Trajectory-User Linking using Higher-order Mobility Flow Representations

Mahmoud Alsaeed
York University
Toronto, Canada
mahmoud2@eecs.yorku.ca

Ameeta Agrawal
Portland State University
Portland, USA
ameeta@pdx.edu

Manos Papagelis
York University
Toronto, Canada
papaggel@eecs.yorku.ca

***Abstract*—Trajectory user linking (TUL) is a problem in trajectory classification that links anonymous trajectories to the users who generated them. TUL has various uses such as identity verification, personalized recommendation, epidemiological monitoring, and threat assessments. A major challenge in TUL modeling is sparse data. Previous TUL research heavily relies on sequence-to-sequence models such as RNNs and LSTMs, with trajectory segmentation to combat sparsity, but segmentation does not sufficiently address the issue and existing models often ignore data skewness, resulting in poor precision and performance. To address these problems, we present TULHOR, a TUL model inspired by BERT, a popular language representation model. One of TULHOR's innovations is the use of higher-order mobility flow data representations enabled by geographic area tessellation. This allows the model to alleviate the sparsity problem and also to generalize better. TULHOR consists of a spatial embedding layer, a spatial-temporal embedding layer and an encoder layer, which encodes properties and learns a rich trajectory representation. It is trained in two steps, first using a masked language modeling task to learn general embeddings, then fine-tuned using a balanced cross-entropy loss to make predictions while handling imbalanced data. Experiments on real-life mobility data show TULHOR's effectiveness as compared to current state-of-the-art models.**

***Index Terms*—trajectory classification, trajectory-user linking, trajectory representation learning, machine learning**

I. Introduction

Motivation. Location-based services (LBS) are systems that provide services to users based on their geographic location. The location information is typically obtained from a GPS-enabled device (such as a cell phone) and is used to provide services such as maps, directions, local search, and location-based advertising. Examples of LBS include ride-hailing apps, food delivery apps, and weather apps, to name a few. LBS can provide more personalized and relevant experiences to users by analyzing user trajectory data, a collection of geographic data points that describe the moving patterns of a user or vehicle, over a period of time. For instance, they can be used to provide location-based recommendations to users based on their past movements and interests, or to predict future traffic conditions for commuting to home or work location. They can also be used in marketing for optimizing the placement and timing of advertisements based on the trajectory of users in a particular area, or for customer segmentation by grouping customers together based on their movement patterns and preferences.

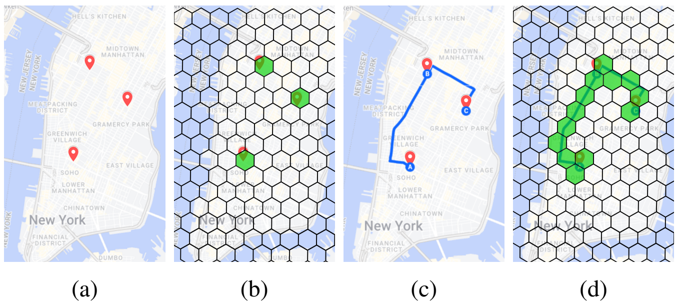

Fig. 1: An illustrative example that shows how sample check-in data from FOURSQUARE-NYC (**a**) can be abstracted to higher-order areas (**b**), and how the sequence of check-ins that infer a trajectory (**c**) can be abstracted to higher-order mobility flow (**d**). We propose TULHOR, a spatiotemporal BERT-based model that learns higher-order mobility flow representations to solve the trajectory-user linking (TUL) problem.

Problem of interest. Associating a particular trajectory with the correct user is known as the *trajectory-user linking* (TUL) problem. In principle, the TUL problem is a trajectory classification problem that aims at classifying anonymous trajectories to the users who generated them. Addressing this problem is essential for LBS because it ensures the privacy and security of user data and enables the provision of accurate and trustworthy location-based services. The main idea in addressing the TUL problem is to examine human mobility patterns to gain insight into the ways in which people move and interact with their surroundings. This analysis can encompass various aspects of human movement, including daily routines, commuting habits, event data (check-ins), and general movement patterns within a specific geographic region.

Current approaches. Current methods to address the TUL problem can be broadly divided into two categories: (i) classical machine learning (ML)-based methods, and (ii) deep learning-based methods. The classical ML-based methods involve traditional trajectory similarity techniques. The most prominent ones are the longest common sub-sequence, dynamic time warping [1], and NeuTraj [2]. These techniques are used to link a user to a trajectory by comparing the similarities between identified and unidentified trajectories. The deep learning-based methods involve modeling the trajectories by learning rich low-level spatial-temporal embeddings of points of interest (POIs), then linking them to the corresponding user based on the spatial-temporal patterns observed in their trajectories. Most existing solutions apply seq2seq models like RNN and LSTM to learn the transition pattern between POIs in trajectories [3]. Miao et al. [4] use LSTM and BiLSTM with attention to learn intra-trajectory relationships. Zhou et al. [5] use RNN with variational auto encoding to learn the temporal pattern in trajectories; they also address the problem by constructing a spatial and a check-in graph, then combining them together and applying GNN to learn POI embeddings [6].

Limitations of current approaches. Despite the promising results of the current approaches, there are some major challenges of the TUL problem:

- *data quality*: trajectory data are typically of low quality, due to low accuracy and/or completeness;
- *data sparsity*: trajectory data are sparse, due to limited and/or missing data;
- *imbalanced data*: the distribution of trajectories across different users (classes) is unequal.

These challenges can hinder the accurate linking of trajectories to their corresponding users. Some of the existing works attempt to resolve the limited data issue by generating new trajectories from existing ones through trajectory segmentation and augmentation methods. While this approach increases the number of available training samples, and can potentially improve model performance, it does not resolve the data quality and sparsity problems. In addition, existing works have overlooked the imbalanced data issue, which can be problematic when developing machine learning models, as it can lead to biased or inaccurate results.

Our approach & contributions. Our approach to the problem aims at addressing some of the main challenges of the TUL problem. The key idea is that we extrapolate *location data* to *higher-order mobility flow data*. Mobility flow refers to the movement of people from one location to another over time. This is in contrast to existing models that incorporate spatial features into learned models by either using pre-trained POI embeddings, or by directly modeling the physical distance between POIs. Note that physical distance does not capture the mobility flow dynamics, as it does not follow the mobility constraints imposed by the map. In addition, instead of defining mobility flow at the granularity of trajectory data points (which are sparse), we learn *higher-order mobility flow representations* based on a regular tessellation of the observation area (map) in hexagons. These high-order representations of trajectories are used to address the TUL problem. A summary of our contributions is provided below:

- We present a method that given location data (as data points), generates higher-order mobility flow data. These data represent sequences of regular hexagons defined on a tessellated observation area (map). The method generalizes and can generate mobility flow data at different levels of tessellation granularity.
- We propose TULHOR (**t**rajectory-**u**ser **l**inking using **h**igher-**o**rder **r**epresentations), a deep learning model based on a spatial-temporal variation of the Bidirectional Encoder Representation from Transformers (BERT) [7] that addresses the TUL problem by learning and utilizing higher-order mobility flow representations.
- We address the problem of imbalanced data that is important for developing fair and accurate TUL models that can effectively handle real-world data with unequal class distributions.
- We demonstrate empirically that our proposed model TULHOR outperforms the state-of-the-art methods and other sensible baselines. We also perform an ablation study and a parameter sensitivity analysis that demonstrate the impact of the different embedding components in the accuracy performance of TULHOR.
- We make our source code publicly available to encourage the reproducibility of our work[1].

TABLE I: Summary of notations

SYMBOL	DESCRIPTION
$\mathcal{M}$	The map of a geographic area
$\mathcal{P}$	A set of points of interest in $\mathcal{M}$
u	User ID
l	Location ID
t	Timestamp
$\langle x, y\rangle$	A tuple of latitude and longitude
r	A check-in record represented by a quadruplet $(u, l, t, \langle x, y\rangle)$
Tr	A check-in trajectory represented as $Tr = \{r_1, r_2, ..., r_m\}$, where r_i is the ith check-in
$\mathcal{G}$	$G = \{g_1, g_2, g_3, ..., g_n\}$ is an hexagonal tessellation of $\mathcal{M}$, where g_i is the ith grid cell ID

Paper organization. The remainder of the paper is organized as follows. Section II presents preliminaries and a formal definition of the problem. Section III presents our method for generating higher-order mobility flow data, and Section IV presents our TULHOR model for addressing the TUL problem. We describe the experimental setup, and present and discuss the results in section V. We review related work in section VI and conclude in section VII.

II. Preliminaries & Problem Definition

In this section, we briefly introduce some definitions and notations (a summary is provided in Table I). Then we formally define the trajectory-user linking problem.

Definition 1: (**Map**) Let $\mathcal{M}$ be a map over a predefined, finite, and continuous geographical area.

Definition 2: (**POI**) Let $\mathcal{P} = \{p_1, p_2, \ldots, p_{|P|}\}$ be a set of points of interests on a map $\mathcal{M}$.

Definition 3: (**Visits** or **Check-ins**) In location-based services, a visit or check-in of a person to a location or place at a particular time is a record represented by a quadruplet $r = (u, l, t, \langle x, y\rangle)$, where u denotes the user, l denotes the location ID, t stands for the time of the visit, and the tuple $\langle x, y\rangle$ represents the latitude and longitude of the visited location. We represent the set of all visits or check-ins by R. In this research, we use the term visit or check-in interchangeably.

[1]https://github.com/theWonderBoy/TULHOR

Definition 4: (**Trajectory**) A temporarily ordered sequence of a user's visits to places (or check-ins), observed during a time period, can be used to describe a trajectory $Tr = \{r_1, r_2, ..., r_m\}$, where m represents the length of the trajectory. We represent the set of all trajectories by $\mathcal{T}$. Since every visit or check-in record relates to a specific point of interest $p \in \mathcal{P}$, a trajectory can also be represented by a sequence of points of interest $Tr = \{p_1, p_2, ..., p_m\}$, where p_i is a point of interest at the location l of the record r_i.

Problem Definition. TUL aims to link anonymous trajectories to the user who generates them. Let $\mathcal{T} = \{Tr_1, Tr_2, ..., Tr_n\}$ be the set of unlinked trajectories and $\mathcal{U} = \{u_1, u_2, u_3, .., u_c\}$ be the set of users who generate them, then TUL is defined as a multiclass classification problem:

$$\min_f \mathbb{E}[\mathcal{L}(f(Tr_i), u_i)] \text{ over } \mathcal{F}, \tag{1}$$

where $\mathcal{F}$ is the set of all classifiers in the hypothesis space, $\mathcal{L}(\cdot)$ is the loss between the predicted label $f(Tr_i) \in \mathcal{U}$ and the true label $u_i \in \mathcal{U}$ of trajectory Tr_i.

III. Generating Higher-order Mobility Flow Representations

In this section, we present details of our method for generating higher-order mobility flow data. We first present the key idea of our method. Then, we can expand on the details and provide explanation and examples to clarify the main points.

Key idea. The check-in data lacks information on the route traveled by a user between consecutive check-ins. However, the routes could provide additional context and show the *flow* of people in a city. We therefore calculate possible routes that connect the check-in locations, consisting of origin, destination, as well as intermediate check-ins (waypoints). We can estimate these routes using publicly available APIs[2]. While these routes are not representing the actual path a user followed, they can largely capture the common routes that connect specific check-in locations. In addition, mobility flow from a check-in to location to another one can also be further abstracted, by considering higher-order abstractions on the map. This include higher-order representations of the check-in locations and the routes connecting them. We achieve this by tessellating the map and translating check-in and route information using higher-order elements of the map's tessellation. The premise of these ideas is that the richer data will help our deep learning model to generalize and avoid overfitting to input training data.

We provide below a formal definition of the key concepts.

Definition 5: (**Grid**) Let $\mathcal{G} \in \{g_1, g_2, \ldots, g_n\}$ be a set of disjoint grid cells that fully tessellate map $\mathcal{M}$. All $g_k \in \mathcal{G}$ are polygons of the same size covering an area. Our work assumes that grid cells are *regular hexagons* that can fill a plane with no gaps, forming a hexagonal tiling (see example in Fig. 1b).

[2]https://developers.google.com/maps/documentation/directions

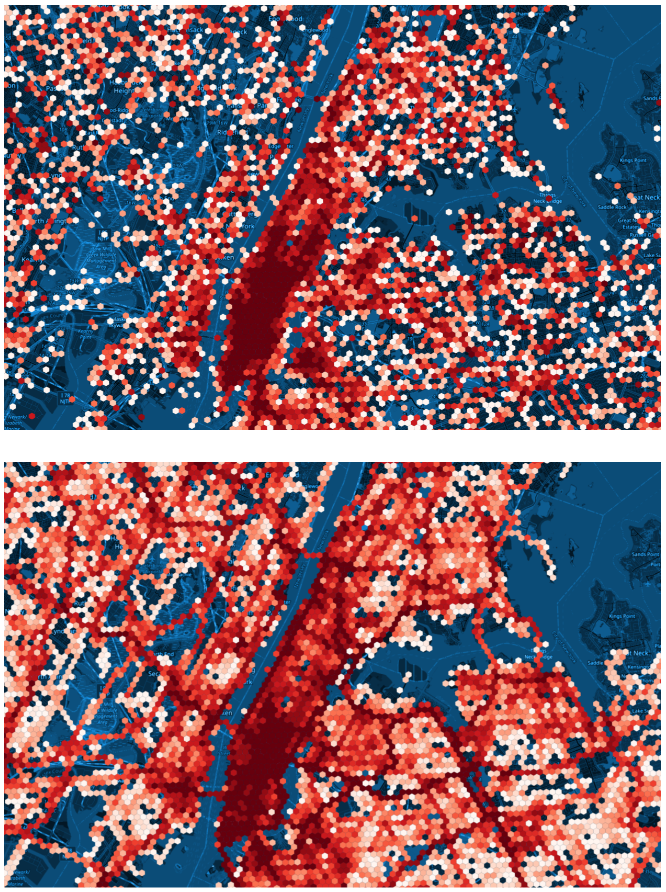

Fig. 2: An illustrative example (Foursquare-NYC) that shows how transforming higher-order check-in data (top) to higher-order mobility flow data (bottom) enriches the trajectory semantics and can help to address the TUL problem.

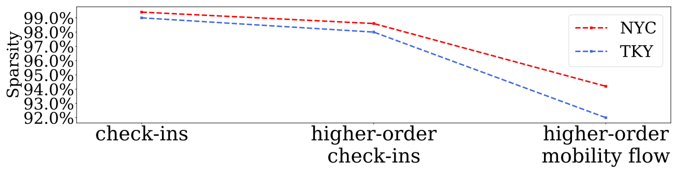

Fig. 3: Impact of higher-order abstraction on sparsity.

Note that the tessellation can happen at different level of resolution, by defining different sizes of the hexagons. The smaller the hexagon size, the higher the resolution.

The tessellation of the map allows to define higher-order semantics for chech-ins, trajectories and mobility flow.

Definition 6: (**Higher-order check-ins**) Since the map $\mathcal{M}$ is fully tessellated, for every visit/check-in and every $p \in \mathcal{P}$ there is a $g \in \mathcal{G}$, such that p is located in g.

Definition 7: (**Higher-order trajectories**) Since every $p \in \mathcal{P}$ belongs in a $g \in \mathcal{G}$, we can translate every trajectory $Tr = \{p_1, p_2, ..., p_m\}$ to a sequence of grid cells $Tr = \{g_1, g_2, ..., g_m\}$, where every $g_i \in \mathcal{G}$.

Definition 8: (**Higher-order mobility flow**) Given a higher-order trajectory $Tr = \{g_1, g_2, ..., g_m\}$, we can define a higher-order mobility flow as a new trajectory $Tr =$

$\{g_1, \langle \ldots \rangle, g_2, \langle \ldots \rangle, ..., \langle \ldots \rangle, g_m\}$, where every $\langle \ldots \rangle$ represents the sequence of grid cells $g \in G$ that need to be traversed between two sequential grid cells of the original trajectory.

Fig. 1 provides an example of how starting from check-in data as input to the problem, we can gradually generate higher-order mobility flow data. In addition, Fig. 2 provides an illustrative example that shows how transforming higher-order check-in data to higher-order mobility flow data enriches the trajectory semantics and can potentially help to improve on the TUL problem. In the figure, we can also observe how higher-order mobility data captures the city's road infrastructure and physical constraints. We also notice that higher-order mobility data exposes new densely visited areas that are missed with the higher-order check-ins. The method is general and can probably be useful in other problems and applications.

Rationale for using higher-order mobility flow. Check-in data is known to be very sparse. Sparsity is characterized by the percentage of zeros in the user-POI interaction matrix. In this matrix, an entry (i, j) is set to 1 if user i visited POI j. Other than being sparse, the data is also very skewed, with most locations having a few check-ins, and only a few locations having many check-ins. This sparsity level can affect the accuracy of modeling trajectories and result in a skewed data representation. Another major challenge of of check-in data is that embedding the longitude and latitude pairs of location data into a machine learning model is challenging due to their continuous nature. They also provide information at a very refined level. A more practical approach is to learn spatiotemporal embeddings at a higher level of granularity, such as at the level of grid cells (of a tessellated map).

We therefore propose to transform the check-ins and mobility flow data to a higher order to address this issue. In practice, this translates the sparse user-POI matrix into a denser user-grid cell matrix. In the user-grid cell matrix, an entry is equal to 1 if the user has visited any place in the corresponding grid cell. This expands the range of interactions from a single locations to multiple locations in a broader area, reducing sparsity. Additionally, multiple POIs can be located within the same cell, further decreasing sparsity. For example, consider the case of three users visiting only one POI each. In this case, the sparsity of the toy user-POI matrix is 66.6% since it has nine entries (3 users x 3 POIs), and only three of them are 1. If we project this to a higher dimension and assume two POIs fall in the same grid cell, then the toy user-grid cell matrix has six entries (3 users x 2 grid cells), and three of them are 1, reducing the sparsity to 50%. Fig. 3 presents the impact of higher-order representations on decreasing the sparsity on two benchmark datasets (FOURSQUARE-NYC and FOURSQUARE-TKY). We observe about a 1% decrease in sparsity when using higher-order check-ins, and more than a 5% decrease for the case of higher-order mobility flow.

IV. MODELING TRAJECTORY-USER LINKING

In this section, we present details of our spatiotemporal deep-learning model, TULHOR, that utilizes higher-order mobility flow data to accurately address the TUL problem.

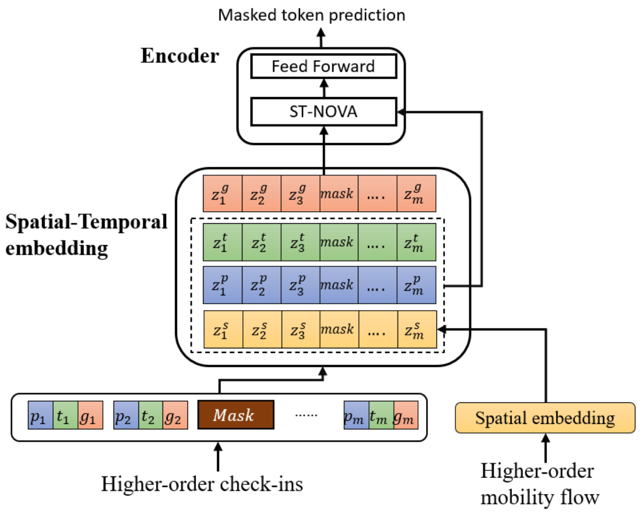
Masked token prediction
Encoder
Feed Forward
ST-NOVA
Spatial-Temporal embedding
z_1^g z_2^g z_3^g mask .... z_m^g
z_1^t z_2^t z_3^t mask .... z_m^t
z_1^p z_2^p z_3^p mask .... z_m^p
z_1^s z_2^s z_3^s mask .... z_m^s
p_1 t_1 g_1 p_2 t_2 g_2 Mask ...... p_m t_m g_m
Spatial embedding
Higher-order check-ins
Higher-order mobility flow

Fig. 4: High-level architecture of TULHOR.

TULHOR is composed of three components: (**A**) a spatial embedding layer, (**B**) a spatial-temporal embedding layer, and (**C**) an encoder (see Fig. 4). TULHOR is based on a Transformer architecture, specifically BERT [7], and uses a masked language modeling task to generate contextual embeddings. TULHOR also benefits from the self-attention mechanism of the Transformer architecture, making it more powerful and efficient than RNN or LSTM models. Unlike BERT, which is trained on sentences, TULHOR is trained on higher-order sequences of check-ins. We also provide information about (**D**) pre-training, and (**E**) fine-tuning the model.

A. *TULHOR's Spatial Embedding Layer*

To learn the spatial relationship of the grid cells, we first construct a graph that captures the spatial proximity of grid cells. Then, we use the higher-order mobility flow data to capture the semantic relationship of grid cells, similarly to the approach followed in [8].

Constructing a hexagon-lattice graph. Given a grid $\mathcal{G}$, we construct an undirected, unweighted graph $G = (V, E)$ of V nodes and E edges, where a node $u \in V$ represents a grid cell $g \in \mathcal{G}$ and an edge $e_{(u,v)} \in E$ indicates that there is a movement from $u \in V$ to $v \in V$ in the mobility flow data. We call this graph a *hexagon-lattice graph* because its nodes and edges are artifacts of a hexagonal tiling of a map.

Learning node representations of the hexagon-lattice graph. In this step, we employ the `node2vec` model [9] to learn the node representations of the grid cells modeled in the hexagon-lattice graph. The `node2vec` model is based on random walks on a graph to learn node representations. Instead of random walks, we use the high-order mobility flow data to represent walks on the graph (i.e., each high-order trajectory represents a walk on the hexagon-lattice graph). By employing these walks we end up learning the semantic relationships between different grid cells on the map. These relationships reflect real-life connections between geographic areas, including constraints imposed by the map (e.g., bridges) and do not solely capture physical proximity. As a result, we learn spatial representations of grid cells.

B. TULHOR's Spatiotemporal Embedding Layer

The spatial-temporal embedding layer converts sparse one-hot encodings of check-in components (grid cells, POI, and timestamps) into a dense representation. POI information is included to differentiate mobility patterns that traverse the same grid cell sequence. For instance, Alice and Bob studying at the same university but in different departments, would have similar grid cell movements but varying POI interactions. The embedding process can be formulated as:

$$z_i^g = \phi_g(g_i, W_g) \tag{2}$$

$$z_i^p = \phi_p(p_i, W_p) \tag{3}$$

$$z_i^s = \phi_s(g_i, W_s) \tag{4}$$

where g_i is the grid cell id, p_i is the point of interest visited in grid cell g_i, and z_i^g, z_i^p, and z_i^s are the embedding of the grid cell g_i, point of interest p_i, and the spatial embedding of g_i, respectively. The three embeddings are calculated through three different layers: $\phi_g(.), \phi_p(.), \phi_s(.)$. The W refers to the learnable parameters optimized during the learning process. W_g and W_p are randomly initialized matrices, while W_s is initialized with the output of the spatial embedding layer. To preserve the unchanging spatial features of cells and avoid unintended modifications during training, W_s is frozen. Equations 2 and 4 use the same input, g_i, but while equation 2 learns the semantic embedding of cells during training, equation 4 reflects the static spatial features that remain constant. The dimensions of W_g, W_p, and W_s are $n \times d_L$, $n_p \times d_L$ and $n \times d_L$, respectively, where n is the number of grid cells in the tessellation, n_p is the number of POIs, and d_L is the embedding dimension.

The timestamp t_i is a continuous feature, and therefore, regarding it directly as an input feature will lead to a loss of information since the embeddings will not scale linearly in the feature space. The aim is to learn timestamp embeddings that preserve the properties of time, such as periodicity. Furthermore, the distance in the embedding space between two timestamps needs to be proportional to the difference between the timestamps, i.e., the relative information between the timestamps must be preserved. Inspired by existing work [10], we design a temporal-aware positional encoding to replace the positional encoding used in the original BERT model with:

$$[z_i^t]_j = \begin{cases} \sin(w_j t_i), & \text{if } j \text{ is odd} \\ \cos(w_j t_i), & \text{if } j \text{ is even} \end{cases} \tag{5}$$

where j is the order of the dimension, w_j is a learnable parameter, and t_i is the timestamp of the ith check-in in the trajectory. To see why this temporal encoding preserves the relative information between the timestamps, we can calculate the distance between two consecutive timestamps as:

$$(z_i^t)(z_{i+1}^t)^\intercal = \sum_{i=1}^{d} \cos(w_i(t_{i+1} - t_i)) \tag{6}$$

where the distance between t_i and t_{i+1} timestamps is the dot product of their respective temporal encoding (z_i^t) and (z_{i+1}^t). We can observe that the distance between the vectors is dependent on the difference between the timestamps $t_{i+1} - t_i$ and on w_i (parameters which the model learns during the training). Thus, the relative and periodic information of time is preserved and learned in this encoding function.

We adopt a non-invasive self-attention mechanism [11] where the side information, like spatial and temporal properties, is passed to the self-attention module directly instead of adding it to the grid cell embeddings. Therefore, the spatial-temporal embedding layer produces two outputs:

$$R^{(id)} = z_1^g, z_2^g, ..., z_m^g \tag{7}$$

$$R = (\{z_1^s, z_2^s, ..., z_m^s\}, \{z_1^p, z_2^p, ..., z_m^p\}, \{z_1^t, z_2^t, ..., z_m^t\}) \tag{8}$$

where $R^{(id)}$ embeddings are passed forward to the encoder, while the R embeddings are passed directly to the self-attention component, as shown in the Figure 4. The $R^{(id)}$ contains the embeddings of the grid cells, while R contains three sets, each one having the embedding of different side information like spatial, temporal, and POIs.

C. TULHOR's Encoder

The encoder block consists of *a multi-head spatial-temporal non-invasive self-attention* mechanism followed by *a position-wise feed-forward layer*. The self-attention enriches each token with *spatial*, *temporal*, and *contextual* information from other tokens in the sequence. By employing multiple heads, the model can capture diverse dependencies simultaneously. A position-wise feed-forward network with `ReLu` activation introduces non-linearity, and a residual connection ensures stable gradient flow during training.

The multi-head spatial-temporal non-invasive self-attention (ST-NOVA) in TULHOR differs from the standard self-attention (SA) found in Transformer models. SA uses invasive attention, requiring additional features like positional information to be incorporated into the input sequence representation. However, this approach poses a drawback as the output of the self-attention layer is fed to the predicting layer, making the searching task more challenging due to the creation of a compounded embedding space To address these problems, we use a non-invasive attention instead, which is represented as:

$$NOVA(R^{(id)}, R) = \sigma(\frac{QK^T}{\sqrt{d_L}})V \tag{9}$$

$$V = R^{id} \times W_V, K = F \times W_k, Q = F \times W_Q \tag{10}$$

$$F = MLP(R^{(id)}||R) \tag{11}$$

where $W_V, W_k, W_Q \in \mathbb{R}^{d_L \times d_n}$, $F \in \mathbb{R}^{m \times d_L}$ and $||$ is the concatenation operation. The ST-NOVA takes two inputs, the input sequence id $R^{(id)}$ and the other side information in R. Then ST-NOVA uses the input sequence id $R^{(id)}$ to calculate the `Values` matrix. Regarding the `Keys` and `Query` matrices, the component concatenates the input sequence id with the additional features and uses a multilayer perceptron (MLP) to unify the dimension; the output of the MLP is used to calculate

TABLE II: Statistics of two FOURSQUARE datasets (FOURSQUARE-NYC and FOURSQUARE-TKY) including the # of trajectories for three user groups (108, 209, all).

DATASET	$\|\mathcal{U}\|$	$\|\mathcal{T}\|$
FOURSQUARE-NYC	108	6795
	209	9,637
	234	10,133
FOURSQUARE-TKY	108	9343
	209	14,151
	451	20,964

the `Keys` and `Query` matrices. ST-NOVA uses the additional features to calculate how tokens are similar. Unlike SA, which infuses the additional features directly into the input sequence, we use the additional features to understand how two tokens are similar.

D. Pre-training TULHOR

BERT-based models are trained following the masked language modeling (MLM) and the next sentence prediction (NSP) training approaches. In our setting, we do not require the NSP task, so we drop it. The main issue with the MLM training is that the model requires many steps to converge because only a percentage of the tokens are masked, which translates to smaller training samples than the autoregressive training task. To address this issue, we increase the percentage of masked tokens. The original BERT model is trained with 15% of tokens masked. However, recent research [12] has shown that increasing the percentage of the masked tokens can boost the model's performance, while also helping it to converge faster. Therefore, we adapted the recommended configuration of 40% of masked tokens. Let $\mathcal{L}_{MLM}$ be the Masked Language Modeling loss, then:

$$\mathcal{L}_{MLM} = \frac{1}{|Tr^m|} \sum_{g_m \in Tr^m} -\log P(g^m = g^{m*} | Tr^{m'}) \quad (12)$$

where Tr is a trajectory and $Tr^{m'}$ is the masked version of it, Tr^m is the set containing the randomly masked items in Tr, and g^{m^*} is the true grid cell for the masked item g^m.

E. Fine-tuning TULHOR

Pre-training our model using the masked language modeling objective, enables it to learn generalized embeddings. Next, we need to fine-tune the model for addressing the trajectory-user linking problem. The first step in fine-tuning our model is to add a classification layer to TULHOR, which will get the probability distribution of users. As in the traditional BERT, a `[CLS]` token is added in the beginning of each trajectory. This token has no temporal-positional information; however, the output of TULHOR for the token `[CLS]` is inferred by all the other steps in the trajectory, so `[CLS]` maintains the spatial-temporal representation of the trajectory. This means that the output of TULHOR for `[CLS]` can still be useful for trajectory-user linking problem. Let h^{Tr} be the `[CLS]` representation of the trajectory Tr. Then the classification layer is formulated as follows:

$$y' = (W_C \cdot h^{Tr} + b_c) \quad (13)$$

where $W_c \in \mathbb{R}^{|\mathcal{U}| \times d_L}$ and $b_c \in \mathbb{R}^{|\mathcal{U}|}$ are the weight matrix and bias of the classification layer, y' is a vector, and y'_i is the probability that the trajectory Tr belongs to user u_i. We apply a softmax activation function to y to transform the values into normalized probabilities:

$$\sigma(y'_i) = \frac{e^{y'_i}}{\sum_{j=1}^{|\mathcal{U}|} e^{y'_j}} \quad for\ i = 1, 2, \ldots, |\mathcal{U}| \quad (14)$$

The function's output is a probability distribution, where all values are between 0 and 1, and the sum of all values is 1. This makes it easy to select the user with the highest probability as the final output with $argmax$.

The final step in fine-tuning is network training, where we apply a balanced cross-entropy loss, balanced by the number of effective samples [13], with backpropagation to train our model. Given unlinked trajectory Tr generated by u_i, then the loss for one sample in the training set is represented as:

$$\mathcal{L}(Tr, u_i) = \frac{1-\beta}{1-\beta^{n_{u_i}}} log(\sigma(y')) \quad (15)$$

where n_{u_i} is the number of trajectories in the training set with user u_i and β is a hyperparameter that controls the balancing factor. Not all samples in the training set have the same impact on performance. Some samples are more crucial, while others may overlap, like in the case of trajectories. Balancing by the effective number of samples considers this factor, whereas inverse class frequency sampling does not.

V. EVALUATION

We perform experiments on two real-world datasets, the New York (NYC) and Tokyo (TKY) check-in datasets from the FOURSQUARE[3] social network, consisting of check-ins from 2012-2013. We do not consider trajectories with less than three check-ins and users with less than five trajectories. To assess model robustness, we evaluate them on three user groups (109, 208, all) from both datasets, using 80% of user trajectories for training and the remaining 20% to evaluate performance. Table II provides the statistics of the datasets.

A. Baselines and Implementation

Baselines. We evaluate TULHOR against the following baselines:

- Conventional ML methods: Decision Tree (DT), Linear Discriminant Analysis (LDA) and Linear Support Vector Machine (SVM). To embed the trajectories, we use Bag-of-Words (BOW) method, followed by applying Singular Value Decomposition (SVD) to reduce the dimensionality of the embeddings.
- TULER [3]: A recurrent neural network model with three variations RNN (TULER), LSTM (TULER-L), and GRU (TULER-G). We reimplement this model in PyTorch.

[3]https://sites.google.com/site/yangdingqi

TABLE III: Results on FOURSQUARE-NYC mobility dataset. The highest performance is indicated in bold and the second best performance has been underlined. ‘Improvement’ denotes the improvement of TULHOR model over the strongest baseline.

	FOURSQUARE-NYC														
	$\vert\mathcal{U}\vert = 108$					$\vert\mathcal{U}\vert = 209$					$\vert\mathcal{U}\vert = 234$				
MODEL	ACC@1	ACC@5	P	R	F1	ACC@1	ACC@5	P	R	F1	ACC@1	ACC@5	P	R	F1
DT	0.884	0.892	0.878	0.867	0.868	0.785	0.788	0.753	0.728	0.730	0.778	0.782	0.722	0.712	0.705
LDA	0.822	0.851	0.962	0.810	0.868	0.746	0.781	0.791	0.687	0.718	0.696	0.752	0.724	0.615	0.650
LINEAR-SVM	0.873	0.929	**0.966**	0.878	0.909	0.776	0.839	0.785	0.702	0.727	0.731	0.798	0.724	0.628	0.657
TULER	0.870	0.929	0.869	0.851	0.852	0.776	0.853	0.749	0.722	0.718	0.768	0.844	0.733	0.707	0.703
TULER-L	0.903	0.942	0.904	0.890	0.890	0.847	0.898	0.828	0.803	0.807	0.845	0.889	0.821	0.806	0.803
TULER-G	0.909	0.949	0.914	0.897	0.898	0.854	0.892	0.835	0.811	0.812	0.846	0.891	0.821	0.805	0.803
ATT-LSTM	0.823	0.896	0.715	0.703	0.709	0.716	0.832	0.554	0.559	0.556	0.712	0.830	0.569	0.557	0.563
ATT-GRU	0.886	0.933	0.779	0.779	0.791	0.835	0.891	0.663	0.680	0.671	0.889	**0.936**	0.741	0.738	0.740
DEEPTUL	0.853	0.923	0.765	0.738	0.751	0.733	0.840	0.614	0.597	0.606	0.789	0.891	0.607	0.617	0.612
TULHOR	**0.940**	**0.966**	0.938	**0.931**	**0.932**	**0.903**	**0.943**	**0.890**	**0.877**	**0.876**	**0.892**	0.932	**0.876**	**0.864**	**0.860**
Improvement	3.42%	1.85%	-2.89%	3.85%	2.53%	5.82%	5.07%	6.58%	7.83%	7.87%	0.35%	-0.49%	6.61%	7.13%	7.19%

TABLE IV: Results on FOURSQUARE-TKY mobility dataset. The highest performance is indicated in bold and the second best performance has been underlined. ‘Improvement’ denotes the improvement of TULHOR model over the strongest baseline.

	FOURSQUARE-TKY														
	$\vert\mathcal{U}\vert = 108$					$\vert\mathcal{U}\vert = 209$					$\vert\mathcal{U}\vert = 451$				
MODEL	ACC@1	ACC@5	P	R	F1	ACC@1	ACC@5	P	R	F1	ACC@1	ACC@5	P	R	F1
DT	0.789	0.793	0.785	0.777	0.775	0.658	0.664	0.629	0.615	0.613	0.522	0.525	0.446	0.437	0.431
LDA	0.853	0.912	0.927	0.847	0.874	0.722	0.808	0.778	0.692	0.713	0.574	0.720	0.553	0.501	0.495
LINEAR-SVM	0.890	0.948	0.923	0.886	0.898	0.769	0.878	0.794	0.736	0.748	0.609	0.761	0.610	0.539	0.550
TULER	0.870	0.933	0.871	0.860	0.860	0.768	0.864	0.762	0.735	0.736	0.637	0.74	0.588	0.554	0.548
TULER-L	0.905	0.952	0.904	0.898	0.897	0.848	0.911	0.837	0.825	0.824	0.739	0.827	0.708	0.675	0.675
TULER-G	0.915	0.954	0.916	0.910	0.909	0.851	0.911	0.842	0.824	0.825	0.738	0.823	0.701	0.672	0.671
ATT-LSTM	0.908	0.966	0.916	0.901	0.908	0.752	0.871	0.795	0.729	0.760	0.407	0.584	0.362	0.326	0.343
ATT-GRU	0.933	**0.975**	0.932	0.928	0.930	0.869	0.937	0.872	0.856	0.864	0.742	0.821	0.715	0.689	0.695
DEEPTUL	0.922	0.966	0.927	0.913	0.920	0.773	0.904	0.820	0.747	0.782	0.660	0.790	0.631	0.587	0.608
TULHOR	**0.939**	0.973	**0.937**	**0.934**	**0.933**	**0.893**	**0.953**	**0.883**	**0.877**	**0.875**	**0.801**	**0.888**	**0.783**	**0.755**	**0.752**
Improvement	0.58%	-0.26%	0.59%	0.71%	0.37%	2.7%	1.77%	1.33%	2.53%	1.30%	7.86%	7.47%	9.52%	9.53%	8.11%

- DeepTUL [4]: A recurrent neural network with historical attention module, this also has three variations RNN (DeepTUL), LSTM (Attn-LSTM) and GRU (Attn-GRU). Unlike TULER, DeepTUL captures the temporal features and is the current state-of-the-art model.

Implementation. TULHOR model is implemented in PyTorch with one encoder layer and 12 attention heads. For pre-training and fine-tuning, we use a batch size of 24 and a learning rate of 0.005 with decays of 0.5. The embedding size is set to 512, β is set to 0.99 and the model is trained for 10 epochs. We also reimplement TULER and its variants in PyTorch and provide the code with our source code. The default settings are used for the baselines.

B. Evaluation Metrics

For evaluating the performance of the TUL models, we use standard metrics from the multiclass classification domain including accuracy@K (ACC@K, with K = 1 and 5), macro precision (P), macro recall (R), and macro F1 score. ACC@K evaluates the accuracy of user-linking classification, while macro F1 provides a comprehensive evaluation of the model's performance across all classes. This is particularly important in the case of TUL datasets, which are characterized by imbalanced data distributions.

C. Overall Performance

In this study, we evaluate the performance of TULHOR and various baseline models on the NYC and TKY datasets, with the results presented in Tables III and IV, respectively. Our findings indicate that TULHOR outperforms all the other baseline models under all the user groups settings and along every metric. In terms of F1 score, on the NYC dataset, TULHOR yields improvements of 4.2%, 7.8%, and 7.1% when $|\mathcal{U}| = 108, 209, 234$, respectively, over the strongest baseline model. Similarly, for the TKY dataset, although the improvement in smaller user settings is moderate, when $|\mathcal{U}| = 451$ setting is considered, TULHOR improves over the strongest baseline by nearly 8.1%. It is worth noting that while TULHOR consistently outperforms the other baselines, the amount of improvement gains considerably increases as the problem becomes more challenging (number of users increases), indicating the scalability and effectiveness of our model. This superior performance can be attributed to TULHOR's ability to capture spatial-temporal patterns in trajectories more effectively than the other methods. Additionally, unlike the baseline models, TULHOR sufficiently addresses the imbalanced data through proper sampling techniques as reflected by the competitive macro recall and macro precision scores for all user groups.

D. Ablation Study

We conduct a comprehensive ablation study to assess each component's significance in the TULHOR model. The full TULHOR model is compared against several modified variations, including: (1) **-HOR**: This eliminates the higher-order generation step, essentially reducing the model to a BERT architecture that processes trajectory data without high-order information, making it comparable to the approach of the other

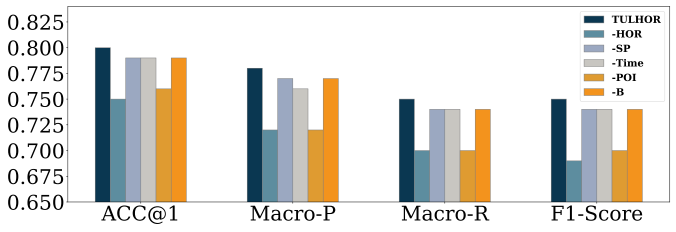

Fig. 5: Results of ablation experiments of TULHOR model for FOURSQUARE-TKY, $|\mathcal{U}| = 451$.

baseline models. (2) **-SP**: This removes the spatial feature extractor step, thus excluding spatial embedding from embedding layer. (3) **-POI**: The point-of-interest information is removed from the higher-order trajectory and subsequently POIs embeddings are removed from the spatial-temporal embedding layer. (4) **-T**: This substitutes the temporal positional encoding with the standard positional encoding used in the original Transformer architecture. (5) **-B**: This omits the balancing factor from the training loss. The results of the ablation study are shown in Figure 5, where we notice that the removal of any of the components results in a decrease in performance of the model, with varying degrees. The results indicate that the inclusion of higher-order information is crucial for the model's performance, as evidenced by the fact that the **-HOR** variation performed the worst among all the experiments. The second worst performance was observed in the **-POI** variation, highlighting the importance of using POI information in conjunction with higher-order information to differentiate between visits to similar grid cells. The results of the **-SP** and **-T** variations demonstrate the significance of spatiotemporal features, respectively, in enhancing the performance of the model. Lastly, the **-B** variation highlights the importance of effective sampling techniques when dealing with imbalanced datasets, which is characteristic of the TUL datasets.

E. Parameter Study

Impact of hyperparameters. We evaluated TULHOR's performance through a parameter study to understand the effect of various hyperparameters: embedding dimension, hidden dimension, number of encoder layers, and number of attention heads, with the experiments conducted using FOURSQUARE-TKY dataset with 451 users. The results are presented in Figure 6. We observe that increasing the embedding and hidden dimensions generally improve TULHOR's performance by allowing the model to store more information in the latent space. Similarly, increasing the number of attention heads leads to improved performance up to a certain point, but adding more than 16 heads results in a decrease in performance, likely due to the limited size of the dataset. Lastly, adding more encoder layers reduces the performance, suggesting that fewer layers may be sufficient for smaller datasets.

Impact of grid cell size. We conduct one more study to test the impact of varying grid cell sizes on TULHOR's performance. We test three different sizes and refer to them

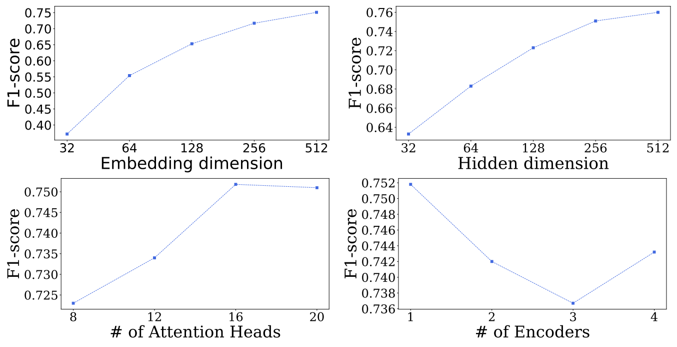

Fig. 6: Results of varying hyperparameters on TULHOR model for FOURSQUARE-TKY, $|\mathcal{U}| = 451$.

TABLE V: Statistics about different tessellations with the # of cells and cell size for each resolution for FOURSQUARE-NYC

RESOLUTION	# OF CELLS	CELL SIZE (km^2)
HEX@7	334	5.160
HEX@8	2,003	0.730
HEX@9	11,036	0.015

as Hex@k, where $k = \{7, 8, 9\}$ is the resolution. The smaller the k is set to, the larger the cell size is, thereby, decreasing the number of cells in the grid. Table V provides the statistics about the different tessellations. The results of this experiment are presented in Table VI. We observe that as the cell size decreases (i.e., the k increases), the performance of the model increases. For the 108 user setting, a 2% decrease in macro F1 is observed for Hex@8 and Hex@7, with slightly higher accuracy for Hex@9. For 209 users, the gap in performance between Hex@9 and other tessellations grows, reaching a 6% difference in macro F1 for Hex@7 and a 3% difference for Hex@8, along with a 3% decrease in accuracy between Hex@8 and Hex@9. In the 451 user setting, while the performance difference between Hex@8 and Hex@9 remained relatively stable, a significant gap in ACC@1 between Hex@9 and Hex@7 is observed. Comparing the 209 and 451 user settings, the difference between Hex@9 and Hex@7 in recall and precision decreased from 6% to 4%, likely due to the use of an effective balanced sampling technique for dealing with imbalanced datasets, like the 451 user setting. To conclude, as the cell size increases, capturing user movement patterns becomes increasingly challenging, as seen in the results.

VI. RELATED WORK

Our research is related to (i) *trajectory data mining*, (ii) *trajectory-user linking*, and (iii) *deep learning for spatiotemporal data*. We cover below some of the most significant efforts relevant to our work. Note that some related work have already been cited throughout the manuscript to keep the discussion focused, so they are mostly omitted here.

A. Trajectory Data Mining

Trajectory data mining involves extracting insights and patterns from large-scale mobility data. It aims to uncover hidden relationships and insights into mobility patterns and

TABLE VI: Results of impact of different grid sizes on the performance of TULHOR on FOURSQUARE-TKY dataset.

	FOURSQUARE-TKY														
	#USERS = 108					#USERS = 209					#USERS = 451				
METHOD	ACC@1	ACC@5	P	R	F1	ACC@1	ACC@5	P	R	F1	ACC@1	ACC@5	P	R	F1
HEX@7	0.923	0.971	0.920	0.911	0.913	0.868	<u>0.943</u>	0.832	0.817	0.815	0.711	0.883	0.734	0.734	0.711
HEX@8	<u>0.926</u>	**0.977**	<u>0.925</u>	<u>0.917</u>	<u>0.917</u>	<u>0.868</u>	0.940	<u>0.862</u>	<u>0.849</u>	<u>0.849</u>	<u>0.790</u>	<u>0.884</u>	<u>0.753</u>	<u>0.740</u>	<u>0.733</u>
HEX@9	**0.939**	<u>0.973</u>	**0.937**	**0.934**	**0.933**	**0.893**	**0.953**	**0.883**	**0.877**	**0.875**	**0.801**	**0.888**	**0.783**	**0.755**	**0.752**

behaviors, with the goal of supporting a wide range of applications, including transportation planning [14], location-based services [15], urban planning, and public health monitoring [16]. Of particular interest are technical problems related to trajectory similarity [17], trajectory clustering [18], anomaly detection in moving objects [19], and graph-related problems, such as finding important nodes in mobility networks [20], and mining interactions of moving objects or people [21]. A couple of comprehensive surveys on trajectory data mining exist that provide a taxonomy of the technical problems, available methods to address them, applications and open research problems [22]–[24].

B. Trajectory-User Linking

Trajectory-user linking (TUL) is a recently introduced trajectory classification problem, where the objective is to link anonymous trajectories to the users that they belong to. Addressing the TUL problem is essential for LBS as it enables personalization, improves data privacy and security, and ensures the accuracy of the services provided to users. Without the ability to accurately link trajectories to users, LBS would not be able to personalize their offerings effectively. For example, if a LBS is not able to accurately identify a user, it may not be able to provide accurate recommendations or advertisements based on their location and movement patterns. By being able to link trajectories to users, LBS can ensure that user data is only used for the intended purpose and is not shared or misused by third parties. TUL can be addressed using either *classical methods* or *machine learning-based methods* (both conventional and deep learning-based ones).

Classical methods. Classical methods rely on trajectory similarity metrics. Typically, these metrics compute the distance between an anonymous trajectory and those of known users, and linking is done based on the smallest distance. The most popular techniques include the longest common sub-sequence, the Hausdorff distance, dynamic time warping [1], and NeuTraj [2]. These methods have limitations in capturing long-term relationships in location data, sensitivity to noise and outliers, and high computational costs, making them unsuitable for real-time or large-scale applications.

Conventional ML methods. Conventional ML-based classification models, such as k-nearest neighbors (KNN) [25] and support vector machines (SVM) [26] can also solve the TUL problem by transforming trajectories into a one-hot vector, treating users as labels, and training the models on trajectories with known users. However, these methods fail to consider spatial-temporal features and often perform inferior to deep learning-based models.

Deep learning-based ML methods. Deep learning-based methods offer several advantages over traditional methods for solving the TUL problem, including improved accuracy, the ability to handle high-dimensional trajectory data, robustness to noise and outliers, and scalability. These advantages make these methods a promising approach for solving the TUL problem. One of the first works in this category [3] used sequence-to-sequence models such as RNN, LSTM, and GRU to learn the check-in embeddings and feed the output to a shallow classification layer. DeepTul [4] improved on this by adding an attention component and considering the temporal dimension. The method generates a representation of a user's historical trajectories for classification and learning multi-periodic patterns. GNNTUL [6] pointed out the high computational cost of previous works and proposed a graph neural network approach that incorporates user visiting intentions in the classification task. TULSN [27] used a siamese neural network to capture semantic information of the trajectories.

C. Deep Learning for Trajectory Data

Trajectory data is a type of spatiotemporal data, which depending on the task, can be treated as sequantial data.

Spatiotemporal models. Deep learning based methods have been proposed for learning representations of spatiotemporal data that are a good fit for several trajectory data mining downstream tasks. A few examples include recommending points of interest (POIs) [28]–[31], clustering trajectories [32], and analyzing movement behavior [33]. The foundation of these approaches involves learning low-dimensional representations of the trajectory data. One of the first methods towards this was "trajectory2vec" [32], which used a sliding window method to capture space- and time-invariant characteristics of trajectories. Similarly, "t2vec" [34] used a customized RNN with a spatial-loss function to address the low sampling rate and noisiness of the data. While other approaches focused solely on capturing the spatial-temporal properties of trajectories, Boonchoo et al. [35] were the first to consider multimodal deep learning that learns representations from data of multiple modalities (images, reviews, and geo-tags). The model learns embeddings by predicting the context of the next trajectory data point, similar to learning sentence representations in language models. A recent trend for learning trajectory representations exploits paths defined on the road networks due to the decrease in sparsity and the ability to learn richer embeddings [36], [37]. A comprehensive review can be found in Wang et al. [38].

Sequence models. For many tasks it is customary to treat trajectory data as a type of sequential data that can be processed by sequence models. For example, Recurrent Neural Networks

(RNNs) [39], [40] have been used to address the problem of human trajectory prediction [41], and to generate trajectories using generative models [42]. Additionally, attention-based models, such as Transformer models [43], can also be used to model trajectory data and capture the long-range dependencies between the different points in the trajectory. For example, Li et al. use a graph-based spatial Transformer-based deep learning model for pedestrian trajectory prediction [44].

VII. Conclusions

In this work, we proposed a novel deep learning framework – TULHOR (trajectory-user linking using higher-order representations) – for modeling the trajectory-user linking problem. To address some of the challenges related to data quality, sparsity, and imbalance, our model was trained by higher-order mobility flow representations extracted by simple check-in data. The results of extensive experiments over two real-world datasets demonstrated the effectiveness of the proposed model, which consistently outperforms several strong baselines. We believe our proposed method and model can have broad and useful applications. Future directions include trajectory-user linking on the roads network and multi-trajectory user linking.

References

[1] D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series." in *KDD workshop*, vol. 10, no. 16. Seattle, WA, USA:, 1994, pp. 359–370.

[2] D. Yao, G. Cong, C. Zhang, and J. Bi, "Computing trajectory similarity in linear time: A generic seed-guided neural metric learning approach," in *ICDE*. IEEE, 2019, pp. 1358–1369.

[3] Q. Gao, F. Zhou, K. Zhang, G. Trajcevski, X. Luo, and F. Zhang, "Identifying human mobility via trajectory embeddings." in *IJCAI*, vol. 17, 2017, pp. 1689–1695.

[4] C. Miao, J. Wang, H. Yu, W. Zhang, and Y. Qi, "Trajectory-user linking with attentive recurrent network," in *AAMAS*, 2020, pp. 878–886.

[5] F. Zhou, Q. Gao, G. Trajcevski, K. Zhang, T. Zhong, and F. Zhang, "Trajectory-user linking via variational autoencoder." in *IJCAI*, 2018, pp. 3212–3218.

[6] F. Zhou, S. Chen, J. Wu, C. Cao, and S. Zhang, "Trajectory-user linking via graph neural network," in *ICC*. IEEE, 2021, pp. 1–6.

[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[8] S. Mehmood and M. Papagelis, "Learning semantic relationships of geographical areas based on trajectories," in *IEEE International Conference on Mobile Data Management (MDM)*. IEEE, 2020, pp. 109–118.

[9] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *ACM SIGKDD*, 2016, pp. 855–864.

[10] Y. Lin, H. Wan, S. Guo, and Y. Lin, "Contrastive pre-training of spatial-temporal trajectory embeddings," *preprint arXiv:2207.14539*, 2022.

[11] C. Liu, X. Li, G. Cai, Z. Dong, H. Zhu, and L. Shang, "Noninvasive self-attention for side information fusion in sequential recommendation," in *Proc. of AAAI*, vol. 35, no. 5, 2021, pp. 4249–4256.

[12] A. Wettig, T. Gao, Z. Zhong, and D. Chen, "Should you mask 15% in masked language modeling?" *arXiv preprint arXiv:2202.08005*, 2022.

[13] Y. Cui, M. Jia, T.-Y. Lin, Y. Song, and S. Belongie, "Class-balanced loss based on effective number of samples," in *Proc. of the IEEE/CVF CVPR*, 2019, pp. 9268–9277.

[14] F. Arasteh, S. SheikhGarGar, and M. Papagelis, "Network-aware multi-agent reinforcement learning for the vehicle navigation problem," in *ACM SIGSPATIAL*, 2022, pp. 1–4.

[15] A. Nematichari, T. Pechlivanoglou, and M. Papagelis, "Evaluating and forecasting the operational performance of road intersections," in *ACM SIGSPATIAL*, 2022, pp. 1–12.

[16] T. Pechlivanoglou, J. Li, J. Sun, F. Heidari, and M. Papagelis, "Epidemic spreading in trajectory networks," *Big Data Research*, vol. 27, p. 100275, 2022.

[17] K. Toohey and M. Duckham, "Trajectory similarity measures," *Sigspatial Special*, vol. 7, no. 1, pp. 43–50, 2015.

[18] J.-G. Lee, J. Han, and K.-Y. Whang, "Trajectory clustering," in *Proc. of the 2007 ACM SIGMOD Intl. Conf. on Manag. of Data*, 2007.

[19] S. Dodge, R. Weibel, and E. Forootan, "Revealing the physics of movement: Comparing the similarity of movement characteristics of different types of moving objects," *Computers, Environment and Urban Systems*, vol. 33, no. 6, p. 419–34, 2009.

[20] T. Pechlivanoglou and M. Papagelis, "Fast and accurate mining of node importance in trajectory networks," in *2018 IEEE Intl. Conf. on Big Data (Big Data)*, 2018, pp. 781–790.

[21] A. Sawas, A. Abuolaim, M. Afifi, and M. Papagelis, "Tensor methods for group pattern discovery of pedestrian trajectories," in *(MDM)*, 2018, pp. 76–85.

[22] Y. Zheng, "Trajectory data mining: An overview," *ACM Trans. Intell. Syst. Technol.*, vol. 6, no. 3, May 2015.

[23] G. Atluri, A. Karpatne, and V. Kumar, "Spatio-temporal data mining: A survey of problems and methods," *ACM Comp. Surv*, vol. 51, no. 4, 2018.

[24] A. Hamdi, K. Shaban, A. Erradi, A. Mohamed, S. K. Rumi, and F. D. Salim, "Spatiotemporal data mining: a survey on challenges and open problems," *Artif. Intel. Review*, vol. 55, no. 2, p. 1441–1488, Feb 2022.

[25] G. Guo, H. Wang, D. Bell, Y. Bi, and K. Greer, "Knn model-based approach in classification," in *On The Move to Meaningful Internet Systems 2003*. Springer, 2003, pp. 986–996.

[26] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intelligent Systems and their applications*, vol. 13, no. 4, pp. 18–28, 1998.

[27] Y. Yu, H. Tang, F. Wang, L. Wu, T. Qian, T. Sun, and Y. Xu, "Tulsn: siamese network for trajectory-user linking," in *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020, pp. 1–8.

[28] Q. Guo, Z. Sun, J. Zhang, and Y.-L. Theng, "An attentional recurrent neural network for personalized next location recommendation," in *AAAI*, vol. 34, no. 01, 2020, pp. 83–90.

[29] D. Lian, Y. Wu, Y. Ge, X. Xie, and E. Chen, "Geography-aware sequential location recommendation," in *ACM SIGKDD*, 2020, pp. 2009–2019.

[30] Q. Liu, S. Wu, L. Wang, and T. Tan, "Predicting the next location: A recurrent model with spatial and temporal contexts," in *Thirtieth AAAI conference on artificial intelligence*, 2016.

[31] J. Li, Y. Wang, and J. McAuley, "Time interval aware self-attention for sequential recommendation," in *WSDM*, 2020, pp. 322–330.

[32] D. Yao, C. Zhang, Z. Zhu, J. Huang, and J. Bi, "Trajectory clustering via deep representation learning," in *2017 international joint conference on neural networks (IJCNN)*. IEEE, 2017, pp. 3880–3887.

[33] W. Yang, Y. Zhao, B. Zheng, G. Liu, and K. Zheng, "Modeling travel behavior similarity with trajectory embedding," in *DASFAA*. Springer, 2018, pp. 630–646.

[34] D. Yao, C. Zhang, Z. Zhu, Q. Hu, Z. Wang, J. Huang, and J. Bi, "Learning deep representation for trajectory clustering," *Expert Systems*, vol. 35, no. 2, p. e12252, 2018.

[35] T. Boonchoo, X. Ao, and Q. He, "Multi-aspect embedding for attribute-aware trajectories," *Symmetry*, vol. 11, no. 9, p. 1149, 2019.

[36] Y. Sang, Z. Xie, W. Chen, and L. Zhao, "Tulrn: Trajectory user linking on road networks," *World Wide Web*, pp. 1–17, 2022.

[37] T.-Y. Fu and W.-C. Lee, "Trembr: Exploring road networks for trajectory representation learning," *ACM TIST*, vol. 11, no. 1, pp. 1–25, 2020.

[38] S. Wang, J. Cao, and P. Yu, "Deep learning for spatio-temporal data mining: A survey," *IEEE TKDE*, vol. 34, no. 08, pp. 3681–700, 2022.

[39] J. Elman, "Finding structure in time," *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990.

[40] A. Graves, "Generating sequences with recurrent neural networks," *arXiv preprint arXiv:1308.0850*, 2013.

[41] A. Alahi, V. Krueger, A. Goel, L. Fei-Fei, and N. Robertson, "Social lstm: Human trajectory prediction in crowded spaces," in *ACM SIGGRAPH*. ACM, 2016, pp. 1–10.

[42] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, "Social gan: Socially acceptable trajectories with generative adversarial networks," in *IEEE CVPR*, 2018, pp. 2255–2264.

[43] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[44] L. Li, M. Pagnucco, and Y. Song, "Graph-based spatial transformer with memory replay for multi-future pedestrian trajectory prediction," in *IEEE/CVF CVPR*, 2022, pp. 2231–2241.