LHMM: A Learning Enhanced HMM Model for Cellular Trajectory Map Matching

Weijie Shi[1], Jiajie Xu* [1], Junhua Fang[1], Pingfu Chao[1], An Liu[1], and Xiaofang Zhou[2]

[1]School of Computer Science and Technology, Soochow University, Suzhou, China
[2]The Hong Kong University of Science and Technology, Hong Kong, China
[1]shiweijie0311@foxmail.com, [1]{xujj, jhfang, pfchao, anliu}@suda.edu.cn, [2]zxf@cse.ust.hk

***Abstract*—Map matching is a problem to align recorded location data to a digital map. It has been well studied to map GPS data collected from vehicles to paths in a road network. The problem of Cellular Trajectory Map-Matching (CTMM) is a new problem that deals with trajectories of cellular-based positioning data. It has a wide range of applications, for example, for telecommunication companies to understand and predict traffic information based on telecom tokens obtained from vehicles. CTMM is a significantly more challenging task that faces much lower data precision and higher positioning errors. While Hidden Markov Model (HMM) based methods can achieve satisfactory results for GPS-based map matching, we show that they cannot be directly applied to the CTMM problem. In this paper, we aim at reducing the impact of positioning errors by incorporating knowledge obtained by neural networks into learned probabilities. A multi-relational graph learning method is developed to generate meaningful embedding, with multi-relational useful information fully preserved in a shared space. An attentive neural network is then designed as the learner for observation probability, incorporating the knowledge of the dynamic correlation between roads and cell towers under varying trajectory contexts. A transition probability learner is used to capture implicit deep features for enhanced transition probability modeling. Finally, the learned observation and transition probabilities are seamlessly integrated into HMM to guide more accurate path-finding. Extensive experiments on two large-scale cellular datasets reveal that our approach achieves high accuracy and robustness on CTMM.**

***Index Terms*—Map-matching, Cellular Trajectory Data, Trajectory Data Pre-processing**

I. Introduction

The widespread use of mobile phones has generated large-scale cellular trajectory data, where each trajectory is a sequence of time-stamped (cell tower) locations sampled via mobile positioning. Cellular trajectories record people's moving history, and have been widely used in various applications, including security tracking [1], traffic management [2], [3], user behavior analysis [4]–[6] and COVID-19 control [7]. To exploit the value behind these data, an essential preprocessing task is Cellular Trajectory Map-Matching (CTMM), which aims to return the traveled path by aligning cellular trajectories onto the road network.

However, existing map-matching methods are mainly designed for GPS trajectories and are unsuitable for the CTMM task. The mainstream map-matching algorithms [8]–[10] adopt the Hidden Markov Model, which treats each road segment as a hidden state, and recovers the path as a road sequence based on the observed trajectory points. Specifically, they rely on an *observation probability* to locate potential road segments of each point, and a *transition probability* to evaluate the likelihood of the object moving from one road to another. Particularly, in existing methods [8], [11], [12], these two probabilities are guided by explicit features (e.g. spatial distance) that can well reflect the point-road correlation, and achieve satisfactory results in GPS trajectory map-matching. Unfortunately, it does not hold for CTMM, since a cellular trajectory depicts the position by the physical location of the interacted cell tower, which usually deviates from the user's actual location by 0.1-3 kilometers [13], [14], and is much higher than the 1-50 meter difference in GPS trajectories. As a result, traditional HMM-based methods would fail due to the absence of robust observation and transition probabilities under high positioning errors.

Recently, deep learning techniques are applied to improve the matching performance for CTMM via capturing useful implicit features. In this line, the state-of-the-art method DMM [15] treats CTMM as a seq2seq problem, and transforms the cell tower sequence to road sequence via RNN inference. However, the seq2seq model faces error propagation [16], exposure bias [17] and hallucination problems [18]. Specifically, utilizing knowledge from previously matched roads for subsequent matching sometimes leads to severe error propagation. The exposure bias and hallucination issues also make the seq2seq model difficult to train and converge. Overall, seq2seq is typically used for sequence generation tasks with monotonous sequential transfer patterns or low accuracy requirements (e.g. QA and machine translation), which are in contrast to CTMM that calls for accurate path generation subject to the road network, physical constraints, and varying mobility patterns.

Example 1. An illustrative example is shown in Figure 1, which includes map-matching results of the HMM-based method (green color) and learning-based method (red line) of the same cellular trajectory. The ground-truth path is the blue line. We can easily observe that the HMM confronts an error on noisy point x_1, which yet can be avoided by the seq2seq-based model since it can locate x_1 to more relevant but farther roads by the learned knowledge. However, for the

*Corresponding author

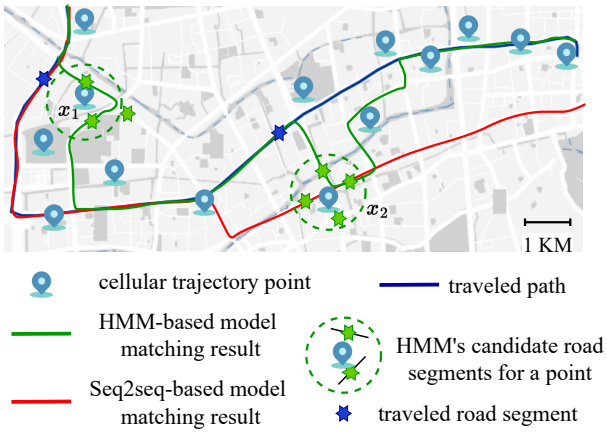


Fig. 1. Illustrative example of cellular trajectory map-matching.

point x_2 with extremely high positioning error, both methods fail to match it to the correct path. The seq2seq-based model suffers from severe error propagation, such that the error of x_2 leads to subsequent matching error. In contrast, the HMM can avoid error propagation and correct mismatches in time. Therefore, the HMM shows strong robustness in ensuring the path roughly follows the cellular trajectory by physical constraints, while learning-based methods are superior in dealing with noisy points.

Therefore, a more practical solution is to integrate the learning capabilities into the HMM framework. We thus adopt the HMM framework as the backbone, which enables us to easily control the map-matching with intuitive physical constraints, and inherit the stability of the HMM in coarse resolution. In addition, deep learning components can be added to the HMM to explore and utilize useful knowledge for more accurate path-finding, particularly for the trajectory with high positioning errors. However, designing such a learning-enhanced model faces several challenges.

First, it requires to model multi-relational information and useful implicit knowledge. Accurate CTMM processing relies on multi-relational information implied in the dataset. For example, to better locate a trajectory point with high positioning error, it is essential to capture the co-occurrence relationship between the cell towers and the roads that are frequently interacted with. Spatial proximity should be preserved to reflect not only the topological structure of the road network, but also the sequentiality of cell towers in trajectories, providing both spatial and moving pattern information. These relations need to be effectively captured and embedded into a shared space, in a balanced and informative manner. Furthermore, based on this, it is important to explore useful knowledge that can boost CTMM processing, such as the implicit and dynamic correlation between the road segment and the cell tower under varying trajectory contexts.

Second, the learned knowledge must be rationally fused into the HMM algorithm by extending the observation and transition probabilities. To accurately locate the road of each point, an effective learned observation probability is required that not only captures the implicit context-aware knowledge for better positioning denoising, but also limits the candidate search space by the explicit features. Besides, the transition probability for path evaluation should be able to decide a better path by not only explicit features (e.g. less distance and turns) but also its hidden relevance to the trajectory, which means a learning-based transition probability is necessary.

Besides, the Viterbi algorithm cannot completely handle the path-finding process for the CTMM task. For efficiency concerns, the HMM-based methods utilize the Viterbi algorithm to choose a globally optimal path, where each point is mapped to some road candidates selected by observation probability. Unfortunately, if all road candidates of a point (e.g. x_2 in Figure 1) do not belong to the ground-truth path, a mismatch will inevitably occur. This phenomenon frequently happens in cellular trajectories due to their high positioning errors. The path-finding process is thus essential to be improved, so as to provide chances to skip such noisy points.

To this end, this paper proposes a learning-enhanced HMM model for CTMM, named LHMM. It incorporates data knowledge of the deep neural network into the HMM framework to guide the evaluation of roads and paths. In the embedding layer, we construct a multi-relational graph and utilize a heterogeneous graph neural network to extract the implicit high-order relations, so that the multi-relational information can be fully considered. To accurately locate the potential position, an attention network is applied to precisely model the dynamic correlation between roads and points for learned observation probability. And the learned transition probability model the hidden relevance between moving paths and trajectories. Furthermore, we integrate these learned probabilities into the HMM path-finding process. To combat noisy points, we design a series of shortcuts into the candidate graph, providing chances to skip such noisy points. The primary contributions are summarized as follows:

- We propose a neuralized HMM algorithm to support accurate CTMM under high positioning errors. To the best of our knowledge, this is the first learning-enhanced HMM map-matching method, which inherits the robustness of the HMM and fuses deep semantic features.
- We propose a multi-relational graph-based representation learning to embed cell towers and road segments with synergistic embeddings, with useful relations and knowledge to be fully captured for the CTMM task.
- We design a learned observation probability with implicit context-aware correlation between roads and points, and a learned transition probability with hidden relevance between moving paths and trajectories. These two learned probabilities then guide the path-finding on an improved candidate graph.
- Extensive experiments are conducted on two large real cellular datasets. The results show that our model achieves state-of-the-art performances.

II. Related Work

GPS Trajectory Map-matching. With the development of traffic sensors and GPS-enabled devices, the task of map-matching has received much attention from the research community [19], [20], which aims to identify the traveled road segment where the user (or vehicle) is/has been driving.

The majority of existing map-matching methods are designed for GPS trajectories. In the literature, classical methods include geometric algorithms [21]–[24], topological algorithms [25], [26], probabilistic algorithms [27], [28], Kalman filter [29], fuzzy logic [30], [31], etc. More advanced methods [8], [10], [11], [32]–[35] adopt HMM to take advantage of its superb abilities in concurrently evaluating multiple hypotheses of the actual mapping to find the eventual maximal likelihood solution. Variants of HMM-based map-matching algorithms are proposed and widely used due to their superior performance. For example, IF-Matching [32] utilizes moving speed to handle many ambiguous cases. RCIVMM [35] proposes a weighted-matrix based interactive voting algorithm to select the best results from a global perspective. MCM [34] models common sub-sequence between the GPS trajectory and the potential routes. Recently, some deep learning-based models [36]–[39] exploit the knowledge (e.g., the mobility pattern) of enormous trajectory big data, with implicit features to improve the locating ability. However, these map-matching methods are designed for GPS trajectories with accurate locations.

Cellular Trajectory Map-matching. Cellular trajectory map-matching (CTMM) aims to transform a trajectory under cellular-based positioning to a path on the road network. Compared to GPS trajectories, cellular trajectories have high positioning errors, making the above works impossible to support CTMM.

In recent years, many efforts have been devoted to the CTMM task. These methods can be generally classified into two categories: (1) the HMM-based methods [12], [40]–[42]. For example, SnapNet [12] applies a series of filters to handle noisy locations and a number of heuristics (e.g. moving direction, fewer turns) to reduce noise interference. THMM [42] considers the geometric relationship between the road segments to constrain the HMM path-finding process. (2) the seq2seq-based method [15]. It adopts a recurrent neural network (RNN) to identify the most-likely path given a sequence of trajectory points. However, seq2seq models are normally used in fields like QA and machine translation, and are relatively weak in supporting sequence generation tasks with the requirement of high accuracy. It always finds low-quality paths very dissimilar from the ground-truth path. In contrast, HMM shows much greater stability due to its utilization of physical hypotheses. This ensures more likely return paths roughly follow the trajectory points by adopting HMM.

Despite the HMM can act as an effective backbone for CTMM, it lacks the ability to consider high-order point-road information to overcome high positioning errors, especially in challenging situations. We thus aim to improve existing methods by proposing neuralized HMM. Different from conventional HMM-based map-matching methods, we leverage the effectiveness of neural networks in learning implicit deep features, and improve the observation and transition probabilities by fusing learned knowledge for enhanced path evaluation.

III. Preliminaries

In this section, we give some useful definitions and formalize the problem of CTMM. Then, we present the basic concept of map-matching with standard HMM.

A. Problem Definition

We first define some key concepts in map matching, and then formalize the problem of this paper.

Definition 1 (Cell Tower). The cell tower has a fixed spatial position, corresponding to a longitude and latitude coordinate. As the mobile phone moves, it connects to cell towers sequentially, forming cellular trajectory data.

Definition 2 (Cellular Trajectory). The cellular trajectory X is a cell tower sampling sequence denoted as $x_1, x_2, \ldots, x_{|X|}$, where $x_i = (pos_i, t_i)$ is a trajectory point under cellular-based positioning, which contains its position and timestamp. Note that pos_i is the location of the interacted cell tower by the smartphone, which has high positioning error relative to the actual location.

Definition 3 (Road Network). Let $G\langle V, E\rangle$ denote a road network, where V is a set of nodes representing intersections or terminal points; E contains a set of directed road segments e_i connecting these nodes. A path $\mathcal{P}$ on road network G is composed of a sequence of consecutive road segments, which is expressed as $e_1 \to e_2 \to \cdots \to e_{|\mathcal{P}|}$.

Problem Formalization. Given a road network G and a cellular trajectory X, the CTMM problem aims to map-match a path $\mathcal{P}$ that is close to the ground-truth path $\mathcal{P}_g$ of X.

B. Map-matching with Standard HMM

Reviewing HMM Algorithm. In the literature, the HMM algorithm is widely used in map-matching due to its good robustness and accuracy. By applying HMM, each point x_i is projected onto a candidate road segment defined as:

Definition 4 (Candidate Road Segments). Candidate road segments of a trajectory point are a set of roads, which are potential locations of the point. For efficiency concerns, a trajectory point is only map-matched to its candidate road segments. We use $c_i^j \in C_i$ to denote the j-th candidate road segment of x_i, where C_i is x_i's candidate road set.

Following previous works [8], [12], [42], a matching path can be transformed into a series of moving between candidate road segments, i.e. $\mathcal{P} : c_1^i \to c_2^j \to \cdots \to c_{|X|}^k$, where $c_{i-1}^j \to c_i^k$ is the shortest path from c_{i-1}^j to c_i^k. Then this path is evaluated by:

$$W(\mathcal{P}) = \sum_{i=2}^{|X|} P_T\left(c_{i-1}^j \to c_i^k\right) \cdot P_O\left(c_i^k | x_i\right) \tag{1}$$

where $P_O(\cdot)$ is the observation probability that denotes the likelihood of placing the point on the candidate road segment;

$P_T(\cdot)$ is the transition probability that denotes the likelihood of moving from one candidate road to another via the shortest path.

Setting probabilities $P_O(\cdot)$ and $P_T(\cdot)$. The key problem of adopting HMM for map-matching is the setting of these two probabilities, which determines the final performance. For observation probability $P_O(\cdot)$, the most popular measure is the spatial distance between a point x_i and its related candidate road c_i^k. Previous works [8], [12], [32], [42] assume that a closer distance implies a higher matching probability:

$$P_O\left(c_i^k|x_i\right) = \frac{1}{\sqrt{2\pi}\sigma_1} e^{-0.5\left(\frac{\text{dist}\left(c_i^k, x_i\right)-\mu_1}{\sigma_1}\right)^2} \tag{2}$$

where $\text{dist}(c_i^k, x_i)$ is a Euclidean distance function. μ_1 and σ_1 respectively denote the mean and standard deviation of Gaussian distribution. For transition probability $P_T(\cdot)$, existing methods assume the length of the moving path is similar to the distance of the corresponding two trajectory points:

$$P_T\left(c_{i-1}^j \to c_i^k\right) = \frac{1}{\sigma_2} e^{\frac{\left|\text{dist}\left(x_{i-1}, x_i\right)-\text{dist}\left(c_{i-1}^j, c_i^k\right)\right|}{\sigma_2}} \tag{3}$$

where $\text{dist}(c_{i-1}^j, c_i^k)$ is the route length of shortest path. σ_2 denotes the standard deviation. Moreover, many heuristics have been adopted to set observation and transition probabilities, including the velocity constraint [8], the moving speed [32], moving direction [12], and moving reachability [34], [42].

Potential Issues. The core of HMM algorithm is to evaluate roads and paths by the observation probability $P_O(\cdot)$ and transition probability $P_T(\cdot)$ respectively. However, different from GPS trajectory map-matching, the CTMM task is much more challenging due to the high positioning error (normally 0.1-3 kilometers) of cell tower positioning, which brings huge ambiguity and uncertainty for map-matching. In such cases, a road closer to a trajectory point is not necessarily more likely to be its actual location. Accordingly, the distance-based features mentioned above fail to guide the effective setting of probabilities $P_O(\cdot)$ and $P_T(\cdot)$ for CTMM, resulting in (1) inaccuracy, due to the inability of accurate evaluation of roads and paths; (2) inefficiency, as it requires much more candidate roads for each point to cover its actual location, leading to expensive evaluation on large-scale paths. To address the above limitations, we design a neuralized HMM algorithm with learned $P_O(\cdot)$ and $P_T(\cdot)$ probability functions to guide more accurate and efficient path-finding for CTMM.

IV. METHODOLOGY

In this section, we present the proposed learning-enhanced HMM model, denoted by LHMM.

A. Overview of LHMM

Based on the general HMM framework, we extend it to the neuralized HMM algorithm by incorporating deep learning. Traditionally, the observation and transition probabilities $P_O(\cdot)$ and $P_T(\cdot)$ are heuristically set and computed, which is not suitable for CTMM. We thus aim to take advantage of neural networks to automatically set the two probabilities by learned knowledge, instead of using heuristics.

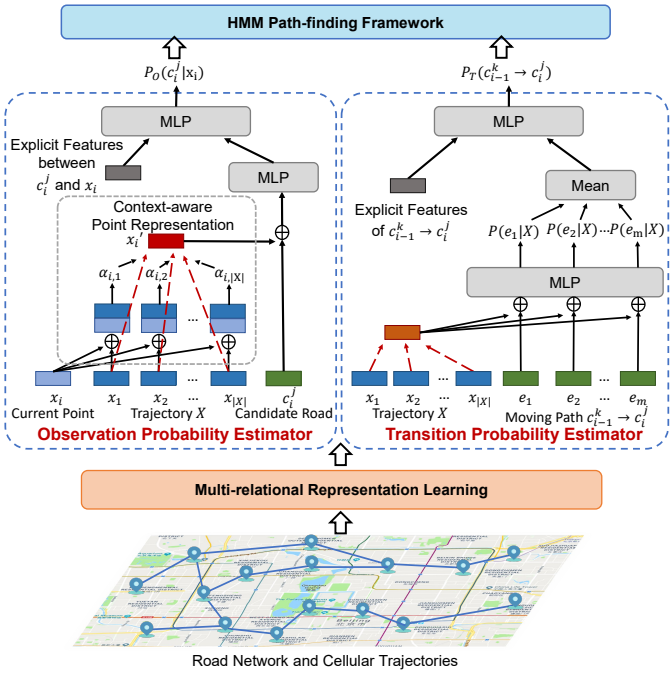


Fig. 2. The architecture of LHMM

The overall framework of our proposed algorithm LHMM is shown in Figure 2. Specifically, it first utilizes a representation learning module to preserve multi-relational semantic information in embeddings. Next, an observation probability learner is designed to derive meaningful $P_O(\cdot)$ by learning knowledge obtained from explicit and implicit features. A transition probability learner is then used to automatically set $P_T(\cdot)$ on top of the learned knowledge. Finally, the two learned probabilities are used to guide the path-finding process under HMM framework.

B. Multi-relational Representation Learning

To enable learned observation and transition probabilities by neural networks, the primary task is to effectively represent all elements (cell towers and road segments) of map-matching. Particularly, it is important to capture the multi-relation among all elements, so that useful semantic information can be provided in representations for subsequent learning of $P_O(\cdot)$ and $P_T(\cdot)$. This requirement is hardly supported by simple representation techniques like the one-hot representation. We thus present a multi-relational representation learning method based on heterogeneous graph modeling.

Multi-relational Graph Construction. There are many types of relationships among cell towers and road segments (e.g. tower-road, tower-tower, and road-road relationships), which provide great opportunities for representation learning and CTMM processing. It is important to precisely denote them by dense data structure for embedding learning.

To describe these relations, we construct a multi-relational graph $\mathcal{G} = (\mathcal{V}_e, \mathcal{V}_{ct}, \mathcal{E})$, where $\mathcal{V}_e$ and $\mathcal{V}_{ct}$ contain all road

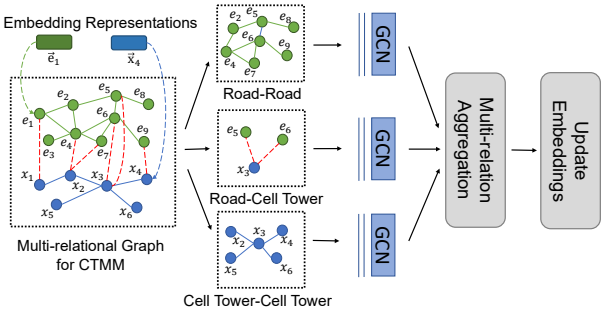


Fig. 3. The Het-Graph Encoder

segments and all cell towers respectively (i.e. $\mathcal{V}_e = G.E$). We use $\mathcal{V}$ to represent all nodes in the graph (i.e. $\mathcal{V} = \mathcal{V}_e \cup \mathcal{V}_{ct}$) when there is no ambiguity. And $\mathcal{E}$ is an edge set involving multiple types of directed edges. Specifically, three types of relations are captured in $\mathcal{G}$, including:

- **Co-occurrence** between roads and cell towers: given a trajectory point $x_i \in X$ and a road segment in corresponding traveled path $e_j \in \mathcal{P}_g$, we say they have a co-occurrence relationship if the closest cell tower in the trajectory to e_j is x_i. The edge $(x_i, CO, e_j) \in \mathcal{E}$ has weights counting the number of co-occurrence. The high co-occurrence correlation means that a trajectory point is more likely to match its frequently interacting road.
- **Sequentiality** between cell towers: given a trajectory $X : x_1, x_2, \ldots, x_{|X|}$, we say they have a sequentially moving relationship in each two neighboring points x_i and x_{i+1}. The edge $(x_i, SQ, x_{i+1}) \in \mathcal{E}$ reflects the mobility patterns among cell towers. When a trajectory point locates its actual location, the necessary calibration information can be provided by trajectory contexts, where points with a strong sequential relation to the current point are likely to contribute more for its location calibration.
- **Topological structure**: there is an edge between two road segments e_i and e_j, where $e_i, e_j \in \mathcal{V}_e$ are adjacent on the road network. This edge is denoted as $(e_i, TP, e_j) \in \mathcal{E}$, which represents the spatial proximity and the reachability between roads.

In this way, each relation is represented by a type of edge, and the graph $\mathcal{G}$ contains multi-relational information, which can benefit the CTMM task. From these relations, higher-order relationships can be mined to support map matching. Next, we discuss the embedding representations for all nodes of $\mathcal{G}$.

Het-Graph Encoder. Based on the multi-relational graph $\mathcal{G}$, we further design the Het-Graph Encoder to effectively embed road segments and cell towers with synergistic representations. It is able to fully preserve multi-relational semantic information, so that the correlation between cell towers and road segments can be easily captured.

Since $\mathcal{G}$ is a multi-relational graph, the representation learning calls for not only enabling strongly correlated nodes in $\mathcal{G}$ to have similar embedded vectors but also jointly representing and balancing multiple relations. This requires: (1) the representation of a node (i.e. a road segment or a cell tower) should consider its neighbors' information; (2) the information sent from different types of neighbors should be processed differently and adaptively. Inspired by R-GCNs [43], we design the multi-relational message propagation as shown in Figure 3. We specifically introduce it in the following.

The first step is to initialize the embedding. For each node $v_i \in \mathcal{V}$, we use $\vec{v_i} \in \mathbb{R}^{|\mathcal{V}|}$ denotes its one-hot representation, which is then converted into a low-dimensional dense vector by a learnable matrix $W_{\text{init}} \in \mathbb{R}^{|\mathcal{V}| \times d}$: $h_i^{(0)} = W_{\text{init}}^{\top} \cdot \vec{v_i}$, where $h_i^{(0)}$ represents the embedding of v_i at layer 0.

The second step is message passing. Each node receives messages from its neighboring nodes, and processes them separately based on different relation types. Given a node v_i, we derive its neighbor group $\mathcal{N}_i^{rel}$ for each relation $rel \in \mathcal{R}$, where $\mathcal{R}$ indicates three types of relations in $\mathcal{G}$. Each neighbor group sends v_i the set of messages of relation rel as

$$z_i^{rel} = \frac{1}{|\mathcal{N}_i^{rel}|} \sum_{v_j \in \mathcal{N}_i^{rel}} W_{rel}^{(l)} h_j^{(l)} \tag{4}$$

where z_i^{rel} is v_i's received message from its neighbor group $\mathcal{N}_i^{rel}$ of relation rel. $h_j^{(l)}$ denotes the embedding of v_j in l-th layer. $W_{rel}^{(l)}$ is a trainable parameter matrix.

The third step is to jointly aggregate the messages from all neighbor groups and update the embedding of node v_i as:

$$h_i^{(l+1)} = \sigma\left(\sum_{rel \in \mathcal{R}} W_{agg} z_i^{rel} + W_0^{(l)} h_i^{(l)} \right) \tag{5}$$

where W_0, W_{agg} are trainable parameters. We take ReLU as activation function $\sigma(\cdot)$. After performing q iterations, we take the node representations of the last step as the embeddings $\vec{x_i}$ and $\vec{e_j}$ for cell towers and road segments respectively. In this way, the embeddings preserve multi-relations in a shared space, so that the downstream component can easily identify the correlation between cell towers and road segments.

C. Observation Probability Learning

This part studies the learning of function $P_O(\cdot)$ for observation probability, which is the likelihood of locating a trajectory point on a candidate road. Existing HMM-based methods typically evaluate it using explicit features with physical meanings (e.g. distance), which are unfortunately unsuitable for the CTMM task due to its high positioning error, as mentioned above. To accurately locate the traveled road of each point, a key issue is to precisely model their high-order correlation from historical data, which is highly implicit and influenced by trajectory contexts. Here, we first capture implicit point-road correlation with contextual knowledge into consideration. Then we further combine some useful explicit features to obtain the observation probability $P_O(\cdot)$.

Implicit Point-Road Correlation. Intuitively, the same cell tower in different trajectories may match different locations, which requires the modeling of dynamic trajectory contexts to overcome location ambiguity and uncertainty. To enable the

current point to be capable of perceiving trajectory contexts, we utilize an attentional network to merge such context information into the current point representation, whereby we can evaluate the high-order correlation between the trajectory point and the candidate road based on contextual knowledge.

Specifically, we feed the current point embedding $\vec{x_i}$ into query of the attention model, and perceive all point embeddings $\vec{x_1}, \vec{x_2}, \ldots, \vec{x_{|X|}}$ as key and value. The attention layer self-adaptively employs the importance weights to summarize point embeddings as the current context-aware point representation $\vec{x'_i}$:

$$\vec{x'_i} = \sum_{j=1}^{|X|} \text{softmax}\left(W_v \cdot \tanh\left(W_q \vec{x_i} \oplus W_k \vec{x_j}\right)\right) \cdot \vec{x_j} \quad (6)$$

where W_q, W_k, and W_v are trainable parameters of attention weights. $\oplus$ denotes concatenate operator. On top of context-aware point representation, we can evaluate the implicit point-road correlation $P(c_i^k|x'_i)$ by a multilayer perceptron:

$$P\left(c_i^k|x'_i\right) = \text{softmax}\left(\text{MLP}\left(\vec{c_i^k} \oplus \vec{x'_i}\right)\right) \quad (7)$$

where $P(c_i^k|x'_i)$ determines whether a candidate road is likely to be the actual location of the current point under varying trajectory contexts.

Learned Observation Probability. Here, we combine implicit features and explicit features between a point and a candidate road to obtain the observation probability. In addition to implicit features, it is also important to consider explicit features. For example, the physical distance is necessary since if a road segment is far away from the current point, it is impossible to become the actual location of the user. Moreover, the current point has a high probability to match a road that frequently interacts with the cell tower. Therefore, we combine the implicit point-road correlation and some explicit features into the learned observation probability $P_O(\cdot)$, which is denoted as:

$$P_O\left(c_i^k|x_i\right) = \text{MLP}\left(P\left(c_i^k|x'_i\right) \oplus D_O\left(x_i, c_i^k\right)\right) \quad (8)$$

where $D_O(x_i, c_i^k)$ indicates the explicit features between the point and the road, including batch-normalized Euclidean distance and co-occurrence frequency.

D. Transition Probability Learning

This part studies the learning of function $P_T(\cdot)$ for transition probability. Formally, given two candidate road segments c_{i-1}^j and c_i^k for two neighboring trajectory points x_{i-1} and x_i respectively, the transition probability evaluates the likelihood of traveled path from x_{i-1} to x_i following the shortest path from c_{i-1}^j to c_i^k. Existing methods take explicit features (e.g. the length of moving path, the number of turns) only to evaluate moving paths at a coarse-grained level. However, these coarse-grained features are difficult to acutely detect if a few roads occur deviations or detours. To accurately identify unreasonable moving paths, we first capture implicit trajectory-path correlation at a fine-grained level. Then we further combine implicit features and some useful explicit features to obtain the transition probability.

Implicit Trajectory-Path Correlation. Intuitively, each road segment of the traveled path should be highly correlated with the overall trajectory. Otherwise, if there are lots of low-relevant road segments to the trajectory, this path is probably not the traveled path. To identify detours and abnormal parts of the moving path at a fine-grained level, we first evaluate the likelihood of each road in the moving path belonging to the trajectory, and further comprehensively consider these likelihoods to evaluate the moving path.

Specifically, we first need to embed trajectory representation. A simple way is to use a mean layer to merge all point embeddings of the trajectory. However, some points are more important to determine whether the road belongs to the trajectory, e.g., points closer to the road, and points that frequently interact with the road. These semantics cannot be captured by the above trajectory representation. Toward this end, we provide an improved trajectory representation approach with an attention-based neural network. For each road e_l, we merge these interacted points in a weighted way to generate trajectory representation $\vec{X_l}$ as:

$$\vec{X_l} = \sum_{i=1}^{|X|} \text{softmax}\left(W'_v \cdot \tanh\left(W'_q \vec{e_l} \oplus W'_k \vec{x_i}\right)\right) \cdot \vec{x_i} \quad (9)$$

Next, a multilayer perceptron is used to predict the likelihood of a road belonging to the trajectory by:

$$P\left(e_l|X\right) = \text{softmax}\left(\text{MLP}\left(\vec{e_l} \oplus \vec{X_l}\right)\right) \quad (10)$$

On top of $P(e_l|X)$, we can easily evaluate the hidden relevance of the moving path to the trajectory:

$$P\left(c_{i-1}^j \to c_i^k\right) = \frac{1}{|sp|} \sum_{e_l \in sp} P\left(e_l|X\right) \quad (11)$$

where sp is the shortest path from c_{i-1}^j to c_i^k , and $|sp|$ is its number of road segments.

Learned Transition Probability. Here, we combine implicit features and explicit features of the moving path to obtain the transition probability. In addition to implicit trajectory-path correlation, some explicit features are also useful for evaluating the moving path. For example, the traveled path should have a similar length compared to the cellular trajectory, since if the matching path is too long, it often means that there are detours in the path. To avoid unnecessary turns on the matching path, we assume that there is a similar number of turns between the trajectory and the matching path. We measure the number of turns based on the sum of angles of every adjacent point or road segment. We denote these two explicit features as $D_T(c_{i-1}^j \to c_i^k)$. Afterwards, the learned transition probability $P_T(\cdot)$ can be set by:

$$P_T\left(c_{i-1}^j \to c_i^k\right) = \text{MLP}\left(P\left(c_{i-1}^j \to c_i^k\right) \oplus D_T\left(c_{i-1}^j \to c_i^k\right)\right) \quad (12)$$

Training Process of $P_O(\cdot)$ and $P_T(\cdot)$. At last, we introduce the detailed training process for the above components. The

neural network requires to be trained using historical cellular trajectories with their traveled paths. The $P_O(\cdot)$ and $P_T(\cdot)$ have their own independent neural networks.

For neuralized observation probability $P_O(\cdot)$, we first assign clear semantics to the implicit point-road correlation by a classification task, which identifies the road segment with positive or negative labels to the current point. Next, we fine-tune the last MLP layer to obtain the observation probability. We sample surrounding road segments of the current point as training data, where the positive road segments that are interacted with the current point, and the rest road segments with negative labels. To balance labels, we take multiple undersampling to perform the training process. For the transition probability $P_T(\cdot)$, we follow the same two steps. First, we train the implicit trajectory-road correlation by classifying road segments in the moving path with positive or negative labels to the trajectory. Second, to evaluate the overall moving path, we fine-tune the last MLP layer by predicting the ratio of traveled roads to the moving path. All training target is to minimize the cross-entropy between outputs and labels. To avoid the overconfidence problem [44], we take the label smoothing technique proposed by [45] in the cross-entropy function.

E. HMM Path-finding Framework

Based on the neuralized observation and transition probabilities defined above, we further introduce the path-finding process in Section IV-E1 and the optimizations to address the detour caused by high positioning errors in Section IV-E2.

1) Path-finding process: Following previous HMM-based methods [8], [12], [32], the path-finding process includes three steps, i.e. *candidate preparation*, *candidate graph construction*, and *viterbi-based path-finding*.

Step 1 (Candidate Preparation). For each point on the trajectory, we first retrieve its possible candidate road segments, which are potential locations of the point. Note that for efficiency concerns, a point is only map-matched to its candidate road segments in the path-finding process.

We select top-k road segments with observation probabilities for the point x_i as candidates, forming x_i's candidate road set C_i. k is a trade-off between efficiency and accuracy. For example, x_3's candidate road segments are c_3^1, c_3^2 and c_3^3 in Figure 4.

Step 2 (Candidate Graph Construction). Given all candidate road sets, a candidate graph $G'(V', E')$ is generated to describe the path-finding space. V' is derived from candidate road sets $C_1 \cup C_2 \cup \cdots \cup C_{|X|}$, and E' is a set of edges representing the shortest path between any two neighboring candidate road segments $c_{i-1}^j \to c_i^k$, as depicted in Figure 4.

As mentioned in Section III-B, the HMM transforms the problem of map-matching into how to search one candidate path $\mathcal{P}_c : c_1^i \to c_2^j \to \cdots \to c_{|X|}^k$ from C_1 to $C_{|X|}$ to match entire trajectory X. With neuralized observation and transition probabilities, the moving between neighboring candidate road segments can be accurately evaluated by the score:

$$W\left(c_{i-1}^j \to c_i^k\right) = P_T\left(c_{i-1}^j \to c_i^k\right) \cdot P_O\left(c_i^k | x_i\right) \tag{13}$$

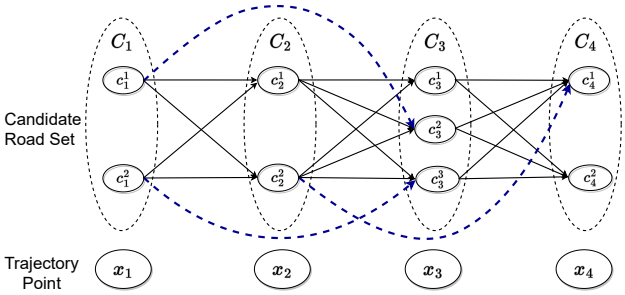


Fig. 4. The candidate graph structure, where the blue edges are shortcuts.

And further, the candidate path can be evaluated by the candidate path score:

$$W\left(\mathcal{P}_c\right) = \sum_{i=2}^{|X|} W\left(c_{i-1}^j \to c_i^k\right) \tag{14}$$

Step 3 (Viterbi-based Path-finding). Among all candidate paths, we aim to search the optimal path $\mathcal{P}$ with the highest candidate path score:

$$\mathcal{P} = \arg\max_{\mathcal{P}_c} W\left(\mathcal{P}_c\right), \qquad \forall \mathcal{P}_c \in G'\left(V', E'\right) \tag{15}$$

We adopt the Viterbi algorithm, which shows excellent performance in previous works [8], [46]. The Viterbi algorithm utilizes dynamic programming, which finds the optimal path to a candidate road segment c_i^k relying on the information of optimal paths to its predecessors (i.e. all $c_{i-1}^j \in C_{i-1}$). Formally, given a candidate road segment c_i^k, we can compute the highest candidate path score into table $f[\cdot]$ and its optimal predecessor into table $pre[\cdot]$:

$$f\left[c_i^k\right] = \max_{c_{i-1}^j \in C_{i-1}} \left(f\left[c_{i-1}^j\right] + W\left(c_{i-1}^j \to c_i^k\right)\right) \tag{16}$$

$$pre\left[c_i^k\right] = \arg\max_{c_{i-1}^j \in C_{i-1}} \left(f\left[c_{i-1}^j\right] + W\left(c_{i-1}^j \to c_i^k\right)\right) \tag{17}$$

According to the highest candidate path score table $f[\cdot]$ and the best predecessor table $pre[\cdot]$, the optimal path can be founded by:

$$c_{|X|}^k = \arg\max_{c_{|X|}^j} f\left[c_{|X|}^j\right], \qquad \forall c_{|X|}^j \in C_{|X|} \tag{18}$$

$$c_{i-1}^j = pre\left[c_i^k\right], \qquad i \in [2, |X|] \tag{19}$$

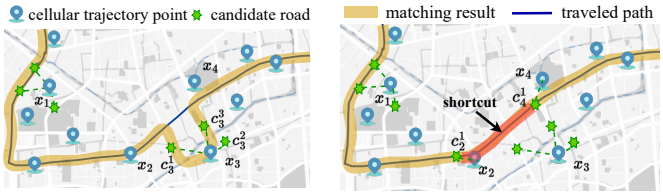
the detail of the procedure *path-finding* is shown in Algorithm 1. By fusing learned knowledge into the HMM to guide the path-finding process, we solve the locating of the noisy point x_1 in Figure 5(a). Unfortunately, we may still face inaccurate matching on noisy points according to Observation 1.

Observation 1. Given a candidate road set C_i, the detour is inevitably by the ordinary path-finding process if C_i has no overlap with the traveled path, i.e. $C_i \cap \mathcal{P}_g = \emptyset$. We call such C_i as an unqualified candidate road set.

Algorithm 1 Path-finding Algorithm

Require: Road network G, a cellular trajectory $X : x_1 \rightarrow x_2 \rightarrow \cdots \rightarrow x_{|X|}$
Ensure: Best matching path $\mathcal{P} : c_1^i \rightarrow c_2^j \rightarrow \cdots \rightarrow c_{|X|}^k$

1: Let $f[\cdot]$ denote the highest score computed so far;
2: Let $pre[\cdot]$ denote the parent of current candidate road;
3: Select candidate road set C_i for each point x_i;
4: **for all** $c_1^k \in C_1$ **do**
5: $\quad f[c_1^k] \leftarrow P_O(c_1^k)$;
6: **end for**
7: **for** $i \leftarrow 2$ to $|X|$ **do**
8: $\quad$ **for all** $c_i^k \in C_i$ **do**
9: $\quad\quad$ Fill highest candidate path score entry $f[c_i^k]$ as Equation (16);
10: $\quad\quad$ Fill best predecessor entry $pre[c_i^k]$ as Equation (17);
11: $\quad$ **end for**
12: **end for**
13: $f[\cdot]$, $pre[\cdot] \leftarrow$ Updated candidate graph by Algorithm 2;
14: $\mathcal{P} =$ Viterbi.backward$(f[\cdot], pre[\cdot])$
15: **return** $\mathcal{P}$



(a) Ordinary path-finding (b) Path-finding with shortcuts

Fig. 5. Illustration of shortcuts

As such, high positioning errors of cellular trajectories see noisy points, which result in unqualified candidate road sets and finally undermine the performance of the HMM. For example, in Figure 5(a), the point x_3 has an excessive positioning error, leading to its all candidate road segments are not in the traveled path (i.e. $c_3^1, c_3^2, c_3^3 \notin \mathcal{P}_g$), making the matching result with a detour.

To avoid the impact of unqualified candidate road sets, a series of shortcuts (skipping edges) are required to provide chances to skip them. For example, the shortcut path directly connects c_2^1 and c_4^1, ignoring the noisy point x_3 to remedy the detour in Figure 5(b). Next, we discuss how to construct the shortcut to skip the noisy point.

2) Optimization with shortcuts: From the above analysis, a series of shortcuts need to be constructed in the candidate graph, so that the negative impact of the unqualified candidate road set can be remedied.

Shortcut Construction. We first pick up a series of candidate road segment pairs to establish the shortcut on the candidate graph. From experimental evaluation, one-hop shortcuts almost avoid the impact of unqualified candidate road sets. Specifically, for each candidate road segment c_i^k, we build K shortcuts with shortest paths from its one-hop road segments

Algorithm 2 Adding Shortcuts into Path-finding Process

Require: Road network G, a cellular trajectory X, the filled highest candidate path score table $f[\cdot]$ and the filled best predecessor table $pre[\cdot]$
Ensure: Updated $f[\cdot]$ and $pre[\cdot]$ with shortcuts

1: **for** $i \leftarrow 3$ to $|X|$ **do**
2: $\quad$ **for all** $c_i^k \in C_i$ **do**
3: $\quad\quad$ Search c_i^k's best one-hop predecessor c_{i-2}^j as Equation (20);
4: $\quad\quad$ Establish the shortcut sp between c_{i-2}^j and c_i^k with shortest path;
5: $\quad\quad$ Obtain the projected road c_{i-1}^u from x_{i-1} to sp;
6: $\quad\quad$ Calculate the shortcut score f' as Equation (21);
7: $\quad\quad$ **if** $f' > f[c_i^k]$ **then**
8: $\quad\quad\quad f[c_i^k] \leftarrow f'$;
9: $\quad\quad\quad pre[c_i^k] \leftarrow c_{i-1}^u$;
10: $\quad\quad\quad pre[c_{i-1}^u] \leftarrow c_{i-2}^j$;
11: $\quad\quad$ **end if**
12: $\quad$ **end for**
13: **end for**
14: **return** $f[\cdot]$, $pre[\cdot]$

$c_{i-2}^j \in C_{i-2}$ to it. The one-hop predecessor is determined as:

$$c_{i-2}^j = \underset{c_{i-2}^j, c_{i-1}^l}{\arg\max} \left(W(c_{i-2}^j \rightarrow c_{i-1}^l) + W(c_{i-1}^l \rightarrow c_i^k) \right) \quad (20)$$

where $c_{i-2}^j \in C_{i-2}$ and $c_{i-1}^l \in C_{i-1}$. The setting of K is a trade-off between computational overhead and accuracy, and the experimental evaluation shows that one shortcut (i.e. $K = 1$) is sufficient.

Shortcut Score Setting. We assign the score to the candidate path with shortcuts. Evaluating the candidate path by Equation (14) requires the same number of transitions (between candidate road segments of consecutive points), but the path with shortcuts has a reduced number since some candidate road sets are skipped. Thus for a fair comparison between candidate paths, we need to restore the road segments of skipped candidate road sets. Specifically, given a shortcut $c_{i-2}^j \rightarrow c_i^k$, we project x_{i-1} to its closest road segment c_{i-1}^u in this shortcut, and use this projected road segment to transform the shortcut to $c_{i-2}^j \rightarrow c_{i-1}^u \rightarrow c_i^k$. The score of the candidate path to c_i^k with the shortcut is defined as:

$$f' = f[c_{i-2}^j] + W(c_{i-2}^j \rightarrow c_{i-1}^u) + W(c_{i-1}^u \rightarrow c_i^k) \quad (21)$$

At last, we present how to establish shortcuts on the candidate graph and how to improve the highest candidate path score table $f[\cdot]$ and the best predecessor table $pre[\cdot]$, whereby the path-finding process enables skipping unqualified candidate road set using shortcuts. The detailed procedure is shown in Algorithm 2, which should be inserted into line 13 of Algorithm 1.

V. EXPERIMENTS

In this section, we conduct extensive experiments to evaluate our approach. We first present the experimental settings. Next,

category	Hangzhou	Xiamen
road segments	92,913	64,828
intersections	67,330	37,591
all cellular trajectory points	3.61 million	1.18 million
all GPS trajectory points	9.73 million	4.98 million
cellular trajectory points per trajectory	34	40
GPS trajectory points per trajectory	81	88
average cellular sampling interval (s)	67	42
maximum cellular sampling interval (s)	247	185
average cellular sampling distance (m)	730	650
median cellular sampling distance (m)	493	455

TABLE I
DATASET CHARACTERISTIC

we describe the criteria that we applied. Then we compare our approach with methods designed for both GPS trajectories and cellular trajectories, and report the major results with analysis for the following questions:

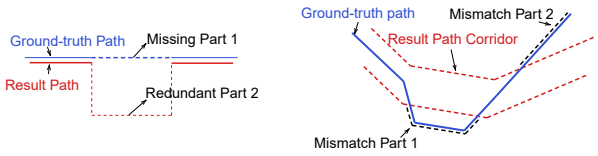
- **Q1:** What is the performance of LHMM compared with existing map-matching methods?
- **Q2:** How does each designed component influence the performance of LHMM?
- **Q3:** What is the performance of LHMM under different data distributions?
- **Q4:** How do the hyper-parameters influence the performance of LHMM?
- **Q5:** How does LHMM work intuitively?

A. Experimental Setting

1) Dataset description: In our experiments, we use two real trajectory datasets, which are located in Hangzhou, China, and Xiamen, China, getting from a mobile communication operator. The dataset contains the cellular trajectory and the corresponding GPS sampling sequence for the same travel path. The ground-truth path is generated by the GPS sampling sequence through a classical HMM algorithm [8]. Before matching, the cellular trajectory initially removes noise and smooths through a series of filters as described in [12], including the speed filter, α-trimmed mean filter, and direction filter. Table I exhibits the main data characteristic.

2) Parameter setup: We set the dimension of embeddings and all latent vectors as 128. All of the trainable parameters are optimized by Adam based on the training set, and hyper-parameters are chosen based on the validation set. The initial learning rate and weight decay rate are 1×10^{-3} and 1×10^{-4} respectively. The label smoothing is 0.1 for cross-entropy loss. To accelerate computation, we implement Het-Encoder with the message passing framework [47] in a parallel manner. Through an experimental comparison, the number of iterations q in Het-Encoder is set as 2 showing great performance. The candidate number k of each point is set to 30 for CTMM and 45 for baselines. The HMM can use a precomputation table to avoid the bottleneck of repeated shortest path searches [11].

3) Evaluation criteria: We evaluate the matching quality, which is measured by the comparison of the ground-truth path and the matching path on road segment level and coarser-grained corridor level, including *precision, recall, RMF, CMF*, *hitting ratio*, and *time*.



(a) RMF (b) CMF

Fig. 6. Illustration of criteria

Precision and Recall. Precision and recall are commonly used accuracy metrics [12], [15]. *Precision* is defined as the ratio of the total length of the correctly-matched path to the total length of the matching path. *Recall* is the ratio of the total length of the correctly-matched path to the total length of the ground truth path.

RMF. Since precision and recall cannot simultaneously account for missing and redundant mismatched road segments, we design an error metric: *Route Mismatch Fraction* (RMF), which is defined by the following:

$$\text{RMF} = \frac{\sum \text{length of mismatched road segments}}{\text{length of the ground-truth path}} \tag{22}$$

A smaller RMF means better accuracy. RMF is the strictest error indicator as shown in Figure 6(a), which focuses on the ratio of the length of missing and redundant mismatched road segments to the ground-truth path length.

CMF. Since the matching path often locates on parallel side roads of the ground-truth path (e.g. urban viaduct and its underlying roads), it can be regarded as a successful matching for some low-demanding applications. Towards this end, we propose coarser-grained *Corridor Mismatch Fraction* (CMF), which uses a x-meter-wide corridor to wrap the matching path and to measure matching accuracy as:

$$\text{CMF} = \frac{\sum \text{corridor uncovered length}}{\text{length of the ground-truth path}} \tag{23}$$

A smaller CMF means better accuracy. As shown in Figure 6(b), the CMF focuses on the ratio of the uncovered ground-truth path length of the matching path corridor to the ground-truth path length. The common corridor radius is 50 meters, recorded as CMF50, which describes the matching path that approximately coincides with the ground-truth path.

Hitting Ratio. To evaluate HMM-based methods' quality of candidate preparation, we propose *Hitting Ratio* (HR), which focuses on the proportion of candidate road sets covering the traveled path. The hitting ratio only suits the HMM-based methods, and reflects the ability of the HMM-based method to locate traveled roads for trajectory points.

Avg Time. To evaluate the efficiency, we use average inference time (Avg Time), which is defined as the average running time required to transform cellular trajectories into matching results.

4) Baselines: We consider the following methods as baselines to compare.

The methods designed for GPS trajectories include:

Dataset	Hangzhou					Xiamen				
Metric	**Precision**	**Recall**	**RMF**	**CMF50**	**Avg Time (s)**	**Precision**	**Recall**	**RMF**	**CMF50**	**Avg Time (s)**
Methods designed for GPS trajectory map-matching										
STM [8]	0.388	0.476	1.237	0.225	0.040	0.411	0.498	1.050	0.198	0.044
IVMM [10]	0.409	0.518	1.125	0.188	0.101	0.428	0.529	0.936	0.172	0.136
IFM [32]	0.430	0.522	1.024	0.178	0.045	0.451	0.537	0.889	0.167	0.048
DeepMM [37]	0.446	0.544	0.881	0.172	0.951	0.478	0.568	0.785	0.158	1.284
MCM [34]	0.449	0.552	0.893	0.169	0.033	0.479	0.572	0.780	0.152	0.039
TransformerMM [38]	0.455	0.552	0.838	0.170	1.667	0.483	0.577	0.769	0.153	1.857
Methods designed for CTMM										
CLSTERS [41]	0.443	0.551	0.922	0.173	0.043	0.470	0.563	0.805	0.154	0.048
SNet [12]	0.446	0.555	0.891	0.169	0.034	0.475	0.565	0.792	0.153	0.041
THMM [42]	0.461	0.562	0.815	0.165	0.041	0.486	0.583	0.767	0.148	0.045
DMM [15]	0.467	0.566	0.784	0.163	0.853	0.489	0.594	0.755	0.145	0.916
Our method										
LHMM	**0.516**	**0.613**	**0.670**	**0.126**	**0.032**	**0.547**	**0.667**	**0.641**	**0.124**	**0.037**
Improved	**10.49%**	**8.30%**	**14.54%**	**22.69%**	**3.03%**	**11.86%**	**12.28%**	**8.79%**	**15.09%**	**5.12%**

TABLE II
OVERALL PERFORMANCE.

- *STM* [8]. ST-Matching (STM in short) is a classical approach designed for low-sampling-rate GPS sequences, which takes the topological structure and the temporal constraints into account.
- *IVMM* [10]. It uses a voting strategy to describe the mutual influence between GPS points.
- *IFM* [32]. IF-Matching (IFM) fuses the surrounding speed to describe moving targets.
- *DeepMM* [37]. It takes LSTM-based seq2seq and attention models for sparse and noisy GPS trajectories.
- *MCM* [34]. It models map-matching as finding a common sub-sequence between the GPS trajectory and the potential routes.
- *TransformerMM* [38]. It takes Transformer instead of LSTM in the seq2seq.

The methods designed for CTMM include:

- *CLSTERS* [41]. It takes a series of calibration manners to smooth cellular trajectories.
- *SNet* [12]. SnapNet (SNet) combines digital map hints and a number of heuristics in the estimation process.
- *THMM* [42]. It designs the geometric, topological, and probabilistic characteristics for CTMM.
- *DMM* [15]. It is relied on the seq2seq model for CTMM, and adopts a reinforcement learning component to enhance the map-matcher.

B. (Q1) Overall Performance

In this section, we validate the superiority of LHMM on two datasets. Table II exhibits the results of our method's performance compared with two types of baselines, including the methods designed for GPS trajectories and CTMM.

Accuracy. According to the results, we note the following key observations. (1) The methods tailored for CTMM (e.g. DMM, THMM) outperform the methods designed for GPS trajectories (e.g. MCM, TransformerMM) since they customize a series of heuristic characteristics from the digital map to overcome high positioning errors. (2) The learning-based methods achieve better results than HMM-based methods on RMF, since the implicit knowledge provides more accurate information to locate the road where the trajectory point is located. (3) While in coarser-grained corridor level, HMM-based methods THMM and MCM achieve similar accuracy to DMM and TransformerMM respectively on CMF50 due to the HMM's high stability.

The LHMM achieves the best accuracy on all metrics. Taking the CMF50 as an example, LHMM achieves 23.63% and 22.69% accuracy gains compared to the strongest HMM-based model THMM and the strongest seq2seq-based model DMM respectively. Such huge improvements demonstrate the effectiveness of combining the HMM and learned knowledge, which integrates the robustness of the HMM and is enhanced by the implicit features to guide the path-finding process.

The possible reasons for the improved accuracy are as follows: (1) LHMM fully captures the multi-relational knowledge useful for the CTMM task, while DMM simply transforms the cell tower to the embedding with auto-encoder and RNN, ignoring abundant semantic information. (2) LHMM considers trajectory context to eliminate ambiguity and embeds learned knowledge into observation and transition probabilities, while other HMM-based approaches only take explicit features to evaluate roads and paths. (3) The shortcut structure provides critical skipping edges on the candidate graph to alleviate the impact of unqualified candidate road sets. With the above components, LHMM achieves state-of-the-art performances. We will further decompose the performance of these components in Section V-C.

Running efficiency. We also evaluate the running efficiency. From Table II, we observe that the HMM-based method (e.g. MCM with 0.033 seconds/per trajectory) is fast than the seq2seq-based method (e.g. DMM with 0.853 seconds/per trajectory). This is because seq2seq methods are limited by the weak parallelism inference of RNN and suffer from large matrix operations. In contrast, the observation and transition probabilities of the HMM can be calculated in parallel, and the HMM path-finding process only requires simple numerical calculations using dynamic programming. We note that

Dataset	Variant	Precision	CMF50	HR
Hangzhou	**LHMM**	**0.516**	**0.126**	**0.953**
	LHMM-E	0.457	0.142	0.931
	LHMM-H	0.489	0.136	0.942
	LHMM-O	0.428	0.178	0.920
	LHMM-T	0.472	0.155	0.926
	LHMM-S	0.484	0.140	0.937
	STM	0.388	0.225	0.874
	STM+S	0.405	0.189	0.911
Xiamen	**LHMM**	**0.545**	**0.125**	**0.965**
	LHMM-E	0.494	0.144	0.938
	LHMM-H	0.517	0.142	0.942
	LHMM-O	0.462	0.158	0.931
	LHMM-T	0.524	0.135	0.952
	LHMM-S	0.516	0.139	0.944
	STM	0.411	0.198	0.882
	STM+S	0.432	0.170	0.915

TABLE III
ABLATIONS RESULTS.

LHMM achieves the fastest matching speed (0.032 seconds/per trajectory). The modeling of implicit features for $P_O(\cdot)$ and $P_T(\cdot)$ improve the ability to locate the traveled locations under high positioning errors, thus allowing LHMM to use a smaller number of candidate roads k for faster matching.

C. (Q2) Ablation Results

We further conduct ablation tests to investigate the effect of all components of LHMM. Table III reports the results of all variant models.

Effect of Het-Encoder. To investigate the effectiveness of Het-Encoder, we design the following variants:

- *LHMM-E*. We replace the graph-based encoder layer with an MLP embedding layer.
- *LHMM-H*. we replace the heterogeneous graph neural network in the Het-Graph Encoder with a homogeneous graph neural network (e.g. GCNs [48]).

From the ablation results, we observe that the performance of LHMM-E falls behind that of LHMM-H, which proves that the multi-relational graph provides useful information for CTMM, and the graph-based encoder is better than the MLP embedding layer to capture the graph's abundant semantic information. The performance of LHMM-H falls behind that of LHMM, indicating that these multi-relations tailored for CTMM need to be embedded in a balanced way, so that the heterogeneity of the multi-relational graph can be fully extracted.

Effect of Learned Observation and Transition Probabilities.. To investigate the influence of implicit features for $P_O(\cdot)$ and $P_T(\cdot)$, we design the following variants:

- *LHMM-O*. We remove the implicit point-road correlation in the observation probability.
- *LHMM-T*. We remove the implicit trajectory-path correlation in the transition probability.

By integrating the implicit features, the performance increase is remarkable (e.g. from LHMM-O 0.178 and LHMM-T 0.155 to LHMM 0.126 on CMF50). This proves the superiority of the learned observation and transition probabilities that are empowered by implicit knowledge.

Effect of Shortcut. To investigate the effectiveness of the shortcut structure, we design the following variants:

- *LHMM-S*. We remove the shortcut structure from LHMM.
- *STM*, *STM+S*. We add the shortcut structure into other HMM-based methods.

From the results of LHMM-S, we observe that the shortcut structure obviously improves the performance from LHMM-S 0.140 to LHMM 0.125 on CMF50, which indicates that the shortcut remedies the impact of unqualified candidate road sets. Comparing the STM and STM+S, we observe that the shortcut is a general component useful for all HMM-based methods, boosting its hitting ratio from 0.874 to 0.911.

D. (Q3) Robustness Analysis

We evaluate the performance against the high positioning error and the sparse sampling rate under different attributes of input cellular trajectories.

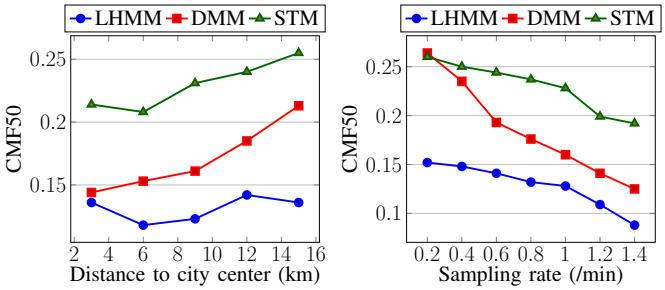
Impact of area of the cellular trajectory. We first evaluate varying areas of cellular trajectories. As we all know, the rural area has few and sparse cell towers, in contrast, the urban area has well-established infrastructure and high-density cell tower distribution, so that the distance to the city center can indirectly reflect the positioning error and the sparsity of the cellular trajectory. We divide the dataset into 5 levels in terms of the distance to the city center. Figure 7(a) reports that LHMM achieves stable performance in both urban and rural areas. In rural areas, although the density of cell towers is sparse, the alternative paths are limited. LHMM utilizes physical features to achieve comparable matching results. While the accuracy of DMM is greatly affected since the seq2seq model suffers from insufficient coverage of historical trajectories. In urban areas, the road network is dense and complex. LHMM locates candidate roads with context-aware knowledge in complex environments, avoiding local optimal matching.

Impact of sampling rate. The sampling density directly influences the matching difficulty. We test LHMM at 7 levels of sampling rates, including 0.2, 0.4, 0.6, 0.8, 1, 1.2, and 1.4 sampling per minute. From Figure 7(b), we observe that as the sampling rate decreases, the matching difficulty also increases continuously. Compared with DMM and STM, LHMM is least affected by the decrease of sampling rate relying on trajectory context and explicit features, while ordinary HMM algorithms cannot combat such sparse cellular trajectories using spatial features only. For the seq2seq-based model DMM, the extremely sparse sampling rate is a fatal blow to the encoder-decoder architecture, since the encoder cannot provide enough guidance for the decoder to fill the long and uncertain path.

E. (Q4) Parametric Analysis

We evaluate the influence of main hyper-parameters and different data scales on LHMM.

Impact of candidate number. We further explore the impacts of the number of candidate road segments k. From Figure 8, we observe that as the number k of candidate roads increases from 10 to 60 for a point, the matching accuracy does not continuous improvement. Because more candidate road segments mean more irrelevant roads, which bring more noise interference, making map-matching more difficult.



(a) Impact of area (b) Impact of sampling rate

Fig. 7. Impact of varying input cellular trajectories

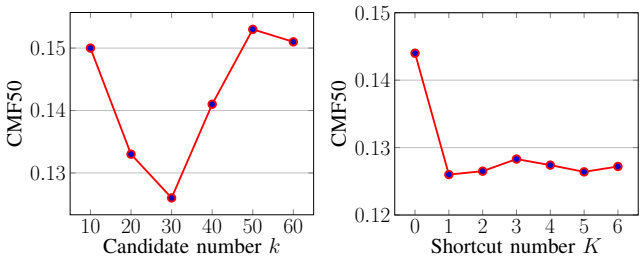
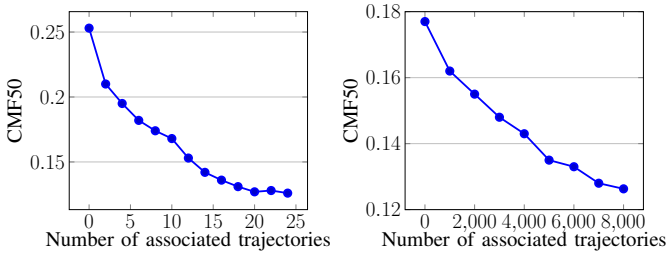


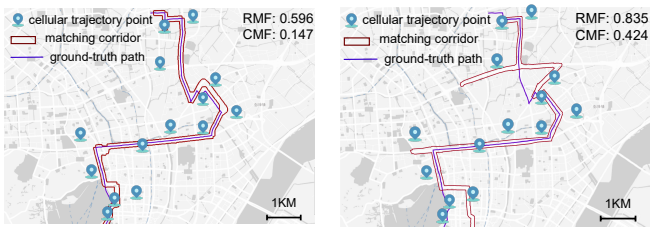
Fig. 8. Impact of candidate number

Fig. 9. Impact of shortcut number



(a) Impact of varying data scales to one cell tower (b) Impact of varying data scales to all cell towers

Fig. 10. Impact of data scale



(a) Matching by LHMM (b) Matching by DMM

Fig. 11. Real cases on challenging situations

Impact of shortcut number. We explore the impacts of the number of shortcuts. For a candidate road segment, we test the K shortcuts with its best one-hop predecessors according to Equation (20). Figure 9 reports that from no shortcut to one shortcut brings a significant boost, since the candidate road sets have chances to be skipped. While there is no steady improvement for more shortcuts according to experiments, this indicates that one shortcut is sufficient to remedy most of the impact of unqualified candidate road sets.

Impact of Data Scale. The data scale of historical trajectories directly affects the performance of the neural network. We test the impact of varying data scales for one cell tower and all cell towers. For one cell tower, Figure 10(a) reports that with the increase of the trajectories that interact with the cell tower, LHMM has more accurate in locating its traveled road. After about 20 associated trajectories, the CMF50 no longer decreases. The possible reason might be that more trajectories follow the same driving pattern, which does not bring more effective localization information. For all cell towers, Figure 10(b) reports that as the number of historical trajectories increases, the embedding quality and locating ability are also improved. More and more areas of the city are covered and explored by historical trajectories, enabling continuous improvement in accuracy.

F. (Q5) Visualization of LHMM

To intuitively observe the effectiveness of our method, we exhibit a real and typical case map matched by LHMM and the strongest baseline DMM. Figure 11(a) reports the performance of LHMM. In this challenging situation, LHMM still maintains high accuracy and robustness (CMF 0.147), which indicates that LHMM overcomes the interference of high positioning errors. In contrast, the DMM does not match properly (CMF 0.424) in sparse and noisy situations as shown in Figure 11(b). The reason is that DMM suffers from error propagation of RNN. Once a road is incorrectly matched, it will mislead the generation of subsequent roads. LHMM has a powerful correction capability of the HMM, and the learned $P_O(\cdot)$ and $P_T(\cdot)$ enable us to accurately locate the traveled positions.

ACKNOWLEDGEMENTS

This work is supported by National Natural Science Foundation of China (Grant No. 62102277, 62272334, 62072125, and 61872258), Natural Science Foundation of Jiangsu Province, China (Grant Nos. BK20210703 and BK20211307), the major project of natural science research in universities of Jiangsu province under grant number 20KJA520005, the priority academic program development of Jiangsu higher education institutions, young scholar program of Cyrus Tang Foundation, and the research work described in this paper is partially conducted in the JC STEM Lab of Data Science Foundations funded by The Hong Kong Jockey Club Charities Trust.

VI. CONCLUSION

In this paper, we have developed a learning-enhanced HMM map-matching approach for cellular trajectories. A representation learning component is designed to fully capture multi-relational information tailored for the CTMM task. A learned observation probability captures the implicit context-aware correlation between roads and points for better positioning denoising, and a learned transition probability models the hidden relevance between moving paths and trajectories. These two probabilities then guide the path-finding process on an improved candidate graph. Extensive experiments on two large real-world datasets show that our approach achieves high accuracy and robustness for the CTMM task.

REFERENCES

[1] S. Prasad, J. Rachna, O. Khalaf, and D.-N. Le, "Map matching algorithm: Real time location tracking for smart security application," *Telecommun Radio Eng*, vol. 79, no. 13, 2020.

[2] S. Brakatsoulas, D. Pfoser, R. Salas, and C. Wenk, "On map-matching vehicle tracking data," in *VLDB*, 2005, pp. 853–864.

[3] A. Fang, X. Peng, J. Zhou, and L. Tang, "Research on the map-matching and spatial-temporal visualization of expressway traffic accident information," in *ICITE*. IEEE, 2018, pp. 23–27.

[4] R. He, J. Cao, L. Zhang, and D. Lee, "Statistical enrichment models for activity inference from imprecise location data," in *IEEE INFOCOM*, 2019, pp. 946–954.

[5] E. Thuillier, L. Moalic, S. Lamrous, and A. Caminada, "Clustering weekly patterns of human mobility through mobile phone data," *IEEE Trans. Mob. Comput.*, vol. 17, no. 4, pp. 817–830, 2017.

[6] R. Becker, R. Cáceres, K. Hanson, S. Isaacman, J. M. Loh, M. Martonosi, J. Rowland, S. Urbanek, A. Varshavsky, and C. Volinsky, "Human mobility characterization from cellular network data," *Commun. ACM*, vol. 56, no. 1, pp. 74–82, 2013.

[7] M. T. Rahman, R. T. Khan, M. R. Khandaker, M. Sellathurai, and M. S. A. Salan, "An automated contact tracing approach for controlling covid-19 spread based on geolocation data from mobile cellular networks," *IEEE Access*, vol. 8, pp. 213 554–213 565, 2020.

[8] Y. Lou, C. Zhang, Y. Zheng, X. Xie, W. Wang, and Y. Huang, "Map-matching for low-sampling-rate gps trajectories," in *ACM SIGSPATIAL GIS*, 2009, pp. 352–361.

[9] P. Newson and J. Krumm, "Hidden markov map matching through noise and sparseness," in *ACM SIGSPATIAL GIS*, 2009, pp. 336–343.

[10] J. Yuan, Y. Zheng, C. Zhang, X. Xie, and G.-Z. Sun, "An interactive-voting based map matching algorithm," in *MDM*, 2010, pp. 43–52.

[11] C. Yang and G. Gidofalvi, "Fast map matching, an algorithm integrating hidden markov model with precomputation," *GIS*, vol. 32, no. 3, pp. 547 – 570, 2018.

[12] R. Mohamed, H. Aly, and M. Youssef, "Accurate real-time map matching for challenging environments," *IEEE TITS*, vol. 18, no. 4, pp. 847–857, 2016.

[13] A. Viel, D. Gubiani, P. Gallo, A. Montanari, A. Dalla Torre, F. Pittino, and C. Marshall, "Map matching with sparse cellular fingerprint observations," in *UPINLBS*. IEEE, 2018, pp. 1–10.

[14] A. Dalla Torre, P. Gallo, D. Gubiani, C. Marshall, A. Montanari, F. Pittino, and A. Viel, "A map-matching algorithm dealing with sparse cellular fingerprint observations," *Geo Spat Inf Sci*, vol. 22, no. 2, pp. 89–106, 2019.

[15] Z. Shen, W. Du, X. Zhao, and J. Zou, "Dmm: fast map matching for cellular data," in *ACM MobiCom*, 2020, pp. 1–14.

[16] L. Yu, W. Zhang, J. Wang, and Y. Yu, "Seqgan: Sequence generative adversarial nets with policy gradient," in *AAAI*, vol. 31, no. 1, 2017.

[17] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, "Scheduled sampling for sequence prediction with recurrent neural networks," *NIPS*, vol. 28, 2015.

[18] S. Jiang and M. de Rijke, "Why are sequence-to-sequence models so dull? understanding the low-diversity problem of chatbots," *arXiv*, 2018.

[19] P. Chao, Y. Xu, W. Hua, and X. Zhou, "A survey on map-matching algorithms," in *ADC*. Springer, 2020, pp. 121–133.

[20] L. Jiang, C. Chen, C. Chen, H. Huang, and B. Guo, "From driving trajectories to driving paths: a survey on map-matching algorithms," *CCF Transactions on Pervasive Computing and Interaction*, pp. 1–16, 2022.

[21] M. A. Quddus, W. Y. Ochieng, and R. B. Noland, "Current map-matching algorithms for transport applications: State-of-the art and future research directions," *Transportation research part c: Emerging technologies*, vol. 15, no. 5, pp. 312–328, 2007.

[22] C. E. White, D. Bernstein, and A. L. Kornhauser, "Some map matching algorithms for personal navigation assistants," *Transportation research part c: emerging technologies*, vol. 8, no. 1-6, pp. 91–108, 2000.

[23] D. Bernstein, A. Kornhauser *et al.*, "An introduction to map matching for personal navigation assistants," 1996.

[24] A. Mosig and M. Clausen, "Approximately matching polygonal curves with respect to the fréchet distance," *Comput. Geom. Theory Appl.*, vol. 30, no. 2, pp. 113–127, 2005.

[25] R. R. Joshi, "A new approach to map matching for in-vehicle navigation systems: the rotational variation metric," in *ITSC*. IEEE, 2001, pp. 33–38.

[26] M. A. Quddus, W. Y. Ochieng, L. Zhao, and R. B. Noland, "A general map matching algorithm for transport telematics applications," *GPS solutions*, vol. 7, no. 3, pp. 157–167, 2003.

[27] M. A. Quddus, R. B. Noland, and W. Y. Ochieng, "Validation of map matching algorithms using high precision positioning with gps," *The Journal of Navigation*, vol. 58, no. 2, pp. 257–271, 2005.

[28] F. Abdallah, G. Nassreddine, and T. Denoeux, "A multiple-hypothesis map-matching method suitable for weighted and box-shaped state estimation for localization," *IEEE TITS*, vol. 12, no. 4, pp. 1495–1510, 2011.

[29] D. Obradovic, H. Lenz, and M. Schupfner, "Fusion of map and sensor data in a modern car navigation system," *VLSI*, vol. 45, no. 1, pp. 111–122, 2006.

[30] M. A. Quddus, R. B. Noland, and W. Y. Ochieng, "A high accuracy fuzzy logic based map matching algorithm for road transport," *J. Intell. Transport. Syst.*, vol. 10, no. 3, pp. 103–115, 2006.

[31] S. Syed and M. E. Cannon, "Fuzzy logic based-map matching algorithm for vehicle navigation system in urban canyons," in *Proceedings of the 2004 National Technical Meeting of the Institute of Navigation*, 2004, pp. 982–993.

[32] G. Hu, J. Shao, F. Liu, Y. Wang, and H. T. Shen, "If-matching: Towards accurate map-matching with information fusion," *IEEE TKDE*, vol. 29, no. 1, pp. 114–127, 2016.

[33] A. Thiagarajan, L. Ravindranath, H. Balakrishnan, S. Madden, and L. Girod, "Accurate,{Low-Energy} trajectory mapping for mobile devices," in *NSDI*, 2011.

[34] W. Li, Y. Wang, D. Li, and X. Xu, "Mcm: A robust map matching method by tracking multiple road candidates," in *ICAAM*. Springer, 2022, pp. 231–243.

[35] Y. Zhang and X. Sui, "Rcivmm: A route choice-based interactive voting map matching approach for complex urban road networks," *TBD*, 2021.

[36] B. Liang, T. Wang, S. Li, W. Chen, H. Li, and K. Lei, "Online learning for accurate real-time map matching," in *PAKDD*, 2016, pp. 67–78.

[37] J. Feng, Y. Li, K. Zhao, Z. Xu, T. Xia, J. Zhang, and D. Jin, "Deepmm: deep learning based map matching with data augmentation," *IEEE TMC*, vol. 21, no. 7, 2022.

[38] Z. Jin, J. Kim, H. Yeo, and S. Choi, "Transformer-based map matching model with limited ground-truth data using transfer-learning approach," *arXiv*, 2021.

[39] C. Chen, X. Zhang, Y. Dong, H. Dong, and F. Rao, "Map-matching based on driver behavior model and massive trajectories," in *IEEE ITSC*, 2014, pp. 2817–2822.

[40] Y. Zhang and Y. He, "An advanced interactive-voting based map matching algorithm for low-sampling-rate gps data," in *ICNSC*. IEEE, 2018, pp. 1–7.

[41] H. Wu, W. Sun, B. Zheng, L. Yang, and W. Zhou, "Clsters: A general system for reducing errors of trajectories under challenging localization situations," *IMWUT*, vol. 1, no. 3, pp. 1–28, 2017.

[42] R. Chen, S. Yuan, C. Ma, H. Zhao, and Z. Feng, "Tailored hidden markov model: A tailored hidden markov model optimized for cellular-based map matching," *IEEE TIE*, vol. 69, no. 12, pp. 13 818–13 827, 2022.

[43] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *ESWC*. Springer, 2018, pp. 593–607.

[44] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *ICML*. PMLR, 2017, pp. 1321–1330.

[45] R. Müller, S. Kornblith, and G. E. Hinton, "When does label smoothing help?" *NeurIPS*, vol. 32, 2019.

[46] G. D. Forney, "The viterbi algorithm," *Proceedings of the IEEE*, vol. 61, no. 3, pp. 268–278, 1973.

[47] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *ICML*. PMLR, 2017, pp. 1263–1272.

[48] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv*, 2016.

[49] M. Srivatsa, R. Ganti, J. Wang, and V. Kolar, "Map matching: Facts and myths," in *ACM SIGSPATIAL GIS*, 2013, pp. 484–487.

[50] Y.-L. Hsueh, H.-C. Chen, and W.-J. Huang, "A hidden markov model-based map-matching approach for low-sampling-rate gps trajectories," in *IEEE SC2*, 2017, pp. 271–274.

[51] W. Wang, W. Zhang, S. Liu, Q. Liu, B. Zhang, L. Lin, and H. Zha, "Beyond clicks: Modeling multi-relational item graph for session-based target behavior prediction," in *WWW*, 2020, pp. 3056–3062.

[52] M. Hashemi and H. A. Karimi, "A weight-based map-matching algorithm for vehicle navigation in complex urban networks," *J. Intell. Transp. Syst. Technol. Plann. Oper.*, vol. 20, no. 6, pp. 573–590, 2016.

[53] X. Liu, K. Liu, M. Li, and F. Lu, "A st-crf map-matching method for low-frequency floating car data," *IEEE TITS*, vol. 18, no. 5, pp. 1241–1254, 2016.

[54] K. Par and O. Tosun, "Parallelization of particle filter based localization and map matching algorithms on multicore/manycore architectures," in *IEEE IV*, 2011, pp. 820–826.

[55] R. R. Joshi, "A new approach to map matching for in-vehicle navigation systems: the rotational variation metric," in *IEEE ITSC*, 2001, pp. 33–38.

[56] C. Feijoo, J. Ramos, and F. Perez, "A system for fleet management using differential gps and vhf data transmission mobile networks," in *IEEE VNIS*, 1993, pp. 445–448.

[57] M. Chen, Y. Liu, and X. Yu, "Predicting next locations with object clustering and trajectory clustering," in *PAKDD*, 2015, pp. 344–356.

[58] J. A. Alvarez-Garcia, J. A. Ortega, L. Gonzalez-Abril, and F. Velasco, "Trip destination prediction based on past gps log using a hidden markov model," *Expert Syst. Appl.*, vol. 37, no. 12, pp. 8166–8171, 2010.

[59] R. Kühne, R.-P. Scbäfer, J. Mikat, K.-U. Thiessenhusen, U. Böttger, and S. Lorkowski, "New approaches for traffic management in metropolitan areas," *IFAC Proc. Vol.*, vol. 36, no. 14, pp. 209–214, 2003.

[60] L. Lv, M. Chen, Y. Liu, and X. Yu, "A plane moving average algorithm for short-term traffic flow prediction," in *PAKDD*, 2015, pp. 357–369.

[61] L. X. Pang, S. Chawla, W. Liu, and Y. Zheng, "On detection of emerging anomalous traffic patterns using gps data," *Data Knowl. Eng.*, vol. 87, pp. 357–373, 2013.

[62] H. Gonzalez, J. Han, X. Li, M. Myslinska, and J. P. Sondag, "Adaptive fastest path computation on a road network: a traffic mining approach," in *VLDB*, 2007, pp. 794–805.

[63] L.-Y. Wei, Y. Zheng, and W.-C. Peng, "Constructing popular routes from uncertain trajectories," in *ACM SIGKDD*, 2012, pp. 195–203.

[64] J. Yuan, Y. Zheng, X. Xie, and G. Sun, "T-drive: Enhancing driving directions with taxi drivers' intelligence," *IEEE TKDE*, vol. 25, no. 1, pp. 220–232, 2011.

[65] Z. Shen, W. Du, X. Zhao, and J. Zou, "Retrieving similar trajectories from cellular data at city scale," *arXiv*, 2019.

[66] S. Shang, L. Chen, Z. Wei, C. S. Jensen, K. Zheng, and P. Kalnis, "Parallel trajectory similarity joins in spatial networks," *VLDB*, vol. 27, no. 3, pp. 395–420, 2018.

[67] C. Chen, Y. Ding, X. Xie, S. Zhang, Z. Wang, and L. Feng, "Trajcompressor: An online map-matching-based trajectory compression framework leveraging vehicle heading direction and change," *TITS*, vol. 21, no. 5, pp. 2012–2028, 2019.

[68] A. Karatzoglou, A. Jablonski, and M. Beigl, "A seq2seq learning approach for modeling semantic trajectories and predicting the next location," in *GIS*, 2018, pp. 528–531.

[69] R. H. Zhang, Q. Liu, A. X. Fan, H. Ji, D. Zeng, F. Cheng, D. Kawahara, and S. Kurohashi, "Minimize exposure bias of seq2seq models in joint entity and relation extraction," *arXiv preprint arXiv:2009.07503*, 2020.

[70] M. Hashemi and H. A. Karimi, "A critical review of real-time map-matching algorithms: Current issues and future directions," *Comput. Environ. Urban Syst.*, vol. 48, pp. 153–165, 2014.

[71] B. Custers, W. Meulemans, M. Roeloffzen, B. Speckmann, and K. Verbeek, "Physically consistent map matching," in *GIS*, 2022, pp. 1–4.