collected with their authorization. For example, as shown in Figure 4.1, by utilizing employees' records collected from the e-commerce Apps, it is possible to develop a data mining method to identify companies' real workplaces.

However, to develop an effective real workplace identification method based on e-commercial data, we must address several challenges. First, company sizes and employee's individual usage patterns in e-commerce platforms vary significantly across companies. This introduces a challenge of generating high-quality locations via clustering for

different companies. Moreover, the generated locations contain complicated observations that cannot be simply captured and interpreted. For instance, there are multiple reasons for users gathering, i.e., going to shopping mall, living in the same residence, going to work, etc. It is challenging to identify valid workplaces from clusters across space over time.

To address the above challenges, in this paper, we design a novel approach named LocRecoginzer to identify real working locations from e-commercial data. Specifically, LocRecognizer contains three main parts: 1) *Data Pre-processing*, which adopts a filter-based method to remove unrelated records from massive e-commercial data. 2) *Candidate Location Generation*, which constructs a two-stage clustering procedure, jointly capturing dependencies from both user and group dimensions, to generate candidate locations. 3) *Real Workplace Discovery*, which proposes a multi-learner deep learning model, fully incorporating spatial-temporal information, to further identify workplaces from location candidates. The main contributions are summarized as follows:

- We formalize the problem of real workplace identification for e-commercial data and identify its unique challenges resulting from the pattern complexity of company and user.
- We propose LocRecoginzer, a novel two-step data mining method, to discover real workplaces of companies from e-commercial data. LocRecoginzer adopts a two-stage clustering method to generate location candidates, and further detect correlated workplaces by using a multi-learner deep learning model.
- The proposed LocRecoginzer method is evaluated extensively over two real-world datasets from an e-commerce platform in Beijing and Nantong. We empirically demonstrate that our LocRecoginzer outperforms six baseline methods on different evaluation criteria.

4.2 Overview

4.2.1 Preliminaries

Definition 11. ***User.*** *We define the users related to a specific company as* $\mathbf{u}_{c_i} = (u_{i,1}, u_{i,2}, \cdots, u_{i,j}, u_{i,n})$*, where* n *is the number of users in the company* c_i*. Moreover, the set of company users is* $\mathcal{U}_{c_i}$ *and the set of all the companies is* $\mathcal{C}$*.*

Definition 12. ***Check-in.*** *A check-in consists of a location in latitude lat, longitude lng, and a timestamp t, denoted as $p = (lat, lng, t)$. For each user $u_{i,j}$ in the company c_i, the check-ins are denoted as $\mathbf{p}_{u_{i,j}} = \langle u_{i,j}, (p_1, p_2, \cdots, p_m) \rangle$, where m is the number of check-ins. The set of users' check-ins in the company c_i is $\mathcal{P}_{c_i}$.*

Definition 13. ***Shipping Location.*** *A shipping location is the geocoded location of a shipping address, denoted as $s = (lat, lng)$. For each user $u_{i,j}$ in the company c_i, the shipping locations are denoted as $\mathbf{s}_{u_{i,j}} = \langle u_{i,j}, (s_1, s_2, \cdots, s_q) \rangle$, where q is the number of shipping locations. The set of users' shipping locations in the company c_i is $\mathcal{S}_{c_i}$.*

Definition 14. ***E-order.*** *An e-order contains the order information when a user purchases online, denoted as $\mathbf{o} = (OID, item, t)$, where OID is the identification number of the order, $item$ contains basic order information, such as category and price, and t is the timestamp. For each user $u_{i,j}$ in the company c_i, the e-orders are denoted as $\mathbf{o}_{u_{i,j}} = \langle u_{i,j}, (o_1, o_2, \cdots, o_r) \rangle$, where r is the number of e-orders. The set of users' e-order information in the company c_i is $\mathcal{O}_{c_i}$.*

Definition 15. ***Workplace.*** *The workplace is a location where the company-related users work, denoted as $w = (lat, lng)$. A company may have one or multiple workplaces and the workplace set of the company c_i is denoted as $\mathcal{W}_{c_i}$.*

Problem Definition. Given the users' check-ins $\mathcal{P}$, shipping locations $\mathcal{S}$ and the e-order information $\mathcal{O}$, we aim to find out all the companies' real workplaces $\mathcal{W}$.

4.2.2 System Framework

The system framework of the proposed LocRecoginzer is illustrated in Figure 4.2. LocRecoginzer consists of three major components: data pre-processing, candidate location generation, and real workplace discovery.

Data Pre-processing. The collected records contain unrelated information in both temporal and spatial dimensions, e.g., active in non-working hours and not in a stable area. Thus, we first take a *Temporal Filter*, which removes users' check-ins during holidays and night; then we adopt a *Moving Filter*, which filters out users' check-ins when users are moving (detailed in Section 4.3).

Candidate Location Generation. The gather of users may be the real workplace of the company, but the variation of user usage behavior and company size make it difficult

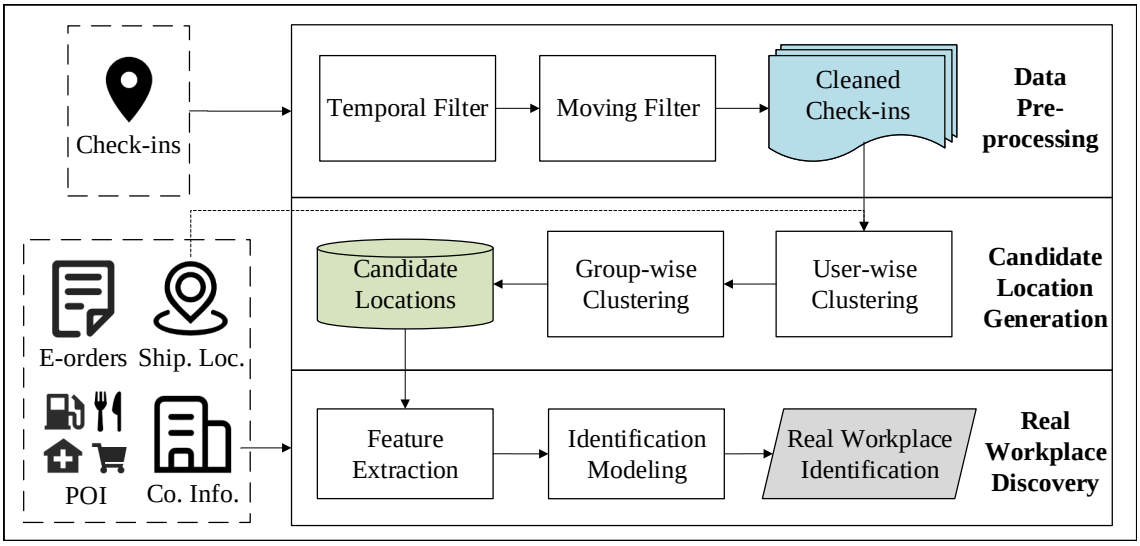


Figure 4.2: System framework.

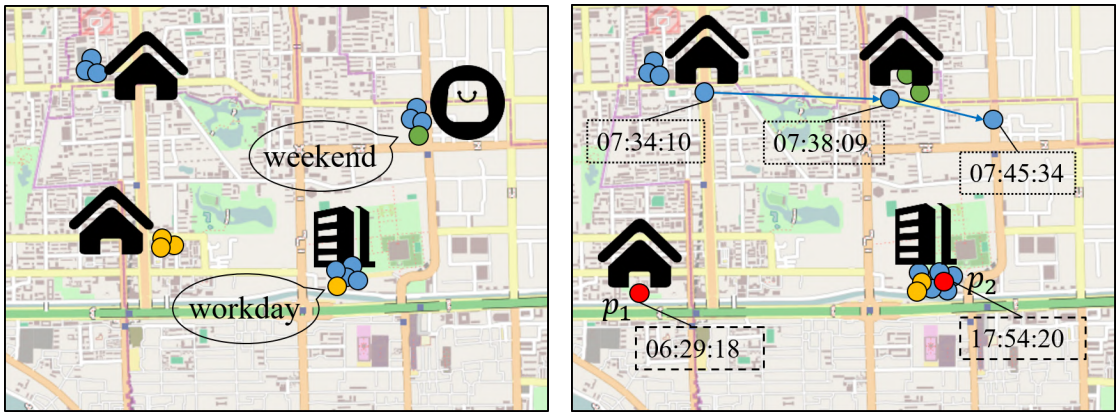
to generate high-quality candidates. This component use prepossessed users' check-ins and shipping locations to generate candidate locations of real workplaces. It includes 1) *User-wise Clustering*, which groups a single user's position records to normalize skewed records among different users; 2) *Group-wise Clustering*, which further gathers the user-wise clusters by an adaptive clustering algorithm among different companies (detailed in Section 4.4).

Real Workplace Discovery. The generated candidates after being clustered, may still be very complex. This makes it difficult for traditional machine learning algorithms to characterize the underlying patterns of such data. To address this challenge, we make use of a multi-learner deep learning model to identify the real workplaces by analyzing users' position records, e-order information, point-of-interest (POI) and company profile information (detailed in Section 4.5).

4.3 Data Pre-processing

To remove unrelated records from e-commercial data from both temporal and spatial dimensions, we propose a filter-based method consisting of 1) *Temporal Filter* and 2) *Moving Filter*.

Temporal Filter. As most users work in day hours during workdays, the check-ins generated in office hours may reveal the company's real workplaces (see Figure 4.3(a)). However, it is difficult to know the exact working hours for each user due to flextime in some



(a) Example of Temporal Filter (b) Example of Moving Filter

Figure 4.3: Data Pre-processing.

companies. In case of moving out the informative records in the data pre-processing stage, we only filter out users' check-ins generated in holidays and the records from 0:00 am to 6:00 am in workdays.

Moving Filter. Since the companies' real workplaces are fixed locations, the user's check-ins should be removed if he/she is moving. However, it is not easy to distinguish whether they are staying or moving from consecutive check-ins since the position records are sparse in different situations. According to the example in Figure 4.3(b), the blue user moves fast in a short time period, while the time range between two consecutive check-ins p_1 and p_2 of the red user is relatively large. Thus, we utilize a heuristic method to filter out moving points. The current evaluated point will be filtered out if its speed is higher than a threshold. It is set to 100m/min since a user would rarely exceed this threshold when staying.

4.4 Candidate Location Generation

In this component, we generate candidate locations for the company's real workplace identification. The main idea is that the places where users gather in might be the workplaces of the company c_i. We gather users' position records, including check-ins $\mathcal{P}_{c_i}$ and shipping locations $\mathcal{S}_{c_i}$ by implementing a two-stage clustering algorithm to generate the candidate locations.

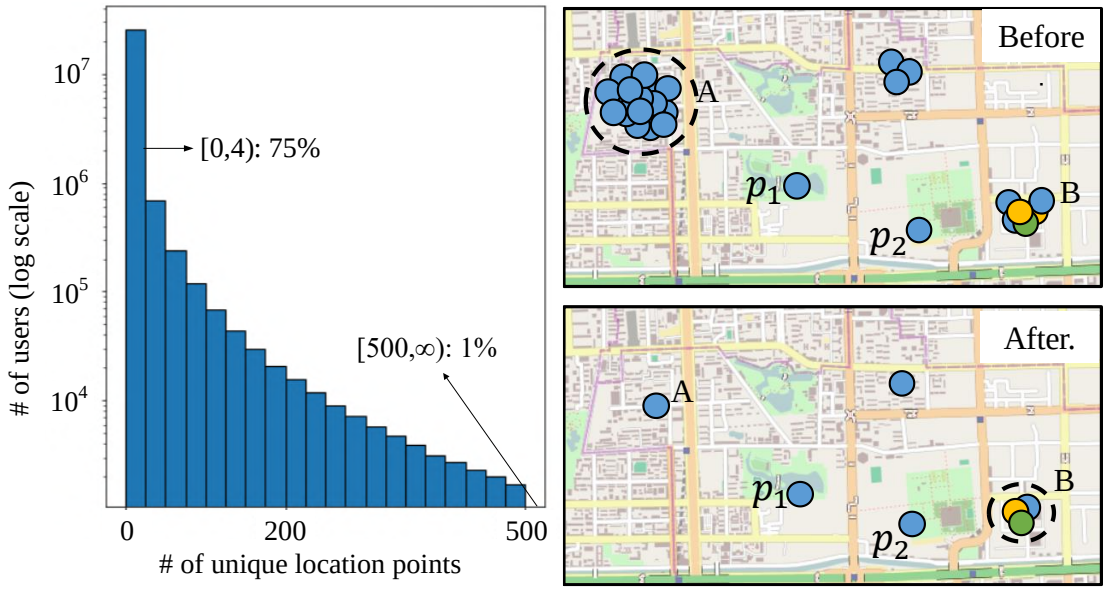
Definition 16. ***Candidate Location.*** *A candidate location is a spatial point, which is the centroid of a group spatially correlated user position. The candidate location is defined*

as $l = (lat, lng)$. *For each company* c_i, *the candidate locations are denoted as* $\mathbf{l}_{c_i} = \langle c_i, (l_1, l_2, \cdots, l_v) \rangle$, *where* v *is the number of candidate locations.*

However, there are two main challenges when generating high quality candidate locations: 1) due to the diverse preference of using the e-commerce App, there are skewed number of position records among different users, which makes the clustering algorithm tend to generate locations where most active users frequently stay, while regard the real workplaces as noises; 2) due to different company sizes, we cannot use the same parameters to cluster users' activity locations for various companies.

To this end, we propose a user-wise clustering algorithm, which normalizes the skewed check-ins among different users. And then we implement a group-wise clustering algorithm, which gathers user group position records by adaptively setting parameters for the clustering algorithm to generate candidate locations for each company.

4.4.1 User-wise Clustering



(a) Users' Check-ins Distribution

(b) Position Profile Comparison

Figure 4.4: Example of user-wise clustering.

Candidate locations should be generated by multiple users. Different users spend various time on the e-commerce App, resulting in the skewed number of check-ins. Figure 4.4(a) demonstrates the distribution of users over their number of check-ins in 2020.

We can see that 75% users have less than 4 unique check-ins, while 1% users have more than 500 unique check-ins. Such skewed distribution affects the quality of the clustering. As an example shown in the top of Figure 4.4(b), the area A may be considered as a candidate location, while the area B may be regarded as noise by the clustering algorithm due to the limited number. If we directly cluster the users' activity locations, we may generate redundant candidate locations and miss the correct one.

To address this issue, we cluster the check-in records user-by-user, so that different check-ins can represent different activity areas of the user. As shown in the bottom of Figure 4.4(b), a high quality candidate location would be generated in the next step. Clustering methods such as the density-based clustering of applications with noise (DBSCAN) [24] and hierarchical clustering [115] could be adopted to find individual locations for each user. Comparing with DBSCAN which removes points like p_1 and p_2, and all the user's activity areas can be kept even with only one record by the hierarchical clustering method. Thus, we implement a hierarchical clustering algorithm to get the unique areas where the user shows up. The Agglomerative proposed in [115] is a "bottom-up" hierarchical clustering algorithm that treats each position record as a cluster, then successively merges two clusters if the distance between centroids is smaller than a threshold D. In this work, we set $D = 100m$ based on our observation and use the centroids of clusters as the normalized check-ins for each user. The significance of user-wise clustering will be evaluated in Section 4.6.

4.4.2 Group-wise Clustering

In this part, we aim to obtain candidate locations for companies' real workplaces based on users' position records, i.e., cleaned check-ins and shipping locations. Considering that the areas of the companies might be in different scales, we generate locations from users by using DBSCAN [24]. Besides, DBSCAN can reduce noise information, which helps us remove redundant candidate locations. DBSCAN groups together points based on a maximum radius of the neighbor measurement ϵ and a minimum number of points $minPts$.

However, due to the different company sizes, we cannot set the same parameters to generate candidate locations in different companies. Figure 4.5 shows the clustering results of setting the same $minPts$ in a large company c_1 and a small one c_2. We can see that there are no candidate locations generated in company c_2 when the $minPts = 10$ (bottom left in Figure 4.5), while there are a large number of redundant candidate loca-

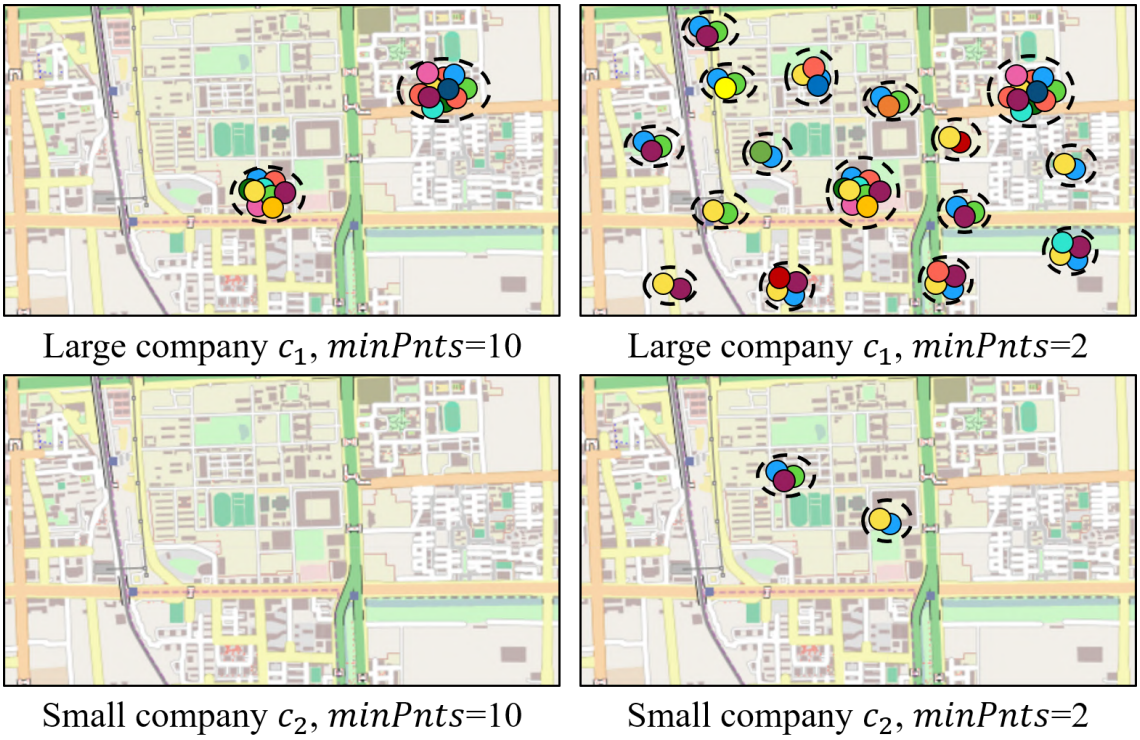


Figure 4.5: Parameters in DBSCAN.

tions in company c_1 when the $minPts = 2$ (top right in Figure 4.5). It is impossible to tune the parameters manually as the number of companies increases.

To overcome this challenge, we set the parameter $minPts$ according to the number of users in a company. It is inspired by the intuition that a large company has a large number of users gathered in, while a small company has a small group of users. As can be seen from Figure 4.5, both company c_1 and company c_2 contain a good number of candidate locations when setting with a large and a small $minPts$ respectively (top left and bottom right). Taking these observations into consideration, we set the parameter $minPts$ as,

$$minPts = max(1, \lceil \log(n) \rceil), \tag{4.1}$$

where n is the number of users in the company. As for the neighbor measurement, we set $\epsilon = 100m$ based on our observation.

4.5 Real Workplace Discovery

In this component, we would like to identify the real workplaces of the companies

from the candidate locations. We propose a multi-learner deep learning model, which fully incorporates spatial-temporal information from multi-source data, i.e., users' position records, e-order information, POI and company profiles to further identify workplaces from candidate locations.

However, there are three main challenges when identifying the real workplaces: 1) *Candidate relationship*. Various features have been extracted for each candidate location, but the relationship among candidate locations is not considered, such as the relative locality among candidate locations and the ranking information of each feature. How to capture the relationship among candidate locations when identifying the real workplaces is a challenge. 2) *Mobility patterns*. Users transition would affect the location identification [8]. However, feature engineering can only capture the static features for each candidate. How to extract such patterns is challenging. 3) *Company Profile*. Users in different companies have diverse performance, leading to divergent weights in each feature. How to capture the influence of companies' profiles is a challenge.

To this end, a multi-learner deep learning model is proposed to identify the real workplaces and the architecture is briefly depicted in Figure 4.6. We apply an attention-based classification method to tackle challenge 1) *candidate relationship* (in the yellow part) and then present two enhancement components: a mobility pattern leaner, which deals with the challenge 2) *mobility patterns* (in the blue part) and a profile feature learner (in the orange part), which solves the challenge 3) *company profile*. Finally, we show how we fuse these three learners to identify the real workplaces (in the green part). In the following, we elaborate them in details.

4.5.1 Candidate Relationship Learner

To tackle challenge 1) capturing the relationship among candidate locations, we extract features from two aspects and then simultaneously consider all the candidate locations in a company. The features we choose belong to two well-defined classes: geographic and purchase behavior features.

Geographic Features $\mathbf{f_g}$. The geographic features encode spatial and temporal information about the properties of the location candidates, which are extracted from users' position records, i.e., the check-ins $\mathcal{P}_{c_i}$ and shipping locations $\mathcal{S}_{c_i}$ and POI information for the company c_i. Details are as follow:

- *Spatial Features:* This class refers to features that describe the environment around the candidate location. The spatial features include: 1) *Locality*, which is the identification number to represent the coordinates of the candidate location, i.e.,$l \rightarrow \langle x, y \rangle$. In this

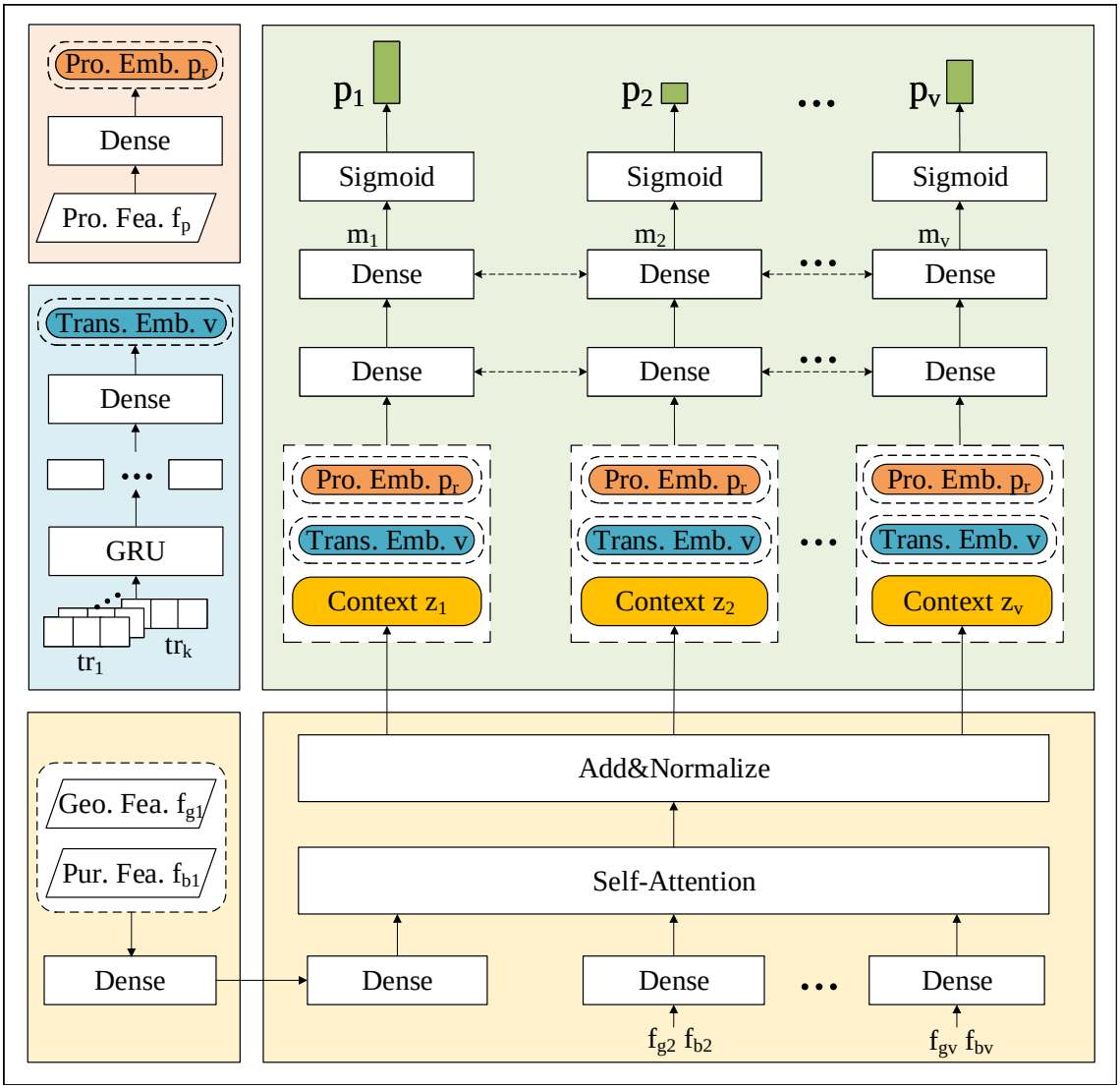


Figure 4.6: Architecture.

paper, we divide the study area into equally-size grid cells and assign the two dimensional grid cell IDs to each candidate location. The locality could show the relative position among different candidates. 2) *Density*, which counts the number of users and the number of position records in the candidate location cluster respectively. Intuitively, a denser area could imply a higher likelihood for being a workplace since the office is the place where all of the users visit. 3) *Surroundings*, which is the POI distribution of the candidate.

- *Temporal Features:* This class extracts features about the users' activity in the candidate location. The temporal features include: 1) *Activeness*, which calculates the number of active days of users in the candidate location. A larger number of activeness in the candidate locations could signify higher confidence for being a residence location. 2) *Heterogeneity*, which is the time distribution of users visiting the candidate location. We discretize [6:00,23:59) into hourly time slots and calculate the visiting time distribution.

Purchase Behavior Features $\mathbf{f_b}$. According to the study of Rensselaer Polytechnic Institute [84], approximately 1.7 million packages are stolen near home each day. Consequently, users usually ship the valuables or important items to companies when ordering online. Besides, office supplies are usually shipped to companies for convenience. Thus, the purchase behavior of users in different candidate locations may help us to identify the real workplaces, where we extract from the e-order information $\mathcal{O}_{c_i}$ for the company c_i. The set of purchase behavior features we devise include:

- *Purchase Density*, which counts the number of orders in the candidate location.
- *Purchase Power*, which computes the average price of items in each candidate location.
- *Purchase Preference*, which is the item distribution purchased in the candidate location. 7 categories (e.g., office supplies, digital products, clothing, etc) are extracted to show the purchase preference of users in the candidate location.

Model. Inspired by Transformer [107], which is capable to parallel learn the relationship among each element in a set, we introduce the multi-head self-attention mechanism into our LocRecognizer. In particular, we remove the positional encoding part since there are no temporal dependencies among candidate locations for a company. To a further step, for each candidate location in a company, the geographic $\mathbf{f_g}$ and purchase behavior features $\mathbf{f_b}$ are fed to a dense layer to get a representation for the candidate. Then the set

of representations are sent to a multi-head self-attention mechanism. The sequence of context vectors is denoted as $\mathcal{Z} = (\mathbf{z}_1, \cdots, \mathbf{z}_v)$, which is a high dimensional representation of each candidate location. Each context vector here has seen features from all other candidate locations (the yellow part in Figure 4.6). Mathematically,

$$\begin{aligned} Attention(\mathbf{Q}, \mathbf{K}, \mathbf{V}) &= softmax(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}})\mathbf{V}, \\ head_i &= Attention(\mathbf{Q}\mathbf{W}_i^{\mathbf{Q}}, \mathbf{k}\mathbf{W}_i^{\mathbf{K}}, \mathbf{V}\mathbf{W}_i^{\mathbf{V}}), \\ \mathcal{Z} &= Concatenate(head_1, \ldots, head_v)\mathbf{W}^{\mathbf{O}} \end{aligned} \tag{4.2}$$

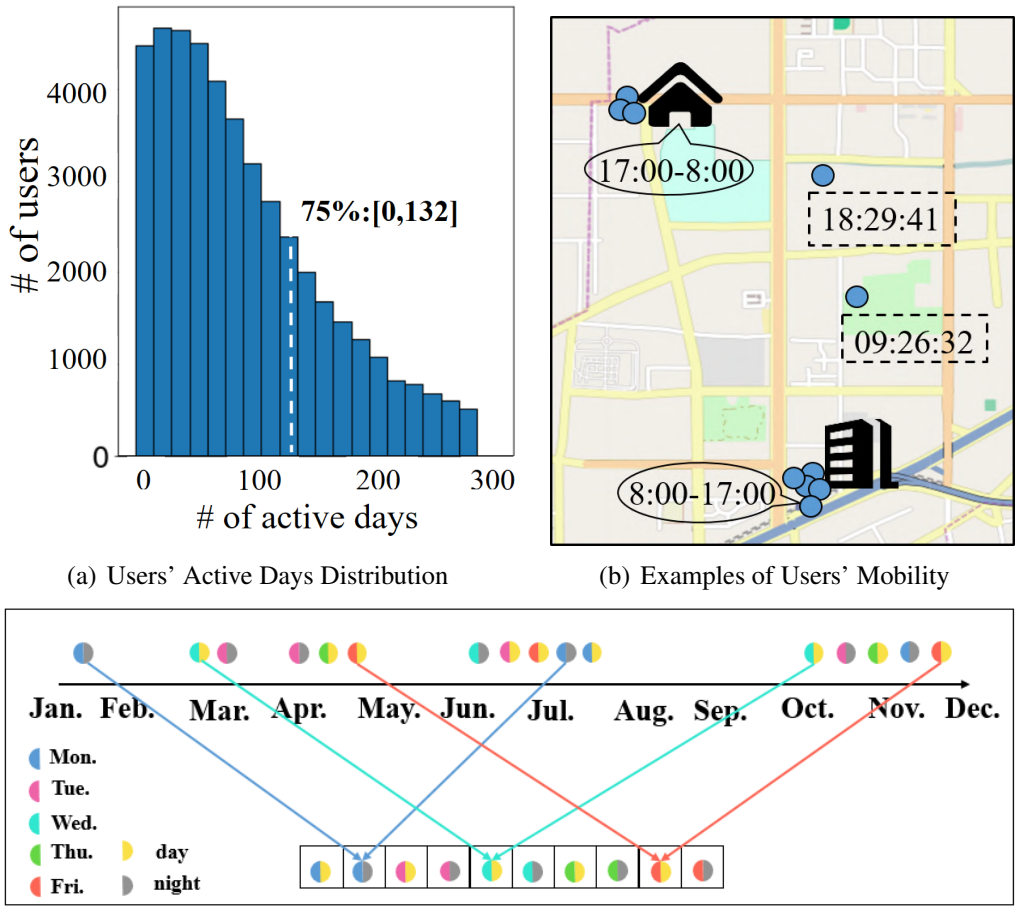
where $\mathbf{Q}$,$\mathbf{K}$,$\mathbf{V}$ are the query, key, and value matrices respectively learned from $\mathbf{f_g}$ and $\mathbf{f_b}$, and d_k is a fixed scaling constant, and W_*^* are appropriately-dimensioned weight matrices.

4.5.2 Mobility Patterns Learner

To solve the challenge 2) *mobility patterns*, we extract the mobility patterns to exploit knowledge about user movements and transitions among candidate locations via neural networks. The main idea is that users commute between home and work regularly on workdays (see Figure 4.7(b)). If we consider such transition in a time period as a sequence, we could learn the semantic meanings of the candidate locations by exploiting such movements.

However, there are other two challenges to generate the transition sequences. 1) Data sparsity. According to Figure 4.7(a), 75% users are active in 132 days during 2020, which is hard to extract informative transition sequence for a whole year. 2) Representation of the candidate locations. As section 4.4 mentioned, candidate locations are generated in different companies. And there is no relationship among candidate locations in various companies, which makes no sense to use the location ID to represent each candidate location.

To tackle the first challenge, for each user, we extract one transition sequence by aggregating his/her check-ins in one year. In particular, we first group the active time into [8:00, 17:00] and (17:00, 8:00), indicating working time and free time from Monday to Friday. Then the candidate location with highest visited frequency by the user in each time slot would be considered as one step in the transition sequence. As elaborated in Figure 4.7(c), each transition sequence consists of ten candidate locations organized chronologically. As for the second challenge, the geographic $\mathbf{f}_g$ and purchase behavior



(a) Users' Active Days Distribution

(b) Examples of Users' Mobility

(c) Example of Transition Sequence Generation

Figure 4.7: Mobility Patterns

features $\mathbf{f}_b$ are a good representation of the candidate locations. In this paper, we use $\mathbf{f}_g$ and $\mathbf{f}_b$ to represent each candidate location. The transition sequence is denoted as $\mathbf{tr} = \langle l'_1, l'_2 \ldots, l'_{10} \rangle$, where $l' = Concatenate(\mathbf{f}_g, \mathbf{f}_b)$.

In order to improve the accuracy, we extract the mobility patterns from transition sequences and learn the profile features of each company. For each company, we select k users transition sequences to capture the mobility patterns. To capture the mobility patterns in transition sequences, we implement the Gated Recurrent Unit (GRU) networks [16], which can learn temporal dependencies for sequential data without performance decay. Each of the transition sequence $\mathbf{tr}_i, \forall i, 1 \leq i \leq k$ would go through the GRU layer to get an embedding vector $\mathbf{e}_i$. Finally, we concatenate the embedding vectors $\mathbf{e}$ together to a dense layer to compress the mobility patterns for multiple transition sequences. The final embedding of transition sequences is denoted as $\mathbf{v}$ (the blue part in

Figure 4.6).

4.5.3 Company Profile Features Learner

The profile features are the property of a company. We assume that users from different companies may have various activeness and purchase preference leading to divergent weights in each feature. The profiles $\mathbf{f}_p$ include:

- *Business life*, which is the age of the company.
- *Type*, which is the industry type of the company, i.e., IT, sales, etc. The type information is a categorical value, converted by one-hot encoding.
- *Size*, which is the registered capital of the company. A large number of registered capital indicates a large size of the company.

In addition, considering that the profile features are a one dimensional vector, we employ a dense layer to learn the patterns (the orange part in Figure 4.6). The embedding of profile features is denoted as $\mathbf{p}_r$.

4.5.4 Prediction and Optimization

To this end, each company consists of one transition embedding vector $\mathbf{v}$, one profile embedding $\mathbf{p}_r$ and several high dimensional representations of candidate locations $\mathcal{Z}$. The green part in Figure 4.6 illustrates how we merge them together. We concatenate the embedding vectors $\mathbf{v}$ and $\mathbf{p_r}$ with each high dimensional representation $\mathbf{z}_i$ and then use a dense layer to fuse. Consequently, each candidate location not only obtains the geographic and purchase behavior features but also contains the mobility patterns and profile features of the company. The merged sequence of vectors is denoted as $\mathcal{M} = (\mathbf{m}_1, \cdots, \mathbf{m}_v)$.

Finally, we convert the problem as a multi-label classification problem. This is because a company may have multiple real workplaces, i.e., we need to identify multiple targets from the candidate locations of a company. Thus, we introduce a dense layer with the sigmoid function to predict whether each of the candidate location is the real workplace. The learning process minimizes the binary cross entropy loss which can help identify the real workplaces for each company. To achieve this goal, we propose the

following optimization problem:

$$\min_{\theta} \sum_{l_i \in \mathcal{L}_c} -(y_i \log(P_\theta(\mathbf{m}_i)) + (1 - y_i) \log(1 - P_\theta(\mathbf{m}_i))), \tag{4.3}$$

where $\mathcal{L}_c$ is the candidate locations of company c, y_i is the binary label of candidate location l_i in $\mathcal{W}_c$, and P_θ represents the neural network for predicting the probability of being a real workplace.

4.6 Experiments

4.6.1 Data Description

Datasets. To validate the effectiveness of our model, we conduct the experiments on a large-scale e-commerce platform in China. The key component of the identification is to find the related users of the company. Such information could be found from the tax sheets, companies' registration information, users' invoice data etc. In this paper, we extract such relationship from invoice information in the e-commerce platform, i.e., a user who issues an invoice with a company name would be considered as the company-related user of the company. Note that all the datasets related to users have been mapped to anonymous users to protect personal privacy. And all the experiments are conducted in the intranet. The detailed description of our datasets is as follows:

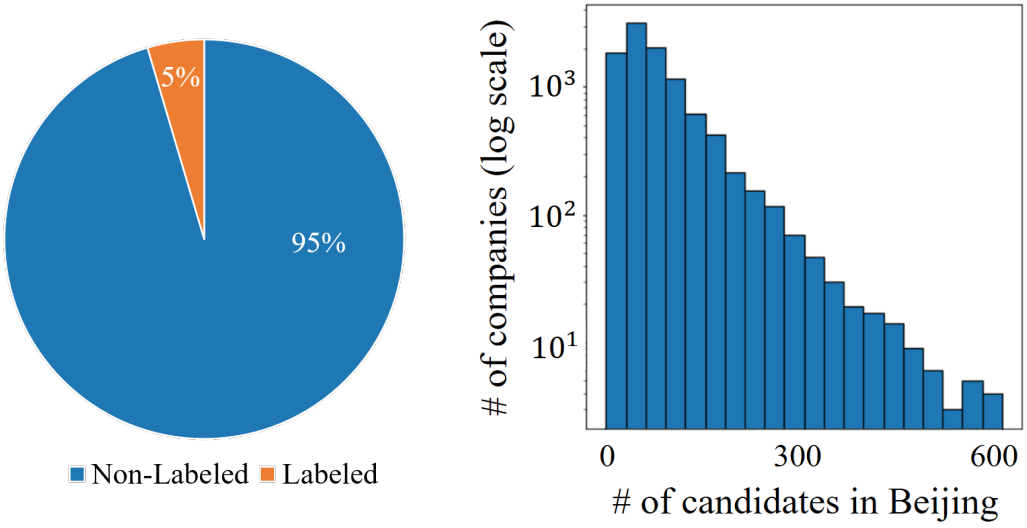
Table 4.1: Statistics of datasets.

Datasets	Beijing	Nantong
# of companies	22,170	2,316
# of users	247,325	9,890
# of check-in records	183,689,776	3,596,099
# of shipping locations	791,542	24,052
# of orders	9,185,611	224,225
# of workplaces	29,817	1,498
# of candidates	2,325,726	46,199
# of POIs	1,907,089	415,639

- **Users' Check-ins.** The check-ins are generated when users browsing the App with the location services authorization. We collect the data from Beijing and Nantong in 2020. The basic statistics of the datasets are shown in Table 4.1. On average, there are

around 11 users in each company with 742 check-ins in Beijing and 4 users with 1552 check-ins of each company in Nantong. Note that the average check-ins is much higher than the results in Figure 4.4(a), since users may use the e-commerce App in the same location and repeated check-ins would be generated.

- **Shipping Location.** The shipping location is extracted from the e-order information. On average, each user has 3.2 shipping locations in Beijing and 2.43 shipping locations in Nantong.
- **E-Order Information.** The E-order information contains an order ID, user ID, item name, category, price. There are around 9 million orders in Beijing and 0.2 million orders in Nantong during 2020.
- **POI.** The POI dataset contains a total number of 1,907,089 POIs in Beijing and 415,639 POIs in Nantong. The POI is categorized into 22 different types, such as hotel, shopping mall, house, etc.
- **Company Information.** The company information includes company name, registration date, registered capital, and business scope provided by the government. There are 22,170 companies in Beijing and 2,316 companies in Nantong in our datasets.



(a) Ratio of Labeled Companies (b) Distribution of Candidate Locations

Figure 4.8: Stats

Ground-Truth Construction. We extract the ground truth from users' shipping addresses via a natural language processing parser[51]. Specifically, if a company name

is included in the shipping address, the Geocoded [2] address location would be considered as the real workplace of the company. However, it cannot identify the company's real workplaces if the users do not fill the company name in the shipping address. As can be seen from Figure 4.8(a), users from 483,771 companies issue invoices from the e-commerce platform in Beijing 2020, while only 22,170 companies, i.e. around 5% of their names are included in users' shipping address. On the contrary, our method is a data driven approach, which can be used without such limits. After training our proposed model with current labels, we can identify other companies' workplaces in the real use.

Candidate locations are generated via DBSCAN and a candidate location will be labelled as the real working location if it is within 200 meters of the ground truth. As shown in Table 4.1, there are 29,817 ground truth labels and around 2 million candidates in Beijing, and 1,498 ground truth labels and around 46 thousands candidates in Nantong. Averagely, companies from Beijing contains 60 candidate locations and companies in Nantong have 20 candidates. The distribution of candidate locations in Beijing is shown in Figure 4.8(b). We split the dataset into training set, validation set and test set with a splitting ratio of 7: 2: 1 based on the companies, ensuring that companies in the testing set would not appear in training and validation sets.

4.6.2 Experimental Settings

Evaluation Metrics. To get a comprehensive evaluation of the proposed method, $Precision$, $Recall$ and F_1 score are adopted as the evaluation metrics. $Precision$ measures how accurate the model is out of the predicted labels. $Recall$ is widely used to measure the models' ability of finding all relevant elements. The F_1 score is a harmonic mean of the $Precision$ and $Recall$. Moreover, to overcome the randomness, the evaluation is repeated for five times, of which the average performance is reported.

Baselines. We compare LocRecoginzer with six baselines:

- Geo: This method simply uses the Geocoding results of the registered address as the real workplaces of companies.
- POI: This method implements the Geocoding results of the company names as the real workplaces of companies.
- MaxU: This is a heuristic method which selects the location among candidates with maximum number of users.

[2] https://en.wikipedia.org/wiki/Address_geocoding

- Random Forest(RF) [34]: RF is a binary classification method. In this baseline model, we independently identify each candidate location with the geographic and purchase behavior features.
- XGBoost(XGB) [15]: XGB is an advanced tree-based classification method, which is also implemented as a binary classification model to identify the workplaces among candidate locations.
- Multilayer Perceptron(MLP) [95]: MLP is a naive neural network framework, which is used to identify the real workplaces.

Variants. To evaluate each component of our proposed model, we also compare LocRecognizer with five different variants.

- L.R.-nG: The geographic features are removed from LocRecognizer to show its importance.
- L.R.-nB: We remove the purchase behavior features to reveal its importance of LocRecognizer.
- L.R.-nT: We remove users' transition sequences to evaluate the significance of mobility patterns learner.
- L.R.-nP: We take off the company profile features learner to evaluate how the profile features influence LocRecognizer.
- L.R.-nU: We train LocRecognizer without user-wise clustering.

Implementations. We train the deep neural network with a machine learning library, Pytorch, version 1.7.1. Our experiments run on a GPU server with 64 GB memory and a Tesla V100 GPU.

4.6.3 Effectiveness Evaluation

Overall Performance. We compare LocRecognizer with baseline models in terms of $Precision$, $Recall$ and F_1 score. The performance of different approaches in two cities for the real workplace identification is presented in Table 4.2. We have the following observations:

Table 4.2: Overall performance comparison. The best result for each evaluation metric is in bold.

Methods	Beijing			Nantong		
	Prec.	Rec.	F_1	Prec.	Rec.	F_1
POI	0.4221	0.3178	0.3626	0.3790	0.3295	0.3525
Geo	0.5632	0.4240	0.4838	0.4718	0.4102	0.4389
MaxU	0.6534	0.5176	0.5776	0.5957	0.5068	0.5476
RF	0.6748	0.6980	0.6862	0.6570	0.7007	0.6781
XGB	0.6741	0.7130	0.6930	0.6737	0.7191	0.6956
MLP	0.7234	0.6306	0.6738	0.7052	0.6479	0.6753
L.R.-nG	0.7149	0.6885	0.7015	0.6692	0.6993	0.6839
L.R.-nB	0.7605	0.7047	0.7315	0.7213	0.6799	0.6999
L.R.-nT	0.7482	0.7067	0.7268	0.7347	0.6863	0.7097
L.R.-nP	0.7681	0.6554	0.7073	0.7430	0.7011	0.7215
L.R.-nU	0.6490	**0.7748**	0.7063	0.6207	**0.7341**	0.6727
LocRec.	**0.7805**	0.7231	**0.7507**	**0.7736**	0.7125	**0.7418**

1. The POI method performs worst since it is based on the coverage of POI database in the Geocoding API. The performance indicates that more than 60% of the companies' real workplaces are not included in the database. The Geocoding results of registered address and the heuristic method have similar performance, which are worse than modeling methods. Both of these two methods have no ability to detect multiple workplaces for the companies, leading to an unsatisfied $Recall$.

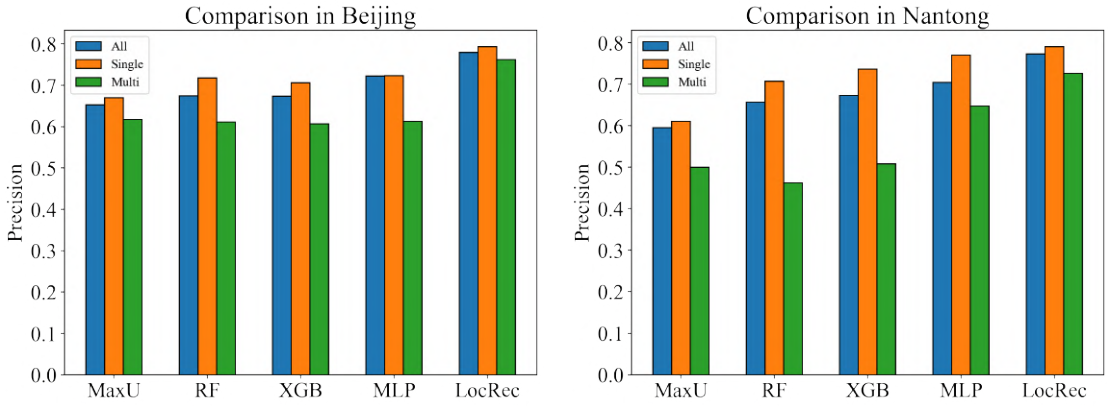
2. Although around 70% real workplaces are correctly identified by the tree-based models, the $Precision$ is not satisfied. In the real world, the government would inspect the company in vain if the $Precision$ is low. The MLP works the best in $Precision$ among all baselines, but it is still worse than our proposed model.

3. LocRecognizer outperforms best on F_1, indicating that our model seeks a good balance between accuracy and coverage. Compared with the best baseline, a 5.7% performance gain is witnessed in Beijing and 4.6% in Nantong in terms of F_1, which shows the significance of the candidate relationship learner.

Importance of Features. To evaluate the importance of each type of features, we compare the variants of L.R.-nG, L.R.-nB, L.R.-nT and L.R.-nP with our proposed model LocRecognizer. As can be seen from Table 4.2, the performance declines significantly when removing the geographic features. Especially, it causes the $Precision$ to decrease 6.6% in Beijing and 10.4% in Nantong, suggesting that the geographic features contribute

most. The results of L.R.-nB and L.R.-nP degrade slightly comparing with LocRecognizer, indicating that the purchase behavior features and company profile information contribute little to our proposed method. There is 3.2% decrease in Beijing and 3.9% decrease Nantong in terms of the $Precision$ after removing the mobility patterns, showing that users' mobility patterns could help us identify companies' real workplaces.

Importance of User-wise Clustering. To evaluate the significance of user-wise clustering, we compare our model LocRecognizer with L.R.-nU. An interesting observation is that the $Precision$ falls but the $Recall$ grows, showing that the user-wise clustering of users' check-ins can help improve the precision but hurt the coverage. One possible reason is that the user-wise clustering discards some of the users' position data, leading to the real workplaces not covered by the clustering algorithm in the first stage. However, a 13% performance gain in Beijing and 15% performance gain in Nantong of the $Precision$ illustrate that the user-wise clustering does reduce the redundant candidates to improve the accuracy of the identification.



(a) Comparison in Beijing

(b) Comparison in Nantong

Figure 4.9: Evaluation for robustness.

Robustness. Since some of the companies contain multiple workplaces, to evaluate the robustness of our model, we compare LocRecognizer with baseline models by testing companies with single and multiple workplaces respectively. As shown in Figure 4.9, the $Precision$ of all the models slightly increases in single workplace detection, indicating that companies with single workplace are easier to be identified. Since the identification of companies with multi workplaces is more complicated, the performance of all the models falls compared with companies with single workplace. However, the decreases of all the baseline models are larger than our LocRecognizer, which shows the robustness of our proposed model.

Inconsistency Detection. In our dataset, around 30% of the companies' registered addresses are inconsistent with the actual office addresses. To evaluate the effectiveness of inconsistency detection, we compare the predicted workplaces with the real ones for these 30% companies. The results show that 85.92% inconsistent companies can be detected in Beijing and 77.88% in Nantong.

4.6.4 Deployment

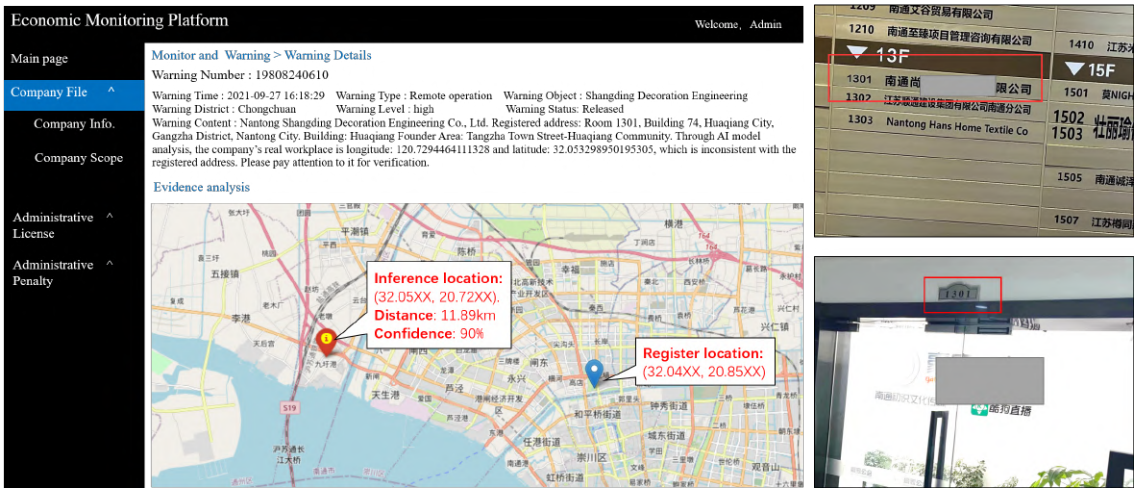


Figure 4.10: System

We further build a developed system to show the effectiveness of LocRecognizer in the real world. On Sep. 27^{th} 2021, an anomalous company was found in Nantong, China via LocRecognizer. As shown in Figure 4.10(on the left), LocRecognizer inferred the real working location of the company X, which is 11.89 km away from its registered address. The officers found that the guidepost on the first floor indicated the company X (figure on the right top in Figure 4.10), while another company Y was cooperated on the address (shown on the right bottom one in Figure 4.10). Both of these two companies were fined due to the location of main office was not consistent with the registered address.

4.7 Related Work

The location identification problem has attracted researchers from a broad spectrum of disciplines in recent years. Related work are summarized in three main areas: 1) geo-

graphic information collection; 2) location-based services data mining; 3) urban computing.

Geographic Information Collection. Geographic information is the collection of information about places and events that occur on the Earth's surface [19]. In early years, there are a large group of volunteers interested in providing geospatial data on public platforms [30], such as OpenStreetMap [31], Wikimapia [52], etc. However, such work relies on human labor which is inefficient and expensive. Recently, with the rapid development of machine learning, lots of researchers have made efforts on collecting geographic information via data mining techniques. Li et al. [54] used geotagged photos to provide a collective view of sense of place, in terms of significance and location. Mariotti et al. [77] investigated the location patterns and the effects of co-working spaces by analyzing the urban context. However, such existing works focus on enriching the information of a known geographic location instead of discovering a new location. With respect to previous work in the general area, in this paper we discovered the real workplaces of companies which were unknown before the detection.

Location-based Services Data Mining. Location-based services often collect users' mobility in different regions. Such information can be leveraged to suggest important locations for the governments, companies or users [47, 86, 126, 43, 142]. In particular, Zhu et al. [142] detected illegal chemical facilities with chemical truck trajectories, which clustered stay points to generate candidates and then proposed a classification model to identify the illegal chemical facilities. However, they implemented the same parameters to generate candidate locations which did not consider the different size of different companies. Karamshuk et al. [47] identified the optimal location for a new retail store by ranking geographic and mobility features of candidate locations. However, the extracted features highly depend on experts experiments, and the relationship among different features was not considered. Jiang et al. [43] used delivery data to identify wrong locations claimed by O2O merchants. They used GBDT to estimate relative distances among merchants, and propose a segment search algorithm to compute the actual location. Different from it, our workplaces are generated by mobility data and identified by an attention-based model.

Urban Computing. From a data mining perspective, we could classify our work in the area of urban computing, which studies the extraction of knowledge from spatial-temporal datasets and aims to improve services and intelligence in the city. Zhao et al. [136] developed a quantitative approach for detecting location spoofing with millions of geo-tagged tweets. Wang et al. [112] detected risky locations by calculating a risk

score based on transportation weight and crowd weight. Huang et al. [40] proposed a convolutional neural networks based method to find micro-seismic event location. Shubina et al. [100] helped Japanese government identify two clusters of the coronavirus disease 2019 (COVID-19) via web search query logs and user location information from location-aware mobile devices. The present paper is well aligned with this stream of work and extends the applicability of the urban computing methods to the field of location detection.

4.8 Conclusion

In this paper, we propose a novel two-step data mining method, i.e., LocRecognizer to identify companies' real workplaces based on users' records from the e-commerce platform. We first adopt a two-stage clustering method to generate candidate locations, and futher detect correlated workplaces by using a multi-learner deep learning model. Experiments show our proposed model significantly outperforms six baselines by at least 5.7% in Beijing and 4.0% in Nantong in terms of F_1 score. A real-world application system has been deployed in Nantong, China since Sep. 2021, which illustrates the effectiveness of our solution. Finally, in future work, we plan to explore the social relationship among users since it is limited to find company-related users via invoice data.

Chapter 5

Illegal Chemical Facility Detection

5.1 Introduction

hemical materials are widely used in every aspects of our daily lives, e.g., energy consumption and manufacture industries. Some of the chemicals can be highly dangerous to the public, e.g., flammable or corrosive. For example, on August 4, 2020, a chemical facility that stored ammonium nitrate at the Port of Beirut, Lebanon exploded, with 204 people killed and thousands injured in the blast [117] (As shown in Figure 5.1). As a consequence, the process, transportation and storage of these hazardous chemical materials are strictly monitored by the government [112]. However, motivated by enormous profit, there are many underground chemical businesses going on, where their storage and process facilities are not registered and lack of proper protection measures. For example, in Jan.2019, five unregistered chemical facilities were found nearby Melbourne, Australia in which stored more than 6 million litres of hazardous chemicals. These facilities will pose great danger to the public safety due to the mismanagement of chemicals, and if it caught fire, the chemicals in it would pose a similar health and environmental risk to the Beirut Explosion [116].

Existing methods to find these illegal chemical facilities either rely on the anonymous tip-off reports, or are based on the active on-field screening over some suspicious areas. However, both of them are time-consuming and achieve limited coverage.

Fortunately, hazardous chemical materials are highly regulated, and allowed to be transported using only the hazardous chemical transportation (HCT) trucks, as demonstrated in Figure 5.2(a). Such vehicles are requires to be equipped with GPS modules, as shown in Figure 5.2(b), and the GPS data is reported to the governments directly. Inspired

Figure 5.1: 2020 Beirut Explosion.

by the fact that HCT trajectories reflect the delivery activities at the chemical facilities, we can implement trajectory data mining method to detect the illegal chemical facilities.

However, it is not a trivial task to locate the illegal chemical facilities from the HCT trajectories directly: 1) there are multiple reasons to generate a stay point in a trajectory, e.g., at gas stations, parking lots, restaurants, etc; and 2) due to the inefficiency of traditional methods, we have very limited labels of illegal chemical facilities.

In this paper, we design a novel approach called ICFinder+ to solve the illegal hazardous chemical facility detection problem. ICFinder+ contains two main parts: 1) *Candidate Location Discovery*, which discovers candidate locations based on the stay points from the HCT trajectories; 2) *Illegal Facility Detection*, which extracts spatio-temporal and contextual features from trajectories and then detect illegal facility locations in a ranking manner. Moreover, we implement a supervision system and put it into real use. The main contributions of the paper are summarized as follows:

- We propose a novel and ubiquitous way to locate the illegal hazardous chemical facilities using HCT trajectories. The proposed method overcomes the drawbacks of coverage and massive human effort in the traditional methods.
- A two-step model is employed to discriminate the illegal hazardous chemical facilities in the paper. The candidate locations are first generated based on the aggregations of

(a) HCT Truck. (b) GPS Equipment.

Figure 5.2: HCT Regulation.

stay points in the trajectories. Each candidate location is modelled based on its spatio-temporal and other contextual features to infer the probability to be an illegal chemical facility.

- The proposed method is evaluated extensively over the real-world HCT trajectories. The results confirm that our proposed method is more effective than the baseline approaches.
- To verify real effect of ICFinder+, we implement a real-world system in Nantong, China since Nov.2020. Extra 20 illegal hazardous chemical facilities was found through our approach as a supplementary support just after people active screening.

5.2 Overview

In this section, we firstly define the important concepts that used extensively in this paper. And then, we outline overall framework of ICFinder+.

5.2.1 Definitions and Problem Statement

Definition 17. ***HCT Trajectory.*** *A HCT trajectory is a sequence of spatio-temporal points, denoted as* $tr = \langle p_1, p_2, \cdots, p_n \rangle$*, where each point* $p = \langle lat, lng, t \rangle$ *consists of a location (e.g., longitude and latitude) at timestamp* t*. Points in a trajectory are organized chronologically, namely,* $\forall i < n : p_i.t < p_{i+1}.t$*. The set of trajectories is* $\mathcal{T}$*.*

Definition 18. ***Stay Point.*** *A stay point is a sub-sequence of a trajectory, which means a moving object stays in a geographic region for a period of time. Mathematically, given a distance threshold D_{max} and a time threshold T_{min}, $\langle p_i, p_{i+1}, ..., p_j \rangle$ is a stay point sp, if $distance(p_i, p_k) \leq D_{max} (\forall k \in [i+1, j])$, $distance(p_i, p_{j+1}) > D_{max}$ (if $i < n$), and $|p_j.t - p_i.t| \geq T_{min}$. The set of stay points is $\mathcal{P}$. The coordinate and time of a sp is estimated using its spatial and temporal centroid:*

$$sp.lat = \frac{\sum_{i=k}^{j} p_k.lat}{j-i+1}, sp.lng = \frac{\sum_{i=k}^{j} p_k.lng}{j-i+1}, \\ sp.t = p_i.t + \frac{p_j.t - p_i.t}{2} \tag{5.1}$$

Definition 19. ***Candidate Location.*** *A candidate location is a spatial area, which is generated by a group of spatially correlated stay points from the trajectories. Location is defined as $l_c = \langle lat, lng \rangle$. The set of candidate location is $\mathcal{L}_c$*

Definition 20. ***White List.*** *The white list is a list of facilities, where hazardous chemicals can be legally manufactured and processed (denoted as $\mathcal{W}$). The type of a facility in $\mathcal{W}$ can be classified into two categories: 1) consumer; and 2) producer.*

Consumers can legally consume the hazardous chemicals, and are denoted as $\mathcal{C}$. Producers can legally produce hazardous chemicals, and are denoted as $\mathcal{P}$. $\mathcal{W} = \mathcal{C} \cup \mathcal{P}$. Each facility in W contains the following properties: 1) name of the facility; 2) permitted hazardous chemical list (producer only); and 3) location , $l_w = \langle lat, lng \rangle$.

Definition 21. ***Illegal Hazardous Chemical Facility.*** *Illegal hazardous chemical facilities store or consume hazardous chemicals without the proper registration. The illegal hazardous chemical facility is denoted as $l_i = \langle lat, lng \rangle$ and the set of illegal hazardous chemical facilities is $\mathcal{I}$.*

Problem Statement: Illegal Hazardous Chemical Facility Detection. *Given locations $\mathcal{L}_c = \{l_j | j \in 1, ..., n\}$ generated from HCT trajectories $\mathcal{T}$ and a white list $\mathcal{W}$, we aim to develop a system to infer the locations of illegal chemical facilities $\mathcal{I}$.*

5.2.2 Overall Framework

The modeling framework of illegal chemical facilities detection is illustrated in Figure 5.3, which consists of two main components:

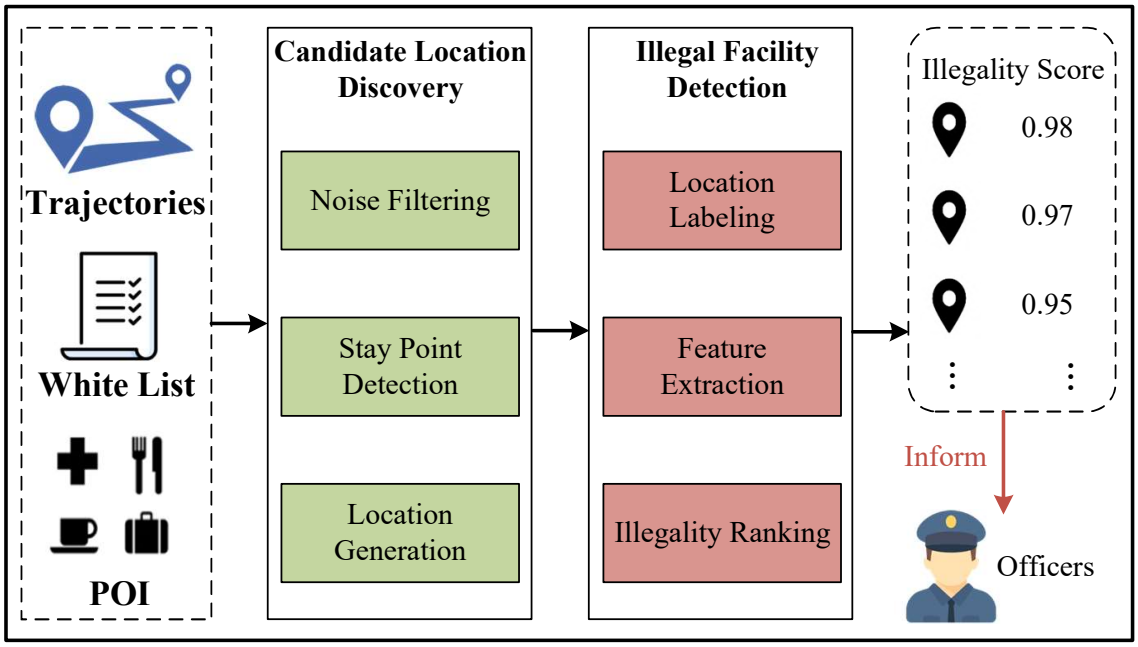


Figure 5.3: Modeling Framework.

Candidate Location Discovery. This component discovers the candidate locations of illegal chemical facilities with HCT trajectories, a white list, points-of-interest (POI). In this comment, three main steps are performed: 1) *Noise Filtering*, which removes the outlier GPS points in the HCT trajectories; 2) *Stay Point Detection*, which extracts all the stay points from trajectories; and 3) *Location Discovery*, which generate the candidate locations based on the clustering results of the stay points (detailed in Section 5.3).

Illegal Facility Detection. This component determines if a candidate location is a illegal chemical facility according to its probability to have loading/unloading events. In this component, we first labels the candidate locations based on the white list and domain knowledge, and then extract features via HCT trajectories and the corresponding contexts. Finally, a ranking model is used to detect illegal hazardous chemical facilities (detailed in Section 5.4).

5.3 Candidate Location Discovery

In this component, we extract the candidate locations from stay points of HCT trajectories which are stored in JUST [56]. The main intuition here is that HCT trucks have to stop, when loading/unloading hazardous chemicals. Thus, the stay points are highly related to

illegal chemical facility locations.

However, there are two main challenges to extract these candidate locations from the trajectories: 1) the GPS points of HCT trajectories contain some noise, during the data collection; 2) there are many random stops for an HCT truck during the chemical transportation.
To this end, in this section, we first process the trajectories with noise filtering techniques to remove the outlier points. After that, we extracts the stay points from the trajectories using a spatial clustering algorithm. Finally, the stay points are clustered to generate the candidate locations, where we can filter the random stops during the transportation.

5.3.1 Trajectory Noise Filtering

Due to the environment features and sensor errors, GPS points in a trajectory may have some shifts, which introduces difficulty and inaccuracy for the further analysis. In our model, we employ a heuristic-based outlier detection algorithm [139] to remove the noise points. In this algorithm, we first calculate the travel speed at each GPS point based on the time interval and moving distance between a point and its precursor. The current evaluated point will be filtered out from the trajectory if its speed is higher than a threshold. In our setting, the speed threshold is set as 130 km/h since the moving speed of a HCT truck would rarely exceed this threshold.

5.3.2 Stay Point Detection

In this step, we identify all the stop events from the HCT trajectories by extracting the stay points using a clustering-based algorithm [55]. The algorithm first checks if the distance between an anchor point and its successors in a trajectory is larger than a given threshold D_{max}. After that, the algorithm calculates the time span between the anchor point and the last successor point within the distance threshold D_{max}. If the time span is larger than a given threshold T_{min}, a stay point is identified and returned. In this problem, based on the experts domain knowledge, we set the parameters as: $D_{max} = 100m$, $T_{min} = 15min$.

5.3.3 Candidate Location Generation

The candidate location is generated by clustering a set of spatially related stay points, as each individual stay point in the trajectory may be generated by various random reasons.

Besides, if the candidate location is the destination for a HCT truck, there must be a set of stay points clustered nearby.

To this end, in this step, the candidate location is discovered from stay points via clustering. The centroid of the cluster is set as the position. Since staying area might be in different scale, we generate locations from stay points by using density-based clustering of applications with noise (DBSCAN) [24]. DBSCAN groups together points based on a maximum radius of the neighbor measurement ϵ and a minimum number of points in an ϵ-neighbor of that point $minPts$. It first finds the points in the ϵ-neighbor of every point, and identifies the core points which are the points if at least $minPts$ points are within distance ϵ of it. Then, DBSCAN selects the connected components of core points on the neighbor graph. Finally, it assigns each non-core point to a nearby cluster if the cluster is an ϵ-neighbor. The centroid of a cluster is the position of the candidate location. In this work, we try different parameter combinations and find that a good candidate location can be generated when $\epsilon = 50m, minPts = 8$.

5.4 Illegal Facility Detection

In this component, we apply a deep learning model to detect if a candidate location is an illegal hazardous chemical facilities.

The most straightforward solution to tackle the problem is classifying the candidate locations directly. However, only 22 illegal chemical facilities were detected in history, which makes it impossible to train the model with such limited labels. On the other side, there usually are some loading/unloading events at the chemical facilities.

Therefore, to detect the illegal chemical facilities, we only need to find all the candidate locations, which have loading/unloading events and are not on the white list.

To this end, we first label the candidate locations, based on the white list and the domain knowledge. Then, we train a model to predict if an uncertain location has the L/U events.

5.4.1 Location Labeling

In this step, we first categorize the locations into three groups: 1) white-listed L/U Locations; 2) Non-L/U Locations; and 3) Uncertain Locations. The details of each group are as follows:

- *White-listed L/U location*, which is a candidate location that has the loading/unloading events near a legal chemical facility. A candidate location is labeled as the *white-listed L/U location* if it is within 50 meters of a location in the white list.

- *Non-L/U location*, which is a candidate location that does not have loading/unloading events. Based on the domain knowledge, it is not possible to have any chemical loading/unloading events near some acceptable POI types, e.g., parking lots, toll stations, and service areas. As a result, a candidate location is labeled as the *non-L/U location* if there is an acceptable POI within 50 meters.

- *Uncertain location*, which are all the candidate locations that are not labeled by the above heuristics. All the potential illegal chemical facilities are included in this set.

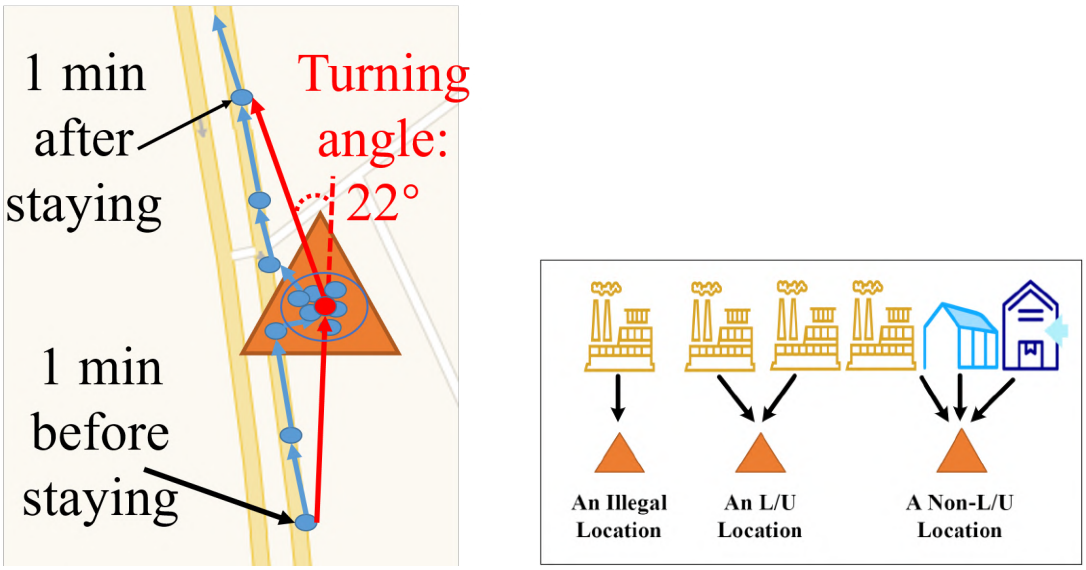
The first two groups of locations are used as positive and negative labels to train the L/U locations detection model. All the uncertain locations are used for illegal chemical facility detection.

5.4.2 Feature Extraction

In this step, we extract representative features to identify candidate locations with loading/unloading events. However, there is a challenging, as both legal and illegal chemical locations have L/U events, but many of their features are very different. Therefore, it will be inaccurate, if we use the features, which are only valid at the legal chemical locations (or white-listed L/U locations), to detect the loading/unloading events at the uncertain locations (i.e., illegal chemical facilities). For example, illegal chemical locations are more likely to locate in rural areas compared with legal chemical locations, and illegal chemical locations tend to be visited by fewer HCT trajectories according to their scale.

To this end, we should only select the features that are shared at both the legal and illegal chemical facilities, but, at the same time, are very different from the Non-L/U locations. Based on our observation, the key difference here is that, comparing to the Non-L/U locations, legal and illegal chemical facilities are destinations of a transportation and consume meaningful chemical combinations. Therefore, we select the features as follows:
Truck Behavior Features. To identify a destination in a trip, we extract the stay duration and turning angles, as follows:

- **Stay Duration.** It is quite intuitive a L/U event usually requires a significant amount of time to proceed. Thus, we extract the average, standard deviation, 20% and 50%



(a) Turning Angle. (b) Origination Comparison.

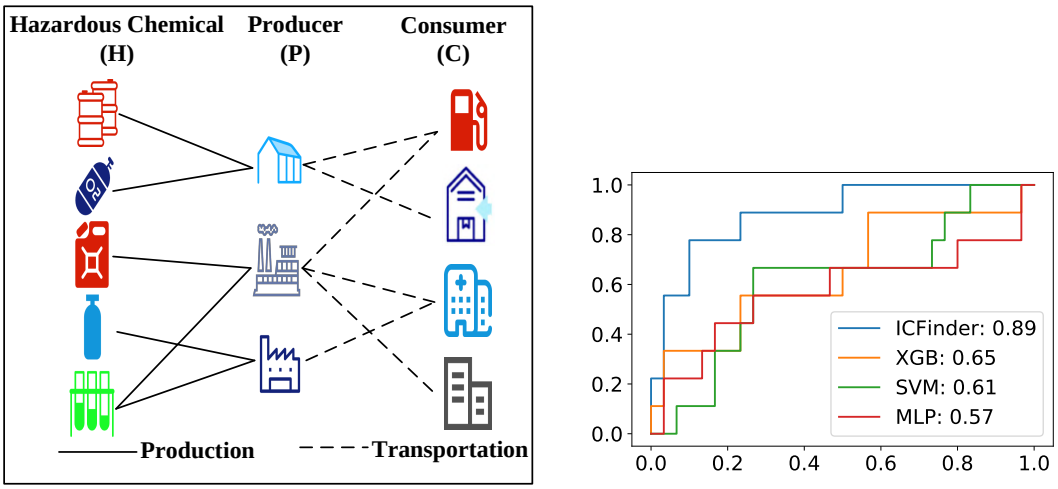
Figure 5.4: Intuitions for Feature Extraction.

percentiles value of stay duration to characterize the stay temporal behavior of HCT trucks at the candidate location.

- **Turning Angles.** It reflects the intention of the stay if it is a destination. Intuitively, HCT trucks will take a U-turn and go back after loading/unloading chemicals. A turning angle is calculated by connecting 1)the GPS points that are one minute before the stay point; 2) the centroid of stay point; and 3) the GPS point that is one minute after the stay point, as shown in Figure 5.4(a). We extract the feature by averaging the turning angles of all stay points at the candidate location to indicate if the visits are intentional.

Goods Features. Based on the domain knowledge, all of the chemical facilities, whether legal or not, require limited combinations of hazardous chemicals. Thus, only limited types of hazardous chemicals will be delivered to L/U locations, while there will be multiple types of hazardous chemicals showing in non-L/U locations. However, we do not know what kind of chemicals are delivered in each HCT trajectory and we do not know whether the HCT trucks contain chemicals or not when stopping by the non-L/U locations. Only the types of permitted chemicals are provided for each producer in L/U locations. Therefore, we need to infer the hazardous chemicals distribution for L/U locations and non-L/U locations via HCT trajectories .

1. **Latent Chemicals Distribution in L/U Location**: In this step, we learn the latent



(a) Hazardous Transport Graph.

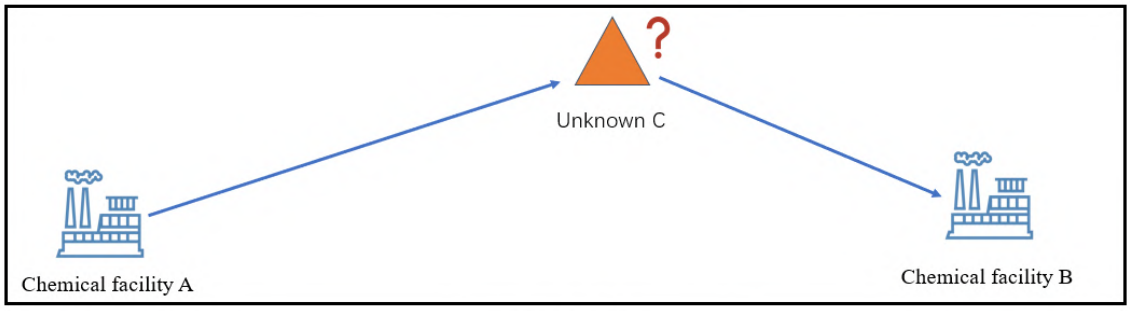
(b) Results.

Figure 5.5: Figures.

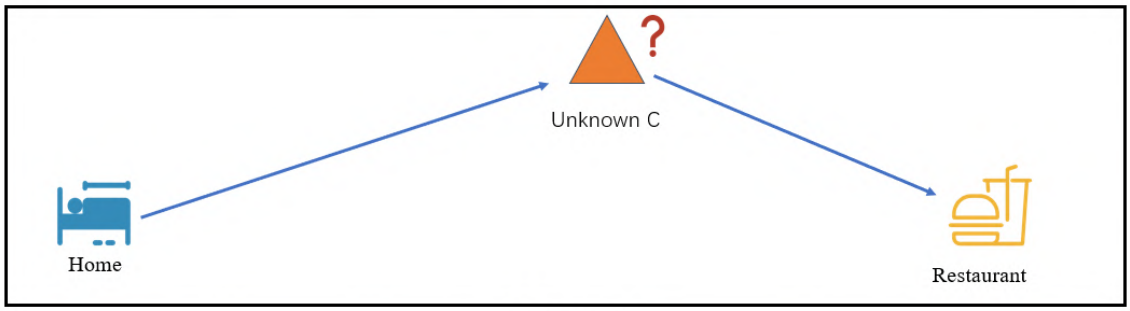
related chemicals for the producers and consumers. First, we leverage the permitted chemicals from producers, and the transportation relation between producers and consumers to construct a heterogeneous graph, i.e., Hazard Chemicals Transportation Graph (HCTG), as shown in Figure 5.5(a). There are three types of nodes in HCTG: hazardous chemicals $\mathcal{H}$, producers $\mathcal{P}$, and consumers $\mathcal{C}$. $\mathcal{P}$ and $\mathcal{C}$ are all derived from the white list, and $\mathcal{H}$ is the set of all hazardous chemicals that have appeared in all the producers. The edges between hazardous chemicals and producers are constructed if a producer can produce a specific chemical, and the weight is set to 1. The edges between producers and consumers are constructed by HCT trajectories. If the HCT trucks stay at a producer and a consumer consecutively, we add a heterogeneous edge between the producer and the consumer. The weight of the edge is set according to the total number of trips from the producer to the consumer. Then, we apply a heterogeneous graph representation learning algorithm Metapath2vec [23] to learn the embeddings of each producer and consumer in the graph, which reflects its latent related chemicals.

2. **Latent Chemicals Distribution in non-L/U Location**: The HCT trucks carry chemicals from facilities in the white list to a-non L/U location to have a rest. Thus, we aggregate by weighted average the embeddings of L/U facilities that each truck has stayed before coming to the current location to capture the latent chemical distribution in the non-L/U locations

Context Information. The HCT trucks may stop by multiple locations. With the context information, we can have a better understanding of the stop's purpose. As shown in Figure 5.6(a), the vehicle usually leaves chemical facility A and arrives at facility B, but one day it stops at an unknown place C. Then we may consider C as an illegal chemical facility. And in case II (shown in Figure 5.6(b)), the vehicle leaves home, stops at an unknown location, and finally arrives at a restaurant. We may consider the unknown place as a normal area since the driver may have a rest.



(a) Case I



(b) Case II

Figure 5.6: Cases of context information

To extract the context information, the most straightforward approach is to use a recurrent neural network to learn the latent representation of a group by trajectories which pass through a target stay area. Usually, we use POI or road network to represent a location. However, the high-dimensional feature sequence will suffer from the curse of dimensionality. Thus, we implement an autoencoder to learn the embedding of the target location. Specifically, we select all the trajectories which pass through the target stay area, and for each trajectory, we implement an LSTM to get the trajectory embedding. Then we merge them all and employ another LSTM with a self-attention module to learn the overall embedding.

5.4.3 Candidate Location Modeling

After extracting features of candidate locations, we train a binary classification model to detect the L/U locations and then rank the probabilities of uncertain locations to detect illegal facilities. During the training phase, we employ L/U locations as the positive labels and Non-L/U locations as the negative labels. During the actual deployment, we predict the probability of L/U locations for all uncertain locations and report the top candidate locations to the government. Figure 5.7 shows the architecture of candidate location selection, named LUInf, which first extracts the location and goods features, and then learns the context information, and finally merges these two parts as the red one to predict the loading/unloading patterns.

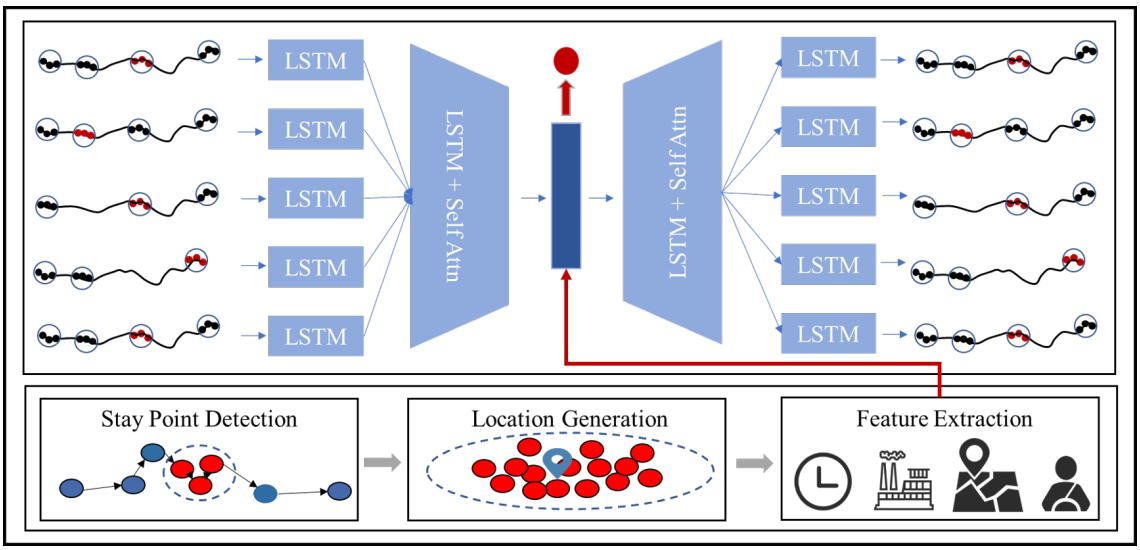


Figure 5.7: Architecture of Candidate Location Selection

5.5 Experiment Evaluation

In this section, we first describe the real dataset used in the experiments. After that, we design three experiments to evaluate the effectiveness of ICFinder: 1) the correction of identifying the L/U patterns at the candidate locations; 2) comparing ICFinder with other traditional end-to-end methods on the same dataset; 3) the coverage of the historical labeled data via ICFinder ranking.

5.5.1 Data Descriptions

The datasets used in experiments are all collected from the City of Nantong, China. The detailed descriptions are as follows:

HCT Trajectories. HCT trajectory datasets contain the 59.74 million GPS points generated by 2,891 HCT trucks which are operated in the City of Nantong, over a period of 5 months (i.e., from June 1^{th} to October 31^{th}, 2020). Each trajectory contains a truck ID, a series of GPS points and the corresponding timestamp. The average sampling time interval is around 2.5 minutes.

White List. The white list $\mathcal{W}$ contains 8512 facilities with 2,011 producer facilities $\mathcal{P}$ and 6,501 consumer facilities $\mathcal{C}$. The total number of hazardous chemical items $\mathcal{H}$ is 1,347 while 35% of them only produced by less than 4 facilities we only use 875 of them.

POI. The POI dataset contains a total of 415,639 POIs, and are categorized into 22 different types.

Ground Truth Labels. Our ground truth labels come from reports of local government screening operations at the same time period of the trajectory data. We have a total of 22 verified illegal hazardous facilities.

5.5.2 Evaluation of L/U Pattern Modeling

In this experiment, we evaluate the effectiveness of different ranking methods. The training data is collected with 3752 Non-L/U locations from the local authority and sampled 3652 L/U locations using the method described in Section 5.4.1. We use 60% locations for training, 10% of locations for validating, and the rest 30% locations for testing.

HCTG Embedding Learning. The embedding of the candidate locations is calculated based on the HCTG. We generate 500 random walks for each node in the length of 50. The embedding size of the node is set as 64. During training process, the learning rate is initialized as 0.025 and adjusted with cosine annealing scheduler.

Evaluation Metric. We use precision, recall, F1-score as the evaluation metrics. Specifically, we count the number of True Positive N_{TP}(correct identification of L/U location), False Positive N_{FP}(incorrect identification of non-L/U location) and False Negative N_{FN}(incorrect identification of L/U location) to calculate them:

Table 5.1: Results of L/U Pattern Modeling

	F1	Recall	Precision
SVM	0.8889	0.8970	0.8810
MLP	0.8631	0.8424	0.8742
XGB-driver_behavior	0.7229	0.7273	0.7186
XGB-goods	0.8704	0.8545	0.8868
XGB	0.8968	0.9212	0.8736
LUInf	0.9351	0.9427	0.9276

$$Precision = \frac{N_{TP}}{N_{TP} + N_{FP}}, \tag{5.2}$$

$$Recall = \frac{N_{TP}}{N_{TP} + N_{FN}}, \tag{5.3}$$

$$F_1 = \frac{2Precision \times Recall}{Precision + Recall}. \tag{5.4}$$

Candidate Models. We compare the effectiveness of different classical machine learning classification methods of Multi-Layer Perception (MLP), Support Vector Machine (SVM), Xgboost (XGB) and LUInf.

Variants. We also compare our methods with three variants:

- XGB-Truck Behavior Features, which only uses truck behavior features and predicts the L/U patterns with XBG.
- XGB-Goods Features, which only uses goods features and predicts the L/U patterns with XBG.
- LUInf-Context Features, which uses all the features without context features and implements the model with XGB. This model is implemented in ICFinder [142].

Results. The results are shown in Table 5.1, where all of the classification models achieve a high level of accuracy and recall. Particularly, LUInf with all the features, achieves the best performance. The results of variants indicate that both driver behavior and goods features are important in improving the effectiveness. Moreover, the context features contribute even more significantly to the results.

Table 5.2: Results of Illegal Facility Detection

	AUC
MLP	0.57
SVM	0.61
XGB	0.65
ICFinder	0.89
ICFinder+	0.93

5.5.3 Evaluation of Illegal Facility Detection

In this experiment, we compare our methods with models that directly trained with the limited the labeled illegal chemical facilities.

Baselines. We employ the same classification model and the following features.

- Spatial Features: 1) POI category distribution; 2) total length of road network; and 3) number of road intersections.
- Temporal Features: 1) the average of stay points per day; 2) the average number of unique drivers per day; 3) the mean and variance of vehicles staying time, and 4) the average turning angles of vehicles.

Baselines are trained on 60% of those labeled data and compared with our method trained on Section 5.5.2. Then the left 40% of data are used for testing.

Evaluation Metric. For this imbalance binary classification task we evaluate the performance by AUC (Area under the ROC Curve) which reflects model performance with different discrimination thresholds.

Results. The results are shown in Table 5.2. As we can see, although our method is not directly trained on the labeled anomaly data, ICFinder is 0.24 higher than the best baseline with XGB and ICFinder+ outperforms ICFinder by 4%. Our method can accurately capture the loading/unloading patterns. Although baselines learned from data of anomaly locations directly and used more features, the anomaly locations vary with each other in many ways and can not be learned well with limited labels. The result confirms that the loading/unloading pattern is the most significant factor, and therefore leads to the best results.

5.5.4 Ranking Results

To objectively demonstrate the effectiveness of our method, we mix the 22 labeled ground truth and 1,142 uncertain locations discovered by our candidate location discovery component, and rank them by the probability of L/U in descending order. Then, we evaluate the number of recovered cases in the top-k list. For top-5, top-10 and top-15 list, there are 1, 3, and 6 cases recovered in the lists, respectively.

It is confirmed that our method effectively detect the facilities that was originally detected by the traditional approaches. For the other high ranked candidate locations, all of them are sent to the government officials as the tips for their next operation, detailed in the next section.

5.6 Related Work

In this section, we summarize the related work in three main areas: 1) HCT problem; 2) trajectory data mining; 3) urban computing.

Hazardous Chemical Transportation Problem. The problem of hazardous chemical transportation (HCT) attracts great attention in urban management since it is ubiquitous and dangerous. In academia, tremendous efforts have been devoted to dealing with HCT problem [112, 46, 108, 109, 88, 26, 87]. Kara et al. [46] addressed the problem of HCT route planning by focusing on the nature of the relationship between the regulator and carriers. Fabiano et al. [26] presented a site-oriented framework for hazardous chemical facilities risk assessment. Planas et al. [87] provided a monitoring system to monitor HCT trucks based on regional responsibilities. Wang et al. [112] proposed a system for risky zones identification based on HCT trajectory data and human mobility data. However, most existing literature rely on that the locations of hazardous chemicals facilities are known, as opposed to our object which the illegal facilities locations are unknown and need to be detected.

Trajectory Data Mining. Trajectory data mining is a general research area which studies discovering various knowledge from massive trajectory data [139], such as map generation [97], driver behavior analysis [92], taxi demand prediction [125], etc. In particular, Lian et al. [63] proposed a Collaborative Exploration and Periodically Returning model to predict whether people will seek new locations to visit. This problem is similar to verify whether a location is visited by HCT drivers to do L/U behavior. However, they extracted a large number of spatial-temporal context features for the prediction, while we cannot

utilize such features which was fully considered in the Section 5.4.2 so that we need to come up with other effective features. Bae et al. [6] used stay points clustering and pattern mining to identify suspicious locations, while they did not consider the correlations among different locations, which was important in hazardous chemicals business.

Urban Computing. Urban computing [140] is a general research area which integrates urban sensing, data management and data analytic together. In particular, a lot of previous works help government with urban planning, such as bike lane planning recommendation [9], traffic estimation for urban deployment plan [135]. In addition, some efforts have been dedicated to public security, such as crime rate inference [111], unregistered taxi drivers identification [92], crowd flow alert [133], fire risk prediction [76]. To improve the scalability and efficiency of urban trajectory data mining applications, some trajectory data management systems are developed [56, 58, 57, 96]. In this work, we study HCT trajectories and provide ICFinder to solve the illegal chemicals facilities problem.

5.7 Conclusion

In this paper, we proposed a novel approach called ICFinder+ to find out unregistered or unqualified hazardous chemical facilities by mining HCT trajectories. We converted the problem of detecting illegal hazardous chemical facilities into that of modeling loading/unloading location patterns and trained the model by incorporating white lists and POI data. Full-fledged trajectory features, including spatio-temporal, HCT-behavioral, goods embedding and contextual features, were taken into consideration in loading/unloading modeling. The feasibility and the ubiquity of ICFinder+ were evaluated with a reasonably big volume of real-world data.

On the basis of ICFinder, we have implemented a real-world application system, which was deployed in Nantong, China since Nov.2020. Since then extra 20 illegal facilities have been found out from the same areas that were fully screened by professional people before.

We are willing to achieve more precise ranking through the following future works: 1) As the label data accumulates, we are to incorporate integrated anomaly detection methods.2) We are to scrutinize the difference between illegal facilities behaviours that might enable us to extract other features or make more particularity models. 3) We are to better enable the fusion of other kinds of data sources and use the integrated data to further increase the precision.

Chapter 6

Learn2Stay

6.1 INTRODUCTION

The stay point detection is a common problem in trajectory data mining. Researchers use stay points to understand the semantics of location and human mobility. However, traditional stay point detection algorithms are highly dependent on the GPS data [72, 114, 48, 120, 141]. Although GPS devices have been widely used to locate various types of moving objects, it is still difficult for city managers to collect all objects' GPS data. In fact, the government only has access to GPS data for a small percentage of objects. Fortunately, there are a large number of sensors which determine the present location installed in urban space, such as cameras, ETC, base stations, etc. Such third-party sensing devices offer a whole new opportunity to solve the problem of stay point detection. Among them, surveillance cameras are the most widely used. By 2018, more than one billion surveillance cameras were deployed worldwide [10]. These cameras help governments manage their cities more efficiently. Compared to GPS devices, surveillance cameras have the following three advantages:

- Easy access: Compared to GPS devices, which are prone to privacy, irregular operation, and equipment failure, surveillance cameras are installed and maintained by city managers in a unified manner, allowing stable collect objects' camera records.

- Wide coverage: All moving vehicles and major areas in the city can be monitored by surveillance cameras. For GPS trajectories, only a portion of vehicle trips can be accessed, which is a barrier to understanding traffic patterns.

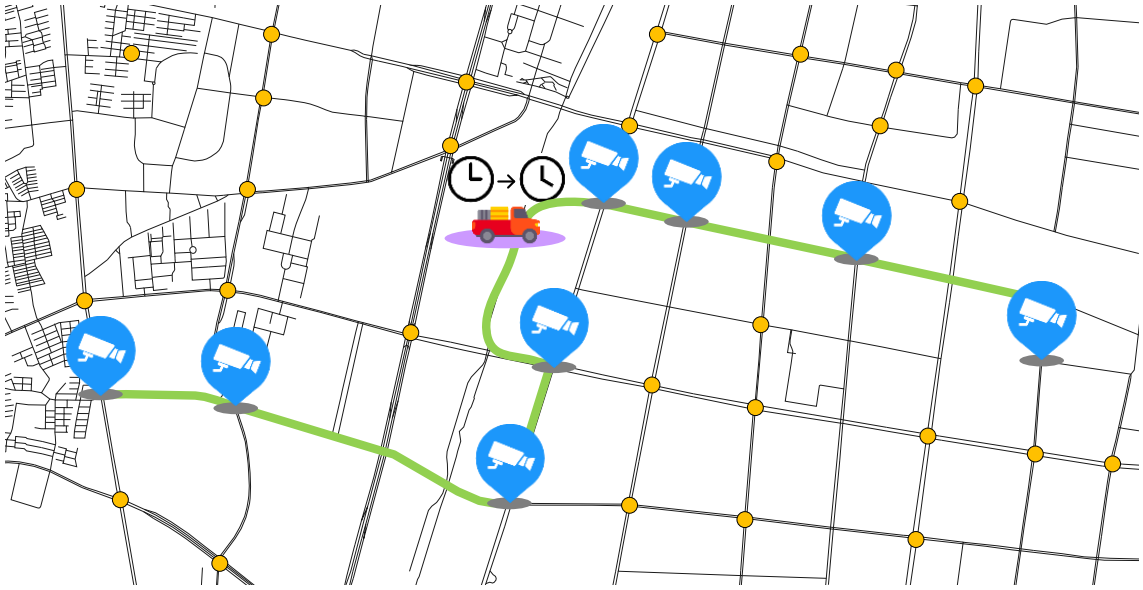


Figure 6.1: Example of Camera Record with Stay Point.

- Private friendly: Surveillance cameras are able to perceive the mobility in the urban space without acquiring details of the object's trajectory.

Due to the development of computer vision, researchers are able to use videos or snapshot data from surveillance cameras to recover the object's trajectory [66, 106, 137, 127], which is the basis of our work. There are rich application scenarios for inferring the stay area based on such sparse trajectories: 1) Manage special vehicles. For special vehicles, stay event often indicates potential safety risks. For example, the stay locations of chemical transport vehicles reveal illegal chemical storage, and the stay locations of construction waste transfer vehicles relate to illegal dumping. 2) Discovering potential popular areas. Because of the wide coverage of surveillance cameras, we are able to discover popular areas of the city without bias compared to the traditional method using taxi trajectory. 3) Mining vehicles' mobility. The stay behavior can reflect the travel intention of the object. However, the semantic information of the camera visited by the vehicle is limited. The sequences of stay areas inferred from surveillance camera records are able to model the mobility of objects better.

Generally, when a vehicle travels on the road network, it is captured by the surveillance camera and generates a vehicle-device binding access record through object detection and license plate recognition technology. Figure 6.1 shows a trajectory with stay points, the green line indicates the real trajectory of the vehicle, and the dots indicate the

camera position. When the vehicle passes the camera in order, some camera records are generated, such as the blue icon. Obviously, the sparse trajectory composed by camera records is a down-sampling of the real trajectory of the object, which presents a huge uncertainty when used in the stay point detection problem. Uncertainty is mainly reflected in three aspects:

1) Stay Event Uncertainty. For two consecutive camera records with stay points, the time interval is divided into two parts, the stay time and the moving time. According to the definition of stay point [139], when the stay time at a location is greater than the threshold, a stay event is considered to occur. However, the stay time within the blind spot of the cameras is difficult to estimate due to the fact that the camera record unlike the GPS trajectory isn't updated according to the fixed frequency, as shown in Figure 6.2(a). It is not possible to detect whether the stay events have occurred from the camera record using the traditional method.

2) Stay Area Uncertainty. Although the cameras have wide coverage, most of them will only be deployed at key nodes of the road due to resource limitations which leads to a blind spot between cameras. We statistics all two consecutive camera records having stay points and estimate the potential stay area using the spatial distribution of real stay points as in Figure 6.2(b). Figure 6.2(b) shows that the greater the range of locations where the vehicle may stay when two consecutive cameras are positioned further apart. And, there are more than 75% records whose potential stay exceeds 10 square kilometers. Traditional stay point detection algorithms cannot be applied to such sparse trajectories.

3) Instability of the surveillance camera. Affected by the environment and detected objects, surveillance cameras make recognition errors once in a while, which are most commonly in false alerts and missed detection. Taking Figure 6.2(c) as an example, when a missed detection occurs, it can further enhance the stay area uncertainty because of the missing next record. And false alerts are produced as noise samples which have a negative impact on modeling stay areas.

To overcome the uncertainty, we design a two-stage approach where the algorithm first detects whether a stay occurs within each of two consecutive camera records, which is used to eliminate the effect of stay event uncertainty. In the second stage, inspired by recommendation systems, we propose a layer-by-layer solution process combining coarse-grained and fine-grained selection to reduce stay area uncertainty. Specifically, the coarse-grained selection module uses the spatial distribution of stay points to generate the candidate region set as initialization, and the fine-grained selection module infers the

exact stay areas by modeling the stay probability of each region within the candidate region set.

To this end, we propose and implement a novel framework for stay area detection, named SAInf (Stay Area Inference). SAInf automatically mines the vehicle stay areas by modeling the stay behavior. The proposed framework contains three main components: 1) data pre-processing, which cleans, detects the stay points from the GPS trajectories, and assigns the stay points to the corresponding camera record pairs. 2) Stay event detection, which detects stay events within camera record pairs by testing travel speed. and 3) stay area identification, which generates candidate regions by modeling the spatial distribution of stay points and selecting the most likely stay area in its candidate region set. The main contributions of the paper are summarized as follows:

- As far as we know, our work is the first research to perform the stay point detection problem on a sparse trajectory. We analyze the stay point detection problem under a sparse trajectory setting and present the problem formulation and challenges in this paper.

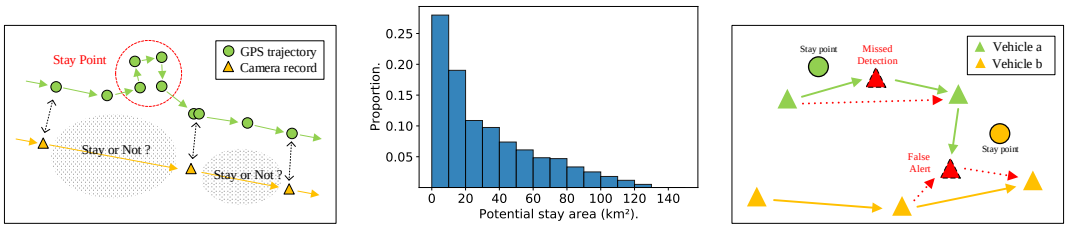
- A novel stay point detection framework is proposed, called SAInf. SAInf provides a standardized pipeline for addressing the uncertainties from camera records. Noteworthy, we also design a model called StayNet for fine-grained stay area identification. StayNet models the complex relationship between candidate regions and stay points, taking account of various factors.

- We evaluated the proposed method using a real dataset from Nantong, China. The results show that our method is efficient, with a performance improvement of about 58% compared to the current state-of-art baseline.

- A service based on SAInf is deployed in real-world applications.

6.2 OVERVIEW

In this section, we first give some preliminaries and used notations, then we define the stay point detection problem under sparse trajectory and outline our solution framework.



(a) Stay Event Uncertainty. (b) Stay Area Uncertainty. (c) Instability of the Surveillance Camera.

Figure 6.2: Challenges to Identify Stay Areas.

6.2.1 PRELIMINARY

Definition 1 *(Region)*: To simplify the problem, we partition a city into disjointed $M \times N$ grids based on latitude and longitude, where a grid denotes a region. All the grids form a region set $R = \{r_{ii}, r_{ij}, \ldots, r_{mn}\}$, where r_{ij} is the cell region in the i-th row and j-th column of the grid map.

Definition 2 *(Camera Check-in Point)*: In the sparse trajectory setting, we call the camera record is camera check-in point. A surveillance camera record is generated when a vehicle passes a camera. The record contains the mobility of the vehicle, denoted as a 3-tuple $ccp =< vid, cid, t >$, where vid denotes vehicle ID, cid indicates the visited camera and t is the visit time. A camera is bound to a fixed location $location(cid) =< lat_{cid}, lng_{cid} >$.

Definition 3 *(Camera Check-in Pair)*: A camera check-in pair consists of two consecutive camera check-in points. The pair is represented as $ccpr =< ccp_i, ccp_{i+1} >$, where ccp_i denotes a camera check-in record. For simplicity, we use the CC pair to indicate it.

Definition 4 *(GPS Trajectory)*: A trajectory is a sequence of GPS points, denoted as $tr =< p_1, p_2, \ldots, p_n >$, where each point $p =< lng, lat, t >$ indicates the longitude and latitude at a location time t. The points in the trajectory are organized chronologically.

Definition 5 *(Stay Event)*: A stay point sp means that a moving object stays in a geographic region for a while, which is a subsequence of GPS trajectory. Formally, given a distance threshold D_{max} and a time threshold T_{min}, $< p_i, p_{i+1}, \ldots, p_j >$ is a stay point sp while $distance(p_i, p_k) \leq D_{max}(\forall k \in [i+1, j])$, $distance(p_i, p_{j+1}) > D_{max}(if \quad j < n)$ and $|p_j.t - p_i.t| \geqslant T_{min}$. The time interval of a sp is $p_j.t - p_j.t$. In our work, we are more concerned with whether and where the moving objects stay than with the time interval of stay points. Therefore, stay points are regarded as stay events, and stay points correspond to stay events one by one.

Problem Statement. Given historical GPS trajectories $TR = \{tr_j | i \in [1, \ldots, p]\}$ and CC pairs $CCPR = \{ccpr_i | i \in [1, \ldots, q]\}$, the stay point detection problem under sparse trajectory setting is to infer regions to stay.

6.2.2 Framework Overview

The system framework of SAInf is elaborated in Figure 6.3, consisting of three components:

Data Pre-processing. This component with the CC pairs and GPS trajectories performs three main tasks.1) *Trajectory Noise Filtering*, which is used to remove outlier GPS points. 2) *Stay Point Detection*, which extracts all stay points from GPS trajectories as ground truth. 3) *Stay Pair Matching*, which matches CC pairs with stay points in chronological order.

Stay Event Detection. This component provides a pipeline for detecting stay events from CC pairs, considering the difference between the two types of CC pair patterns. Two main tasks are performed:1) *CC pair Modeling* , which characterizes the CC pairs by modeling the travel speed. 2) *Stay Query Pair Detection*, which builds a stat model for two types of CC pairs to detect if the stay event occurs within CC pairs.

Stay Area Identification. This component receives CC pairs having stay events. Two steps are designed in this component to estimate the stay area: 1) *Candidate Regions Generation*, which generates a candidate region set for initialization by modeling the spatial distribution of real stay points. 2) *Stay Area Selection*, which builds a deep learning model to estimate stay probability in each candidate region, and output the top-k regions with the highest probability.

Specifically, CC pairs and GPS trajectories are fed into the framework in the offline learning phase, then the SAInf detects the stay events from the GPS trajectories as ground truth. SAInf extracts stay patterns by learning the relationship between CC pairs and their corresponding stay events. In the online inference phase, the SAInf receives CC pairs and uses the trained stay detection module and stay area identification module to infer the regions where vehicles stay.

6.3 Data Pre-processing

In this component, trajectories are cleaned, and stay points are extracted. Then the stay points and CC pairs are matched in chronological order.

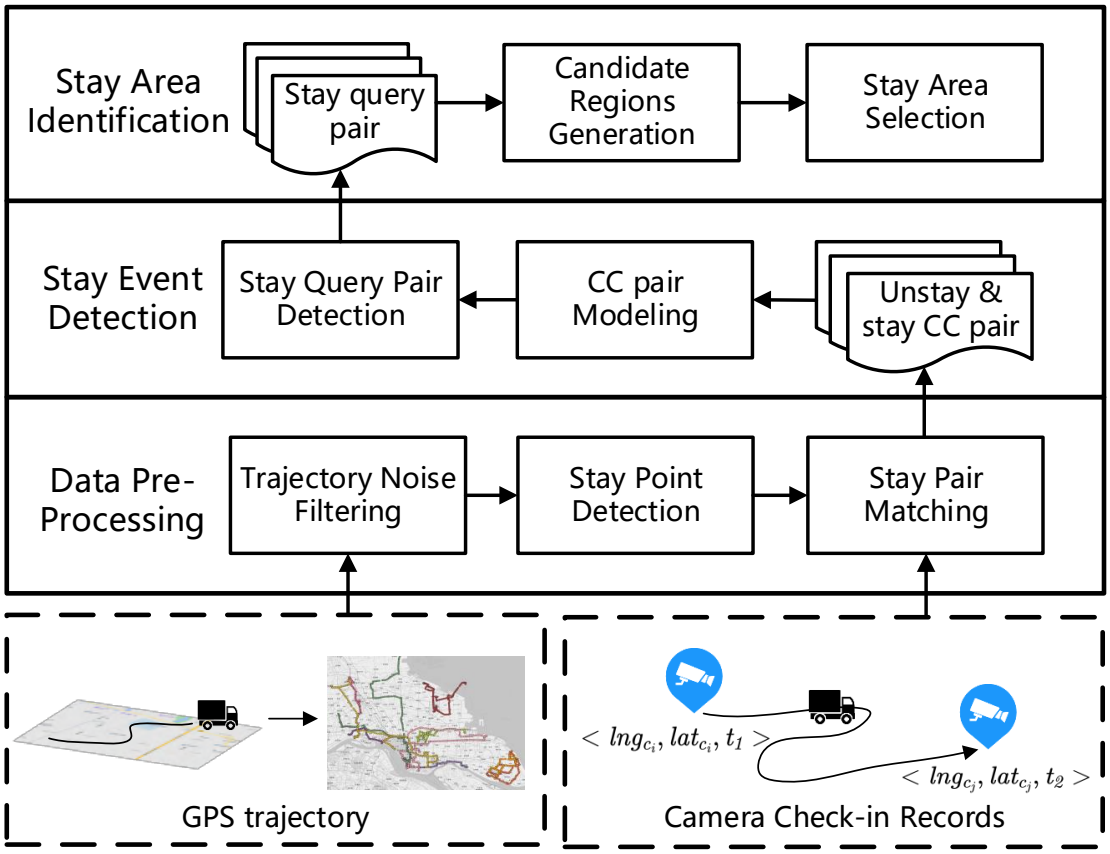


Figure 6.3: System framework.

6.3.1 Trajectory Noise Filtering

The GPS trajectories generated by a vehicle usually contain noise points. For example, as shown in Figure 6.4(a), the error of p_4 and p_6 might be several hundred meters away from its true location due to the urban canyon effect. Such noise points would affect the quality of stay point detection. A heuristic-based approach proposed in [139] is used to filter noise points in trajectories. The algorithm sequentially calculates the traveling speed for each point in a trajectory based on its precursor and itself. If the speed is larger than the threshold, the current examined point is removed from the trajectory. In this example, if v_{34} and v_{56} are larger than the speed threshold, they are removed from the trajectory. The speed threshold is set to 72km/h since the moving speed of a vehicle in urban space would rarely exceed this threshold.

6.3.2 Stay Point Detection

Based on the cleaned trajectories, we extract all stay points from them. We use stay points as ground truth for modeling the relationship between stay points and CC pairs. The stay point detection algorithm proposed in [55] is employed. The algorithm first checks if the distance between an anchor point and its successors in a trajectory is larger than a given threshold D_{max}. In the example shown in Figure 6.4(b), p_3 is the current anchor point, and p_4 to p_7 are its successors within D_{max}. It then calculates the duration between the anchor point and the last successor within D_{max} (p_3 and p_6). If the duration is larger than the given temporal threshold T_{min}, a stay point is detected (p_3 to p_6), and the anchor point moves to the next point after the current stay point (p_7). Otherwise, the anchor point moves forward by one (p_4). This process is repeated until the anchor point moves to the end of the sequence. The algorithm has the chance to generate stay points that are temporally consecutive, which makes little difference for stay area detection. Therefore, we also merge those consecutive stay points. We tried different parameter combinations and found that most delivery time of waybills can be included in stay points when we set D_{max} = 50m and T_{min} = 300s.

6.3.3 Stay Pair Matching

In order to mine the relationship between the CC pair and the stay location, we need to match the corresponding stay points for each CC pair. Referring to the process of a vehicle being recorded by the camera while driving on the road network, the vehicle first is caught by camera A, then moves into the blind spot where may stay, and finally is caught by camera B to generate a complete CC pair. A natural idea is to match the stay points in the CC pair time range according to the temporal order. We propose to use the visit time of the CC pair and the start and end times of the stay points to match the stay points sandwiched in the corresponding CC pair. After the stay pair module, stay CC pairs and non-stay CC pairs are generated for stay detection.

6.4 Stay pair Detection

In this component, We detect whether stay occurs within each CC pair so that we can more accurately model the stay area based on the detected CC pairs in the later component.

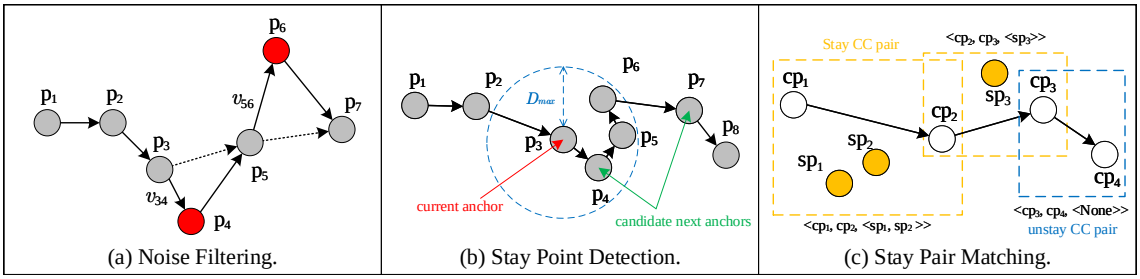


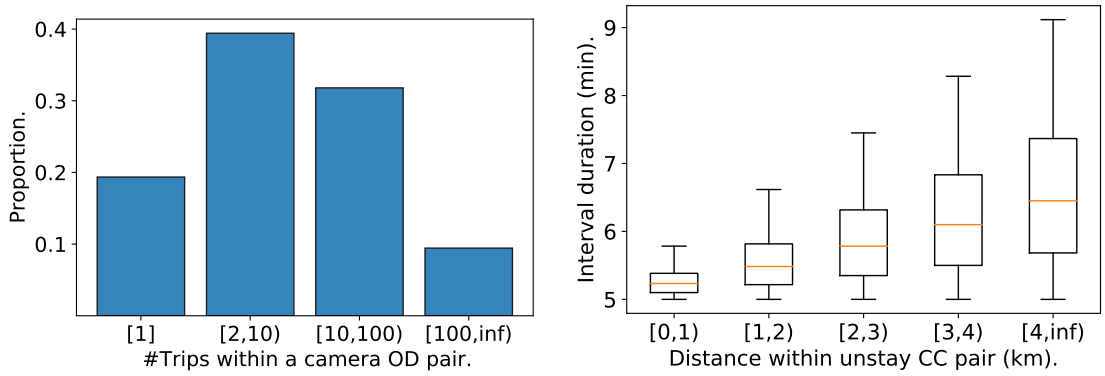
Figure 6.4: Example of Pre-Processing.

6.4.1 Motivation

The stay pair detection mainly consists of two steps: 1) CC pair modeling, which models all CC pair based on travel speed; 2) Stay query pair detection, which detects stay points. This is inspired by two insights discovered in the dataset:

Unbalanced data distribution in the camera OD pair: In general, an intuitive idea is that stay events are based on the distribution of historical travel duration for each CC pair, since vehicles spend more time cost in passing through a CC pair with a stay than a CC pair without a stay. It requires us to model each possible camera OD pair separately. However, it is difficult to model each CC pair individually due to the unbalanced data distribution corresponding to the camera OD pair referring to Figure 6.5(a). Figure 6.5(a) shows that 60% trips under camera OD pair are less than 10 and 90% trips are less than 100. It shows that it is difficult for us to capture the transportation pattern under each camera OD pair in its entirety. To make the most of data, we need to model the CC pairs under all camera OD pairs uniformly.

Positive correlation between distance and travel duration: Because the travel distance between different camera OD pairs is different, the travel time is still affected by the distance when the vehicle is driving normally. As an example, there are two CC pairs, named $ccpr_1$ and $ccpr_2$, the shortest reachable distance between two pairs is 2000 miles and 100 miles respectively, and the travel time is 7 minutes for both. This is because the vehicle spends at least 400 seconds passing the two cameras from $ccpr_1$ and the remaining 20 seconds are not enough to generate a stay point that has at least 300 seconds. In contrast, $ccpr_2$ has a higher probability to generate a stay event due to the actual travel duration being much longer than the travel duration without a stay event. Figure 6.5(b) demonstrates the fact that the travel time increases with distance in the camera OD pair, and the distance used in the figure is the Euclidean distance. Therefore, we need to eliminate the effect of distance on travel time when modeling travel time uniformly.



(a) #Trips Distribution.

(b) Distance Distribution.

Figure 6.5: Insights of Stay Pair Detection.

6.4.2 Main idea

Inspired by these two insights, we model all CC pairs uniformly and use distance to normalize the travel time within each pair by dividing the distance to reduce the effect of skewed data distribution and distance on travel time. As shown in Fig. 5(b), considering that the travel duration within CC pairs is proportional to the corresponding distance, we normalize the travel duration by dividing the distance. The equation is shown as eq. (6.1). Equation eq. (6.1) describes the statistic s_i for the CC pair $ccpr_i$, $ccpr_i$ contains two camera check-in points ccp_m and ccp_n.

$$s^* = \frac{1}{s} \tag{6.1}$$

However, we take the reciprocal of the statistic s^* instead of s due to the division by zero problem. Statistic s^* describes the stay events within the CC pairs in numerical form which is similar to s. When a stay event occurs, the travel duration becomes longer under the same travel distance setting and s^* becomes smaller, and vice versa. Another advantage is that s^* describes how fast the vehicle moves between cameras, which is easier to understand.

In practice, the real routing of the vehicle is unknown due to the problem set. For simplicity, we use the geographic distance between cameras instead of the real travel distance. Finally, speed within CC pairs is the statistic for detecting whether the stay event occurs between cameras.

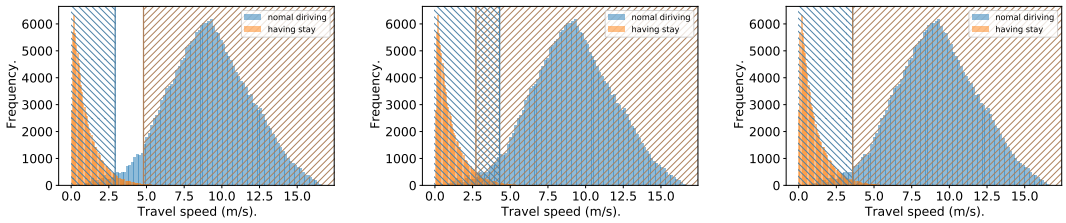
With the visual analysis in the statistic s^*, it is found that the complexity is less for the stay pair detection problem. We directly use hypothesis testing [90] to calculate the

Algorithm 3 Stay Event Detection

Input: stay CC pairs $\mathcal{D}_{stay}$; unstay CC pairs $\mathcal{D}_{unstay}$.
Output: stay query pair $\hat{\mathcal{D}}_{stay}$.
Initialize: $\hat{\mathcal{D}}_{stay} \leftarrow \varnothing$; $\alpha \leftarrow 0.05$

1: **for** $ccpr_i \leftarrow \mathcal{D}_{stay} \cup \mathcal{D}_{unstay}$ **do**
2: $v_i \leftarrow speed(ccpr_i)$;
3: **end for**
4: $V_{stay} \leftarrow \{v_i | i \in N_{stay}\}$;
5: $V_{unstay} \leftarrow \{v_i | i \in N_{unstay}\}$;
6: $ts_1 \leftarrow hypothesis\_testing(V_{stay}, \alpha)$;
7: $ts_2 \leftarrow hypothesis\_testing(V_{unstay}, \alpha)$;
8: **while** $ts_2 - ts_1 > \tau$ **do**
9: $\alpha \leftarrow \alpha - \delta$;
10: $ts_1 \leftarrow hypothesis\_testing(V_{stay}, \alpha)$;
11: $ts_2 \leftarrow hypothesis\_testing(V_{unstay}, \alpha)$;
12: **end while**
13: $v* \leftarrow (ts_2 + ts_1)/2$;
14: **for** $v_i \leftarrow V_{stay} \cup V_{unstay}$ **do**
15: **if** $v_i \leq v*$ **then**
16: $\hat{\mathcal{D}}_{stay} \leftarrow \hat{\mathcal{D}}_{stay} \cup \{ccpr_i\}$;
17: **end if**
18: **end for**
19: **return** $\hat{\mathcal{D}}_{stay}$

threshold of the statistic s^* to detect stay events. The pseudo-code of the stay pair detection is illustrated in Algorithm 3, which takes stay CC pairs D_{stay} and unstay CC pair D_{unsaty}, and returns the stay query pair $\hat{D}_{stay}$ from the D_{stay}. First, algorithm 1 calculates the speed of each CC pair from the stay CC pair and the unstay CC pair (line 1-2). The speed from the two types of CC pairs is sampled as the population V_{stay} and V_{unstay}, respectively. Then, the hypothesis testing initialization in the statistics V_{stay} and V_{unstay}, respectively (line 3-5). Next, we perform hypothesis testing on the statistics V_{stay} and V_{unstay}, respectively. For hypothesis test T_1 at v_{stay}, each of the two competing models, the null model, and the alternative model, is separately fitted to the data less than or equal to threshold $v*$. For the hypothesis test T_2 in V_{unstay}, the two models are reversed. Note that we did not choose the common significance levels α like 0.05 and 0.01 because we found confusion interval when α was tested using these two values. Confusion interval is part that falls into either the rejection or acceptance regions of both tests at the same time making it impossible to discern the test results. Figure 6.6(a) shows the confusion interval



(a) $\alpha = 0.01$ (b) $\alpha = 0.05$ (c) United Hypothesis testing

Figure 6.6: Examples of Different Significance Levels.

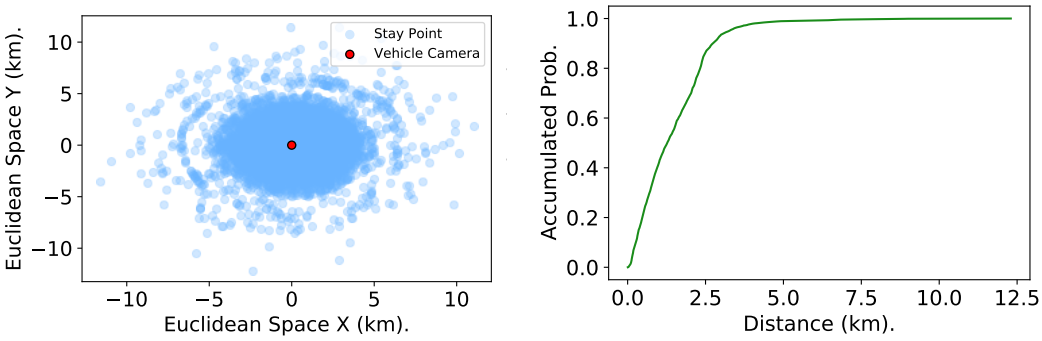
generated at $\alpha = 0.01$, where the blank part belongs to the rejection region of both tests, and thus the data in it are neither stay nor normal driving. Similarly, Figure 6.6(b) shows the confusion interval generated when $\alpha = 0.05$, where the overlapping part belongs to the rejection domain of both tests and is rejected by both tests. The data in the confusion interval in both examples are not discriminable. Therefore, we chose a compromise by having the acceptance region of both tests be the rejection region of the other. We call the method United Hypothesis testing. Figure 6.6(c) illustrates such an example (line 5). More specifically, we use the iterative method to calculate the boundary position of the united hypothesis test and return the threshold $v*$ when the difference between the reject region position of both tests is less than the threshold τ. Eventually, CC pair $\hat{D}_{stay}$ with stay events is detected and returned based on threshold $v*$ (line 11-14).

6.5 Stay Area Identification

Due to the high uncertainty in temporal and spatial aspects, in order to ensure the performance and computational efficiency of the method, we regard the stay region selection problem as the recommendation problem, inspired by the recommender system [93]. Stay Area Identification performs coarse and fine selecting in the potential stay areas of each stay query pair. Specifically, in the coarse selecting process, the algorithm determines the candidate region set from the stay query pair by modeling the reachable range. In the refined selecting stage, the algorithm selects the most likely regions from the candidate region set.

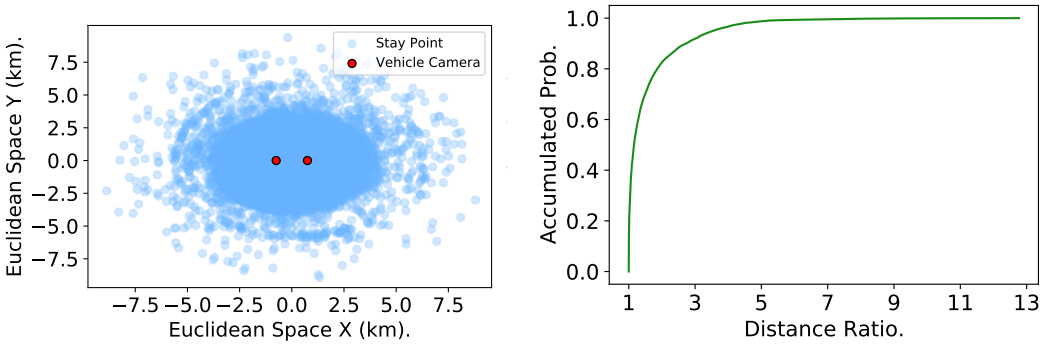
6.5.1 Candidate Regions Generation

The candidate area generation problem for CC pairs is essentially estimating the reachable range at a fixed start and end location. We denote the start camera check-in location as A,



(a) Spatial distribution of case I.

(b) CDF of case I.



(c) Spatial distribution of case II.

(d) CDF of case II.

Figure 6.7: Visualization at Stay Points within CC Pairs

and the end camera check-in location as B, there are two possible cases to calculate the candidate region set:

Case I: A and B are in the same location. In this case, the candidate area is a circle. The vehicle starts from the center of the circle and then returns to the center again. The radius of the circle is the maximum distance that each CC pair can move at the travel duration.

Case II: A and B are in a different location. The candidate area is an ellipse with A and B as the foci, and distance 2a is the distance that can be traveled by the travel duration within each CC pair. The distance 2c is the geographical distance between two cameras in the CC pair.

In both cases, the traditional solution is to set a travel speed based on experience to calculate the maximum travel distance within the travel duration from each CC pair. However, due to the complexity of vehicle mobility and the diversity of spatial-temporal context on CC pairs, the travel speed is expected to be dynamic for different cc pairs. Blindly using a non-universal empirical value can lead to a redundant candidate region set obtained by the coarse selecting stage, which in turn affects the performance of the fine selecting stage. To explore the spatial distribution of stay points in the two cases, we visualized the stay points and camera positions. In particular, for Case II, we normalize the camera positions so that the camera positions of all CC pairs overlap. Visualization results are shown in Figure 6.7(a) and 6.7(c). It is found that the stay points are overall two-dimensional Gaussian distribution around the camera locations. Therefore, we propose a KDE-based method [121] to select candidate regions. Kernel Density Estimation (KDE) find potential stay candidate region through fitting the spatial distribution of stay points in both two types of case.

KDE is a non-parametric method to estimate the probability density function of a random variable based on kernels as weights. Following the kernel density model, we define the density score for a stay point x with respect to its CC pair c_i as follows:

$$f_h(x) = \frac{1}{n}\sum_{i=1}^{n} K_h\left(x - c_i\right) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x - c_i}{h}\right) \tag{6.2}$$

where h represents the bandwidth, x stands for a stay point, c_i represents the i-th camera. Specifically, c_i represents the center of the circle in Case I. In case 2, c_i represents the midpoint of the two camera positions. And n represents the number of samples. In our problem, the samples are the location of stay point, $(x - c_i)$ denotes the geographic distance between the stay point x and the camera c_i, and $K(.)$ is the kernel function. We

Algorithm 4 Candidate Regions Generation

Input: stay query pair $\hat{D}_{stay}$; region set R
Output: candidate region set S.
Initialize: $\hat{D}^1_{stay} \leftarrow \varnothing$; $\hat{D}^2_{stay} \leftarrow \varnothing$; $S \leftarrow \varnothing$

for $ccpr_i \leftarrow \hat{D}_{stay}$ **do**
  if $dist(ccpr_i) = 0$ **then**
    $\hat{D}^1_{stay} \leftarrow \hat{D}^1_{stay} \cup \{ccpr_i\}$;
  else
    $\hat{D}^2_{stay} \leftarrow \hat{D}^2_{stay} \cup \{ccpr_i\}$;
  end if
end for
$\mathcal{F}_1 \leftarrow KDE(\hat{D}^1_{stay})$;
$\tau_d \leftarrow mean(percentile(\mathcal{F}_1, 95))$;
$\mathcal{F}_2 \leftarrow KDE(\hat{D}^2_{stay})$;
$\tau_{dr} \leftarrow mean(percentile(\mathcal{F}_2, 95))$;
for $ccpr_i \leftarrow \hat{D}_{stay}$ **do**
  if $dist(ccpr_i) = 0$ **then**
    $poly_i \leftarrow circle(\tau_d)$;
  else
    $d \leftarrow dist(ccpr_i)$;
    $poly_i \leftarrow ellipse(d * \tau_{dr}, d)$;
  end if
  $S[ccpr_i] \leftarrow spatial\_overlap\_query(poly_i, R)$;
end for
return S;

use the Gaussian kernel function in the paper, which is expressed as follows:

$$K(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2} \tag{6.3}$$

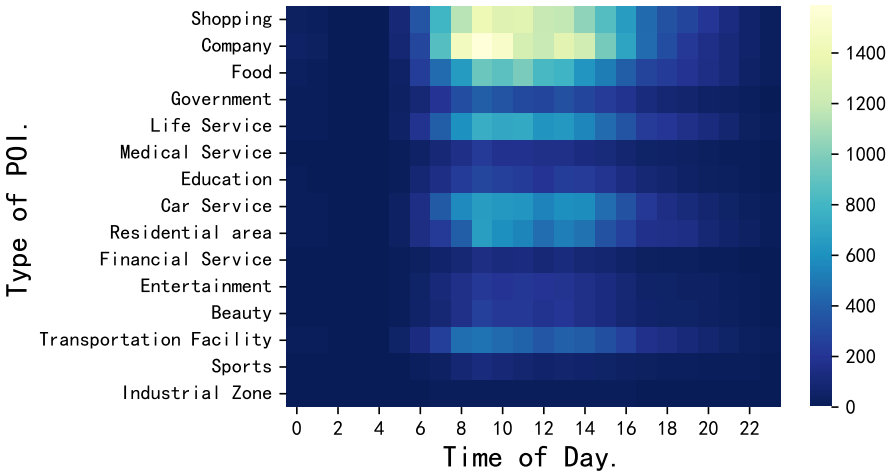
Note that, Figure 6.7(a) and 6.7(c) show some outliers, and they are mostly noise generated by camera false alert and missed detection. Furthermore, Figure 6.7(b) and 6.7(d), demonstrate the fact that more than 90% x in both type cases are concentrated within a certain range of the center c_i. Because distance cannot be directly compared in the normalized spatial distribution, the distance ratio in Figure 6.7(d) is used instead of distance, which refers to the ratio of a to c in the ellipse. Based on the above two facts, we used the 95th percentile of the obtained two-dimensional Gaussian distribution as the maximum boundary of the potential stay region after fitting the spatial distribution with KDE.

Specifically, for Case I, we use the 95th percentile as the boundary of the candidate region. The polygon corresponding to the boundary is used as the radius of the candidate circle region according to the average distance from each vertex to the center for simplicity (line 6-7). Similarly, for Case II, Algorithm 2 use the 95th percentile as the boundary of the candidate ellipse region, and return polygon corresponding to the boundary (line 8-9). The average distances which from each vertex to the two focis is 2a and the distance between the two normalized cameras is 2c in the ellipse. To ensure the dynamics of the boundary, we use the distance rate a/c to calculate the adaptive boundary of the candidate region instead of the fixed spatial threshold. In this setup, the 2a in candidate ellipse region is able to vary with the distance between the two cameras in the CC pair. Finally, the algorithm 2 preform spatial overlap query between the candidate circle and ellipse regions and the predefined regions, and return the overlapping regions as candidate sets (line 10-17).

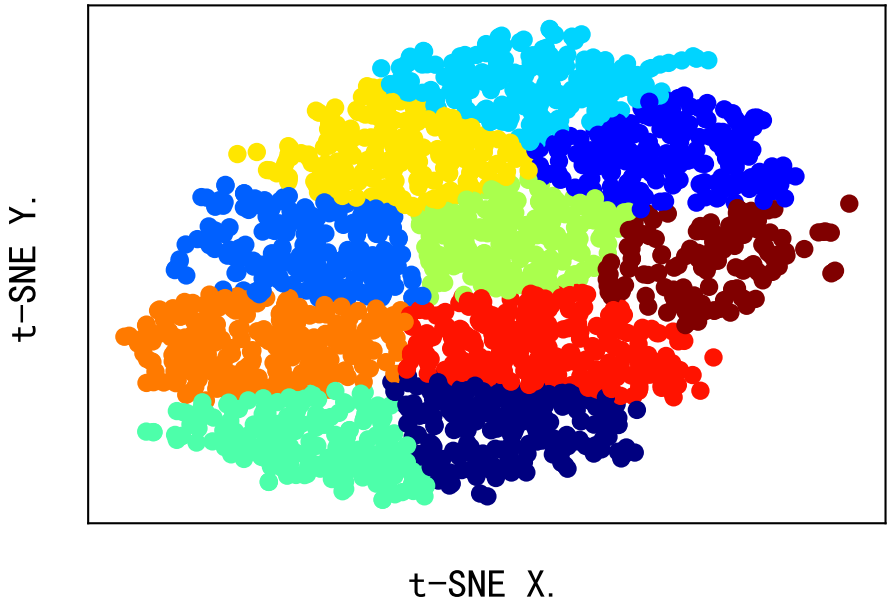
6.5.2 Stay Area Selection

In the previous process, the candidate region generation module estimates adaptive range thresholds to generate the candidate region set for each stay query pair. Then, our task is to select n stay regions from the candidate region set as the output in the stay area selection module. However, because of the complex the stay pattern, three challenges arise. The first challenge is how to model the relationship between spatial regions and stay points. Second, representing the spatio-temporal context of the stay query pair is the second challenge considering the rich spatio-temporal relationships in CC pairs. Finally, how to effectively fuse these information to model the stay probability in each region from candidate region set is another challenge.

Main Idea. To overcome these problems, we obtained three insights as follows by analyzing the historical stay points. The heat map of vehicle stay at various POIs and periods is presented in Figure 6.8(a). It indicates that stay pattern in same type of vehicles tends to be similar, similar in the sense that vehicles will stay at similar locations during similar time periods. For example, hazardous chemical vehicles need to transport chemicals, they often stay near companies, stores, medical and other facilities centrally. And taxis are more inclined to visit hot spots frequently. We propose to capture the spatio-temporal information of the stay query pairs through time periods, passing camera locations and weather, etc. Furthermore, Vehicles' stay behavior is purposeful and is relevant to region semantically. Relying on the fact that semantic information of the region is described by



(a) Stay Point Heatmap.



(b) Historical Stay Preferences.

Figure 6.8: Insight of Stay Area Selection.

the contained POI, we naturally use POI to represent regions to further model the relationship between stay points and regions. Figure 6.8(b) shows the average POI vector for each vehicle's historical stay region, which is revealed that vehicles' stay preferences are personalized. So the historical stay points affect where the vehicle will stay in the future, because historical data contains vehicle stay preferences. Overall, we transform the selecting problem in the set into the ranking problem with the stay probability in each

region, and the ground truth of the stay point is used as the supervisory signal.

Model Overview. In practice, we designed a stay region selection model, called StayNet. Figure 6.9 depicts the structure of StayNet, which consists of four components to select the top-k regions for each stay query pair:

- **Candidate Region Representation**, which models all regions in the candidate region set uniformly using TF-IDF value of POIs, and output variable representation of each region;
- **Vehicle Representation**, which explores each vehicle's preference representation through mining historical stay behaviors.
- **Spatio-temporal Context Encoding**, which extracts and embeds spatio-temporal features of each stay query pair, and combines the embeddings and vehicle historical representation to generate a hidden representation which indicates dynamics.
- **Knowledge Fusion**, which fuse and further enhances each region representation with the spatio-temporal context representation in an adaptive manner so that an ideal prediction result can be achieved.

Candidate Grid Representation. The function of the location is key to whether the vehicle stays. To weaken the influence of invariant features in the region and enhance the features that can be distinguished from each other in the candidate region, we expect that candidate region set is observed uniformly. Considering the uniform modeling of variable-length collections, we use Transformer.

Transformer [107] has since been widely used in natural language processing, computer vision, and sequence modeling, which follows the architecture of the encoder-decoder structure. Both encoder and decoder are composed of stack identical Transformer Block (TB) layers. Each transformer block has two sub-layers, including a multi-head self-attention mechanism (MultiHead(.)), and a simple position-wise fully connected feed-forward network (FFN(.)). The residual connection [33] is used around each of the two sub-layers, followed by a layer normalization [5].

The MultiHead(.) is an ensemble of h single-head attention, which concatenates all heads in parallel and projects them once again. A single-head attention mechanism is described in the form of formula eq. (6.4), which calculates the weighted sum of the values, where the weights are obtained by interacting with the corresponding query and key using softmax function:

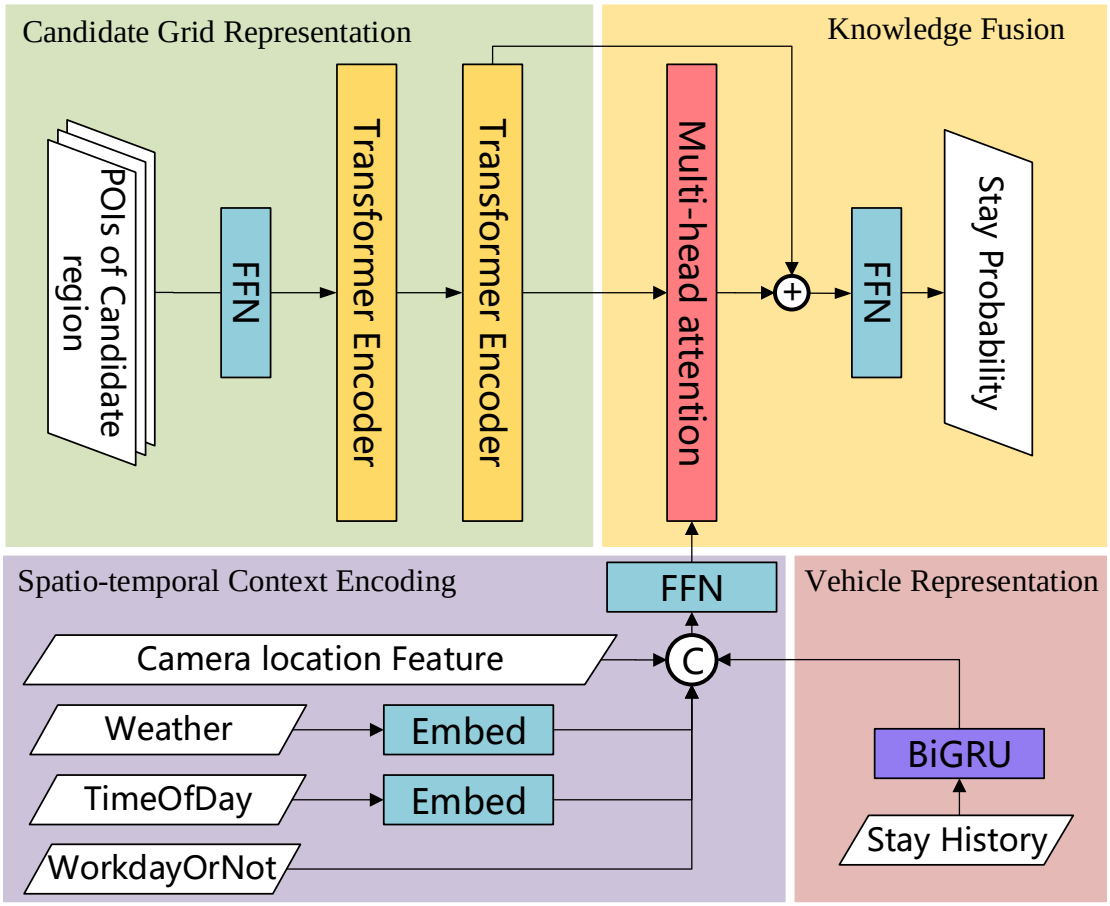


Figure 6.9: The Structure of StayNet.

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V} \tag{6.4}$$

where $\frac{1}{\sqrt{d_k}}$ is the scaling factor, $Q, K \in \mathbb{R}^{L \times d_{model}}$ is the query matrix and the key matrix, $V \in \mathbb{R}^{L \times d_{model}}$ is the key matrix which from multiple queries, keys and values packed together, respectively. L is the collection length of queries, keys and values. Further, the MultiHead(.) is as follows:

$$\begin{aligned} \text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) &= [\text{head}_1; \ldots; \text{head}_h]\mathbf{W}^O \\ \text{where head}_i &= \text{Attention}(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V) \end{aligned} \tag{6.5}$$

where $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$,$W_i^K \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$, $W^O \in \mathbb{R}^{hd_v \times d_{model}}$ are

the learnable parameter matrices.

The FFN(.) layer contains a fully connected feed-forward network, which is applied to each position separately and identically. It consists of two linear transformations with a ReLU activation with learnable parameters W_1, W_2, b_1, and b_2:

$$\mathrm{FFN}(\mathbf{X}) = \max(\mathbf{0}, \mathbf{X}\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2, \tag{6.6}$$

Consider a set $X \in \mathbb{R}^{L\times d}$ with L elements and each element with d-dimensional features, the process of passing through Transformer Encoder with multiple Transformer Block layers is as follows:

$$\begin{aligned}
&\mathbf{X}^0 = \mathrm{X} \\
&\tilde{\mathbf{X}}^{j+1} = \mathrm{LayerNorm}(\mathrm{MultiHead}(\mathbf{X}^j, \mathbf{X}^j, \mathbf{X}^j) + \mathrm{X}^j) \\
&\mathbf{X}^{j+1} = \mathrm{LayerNorm}(\mathrm{FFN}(\tilde{\mathbf{X}}^{j+1}) + \tilde{\mathbf{X}}^{j+1})
\end{aligned} \tag{6.7}$$

where m is the number of transformer block layers. The S^m is the final output of the transformer block. Note that elements in the set are disordered so that the positional encoding layer is removed. In this component, we use the vector composed of TF-IDF values for 16 types of POI to indicate the functional features of each region, and combine all regions in a candidate region set as the input X_r, $X_r \in \mathbb{R}^{L\times 16}$. It is expected that the differential representation between regions is captured through double transformer block layers as encoder after processing with feed-forward network:

$$\begin{aligned}
\tilde{X}_r &= \mathrm{TransEnc}(\mathrm{FFN}(X_r)) \\
E_r &= \mathrm{TransEnc}(\tilde{X}_r)
\end{aligned} \tag{6.8}$$

Vehicle Representation.

To parameterize vehicle preferences, we propose to use the vehicle's historical stay points to learn the corresponding embedding. Considering the BiGRU is computationally cheaper, we use BiGRU to model the vehicle's historical stay points instead of Transformer. BiGRU is used in sequence modeling to extract context information from temporal vector sequences. BiGRU is composed of a forward GRU unit and a backward GRU unit. The hidden layer output of BiGRU at time t is spliced through the hidden layer output of the forward GRU unit and backward GRU unit, as shown in formula eq. (6.9):

$$
\begin{aligned}
\overrightarrow{h_t} &= \mathrm{GRU}(x_t, \overrightarrow{h_{t-1}}) \\
\overleftarrow{h_t} &= \mathrm{GRU}(x_t, \overleftarrow{h_{t-1}}) \\
h_t &= [\overrightarrow{h_t}, \overleftarrow{h_t}]
\end{aligned}
\tag{6.9}
$$

where GRU consists of an updated gate and a reset gate. Gate structure selectively remembers and update cell states and current inputs to solve the problem of RNN gradient disappearance or explosion. For time t, the GRU cell state calculation formula is as follows:

$$
\begin{aligned}
r_t &= \sigma\left(w_r\left[h_{t-1}, x_t\right] + b_r\right) \\
z_t &= \sigma\left(w_z\left[h_{t-1}, x_t\right] + b_z\right) \\
\tilde{h}_t &= \tanh\left(w_h\left[r_t h_{t-1}, x_t\right] + b_h\right) \\
h_t &= \left(1 - z_t\right) h_{t-1} + z_t \tilde{h}_t
\end{aligned}
\tag{6.10}
$$

where σ and $\tanh$ are the Sigmoid function and Tanh function, respectively. w_r,w_z,w_h,b_r,b_z and b_h are learnable parameters. x_t is the input vector of time t, h_t is the hidden state, and is also the output vector, containing all the valid information of the previous t time step. r_t is the reset gate, which adds the current input x_t on hidden state of the previous unit h_{t-1} in a targeted manner and return current state $\tilde{h}_t$. z_t is the update gate, which controls the proportion of hidden state of the previous unit h_{t-1} updates to the current state $\tilde{h}_t$.

The outputs of the BiGRU are sent to a vehicle-level sum pooling to obtain the preferences representation of each vehicle, denoted as e_h:

$$
e_h = \mathrm{BiGRU}(X_h) \tag{6.11}
$$

where $X_h \in \mathbb{R}^{M*16}$, M is the length of historical stay point records. Similar to X_r, the vector composed of TF-IDF values for 16 types of POI is used to characterize the stay region. In practice, we use stay points generated from GPS trajectories in the past for a fixed period of time for vehicle representation, and strictly control the respective collection time of stay points used for vehicle representation and stay points used as ground truth to prevent information leakage. For example, we use stay point data generated by

GPS trajectory from Jan-Mar in the vehicle representation component and stay point data from Apr-Sep as ground truth for the overall stay area identification framework. Also, the mean vector of preference vectors is used for unobserved vehicles. When the service is deployed, the vehicle representation module inputs the sequence of stay areas predicted by the model.

Stay Query Pair Representation.

Because the stay event is also related to its own spatio-temporal context, we extract three types of features in the stay query pair to depict it including spatial features, temporal features, and extra features. The spatial features x_s are composed of the latitude, longitude, and position embedding of the two cameras visited consecutively and the corresponding grid coordinates. Camera position embedding is pre-trained on camera check-in trajectory by DeepWalk [85]. There are two features that belong to the temporal type. One is the index of the discrete-time bin x_b corresponding to the start and end of the stay query pair, the other is the binary value x_d indicating whether the time slot is on workdays or weekends. Finally, extra feature x_w is the weather type during the occurrence in each stay query pair. During processing, the x_d and x_w are fed firstly into an embedding layer to obtain the dense representation. Then, we concatenate them together and send them to a feed-forward network to obtain the stay query pair representation e_{st}. Details are shown in formula 12:

$$\begin{aligned} e_w &= \mathrm{Embed}(x_w) \\ e_b &= \mathrm{Embed}(x_b) \\ e_{st} &= \mathrm{FFN}([x_s; e_w; e_b; x_d; e_h]) \end{aligned} \tag{6.12}$$

Knowledge Fusion.

The knowledge fusion component takes candidate region representation E_r and stay query pair representation e_{st} as the input, enhances E_r using cross attention based on e_{st}, and returns stay probability of each candidate region. Moreover, residual connections are used o ensure the stability of the training. Knowledge fusion is formally defined as follows:

$$\begin{aligned} E_{st} &= \mathrm{MultiHead}(E_r, e_{st}, e_{st}) + E_r \\ \hat{y} &= \mathrm{FFN}(E_{st}) \end{aligned} \tag{6.13}$$

In the training phase, we use BCEloss to optimize the optimized StayNet:

$$loss = \text{BCELoss}(\hat{y}, y) \tag{6.14}$$

where $\hat{y}$ is predicted result, y is corresponding ground truth. It should be noted that the proportion of negative samples far exceeds that of positive samples due to the imbalance of positive and negative samples in the candidate region set. In practice, we use the negative to positive sample ratio to weight the loss of positive samples in BCEloss.

6.6 EXPERIMENTS

6.6.1 Experimental Settings

We use a real world dataset which contains camera check-in records and GPS trajectory data for 2,182 vehicles that were collected from Nantong, China from March 1, 2021 to March 18, 2022.

- **Camera Check-in Record.** A total of 14,058 smart surveillance cameras are deployed at intersections in Nantong. When a vehicle passes them, a log record containing timestamp, longitude, latitude, license plate number and the corresponding camera ID is generated. The dataset contains 1.22 million records generated by 2182 vehicles in 175 days.

- **GPS Trajectory.** They are raw GPS logs generated by the vehicle's navigation device, each record contains license plate number, longitude, latitude and timestamp. In the paper, 230 million data for 265 days from 2182 vehicles were used. These data are pre-processed, where the first 90 days of stay points from the trajectories are used for vehicle representation, and the stay points detected from the rest of data are used as ground truth.

- **POI.** We collected 15 categories of POI data from Nantong for characterizing the area using services from Baidu Map.

- **Weather.** Weather data is collected from open API services to model the context of stay behavior.

In practice, after the GPS trajectory and camera check-in records are processed by the data pre-processing module, the stay points detected from the GPS trajectory and the

camera check-in records are combined in chronological order to obtain two types of CC pairs for stay area inference. The POI data is projected onto a preset grid and the tf-idf vector of region from grid is calculated. Last, a total of 329.3 thousand CC pairs are generated, of which 60% are used for training, 20% for validation, and finally 20% for testing.

Evaluation Metrics. We choose hit ratio, which measures how many real stay regions can recall successfully from its k query results. It is formulated as

$$hit@k = \frac{\text{\#hits of topK}}{\text{\#stay region}} \tag{6.15}$$

To uniformly evaluate the performance of the method, we construct the sum of different hit@k called SUM. In addition, A weighted ACC is used for evaluating the performance of stay event detection. we use ratio of unstay and stay to balance because it the number of CC pairs in the two categories is significantly disproportionate:

$$ACC = \frac{w * \text{\#right stay CC pair} + \text{\#right unstay CC pair}}{\text{\#stay CC pair} + \text{\#unstay CC pair}} \tag{6.16}$$

Hyperparameters. The stay point detection algorithm uses the two thresholds are 50 meters and 300 seconds. The region size is set to 1km $\times$ 1km. The 2-layer transfomer encoder is used in StayNet. 4 and 8 headers are used in the transformer and attention mechanisms, respectively. The hidden size of FC layer in FFN, BiGRU, transformer and self-attention are all set to 128. Weather type and time of day index is embedded to $\mathbb{R}^2$. Beside, we used Sigmoid function as the activation function for the fully connected layer in the knowledge fusion component. In training phase, we adopt Adam optimizer with β_1= 0.9 and β_2= 0.999 and use an initial learning rate 7e-3. The training epoch is set to 100 rounds.

Implementations. Our algorithm and other baselines are completely implemented in Python. The algorithm with neural networks is implemented using the PyTorch. Experiments are conducted on a workstation with an AMD Ryzen9 5900X @ 4.2GHz, 32GB memory, NVIDIA RTX3090 and Windows 10 OS.

6.6.2 Comparison of Frameworks.

Baseline. To the best of our knowledge, there is no solution that can be used to solve the problem well. Therefore, we designed three baselines based on the enforcement experi-

ence of city managers.

- **Random Infer (RInf).** For each CC pair, we do not detect whether the stay event occurs or not, and randomly select k regions in the whole city as the result.
- **Spatial Heuristic Infer (SHInf).** Without detecting stay events, the framework determines the candidate range based on the empirical parameter and performs spatial heuristic selection in candidate region set within the range. Spatial heuristic selection sorts the candidate region set in descending order by stay frequency, and returns the top k regions
- **Velocity Heuristic Infer (VHInf).** The empirical velocity is used as the threshold for detecting stay events. We use the empirical velocity and the interval of the CC pair to calculate the reachable range and perform spatial heuristic selection on the candidate region set within the reachable range.

Variants. We also compare SAInf with its three variants:

- **SAInf-nD.** This variant uses the empirical velocity as a threshold to detect stay events. Settings of stay area inference are consistent with SAInf
- **SAInf-nS.** This variant uses fixed empirical spatial threshold to determin the range of candidate region set instead of adaptive stay region generation.
- **SAInf-nC.** Instead of using StayNet as the selection model, this variant uses spatial heuristic selection.

Evaluation. Table 6.1 shows the performance of the proposed framework as compared to all other competing framework. Our proposed SAInf significantly outperforms all competing baselines by achieving the highest all metric. Overall 58.03% performance improvement in SAInf compared to the current best method. And, SAInf in hit@1, hit@3hit@3 improved by 95.56%, 57.06% and 43.41% respectively. The ACC is only improved by 0.8%, which is related to the fact that the empirical speed parameters are taken close to the adaptive threshold. Also, because ACC is a comprehensive evaluation metric that distinguishes between stay and unstay CC pairs, although it is a small improvement, it can largely affect the final inference results. RInf and SHInf do not detect stay events, but perform stay region selection for all input CC pairs. This rough processing results in part stay CC pairs being lost, which seriously affects the subsequent stay

Table 6.1: Stay Area Infer Evaluation.

Methods	**hit@1**	**hit@3**	**hit@5**	**Sum**	**ACC**
RInf	0.0001	0.0004	0.0007	0.0012	0.5000
SHInf	0.1966	0.4023	0.5055	1.1044	0.5000
VHInf	0.2141*	0.4155*	0.5218*	1.1514*	0.9704*
SAInf-nD	0.4170	0.6430	0.7263	1.7863	0.9704
SAInf-nC	0.4084	0.6224	0.7140	1.7448	0.9784
SAInf-nS	0.2785	0.4839	0.5806	1.3430	0.9784
SAInf(Ours)	**0.4373**	**0.6659**	**0.7477**	**1.8509**	**0.9784**

region inference. And even worse, the large and invalid results generated consume computational and storage resources. VHInf performs stay event detection on CC pairs by empirical speed to avoid losting CC pair with stay point. In contrast, the candidate region set for VHInf depends on the empirical velocity and the duration of the CC pair. It suggests that candidate region set that vary dynamically with the CC pair are more helpful in the stay region selection. VHInf obtains the optimal performance in addition to our framework.

For the three variants of SAInf, the performance improvement of SAInf-nS is the most significant, which demonstrates the effectiveness of StayNet. Moreover, it can be found that the performance of both SAInf-nC and SAInf-nV using empirical parameters is lower than our framework, which indicates the superiority of our proposed stay event detection algorithm and candidate region generation algorithm. Specifically, in the candidate region generation, the recall of the algorithm and the number of candidate region are trade-off. In general, to recall all possible resident regions would increase the number of candidate regions, resulting in a poor selection phase. When the edge length of the region is 1000m, our candidate region generation algorithm can guarantee the average number of candidate regions to be 32 under the condition that 91% of the results are recalled, but the algorithm based on empirical parameters makes the average number of candidate regions rise to 101 under the condition that 94% of the results are recalled. The experimental results prove that we have reached a better balance. Similarly, the adaptive stay event detection algorithm effectively reduces the effect of confuse intervals leading to better experimental performance than empirical parameters.

Parameter Sensitivity. We conducted experiments in a square region with side lengths of 500m, 1000m and 1500m respectively, and the results of the experiments are presented in Figure 6.10. Figure 6.10 shows the best baseline and our method and its variants. Overall, all variants showed improvement compared to the baseline method, and the most

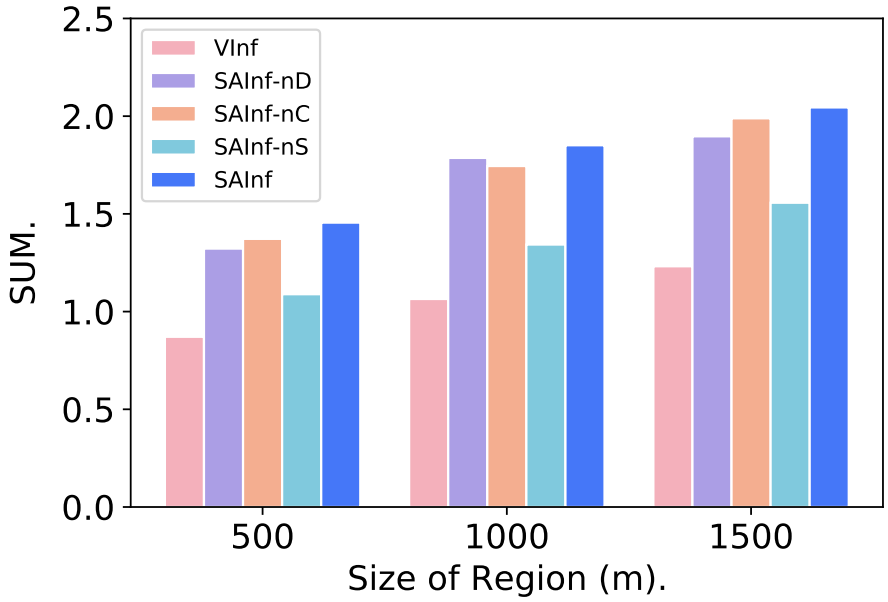


Figure 6.10: Experimental Results of Different Framework.

significant improvement was seen at the 1500m setting. The performance improvement becomes larger as the side length of the region becomes larger. This is similar to our intuition that the larger the granularity of the region the better the performance of the model, but the more difficult it is to apply in practice. To balance the performance and management, we choose a region with a side length of 1000m as the minimum unit when it is deployed.

6.6.3 Comparison of Selection Model.

Baseline. The baselines for selection model falls into three types overall, which are heuristics, instance modeling, and joint modeling. The heuristic method contains both spatial heuristic selection (SHS) and spatio-temporal heuristic selection (STHS). The spatial heuristic selection returns the k regions with the highest frequency in the candidate region set. The spatio-temporal heuristic selection takes into account the temporal dimension which is start time of the CC pair. The instance modeling approach models each region in the candidate region set individually as a binary classification problem, and returns the k regions with the highest stay probability. Such methods include logistic regression (LR), eXtreme Gradient Boosting (XGBOOST) and multilayer perceptron (MLP). The joint modeling combines all regions from the candidate region set together

Table 6.2: Area Selection Evaluation.

Methods	**hit@1**	**hit@3**	**hit@5**	**Sum**
Random	0.1063	0.2520	0.3426	0.7009
SHS	0.3303	0.5740	0.6887	1.5930
STHS	0.3394	0.5614	0.6596	1.5604
LR	0.2977	0.5999	0.7526	1.6502
XGBOOST	0.4737*	0.7191	0.8314	2.0242
MLP	0.4182	0.7107	0.8232	1.9521
RNN	0.3285	0.6124	0.7536	1.6945
Transformer	0.4663	0.7434*	0.8487*	2.0584*
StayNet-nGlo	0.4777	0.7559	0.8598	2.0935
StayNet-nFus	0.5149	0.7839	0.8856	2.1843
StayNet-nVeh	0.5135	0.7858	0.8882	2.1874
StayNet (Ours)	**0.5187**	**0.7899**	**0.8869**	**2.1954**

into the model, and return the stay probabilities in each region to increase the information interaction among regions. RNN and Transformer both belong to the this type.

Variants. Three variants are listed as follows:

- **SatyNet-nFus.** Attention mechanism is replaced by addition in SatyNet-nFus.
- **SatyNet-nGlo.** We remove the spatio-temporal context encoding and use only the output of the vehicle representation for knowledge fusion.
- **SatyNet-nHis.** This variant removes the vehicle representation component.

Evaluation. The results of area selection are presented in Table 6.2. It can be seen that compared to the current best algorithm, proposed method has an overall improvement of 6.7% and in hit@1,hit@3 and hit@5 improved by 11.2%, 6.3% and 4.5%, respectively. Specifically, with the exception of StayNet and its variants, XGBOOST achieved the highest hit@1, demonstrating the superiority of instance modeling. While transformer achieved the highest hit@3,hit@5 and sum result. This may be related to the fact that the joint modeling approach is able to better capture the functional differences among regions within the candidate region set, and such differences describe more essentially the interaction of the functionality of regions on the stay behavior. From the different types of methods, the all heuristics show a significant improvement compared to random selection, which indicates that the functionality and spatio-temporal context of region has a strong correlation with the stay behavior. It is worth noting that although STHS has an

improvement in hit@1 compared to SHS, it then decreases in overall performance. Only STHS takes into account the effect of temporal dimension, but due to the limitation of observation data, STHS cannot accurately model the effect of regions on stay behavior. The instance modeling approach can make more efficient use of historical observation data, and achieves better results than heuristics by modeling the nonlinear correlation between regions and the stay behavior pattern. It is important that in the instance modeling approach, each region contains a POI tf-idf vector of the region and spatio-temporal context information. However, due to the fact that the instance modeling uses individual view of each region in the candidate region set, it cannot accurately quantify the differences among regions and leads to a lower differentiation in stay probabilities. MLP and XGBOOS have similar results in hit@3 and hit@5, and XGBOOST achieves the best results for hit@1 except for our method. It indicates that XGBOOST can effectively attenuate the effect of useless features in the region on the results, because of its powerful feature filtering ability. In fact, this is similar in implication to the purpose of joint modeling, which is to better differentiate features among regions. In contrast, the joint modeling approach can uniformly observe all instances in the candidate region set and effectively improve the accuracy of ranking. Among them, transformer achieves competitive results in most metrics. RNN degrades the performance because the sequential modeling approach cannot process the instances in the stay region set in parallel. Comparing with variants of our proposed method, we found that the biggest improvement was achieved by encoding spatio-temporal contextual information separately and then performing knowledge fusion. From StayNet-nFus to StayNet, the knowledge fusion component of StayNet using the attention mechanism is able to better fuse spatial-temporal context encoding and region representation. Finally, the results of StayNet-nVeh show that the introduction of vehicle representation enhances the ranking ability of the model to some extent, further improving the overall performance. It is worth noting that although the vehicle indicates limited improvements in performance, it will accelerate model convergence.

Parameter Sensitivity. In addition, we also evaluated the performance of different selection models under the three region settings. The best baseline, our model and its variants are compared in Figure 6.11. Contrary to the results of the framework, our method and its variants achieve the maximum boost on a region with a 500m side length. This is related to the number of candidate regions and the distribution of POIs; when the number of candidate regions becomes larger, the task becomes more difficult and the model performance decreases. Since POIs are spatially clustered, when the size of the region

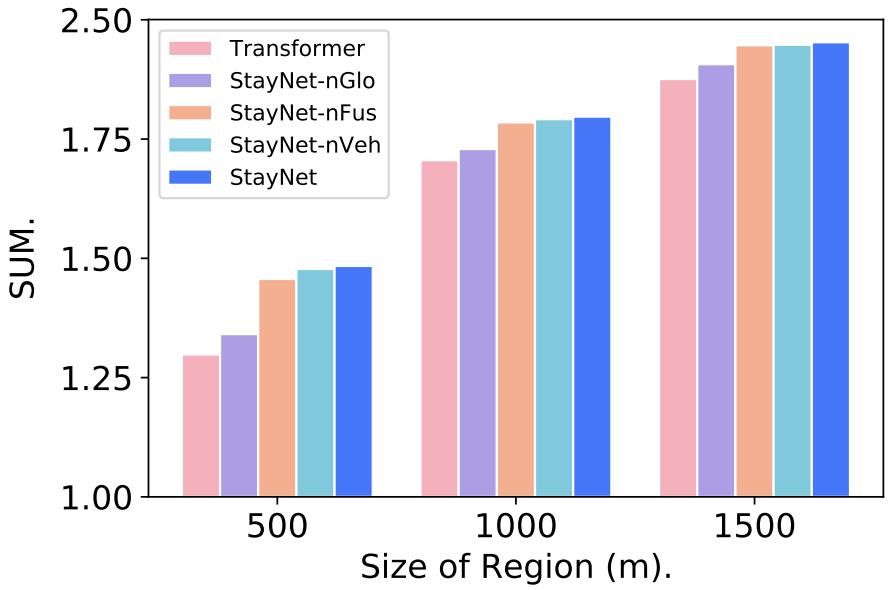


Figure 6.11: Experimental Results of Different Selection Model.

becomes small, the POI loses its ability to characterize the regions lacking significant POI. At the same time, as the regions become larger, the number of candidate regions becomes smaller, the difficulty of the task decreases, and the performance improvement of our model diminishes.

6.6.4 Case Study.

We further give a case study to test the effectiveness of SAInf in a real world setting, shown in Figure 6.12. We use SAInf to infer the potential stationing area of the vehicle on the camera check-in logs collected in real time and compare the results with the stay point of the corresponding vehicle equipped with a GPS device. It was found that 3 of the top 5 regions output by SAInf had vehicles actually staying. Based on the results of the SAInf, government managers can investigate these regions, for example they can discover illegal construction waste dumping sites.

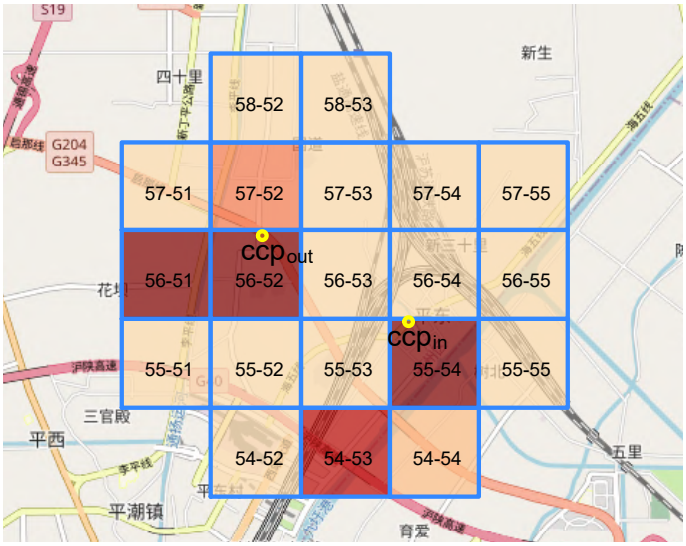


Figure 6.12: Case Study in Real World Environment.

6.7 RELATED WORK

6.7.1 Stay Point Detection

The stay of mobile objects is often accompanied by rich semantics. Mining stay points can help us understand the mobility of objects and the functionality of location. In the past decades, with the popularity of GPS devices, a large number of trajectory mining research efforts have emerged. These works usually use stay points for mining. It is divided into three main areas of work. 1) Location Discovery. Wang [114] proposed a clustering algorithm based on stay points and grid density to detect hotspot areas in cities from GPS trajectory data. Zheng [141] used the user's sequence of stay points to learn the correlation of location. 2) Event Discovery. Ruan [98] propose a framework, where the cluster of stay points in the delivery order's trajectory is annotated to infer the delivery location and delivery time. Zhu [142] correlated the cluster of stay points with loading and unloading events to identify illegal chemical facilities by ranking the unregistered locations. Yu [128] proposed a congestion location detection method based on stay point clustering, which uses the directional residuals and velocities to mine the road congestion locations. 3) Mobility Understanding. Ji [42] modeled the user's location history and analyzed their daily behavior based on a sequence of historical stay points. Liu et al.[68] designed a recommender system that uses common activities to annotate a user's stay points to learn the transfer probabilities between activities and the generation probabilities between ac-

tivities and stay points. However, these efforts are designed based on GPS trajectory data. Limited by the privacy and coverage of GPS device, we cannot capture the comprehensive behavioral patterns of all moving objects in the city. Unlike them, a new kind of data is used by us, vehicle camera recordings. With three advantages of wide spatial distribution, full vehicle coverage and privacy-friendly characteristics, vehicle camera records can provide a more comprehensive perception of moving objects. However, limited by resources, cameras are only deployed at key intersections, which leads to significant uncertainties, including stay event uncertainty, stay location uncertainty and identification uncertainty. To address these problems, we propose a two-stage framework to infer the vehicle's stay area based on camera records. It provides the underlying support for subsequent semantic mining using camera records.

6.7.2 Surveillance Camera Records Mining

Surveillance camera records mining refers to the use of camera-captured vehicle records to discover various knowledge. In order to monitor the traffic status of the whole urban, Yu [129] proposed a framework to infer citywide traffic flow based on surveillance camera recordings. Qin [89]. developed a robust and interpretable traffic state attribution framework based on surveillance camera recordings for complementary road states. In order to achieve a more fine-grained location-based service, Chen [14] integrated the spatio-temporal characteristics of vehicles in urban traffic monitoring systems and used OD pairs from camera recordings to detect potential community for each vehicle. However these works tend to address the problem of unbalanced distribution of camera recording data and missing data due to the instability of the devices. In our work, we focus on solving the problem of uncertainty in camera recording mining. Although Chen attempt to solve the problem, our problem setting is more difficult because the real deployment granularity of cameras is actually more sparse and they do not essentially solve the problem of stay detection.

6.8 Conclusion

In this paper, we explore for the first time the stay point detection problem under camera records data setting and design a framework SAInf to solve it. SAInf models the effect of specific spatio-temporal context on stay behavior by learning the relationship between CC pairs and stay points. In practice, SAInf detect stay point through a two-stage design, first

discovering the CC pair containing stay points through stay event detection module, then estimating the exact locations of the stay points with the stay area identification module. The data pre-processing module provides support for offline learning. Experiments show SAInf outperforms baselines by 58% on real-world dataset. Finally, an intelligent stay area inference system based on SAInf is deployed and used internally. In the future, we will further explore the design of other modules in the framework to achieve end-to-end stay point detection under the camera records data setting. One future direction is that obtaining the more robust potential stay candidate set by predicting the stay interval time within CC pairs to improve the system performance.

Bibliography

[1] Administrative Regulations of The People's Republic of China Governing the Registration of Legal Corporations.

[2] Company addresses. https://companies-register.companiesoffice.govt.nz/help-centre/starting-a-company/company-address/.

[3] OpenStreetMap. http://www.openstreetmap.org/.

[4] Taxi, Uber, and Lyft Usage in New York City. http://toddwschneider.com/posts/taxi-uber-lyft-usage-new-york-city/.

[5] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[6] S. Bae, A. Ravi, K. Sangaralingam, N. Verma, A. Datta, and V. Chugh. Suspicious location detection using trajectory analysis & location backfilling–a scalable approach. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 2007–2010. IEEE, 2019.

[7] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv:1409.0473*, 2014.

[8] A. Ballatore, M. Bertolotto, and D. C. Wilson. Geographic knowledge extraction and semantic similarity in openstreetmap. *Knowledge and Information Systems*, 37(1):61–81, 2013.

[9] J. Bao, T. He, S. Ruan, Y. Li, and Y. Zheng. Planning bike lanes based on sharing-bikes' trajectories. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1377–1386, 2017.

[10] P. Bischoff. Surveillance camera statistics: which city has the most cctv cameras? may 2021.

[11] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah. Signature verification using a" siamese" time delay neural network. In *Advances in neural information processing systems*, pages 737–744, 1994.

[12] R. Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

[13] J. Chen, R. Wang, j. Luan, and J. Zuo. Enterprise's address recognition methods and identifying system, 2017. CN107967332A.

[14] K. Chen, Y. Yu, P. Song, X. Tang, L. Cao, and X. Tong. Find you if you drive: Inferring home locations for vehicles with surveillance camera data. *Knowledge-Based Systems*, 196:105766, 2020.

[15] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[16] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv:1406.1078*, 2014.

[17] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *CVPR*, pages 539–546. IEEE, 2005.

[18] A. Chowdhury, T. Chakravarty, A. Ghose, T. Banerjee, and P. Balamuralidhar. Investigations on driver unique identification from smartphone's gps data alone. *Journal of Advanced Transportation*, 2018, 2018.

[19] G. O. T. Collective. Tools for spatial data collection and utilization. *British Columbia in a Global Context*, 2014.

[20] G. Cui, J. Luo, and X. Wang. Personalized travel route recommendation using collaborative filtering based on gps trajectories. *International journal of digital earth*, 11(3):284–307, 2018.

[21] Y. Ding, Y. Li, K. Deng, H. Tan, M. Yuan, and L. M. Ni. Detecting and analyzing urban regions with high impact of weather change on transport. *IEEE Transactions on Big Data*, 2016.

[22] W. Dong, J. Li, R. Yao, C. Li, T. Yuan, and L. Wang. Characterizing driving styles with deep learning. *arXiv:1607.03611*, 2016.

[23] Y. Dong, N. V. Chawla, and A. Swami. metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD*, pages 135–144, 2017.

[24] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231, 1996.

[25] S. Ezzini, I. Berrada, and M. Ghogho. Who is behind the wheel? driver identification and fingerprinting. *Journal of Big Data*, 5(1):9, 2018.

[26] B. Fabiano, F. Currò, A. P. Reverberi, and R. Pastorino. Dangerous good transportation by road: from risk analysis to emergency planning. *Journal of Loss Prevention in the process industries*, 18(4-6):403–413, 2005.

[27] X. Fang, J. Huang, F. Wang, L. Zeng, H. Liang, and H. Wang. Constgat: Contextual spatial-temporal graph attention network for travel time estimation at baidu maps. In *Proc. of the 26th ACM SIGKDD*, pages 2697–2705, 2020.

[28] J. Feng, Y. Li, C. Zhang, F. Sun, F. Meng, A. Guo, and D. Jin. Deepmove: Predicting human mobility with attentional recurrent networks. In *Proc. of the 2018 WWW conference*, pages 1459–1468, 2018.

[29] Y. Gal and Z. Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. *arXiv:1512.05287*, 2015.

[30] M. F. Goodchild. Citizens as sensors: the world of volunteered geography. *GeoJournal*, 69(4):211–221, 2007.

[31] M. Haklay and P. Weber. Openstreetmap: User-generated street maps. *IEEE Pervasive computing*, 7(4):12–18, 2008.

[32] D. Hallac, A. Sharang, R. Stahlmann, A. Lamprecht, M. Huber, M. Roehder, J. Leskovec, et al. Driver identification using automobile sensor data from a single turn. In *2016 IEEE 19th ITSC*, pages 953–958. IEEE, 2016.

[33] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[34] T. K. Ho. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pages 278–282. IEEE, 1995.

[35] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[36] E. Hoffer and N. Ailon. Deep metric learning using triplet network. In *International Workshop on Similarity-Based Pattern Recognition*. Springer, 2015.

[37] H. Hong, Y. Lin, X. Yang, Z. Li, K. Fu, Z. Wang, X. Qie, and J. Ye. Heteta: Heterogeneous information network embedding for estimating time of arrival. In *Proc. of the 26th ACM SIGKDD*, pages 2444–2454, 2020.

[38] S. Hoteit, S. Secci, S. Sobolevsky, C. Ratti, and G. Pujolle. Estimating human trajectories and hotspots through mobile phone data. *Computer Networks*, 64:296–307, 2014.

[39] B. Hu, Z. Lu, H. Li, and Q. Chen. Convolutional neural network architectures for matching natural language sentences. In *NIPS*, pages 2042–2050, 2014.

[40] L. Huang, J. Li, H. Hao, and X. Li. Micro-seismic event detection and location in underground mines by using convolutional neural networks (cnn) and deep learning. *Tunnelling and Underground Space Technology*, 81:265–276, 2018.

[41] J. Yuan, Y. Zheng, L. Zhang, X. Xie. T-Finder: A Recommender System for Finding Passengers and Vacant Taxis. *IEEE TKDE*, 25(10):2390–2403, 2013.

[42] Y. Ji, C. Zhang, Z. Zuo, and J. Chang. Mining user daily behavior based on location history. In *2012 IEEE 14th International Conference on Communication Technology*, pages 881–886. IEEE, 2012.

[43] D. Jiang, Y. Ding, H. Zhang, Y. Liu, T. He, Y. Yang, and D. Zhang. Alwaes: an automatic outdoor location-aware correction system for online delivery platforms. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 5(3):1–24, 2021.

[44] R. R. Joshi. A new approach to map matching for in-vehicle navigation systems: the rotational variation metric. In *Proc. of ITSC 2001*, pages 33–38. IEEE, 2001.

[45] H. Kamper, W. Wang, and K. Livescu. Deep convolutional acoustic word embeddings using word-pair side information. In *ICASSP*, pages 4950–4954. IEEE, 2016.

[46] B. Y. Kara and V. Verter. Designing a road network for hazardous materials transportation. *Transportation Science*, 38(2):188–196, 2004.

[47] D. Karamshuk, A. Noulas, S. Scellato, V. Nicosia, and C. Mascolo. Geo-spotting: mining online location-based services for optimal retail store placement. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 793–801, 2013.

[48] S. Khetarpaul, R. Chauhan, S. Gupta, L. V. Subramaniam, and U. Nambiar. Mining gps data to determine interesting locations. In *Proceedings of the 8th International Workshop on Information Integration on the Web: in conjunction with WWW 2011*, pages 1–6, 2011.

[49] A. V. Khezerlou, X. Zhou, L. Li, Z. Shafiq, A. X. Liu, and F. Zhang. A traffic flow approach to early detection of gathering events: Comprehensive results. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 8(6):74, 2017.

[50] T. Kieu, B. Yang, C. Guo, and C. S. Jensen. Distinguishing trajectories from different drivers using incompletely labeled trajectories. In *Proceedings of the 27th ACM International CIKM*, pages 863–872. ACM, 2018.

[51] P. Kordjamshidi, M. Van Otterlo, and M.-F. Moens. Spatial role labeling: Towards extraction of spatial relations from natural language. *ACM Transactions on Speech and Language Processing (TSLP)*, 8(3):1–36, 2011.

[52] A. Koriakine and E. Saveliev. Wikimapia https://en.wikipedia.org/wiki/wikimapia, 2006.

[53] T. Kurashima, T. Iwata, G. Irie, and K. Fujimura. Travel route recommendation using geotags in photo sharing sites. In *Proc. of the 19th ACM CIKM*, pages 579–588, 2010.

[54] L. Li and M. F. Goodchild. Constructing places from spatial footprints. In *Proceedings of the 1st ACM SIGSPATIAL international workshop on crowdsourced and volunteered geographic information*, pages 15–21, 2012.

[55] Q. Li, Y. Zheng, X. Xie, Y. Chen, W. Liu, and W.-Y. Ma. Mining user similarity based on location history. In *Proceedings of the 16th ACM SIGSPATIAL international conference on Advances in geographic information systems*, pages 1–10, 2008.

[56] R. Li, H. He, R. Wang, Y. Huang, J. Liu, S. Ruan, T. He, J. Bao, and Y. Zheng. Just: Jd urban spatio-temporal data engine. In *ICDE*, pages 1558–1569. IEEE, 2020.

[57] R. Li, H. He, R. Wang, S. Ruan, T. He, J. Bao, J. Zhang, L. Hong, and Y. Zheng. Trajmesa: A distributed nosql-based trajectory data management system. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[58] R. Li, H. He, R. Wang, S. Ruan, Y. Sui, J. Bao, and Y. Zheng. Trajmesa: A distributed nosql storage engine for big trajectory data. In *36th IEEE International Conference on Data Engineering, ICDE 2020, Dallas, TX, USA, April 20-24, 2020*, pages 2002–2005. IEEE, 2020.

[59] X. Li, K. Zhao, G. Cong, C. S. Jensen, and W. Wei. Deep representation learning for trajectory similarity computation. In *Proc. of IEEE 34th ICDE*, pages 617–628. IEEE, 2018.

[60] Y. Li, J. Luo, C.-Y. Chow, K.-L. Chan, Y. Ding, and F. Zhang. Growing the charging station network for electric vehicles with trajectory data analytics. In *ICDE*, 2015.

[61] Y. Li, M. Steiner, J. Bao, L. Wang, and T. Zhu. Region sampling and estimation of geosocial data with dynamic range calibration. In *ICDE*, 2014.

[62] D. Lian, Y. Wu, Y. Ge, X. Xie, and E. Chen. Geography-aware sequential location recommendation. In *Proc. of the 26th ACM SIGKDD*, pages 2009–2019, 2020.

[63] D. Lian, X. Xie, V. W. Zheng, N. J. Yuan, F. Zhang, and E. Chen. Cepr: A collaborative exploration and periodically returning model for location prediction. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 6(1):1–27, 2015.

[64] Y. Liang, S. Ke, J. Zhang, X. Yi, and Y. Zheng. Geoman: Multi-level attention networks for geo-sensory time series prediction. In *IJCAI*, pages 3428–3434, 2018.

[65] B. Liao, J. Zhang, C. Wu, D. McIlwraith, T. Chen, S. Yang, Y. Guo, and F. Wu. Deep sequence learning with auxiliary information for traffic prediction. In *Proc. of the 24th ACM SIGKDD*, pages 537–546, 2018.

[66] Z. Lin, G. Zhang, Z. He, J. Feng, W. Wu, and Y. Li. Vehicle trajectory recovery on road network based on traffic camera video data. In *Proceedings of the 29th International Conference on Advances in Geographic Information Systems*, pages 389–398, 2021.

[67] C. Liu, K. Deng, C. Li, J. Li, Y. Li, and J. Luo. The optimal distribution of electric-vehicle chargers across a city. In *ICDM*. IEEE, 2016.

[68] H. Liu, G. Wu, and G. Wang. Tell me where to go and what to do next, but do not bother me. In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 375–376, 2014.

[69] Y. Liu, K. Zhao, G. Cong, and Z. Bao. Online anomalous trajectory detection with deep generative sequence modeling. In *36th ICDE*, pages 949–960. IEEE, 2020.

[70] J. O. López, A. C. C. Pinilla, et al. Driver behavior classification model based on an intelligent driving diagnosis system. In *15th ITS*, pages 894–899. IEEE, 2012.

[71] Y. Lou, C. Zhang, X. Xie, Y. Zheng, W. Wang, and Y. Huang. Map-matching for low-sampling-rate gps trajectories. In *Proc. of 18th ACM SIGSPATIAL*, 2009.

[72] C.-T. Lu, P.-R. Lei, W.-C. Peng, and I.-J. Su. A framework of mining semantic regions from trajectories. In *International Conference on Database Systems for Advanced Applications*, pages 193–207. Springer, 2011.

[73] B. Lyu, S. Li, Y. Li, J. Fu, A. C. Trapp, H. Xie, and Y. Liao. Scalable user assignment in power grids: a data driven approach. In *SIGSPATIAL GIS*. ACM, 2016.

[74] M. Qu, H. Zhu, J. Liu, G. Liu, H. Xiong. A Cost-Effective Recommender System for Taxi Drivers. In *The 20th SIGKDD'14*, pages 45–54, New York, NY, 2014. ACM.

[75] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[76] M. Madaio, S.-T. Chen, O. L. Haimson, W. Zhang, X. Cheng, M. Hinds-Aldrich, D. H. Chau, and B. Dilkina. Firebird: Predicting fire risk and prioritizing fire inspections in atlanta. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 185–194, 2016.

[77] I. Mariotti, C. Pacchi, and S. Di Vita. Co-working spaces in milan: Location patterns and urban effects. *Journal of Urban Technology*, 24(3):47–66, 2017.

[78] A. MARSHALL. Uber's New Features Put a Focus on Rider Safety . https://www.wired.com/story/ubers-new-features-focus-rider-safety/, 09 2019.

[79] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in NeurIPS*, pages 3111–3119, 2013.

[80] P. Newson and J. Krumm. Hidden markov map matching through noise and sparseness. In *Proc. of the 17th ACM SIGSPATIAL*, pages 336–343, 2009.

[81] M.-h. Oh and G. Iyengar. Sequential anomaly detection using inverse reinforcement learning. In *Proceedings of the 25th ACM SIGKDD*, pages 1480–1490, 2019.

[82] M. Pan, Y. Li, X. Zhou, Z. Liu, R. Song, H. Lu, and J. Luo. Dissecting the learning curve of taxi drivers: A data-driven approach. In *Proceedings of the 2019 SIAM SDM*, pages 783–791. SIAM, 2019.

[83] S. H. Park, B. Kim, C. M. Kang, C. C. Chung, and J. W. Choi. Sequence-to-sequence prediction of vehicle trajectory via lstm encoder-decoder architecture. In *2018 IEEE IV*, pages 1672–1678. IEEE, 2018.

[84] D. Parker. Package theft: What you need to know. https://www.sourcetoday.com/distribution/article/21139045/package-theft-what-you-need-to-know.

[85] B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014.

[86] T.-A. N. Pham, X. Li, and G. Cong. A general model for out-of-town region recommendation. In *Proceedings of the 26th International Conference on World Wide Web*, pages 401–410, 2017.

[87] E. Planas, E. Pastor, F. Presutto, and J. Tixier. Results of the mitra project: Monitoring and intervention for the transportation of dangerous goods. *Journal of hazardous materials*, 152(2):516–526, 2008.

[88] G. Purdy. Risk analysis of the transportation of dangerous goods by road and rail. *Journal of Hazardous materials*, 33(2):229–259, 1993.

[89] H. Qin, X. Zhan, Y. Li, X. Yang, and Y. Zheng. Network-wide traffic states imputation using self-interested coalitional learning. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 1370–1378, 2021.

[90] A. E. Raftery, W. Gilks, S. Richardson, and D. Spiegelhalter. Hypothesis testing and model. *Markov chain Monte Carlo in practice*, pages 165–187, 1995.

[91] M. Rahmani, E. Jenelius, and H. N. Koutsopoulos. Route travel time estimation using low-frequency floating car data. In *16th International IEEE ITSC*, pages 2292–2297. IEEE, 2013.

[92] H. Ren, M. Pan, Y. Li, X. Zhou, and J. Luo. St-siamesenet: Spatio-temporal siamese networks for human mobility signature identification. In *Proc. of the 26th ACM SIGKDD*, pages 1306–1315, 2020.

[93] F. Ricci, L. Rokach, and B. Shapira. Introduction to recommender systems handbook. In *Recommender systems handbook*, pages 1–35. Springer, 2011.

[94] H. Rong, X. Zhou, C. Yang, Z. Shafiq, and A. Liu. The rich and the poor: A markov decision process approach to optimizing taxi driver revenue efficiency. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 2329–2334. ACM, 2016.

[95] F. Rosenblatt. Principles of neurodynamics. perceptrons and the theory of brain mechanisms. Technical report, Cornell Aeronautical Lab Inc Buffalo NY, 1961.

[96] S. Ruan, R. Li, J. Bao, T. He, and Y. Zheng. Cloudtp: A cloud-based flexible trajectory preprocessing framework. In *34th IEEE International Conference on Data Engineering, ICDE 2018, Paris, France, April 16-19, 2018*, pages 1601–1604. IEEE Computer Society, 2018.

[97] S. Ruan, C. Long, J. Bao, C. Li, Z. Yu, R. Li, Y. Liang, T. He, and Y. Zheng. Learning to generate maps from trajectories. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 890–897, 2020.

[98] S. Ruan, Z. Xiong, C. Long, Y. Chen, J. Bao, T. He, R. Li, S. Wu, Z. Jiang, and Y. Zheng. Doing in one go: Delivery time inference based on couriers' trajectories. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2813–2821, 2020.

[99] W. S. Ma, Y. Zheng. A large-scale dynamic taxi ridesharing service. In *The 29th ICDE'13*, pages 410–421, New York, NY, 2013. IEEE.

[100] V. Shubina, S. Holcer, M. Gould, and E. S. Lohan. Survey of decentralized solutions with mobile devices for user location tracking, proximity detection, and contact tracing in the covid-19 era. *Data*, 5(4):87, 2020.

[101] F. Siddiqui. Uber makes changes amid swarm of criticism over rider safety. `https://www.washingtonpost.com/technology/2019/09/26/uber-makes-safety-changes-amid-swarm-criticism-over-protection-ride` 19.

[102] H. Su, K. Zheng, J. Huang, H. Wang, and X. Zhou. Calibrating trajectory data for spatio-temporal similarity analysis. *The VLDB Journal*, 24(1):93–116, 2015.

[103] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. *Advances in NeurIPS*, 27:3104–3112, 2014.

[104] G. Synnaeve, T. Schatz, and E. Dupoux. Phonetics embedding learning with side information. In *2014 IEEE SLT*, pages 106–111. IEEE, 2014.

[105] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE conference on CVPR*, pages 1701–1708, 2014.

[106] P. Tong, M. Li, M. Li, J. Huang, and X. Hua. Large-scale vehicle trajectory reconstruction with camera sensing network. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, pages 188–200, 2021.

[107] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *arXiv:1706.03762*, 2017.

[108] M. Verma. Railroad transportation of dangerous goods: A conditional exposure approach to minimize transport risk. *Transportation research part C: emerging technologies*, 19(5):790–802, 2011.

[109] M. Verma and V. Verter. Railroad transportation of dangerous goods: Population exposure to airborne toxins. *Computers & operations research*, 34(5):1287–1303, 2007.

[110] D. Wang, J. Zhang, W. Cao, J. Li, and Y. Zheng. When will you arrive? estimating travel time based on deep neural networks. In *Proc. of the AAAI*, volume 18, pages 1–8, 2018.

[111] H. Wang, D. Kifer, C. Graif, and Z. Li. Crime rate inference with big data. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 635–644, 2016.

[112] J. Wang, C. Chen, J. Wu, and Z. Xiong. No longer sleeping with a bomb: a duet system for protecting urban safety from dangerous goods. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1673–1681, 2017.

[113] J. Wang, N. Wu, X. Lu, X. Zhao, and K. Feng. Deep trajectory recovery with fine-grained calibration using kalman filter. *IEEE TKDE*, 2019.

[114] X. Wang, Z. Zhang, and Y. Luo. Clustering methods based on stay points and grid density for hotspot detection. *ISPRS International Journal of Geo-Information*, 11(3):190, 2022.

[115] J. H. Ward Jr. Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, 58(301):236–244, 1963.

[116] Wikipedia. 2020 beirut explosion. https://www.abc.net.au/news/2019-01-02/illegal-chemicals-stored-in-melbourne-warehouses-epa-claims/10678258, 2019.

[117] Wikipedia. 2020 beirut explosion. https://en.wikipedia.org/wiki/2020_Beirut_explosion, 2020.

[118] D. Xi, F. Zhuang, Y. Liu, J. Gu, H. Xiong, and Q. He. Modelling of bi-directional spatio-temporal dependence and users' dynamic preferences for missing poi check-in identification. In *Proc. of the AAAI*, volume 33, pages 5458–5465, 2019.

[119] T. Xia, Y. Qi, J. Feng, F. Xu, F. Sun, D. Guo, and Y. Li. Attnmove: History enhanced trajectory recovery via attentional network. *arXiv:2101.00646*, 2021.

[120] X. Xiao, Y. Zheng, Q. Luo, and X. Xie. Finding similar users using category-based location history. In *Proceedings of the 18th SIGSPATIAL international conference on advances in geographic information systems*, pages 442–445, 2010.

[121] Z. Xie and J. Yan. Kernel density estimation of traffic accidents in a network space. *Computers, environment and urban systems*, 32(5):396–406, 2008.

[122] T. Xu, H. Zhu, X. Zhao, Q. Liu, H. Zhong, E. Chen, and H. Xiong. Taxi driving behavior analysis in latent vehicle-to-vehicle networks: A social influence perspective. In *Proceedings of the 22nd SIGKDD*, pages 1285–1294. ACM, 2016.

[123] Y. Ge and H. Xiong and A. Tuzhilin and K. Xiao and M. Gruteser. An energy-efficient mobile recommender system. In *The the 16th International Conference on KDD*, pages 899–908, New York, NY, 2010. ACM.

[124] Y. Ge, C. Liu, H. Xiong, J. Chen. A Taxi Business Intelligence System. In *The 17th International Conference on KDD*, pages 735–738, New York, NY, 2011. ACM.

[125] H. Yao, F. Wu, J. Ke, X. Tang, Y. Jia, S. Lu, P. Gong, J. Ye, and Z. Li. Deep multi-view spatial-temporal network for taxi demand prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[126] D. Yu, K. Xu, and D. Wang. Modeling user contextual behavior semantics with geographical influence for point-of-interest recommendation. In *SEKE*, pages 373–484, 2019.

[127] F. Yu, W. Ao, H. Yan, G. Zhang, W. Wu, and Y. Li. Spatio-temporal vehicle trajectory recovery on road network based on traffic camera video data. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 4413–4421, 2022.

[128] Q. Yu, Y. Luo, C. Chen, and X. Zheng. Road congestion detection based on trajectory stay-place clustering. *ISPRS International Journal of Geo-Information*, 8(6):264, 2019.

[129] Y. Yu, X. Tang, H. Yao, X. Yi, and Z. Li. Citywide traffic volume inference with surveillance camera records. *IEEE Transactions on Big Data*, 7(6):900–912, 2019.

[130] J. Yuan, Y. Zheng, C. Zhang, X. Xie, and G.-Z. Sun. An interactive-voting based map matching algorithm. In *Proc. of 11th MDM*, pages 43–52. IEEE, 2010.

[131] J. Yuan, Y. Zheng, L. Zhang, X. Xie, and G. Sun. Where to find my next passenger. In *Proceedings of the 13th Ubicomp*, pages 109–118, New York, NY, 2011. ACM.

[132] Z. Yuan, X. Zhou, and T. Yang. Hetero-convlstm: A deep learning approach to traffic accident prediction on heterogeneous spatio-temporal data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 984–992. ACM, 2018.

[133] J. Zhang, Y. Zheng, and D. Qi. Deep spatio-temporal residual networks for citywide crowd flows prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

[134] X. Zhang, X. Zhao, and J. Rong. A study of individual characteristics of driving behavior based on hidden markov model. *Sensors & Transducers*, 167(3):194, 2014.

[135] Y. Zhang, Y. Li, X. Zhou, X. Kong, and J. Luo. Curb-gan: Conditional urban traffic estimation through spatio-temporal generative adversarial networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 842–852, 2020.

[136] B. Zhao and D. Z. Sui. True lies in geospatial big data: detecting location spoofing in social media. *Annals of GIS*, 23(1):1–14, 2017.

[137] M. Zheng, S. Karanam, and R. J. Radke. Towards automated spatio-temporal trajectory recovery in wide-area camera networks. *IEEE Transactions on Biometrics, Behavior, and Identity Science*, 3(1):59–71, 2020.

[138] S. Zheng, Y. Yue, and J. Hobbs. Generating long-term trajectories using deep hierarchical networks. *Advances in NeurIPS*, 29:1543–1551, 2016.

[139] Y. Zheng. Trajectory data mining: an overview. *ACM TIST*, 6(3):1–41, 2015.

[140] Y. Zheng, L. Capra, O. Wolfson, and H. Yang. Urban computing: concepts, methodologies, and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 5(3):1–55, 2014.

[141] Y. Zheng and X. Xie. Learning location correlation from gps trajectories. In *2010 Eleventh International Conference on Mobile Data Management*, pages 27–32. IEEE, 2010.

[142] Z. Zhu, H. Ren, S. Ruan, B. Han, J. Bao, R. Li, Y. Li, and Y. Zheng. Icfinder: A ubiquitous approach to detecting illegal hazardous chemical facilities with truck trajectories. In *Proceedings of the 29th International Conference on Advances in Geographic Information Systems*, 2021.