Trajectory-User Linking using Higher-order Mobility Flow Representations

by

Mahmoud Alsaeed

A THESIS SUBMITTED TO THE FACULTY OF GRADUATE STUDIES IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

GRADUATE PROGRAM IN ELECTRICAL ENGINEERING AND COMPUTER SCIENCE
YORK UNIVERSITY

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 recurrent neural networks 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.

To Mom, Dad and my grandparents

Acknowledgements

Praise be to Allah, the Most Gracious, the Most Merciful. Prayer and peace be upon the Prophet Muhammad. First of all, I am grateful to my supervisor, Manos Papagelis, for his unwavering guidance, remarkable patience, tireless support, and invaluable advice throughout my studies. His contributions played a significant role in the success of this work, and I would not have been able to complete it without his help. Thanks to Ameeta Agrawal for her help and guidance for the past two years. I would like to thank committee members Aijun An and Mehdi Nourinejad for taking the time to read my thesis and for providing feedback. I would like to express my deepest gratitude to my parents, grandparents, brothers, and sisters for their love and support throughout this journey. I would like to extend my sincere appreciation to my friends, whose company may have delayed my graduation and reduced my productivity in terms of published works. However, their friendship was invaluable and made the journey worthwhile. In addition, I would like to express my gratitude to my cats who provided me with emotional support during the course of my studies. Although they could be a distraction at times, their positive impact cannot be overstated.

Table of Contents

Abstract ii

Dedication iii

Acknowledgments iv

Table of Contents vii

List of Tables viii

List of Figures x

1 Introduction 1

1.1 Problem of Interest 1

1.2 Current Approaches 2

1.3 Limitations of Current Approaches 3

1.4 Thesis Contributions 3

1.5 Thesis Organization 5

2 Related Work 6

2.1 Trajectory Data Mining 6

2.2 Trajectory Classification 7

2.2.1 General Classification 7

2.2.2 Problem-specific Classifications 8

2.2.2.1 Transportation Mode 8

2.2.2.2 Trajectories on Road Network 9
2.3 Trajectory-user Linking 10
2.3.1 Classical Methods 10
2.3.2 Conventional ML Methods 11
2.3.3 Deep learning-based ML Methods 11
2.4 Statistical Models for Trajectory Data 12
2.5 Deep Learning for Trajectory Data 12
2.5.1 Sequential Models 12
2.5.2 Spatiotemporal Models 13

3 Preliminaries & Problem Definition **15**
3.1 Preliminaries 15
3.2 Problem Definition 16

4 Generating Higher-order Mobility Flow Representations **18**
4.1 Key Idea 18
4.2 Rationale for using Higher-order Mobility Flow 22
4.3 Implementation Considerations 23
4.3.1 Google Maps 23
4.3.2 H3 26

5 Modeling Trajectory-user Linking **28**
5.1 TULHOR's Spatial Embedding Layer 28
5.1.1 Constructing a Hexagon-lattice Graph 29
5.1.2 Learning Node Representations of the Hexagon-lattice Graph . 29
5.2 TULHOR's Spatiotemporal Embedding Layer 30
5.3 TULHOR's Encoder 32
5.4 Pre-training TULHOR 33
5.5 Fine-tuning TULHOR 34

6 Experimental Evaluation **36**

6.1 Datasets 36

6.2 Baselines Methods 39

6.2.1 Implementation Details 40

6.3 Evaluation Metrics 40

6.4 Experimental Results 42

6.4.1 Accuracy Performance 42

6.4.2 Ablation Study 45

6.4.3 Impact of Hyperparameters 49

6.4.4 Impact of Grid Cell Size 51

6.4.5 Impact of Sparsity 54

7 Conclusions **56**

7.1 Future Work 57

7.1.1 Multi-trajectory User Linking 57

7.1.2 Trajectory-user Linking on the Roads Network 57

7.1.3 Point-of-Interest Recommendation 57

7.1.4 Mixed Tessellations 58

List of Tables

3.1 Summary of notations 17

6.1 Statistics of two FOURSQUARE datasets (FOURSQUARE-NYC and FOURSQUARE-TKY) including the # of trajectories for three user groups (108, 209, all). 38

6.2 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. 43

6.3 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. 44

6.4 Statistics about different tessellations with the # of cells and cell size for each resolution for FOURSQUARE-TKY 50

6.5 Results of impact of different grid sizes on the performance of TULHOR on FOURSQUARE-TKY dataset. 50

6.6 Statistics about different tessellations with the # of cells and cell size for each resolution for FOURSQUARE-NYC 50

6.7 Results of impact of different grid sizes on the performance of TULHOR on FOURSQUARE-NYC dataset. 50

List of Figures

4.1 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. 19

4.2 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. 20

4.3 Impact of higher-order abstraction on sparsity. 21

4.4 Trajectories for FOURSQUARE-NYC 24

4.5 Trajectories for FOURSQUARE-TKY 25

5.1 High-level architecture of TULHOR. 29

6.1 snippet of Foursquare-NYC 36

6.2 Results of ablation experiments of TULHOR model for FOURSQUARE-TKY, $|\mathcal{U}| = 451$ on different tessellation levels 46

6.3 Results of varying hyperparameters on TULHOR model for FOURSQUARE-TKY, $|\mathcal{U}| = 451$. 49

6.4 Different tessellations levels for FOURSQUARE-NYC dataset. 52

6.5 Different tessellations levels for FOURSQUARE-TKY dataset. 53
6.6 The sparsity of check-ins, High-order check-ins, and High-order mobility flow across different tessellations For Foursquare-NYC & Foursquare-TKY 54
6.7 The F1 score of -HOR and TULHOR across different tessellations. The -HOR variation utilizes raw check-ins and acts as an indicator of the impact of sparsity. 54

Chapter 1

Introduction

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.

1.1 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.

1.2 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].

1.3 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.

1.4 Thesis 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].

- This work has been published at the 24th IEEE International Conference on Mobile Data Management (IEEE MDM 2023) [8].

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

1.5 Thesis Organization

The remainder of the thesis is organized as follows. Chapter 3 presents preliminaries and a formal definition of the problem. Chapter 4 presents our method for generating higher-order mobility flow data, and chapter 5 presents our TULHOR model for addressing the TUL problem. We describe the experimental setup, and present and discuss the results in chapter 6. We review related work in chapter 2 and conclude in chapter 7.

Chapter 2

Related Work

Our research is related to (i) *trajectory data mining*, (ii) *trajectory classification*, 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.

2.1 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 behaviors, with the goal of supporting a wide range of applications, including transportation planning [9], location-based services, urban planning [10], and public health monitoring [9, 11, 12]. Of particular interest are technical problems related to trajectory similarity [13], trajectory clustering [14], anomaly detection in moving objects [15], and graph-related problems, such as finding important nodes in mobility networks [16], and mining interactions of moving objects or people [17–19]. 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 [20–22]. Recently, deep learning approaches for spatiotemporal data representation learning have gained increasing attention (see survey by Wang et

al. [23] and discussion in paragraph 2.5).

2.2 Trajectory Classification

A prevalent data mining task is trajectory classification which aims to predict labels or classes of moving objects based on their trajectories. Works in this field focus on discovering and studying the optimal methods to extract features from trajectories. These features are divided into global features (such as maximum and minimum speed), obtained by studying the entire trajectory, and local features (obtained by examining subparts of the trajectory independently). Trajectories can also be classified into two categories: raw trajectories, which only contain spatial and temporal data, and multi-aspect trajectories, which contain semantic information - sometimes called semantic dimension - about visited places, such as their name, category, and opening hours. We classify the work in trajectory classification into two main categories, (i) *General classification* (ii) *Problem-Specific Classifications*. General classification approaches aim to develop techniques applicable to various trajectory classification tasks. In contrast, problem-specific classification approaches focus on developing techniques tailored to a particular application or problem domain. In the next section, we provide an overview of each trajectory classification category before we examine Trajectory-user linking in depth, which is considered a problem-specific classification.

2.2.1 General Classification

One of the earlier's works in General classification is Trajclass [24] which proposed a classification method using hierarchical Region-based and Trajectory Based clustering. After preprocessing data through the partitioning and grouping stage, Trajclass constructs a grid structure to cover the spatial dimension. Each trajectory is then assigned a cell; if the cell is non-homogeneous (i.e., there is an imbalance in the class distribution of trajectories within the cell), the cell is split further into the minor part until the ideal homogeneity and conciseness are reached. Some cells will not

reach the ideal level of homogeneity, and they will be passed to the next step, where the trajectory-based metric is used for clustering. A significant limitation of Trajclass is that the method only considers the spatial dimension of the trajectory. [15] tries to address the issue by proposing a trajectory feature extraction method that captures spatial and temporal features along with speed, velocity, and acceleration while also learning the local and global feature. Previous works split the trajectories based on some predefined metrics, Trajeclass based on homogeneity while based [15] on the change in movement parameters. [25] proposed an algorithm to split trajectories without relying on any predefined notions. It works by splitting trajectories into smaller sub-trajectories called Movelets, short for "Moving Elements of Trajectories". These MOVELETS represent local features of the trajectory and capture its distinctive characteristics. MOVELETS then extracts features from these MOVELETS using a set of statistical measures such as mean, standard deviation, and entropy. Next, MOVELETS uses a feature selection algorithm to select the most relevant MOVELETS that are most discriminative in distinguishing between different classes of trajectories. The selected MOVELETS are then used to construct a feature vector representing the trajectory. [25] Extended the idea of MOVELETS to work for Multi-aspect trajectories, where they take into account semantic features such as weather and POI category while generating the subtrajectoreis.

2.2.2 Problem-specific Classifications

We consider three problem-specific classification tasks (i) *transportation mode*, (ii) *trajectories on Road Network*, and (iii) *trajectory-user linking*.

2.2.2.1 Transportation Mode

Transportation mode classification is a trajectory classification problem where the goal is to infer/predict the mode of transportation of individuals through their movement patterns. The transportation mode classification task has numerous applications

in planning, traffic management, and urban mobility. The datasets used for transportation mode classification are divided into two primary categories: single-mode-based trajectories (SMT) and multi-mode-based trajectories (MMT). SMT consists of trajectories collected using only one transportation mode, while MMT consists of trajectories collected with multiple modes of transportation used by the user. [26] provides a review of significant works in the transportation mode classification field. Among notable works in transportation mode classification is [27], which introduced a technique to extract intricate features that are more effective in discriminating between modes of transportation than conventional features like acceleration and speed. The features include heading change rate (HCR), stop rate (SR), and velocity change rate (VCR). After feature extraction, a post-processing algorithm is used with a supervised learning model for classification. [27]'s the main limitation is the small and temporally sparse dataset used. Subsequent works, such as [28], addressed this problem by collecting additional data using smartphones. To better understand the impact of each feature in trajectory and its role in classification, [29] conducted a comprehensive study on trajectory features to study their impact on the classification task. Trajectories are prone to error or inaccuracies due to various factors, such as GPS signal interference, low-quality GPS devices, or inadequate sampling rates. The noise in the trajectory data can affect the accuracy and reliability of transportation mode classification and other analysis tasks that rely on trajectory data. [30] addressed this issue by using a noise removal technique.

2.2.2.2 Trajectories on Road Network

Trajectories on road network classification refers to categorizing or labeling the movement patterns of vehicles or individuals as they move through a road network. Trajectories on road network classification distinguishes itself from other trajectory classification tasks primarily because of its data. In trajectories on road network, the trajectory is mapped onto the road network, thereby converting it from a sequence of

spatial-temporal points to a sequence of road segments. The work [31] formally introduced the road vehicle classification problem and suggested a frequent pattern-based model tackling the classification task.[32] focused on capturing the sequential features of trajectories on a road network and proposed a neural network based approach

2.3 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).

2.3.1 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]. Despite their widespread use, these methods have limited capability of capturing long-term relationships between location data points, they are sensitive to noise and outliers in the data, which can lead to inaccurate results, and they are computationally expensive, therefore not suitable for real-time or large-scale

applications.

2.3.2 Conventional ML Methods

Conventional ML-based classification models, such as k-nearest neighbors (KNN) [33] and support vector machines (SVM) [34] 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.

2.3.3 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 [35] used a siamese neural network to capture semantic information of the trajectories.

2.4 Statistical Models for Trajectory Data

Statistical models are helpful in trajectory mining because they can capture relationships between spatial-temporal data points that do not occur sequentially. Statistical models, such as Collaborative Filtering are used to address human trajectory prediction task [36, 37]. These models are instrumental in applications where the data is high-dimensional and sparse, as they can effectively handle large amounts of data, which is the case for most trajectories datasets. Most trajectories data mining works that apply a classical model are focused on human trajectory prediction tasks, where the goal is the predict the next spatial-temporal point - referred to as the Point of Interest- that user will interact with.

Matrix Factorization is the most popular collaborative filtering method, which projects users and spatial-temporal points into a shared vector space. MF then estimates users' preference for a point by calculating the inner product between the user vector and the spatial-temporal point vector [38–40]. Another famous line of work is item-based neighborhood methods [41–43], which estimate a user's interest in a point via measuring its similarities with the user interactions history.

In recent years, deep learning-based methods have been developed to learn trajectory embeddings by leveraging neural networks. Two-layer Restricted Boltaman Machines was one of the earliest works to use Deep Learning for Collaborative Filtering

2.5 Deep Learning for Trajectory Data

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

2.5.1 Sequential Models

For many tasks it is customary to treat trajectory data as a type of sequential data that can be processed by sequence models. One of the earliest techniques to cap-

ture sequential patterns relied on Markov chains. For instance, [44] formalized the sequential recommendation as an optimization problem and then employed Markov Decision Proccies to address it. Other works tried to combine Matrix Factorisation with Markov chains into Factorzaing Personalized Markove Chaines to model sequential behaviors [45]. Also, high-order Markov chains were used to expand the size of the lookback window [46, 47].

Recurrent Neural Networks and, more specifically, Long Short -Term Memory (LSTM) became a popular solution for modeling user behavior sequences [48–50], and to generate trajectories using generative models [51].

Besides RNNs, various deep learning models are proposed for capturing sequential features and apply them in recommendations tasks [52–55]. The most famous ones are [55] which offer a convolutional sequence Model to learn the sequential patterns. [52] Uses memory network to enhance the performance of sequential recommendation. Additionally, attention-based models, such as Transformer models [56], 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 [57].

2.5.2 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) [58–61], clustering trajectories [62, 63], and analyzing movement behavior [64]. The foundation of these approaches involves learning low-dimensional representations of the trajectory data. One of the first methods towards this was “trajectory2vec” [62], which used a sliding window method to capture space- and time-invariant characteristics of trajectories. Similarly, “t2vec” [63] used a customized RNN with a spatial-loss function to address the low sampling rate and noisiness of the data. While other ap-

proaches focused solely on capturing the spatial-temporal properties of trajectories, Boonchoo et al. [65] 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 [66, 67]. A comprehensive review can be found in Wang et al. [23]. [68].

Another famous line of work is integrating deep learning-based methods with auxiliary information, e.g., text [69, 70] and images [71, 72].

Chapter 3

Preliminaries & Problem Definition

In this chapter, we briefly introduce some definitions and notations (a summary is provided in Table 3.1). Then we formally define the trajectory-user linking problem.

3.1 Preliminaries

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.*

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.

3.2 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 where an instance (a trajectory) belongs to one of the many classes (i.e., one user out of c):

$$\min_{f} \mathbb{E}[\mathcal{L}(f(Tr_i), u_i)] \text{ over } \mathcal{F}, \tag{3.1}$$

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.

Table 3.1: 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

Chapter 4

Generating Higher-order Mobility Flow Representations

In this chapter, 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.

4.1 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[1]. 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

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

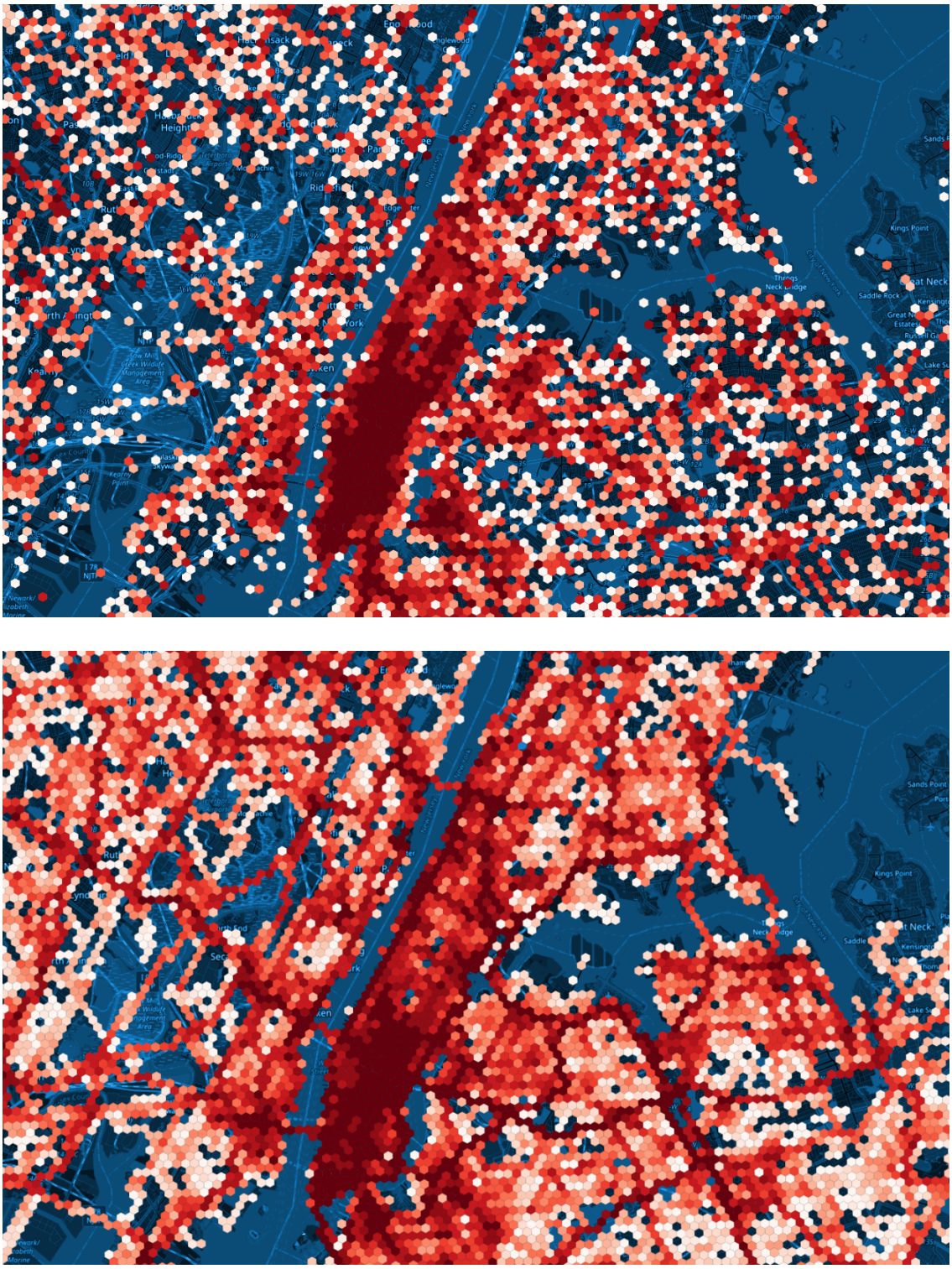
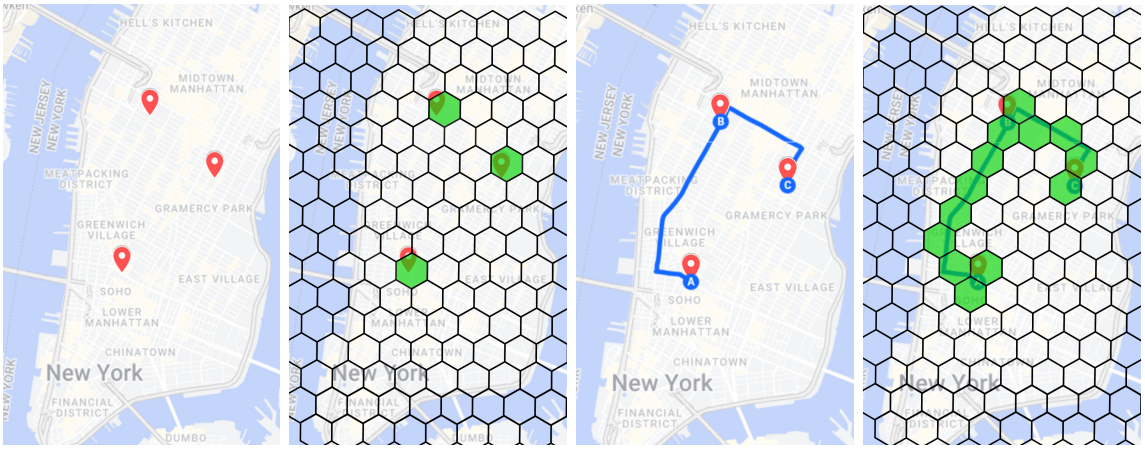


Figure 4.1: An illustrative example (FOURSQUARE-NYC) that shows how transforming higher-order check-in data (top) to higher-order mobility flow data (bottom) enriches the trajectory semantics and can help to address the TUL problem.



(a) (b) (c) (d)

Figure 4.2: 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.

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. 4.2b).*

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. Therefore, g represents a higher-order check-in.*

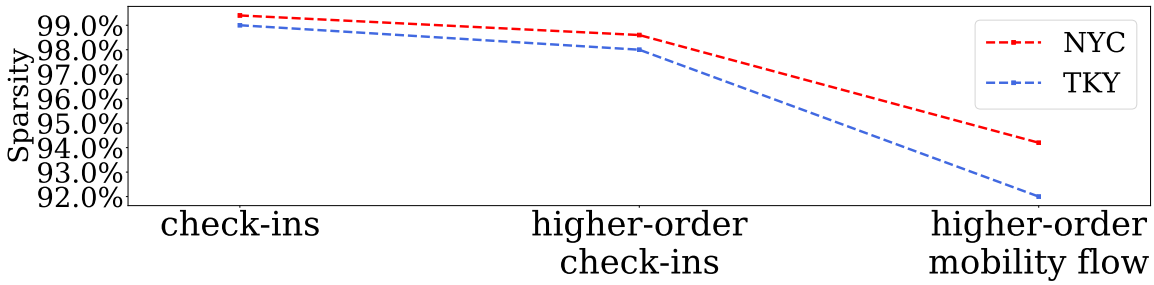


Figure 4.3: Impact of higher-order abstraction on sparsity.

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 $g_i \in \mathcal{G}$. Therefore, $Tr = \{g_1, g_2, ..., g_m\}$ represents a higher-order trajectory.*

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. 4.2 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. 4.1 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.

4.2 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. 4.3 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.

4.3 Implementation Considerations

This section provides detailed information about the tools used in this work.

4.3.1 Google Maps

Google Maps API is a powerful toolset provided by Google that enables developers to integrate maps, location-based services, and geolocation features into web or mobile applications. One feature of interest to our work is the Directions service, an API to get the route between two or more locations. The route generated is provided as an Encoded Polyline, which is a Google-specific method of representing polyline data in a compressed format that is easily transmitted over the internet. Instead of using a series of latitude and longitude coordinates to define the points along a polyline, encoded polylines use a series of encoded strings representing the offsets between consecutive points on the polyline. The dataset we have at hand includes only the check-ins made by users and does not encompass any information about the users' movements between those check-ins. To address this, we have utilized the Directions service to produce the trajectories, as depicted in figures 4.4 & 4.5.

A significant limitation of Google API is privacy laws prohibiting sharing the generated routes. Additionally, the API incurs a cost and is not free of charge. To this end, we suggest using open-source routing APIs, such as OSRM[2] and Graphhopper[3].

[2]https://project-osrm.org/
[3]https://www.graphhopper.com/

Figure 4.4: Trajectories for Foursquare-NYC

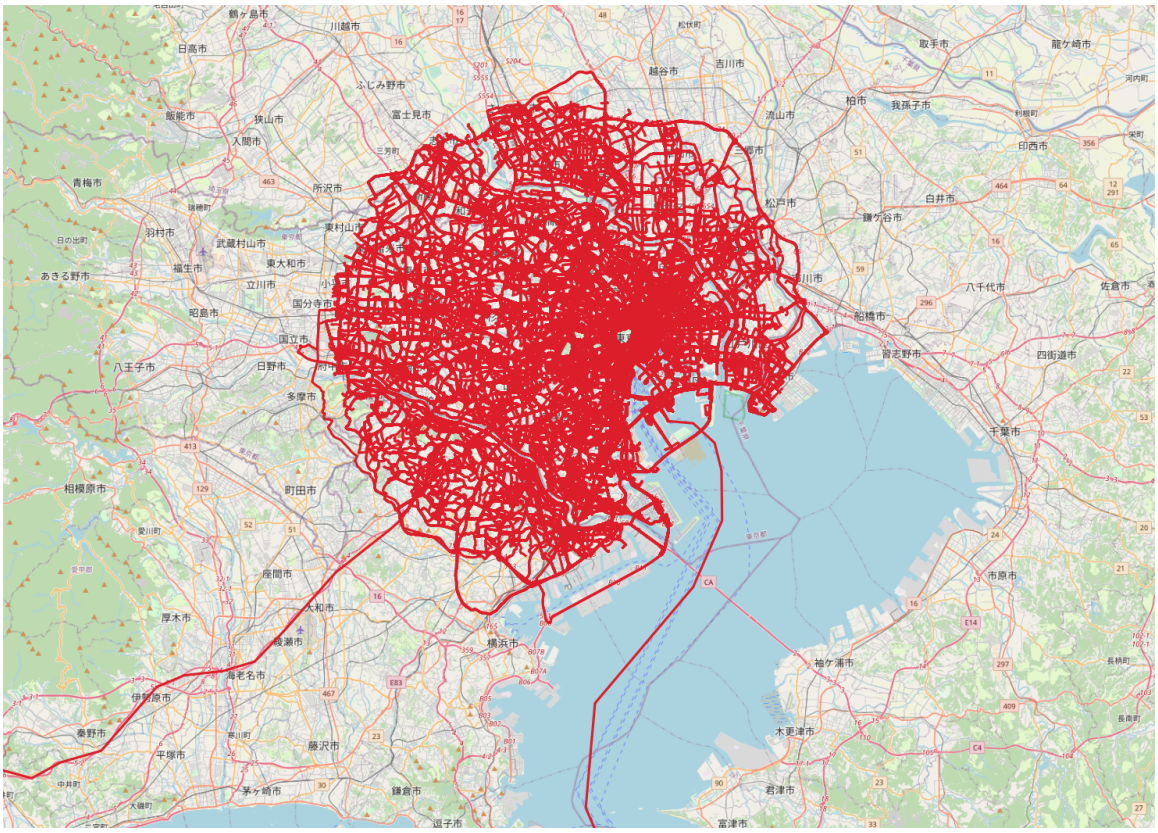
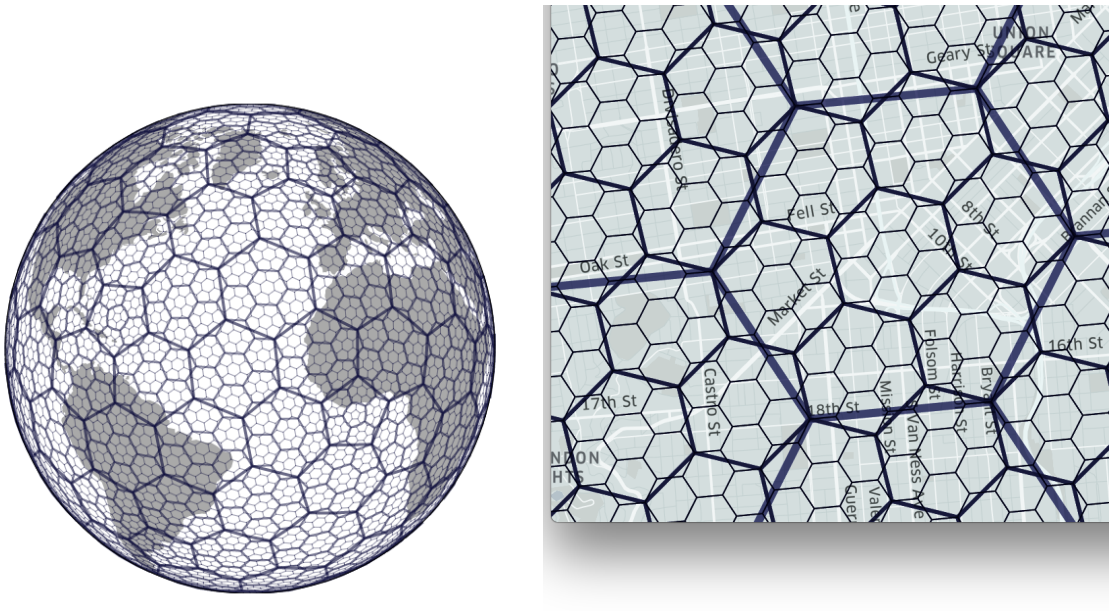


Figure 4.5: Trajectories for FOURSQUARE-TKY

4.3.2 H3



(a) H3 partition of the globe into Hexagons.

(b) H3 Hexagons hierarchy

H3[4] is an open-source geospatial indexing system developed by Uber. The library provides a way to represent and organize geographic coordinates into a hierarchical grid system. H3 divides the earth's surface into hexagons of various sizes, which can be nested within each other to create a hierarchical structure. The hexagons are defined by a unique index, which can be used to identify a specific location on the earth's surface. H3 provides a range of tools and functions to manipulate and analyze data based on the hexagonal grid system.

We use H3 to obtain the higher-order check-ins and higher-order mobility flow. Specifically, we utilize the `latLngToCell(lat, lng, resolution)` function, which takes as inputs a latitude, longitude, and resolution and outputs the hexagon that encompasses the corresponding location at the specified resolution. Generating higher-order check-ins is easy compared to higher-order mobility flow. As mentioned previously, the trajectory data consists of encoded polylines. Regrettably, H3 does not have built-in compatibility for this format. Nevertheless, we have leveraged a community-

[4]https://h3geo.org/

developed solution[5] that expands the functionality of the H3 library to enable the intersection of polylines with H3 hexagons. This capability is necessary for generating the higher-order mobility flow.

[5]https://github.com/DahnJ/H3-Pandas/issues/11

Chapter 5

Modeling Trajectory-user Linking

In this chapter, we present details of our spatiotemporal deep-learning model, TULHOR, that utilizes higher-order mobility flow data to accurately address the TUL problem. TULHOR is composed of three components: (**A**) a spatial embedding layer, (**B**) a spatial-temporal embedding layer, and (**C**) an encoder (see Fig. 5.1). 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.

5.1 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 [73].

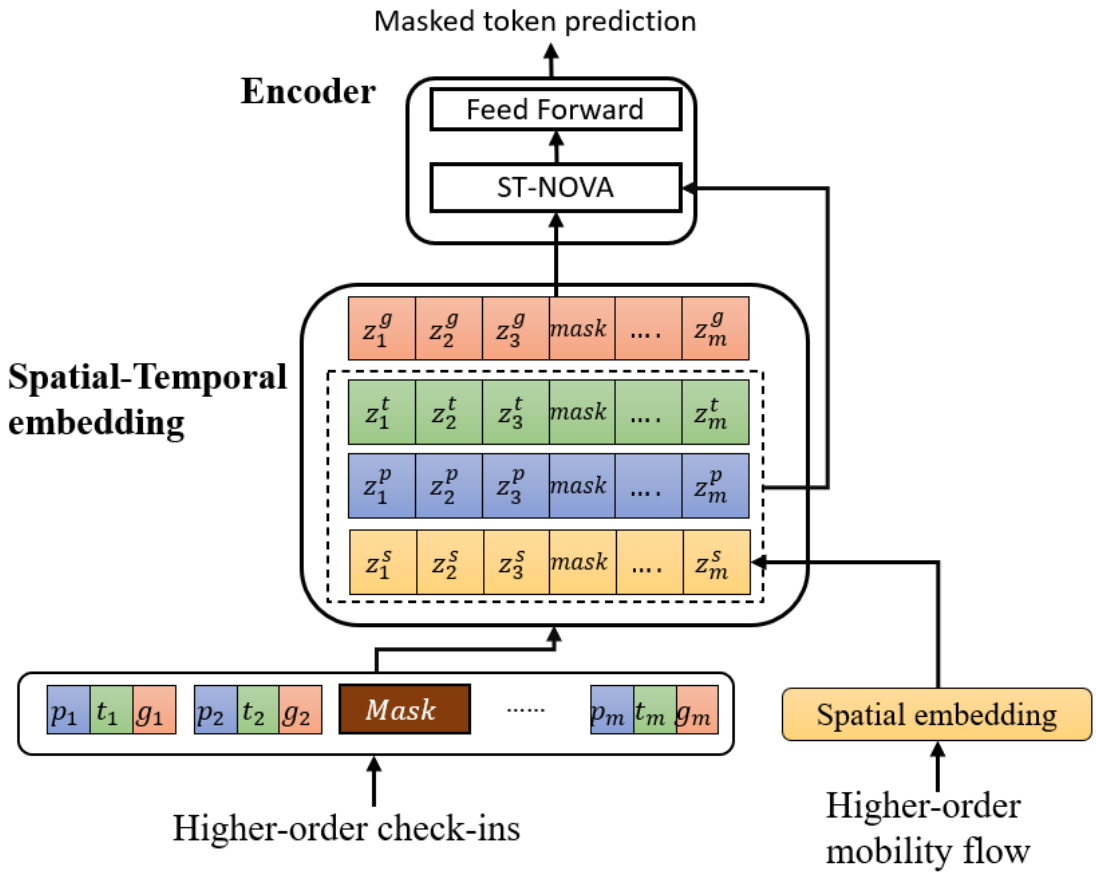


Figure 5.1: High-level architecture of TULHOR.

5.1.1 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.

5.1.2 Learning Node Representations of the Hexagon-lattice Graph

In this step, we employ the `node2vec` model [74] 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.

5.2 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:

$$z_i^g = \phi_g(g_i, W_g) \tag{5.1}$$

$$z_i^p = \phi_p(p_i, W_p) \tag{5.2}$$

$$z_i^s = \phi_s(g_i, W_s) \tag{5.3}$$

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 5.1 and 5.3 use the same input, g_i, but while equation 5.1 learns the semantic embedding of cells during training, equation 5.3 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 [75], we design a temporal-aware positional encoding to replace the positional encoding used in the original BERT model with:

$$[z_i^t]_j = \begin{cases} \sin(w_j t_i), & \text{if } j \text{ is odd} \\ \cos(w_j t_i), & \text{if } j \text{ is even} \end{cases} \tag{5.4}$$

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:

$$(z_i^t)(z_{i+1}^t)^\intercal = \sum_{i=1}^{d} \cos(w_i(t_{i+1} - t_i)) \tag{5.5}$$

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 [76] 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:

$$R^{(id)} = z_1^g, z_2^g, ..., z_m^g \tag{5.6}$$

$$R = (\{z_1^s, z_2^s, ..., z_m^s\}, \{z_1^p, z_2^p, ..., z_m^p\}, \{z_1^t, z_2^t, ..., z_m^t\}) \tag{5.7}$$

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 5.1. 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.

5.3 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. For the model to attend to all these different dependencies, the self-attention mechanism uses multiple heads, allowing the model to capture various dependencies in parallel. Following the multi-head self-attention component, there is a position-wise feed-forward network with `ReLu` activation function to introduce non-linearity. Next, there is a residual connection, which allows the gradient to flow through the model without exploding or vanishing, making the training stable. The training is further stabilized using layer normalization.

Note that the multi-head spatial-temporal non-invasive self-attention (ST-NOVA) in TULHOR differs from the standard self-attention (SA) found in Transformer models. The standard self-attention is represented as:

$$SA(Q, K, V) = \sigma(\frac{QK^T}{\sqrt{d_n}})V \tag{5.8}$$

where $Q, K, V \in \mathbb{R}^{m \times d_n}$, d_n is the hidden state embedding dimensions, and σ is the softmax operation. SA calculates the weighted average for each token in the sequence based on its corresponding similarity with the other tokens in the same sequence. SA

uses an invasive-attention, implying that any additional features, such as positional information, must be infused into the input sequence representation. This has a major drawback because the output of the self-attention layer is fed to the predicting layer, which tries to search the token ID space; if we were to add additional features to the sequence representation, then we would end up creating a compounded embedding space, which makes the searching task harder. To address these problems, we use a non-invasive attention instead, which is represented as:

$$NOVA(R^{(id)}, R) = \sigma(\frac{QK^T}{\sqrt{d_L}})V \tag{5.9}$$

$$V = R^{id} \times W_V, K = F \times W_k, Q = F \times W_Q \tag{5.10}$$

$$F = MLP(R^{(id)}||R) \tag{5.11}$$

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 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.

5.4 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 [77] 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:

$$\mathcal{L}_{MLM} = \frac{1}{|Tr^m|} \sum_{g_m \in Tr^m} -\log P(g^m = g^{m*} | Tr^{m'}) \tag{5.12}$$

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.

5.5 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:

$$y' = (W_C \cdot h^{Tr} + b_c) \tag{5.13}$$

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:

$$\sigma(y'_i) = \frac{e^{y'_i}}{\sum_{j=1}^{|\mathcal{U}|} e^{y'_j}} \quad for\ i = 1, 2, \ldots, |\mathcal{U}| \tag{5.14}$$

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 [78], with backpropagation to train our model. Given unlinked trajectory Tr generated by u_i, then the loss is represented as:

$$\mathcal{L}(Tr, u_i) = \frac{1-\beta}{1-\beta^{n_{u_i}}} log(\sigma(y')) \tag{5.15}$$

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.

Note as well that the TULHOR is trained on a set of existing users (i.e., classes) and its purpose is to classify new instances into one of the existing users (i.e., class). If the TULHOR encounters a new instance that doesn't fit into any of the existing users (i.e., classes), it cannot make a prediction. In some cases, it is possible to retrain the our model with new users (classes) or to modify the existing users (classes) to include the new instance. However, this would require retraining the model with new data and modifying its architecture accordingly.

Chapter 6

Experimental Evaluation

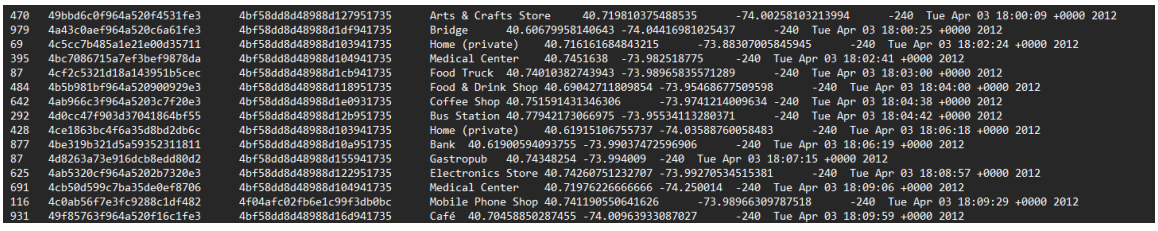
This section provides an overview of the datasets, then presents the baseline methods and elaborates on implementation details.

6.1 Datasets

We perform experiments on two real-world datasets, the New York (NYC) and Tokyo (TKY) check-in datasets from the FOURSQUARE[1] social network, consisting of check-ins from 2012-2013. The dataset is in tab-separated values(tsv) format, where each row represents a check-in/record, as seen in 6.1. Each check-in contains:

- User ID (anonymized)
- Venue ID (Foursquare)
- Venue category ID (Foursquare)
- Venue category name (Foursquare)

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



470	49bbd6c0f964a520f4531fe3	4bf58dd8d48988d127951735	Arts & Crafts Store	40.719810375488535	-74.00258103213994	-240	Tue Apr 03 18:00:09 +0000 2012
979	4a43c0aef964a520c6a61fe3	4bf58dd8d48988d1df941735	Bridge	40.606799581406 43	-74.04416981025437	-240	Tue Apr 03 18:00:25 +0000 2012
69	4c5cc7b485a1e21e00d35711	4bf58dd8d48988d103941735	Home (private)	40.716161684843215	-73.88307005845945	-240	Tue Apr 03 18:02:24 +0000 2012
395	4bc7086715a7ef3bef9878da	4bf58dd8d48988d104941735	Medical Center	40.7451638	-73.982518775	-240	Tue Apr 03 18:02:41 +0000 2012
87	4cf2c5321d18a143951b5cec	4bf58dd8d48988d1cb941735	Food Truck	40.74010382743943	-73.98965835571289	-240	Tue Apr 03 18:03:00 +0000 2012
484	4b5b981bf964a520900929e3	4bf58dd8d48988d118951735	Food & Drink Shop	40.69042711809854	-73.95468677509598	-240	Tue Apr 03 18:04:00 +0000 2012
642	4ab966c3f964a5203c7f20e3	4bf58dd8d48988d1e0931735	Coffee Shop	40.751591431346306	-73.9741214009634	-240	Tue Apr 03 18:04:38 +0000 2012
292	4d0cc47f903d37041864bf55	4bf58dd8d48988d12b951735	Bus Station	40.779421730666975	-73.95534113280371	-240	Tue Apr 03 18:04:42 +0000 2012
428	4ce1863bc4f6a35d8bd2db6c	4bf58dd8d48988d103941735	Home (private)	40.61915106755737	-74.03588760058483	-240	Tue Apr 03 18:06:18 +0000 2012
877	4be319b321d5a59352311811	4bf58dd8d48988d10a951735	Bank	40.61900594093755	-73.99037472596906	-240	Tue Apr 03 18:06:19 +0000 2012
87	4d8263a73e916dcb8edd80d2	4bf58dd8d48988d155941735	Gastropub	40.74348254	-73.994009	-240	Tue Apr 03 18:07:15 +0000 2012
625	4ab5320cf964a5202b7320e3	4bf58dd8d48988d122951735	Electronics Store	40.74260751232707	-73.99270534515381	-240	Tue Apr 03 18:08:57 +0000 2012
691	4cb50d599c7ba35de0ef8706	4bf58dd8d48988d104941735	Medical Center	40.71976226666666	-74.250014	-240	Tue Apr 03 18:09:06 +0000 2012
116	4c0ab56f7e3fc9288c1df482	4f04afc02fb6e1c99f3db0bc	Mobile Phone Shop	40.741190550641626	-73.9896630978 7518	-240	Tue Apr 03 18:09:29 +0000 2012
931	49f85763f964a520f16c1fe3	4bf58dd8d48988d16d941735	Café	40.70458850287455	-74.00963933087027	-240	Tue Apr 03 18:09:59 +0000 2012

Figure 6.1: snippet of Foursquare-NYC

- Latitude
- Longitude
- Timezone offset in minutes (The offset in minutes between when this check-in occurred and the same time in UTC)
- UTC time

We form a check-ins trajectory by grouping user check-ins for each date where "date" refers to the year, month, and day. When creating a check-ins trajectory, the user check-ins are first grouped by date, which means that all the check-ins that occurred on the same day are combined and then merged in a temporally ascending order starting with the earliest check-in and ending with the latest check-in. 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 6.1 provides the statistics of the datasets.

Table 6.1: Statistics of two FOURSQUARE datasets (FOURSQUARE-NYC and FOURSQUARE-TKY) including the # of trajectories for three user groups (108, 209, all).

Dataset	$\lvert\mathcal{U}\rvert$	$\lvert\mathcal{T}\rvert$
Foursquare-NYC	108	6795
	209	9,637
	234	10,133
Foursquare-TKY	108	9343
	209	14,151
	451	20,964

6.2 Baselines Methods

We evaluate TULHOR against the following baselines:

- Linear Discriminant Analysis (LDA): Is a supervised machine-learning method for classification which aims to project the high-dimensional input data onto a lower-dimensional space while preserving class-discriminatory information. We use Bag-of-Words (BOW) to embed the trajectories and then apply Singular Value Decomposition to reduce the dimensionality of the embeddings (SVD).

- Decision Tree (DT):The decision tree algorithm is a non-parametric method for machine learning. It learns decision rules based on the data features and can be used for classification purposes.

- Support Vector Machine (SVM): is a supervised machine learning technique that seeks to learn a hyperplane capable of separating data points. It accomplishes this by mapping the data to a higher dimensional space and then learning the hyperplane. There are three distinct variations of SVM: Linear-SVM, which learns a linear hyperplane, poly-SVM, which learns a polynomial hyperplane, and RBF-SVM, which learns a radius-based hyperplane.

- 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.

- 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.

6.2.1 Implementation Details

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.

6.3 Evaluation Metrics

For evaluating the performance of the TUL models, we use the following metrics from the multiclass classification domain:

- Accuracy@k: Metric to measure how well a TUL model can identify and label unknown trajectories. It measures the proportion of correct predictions within the top k-ranked labels. For this work, we use acc@k with k=1 and k=5, represented by the formula:

$$ACC@k = \frac{(\text{\# of correct predictions within top k ranked items or labels})}{k} \tag{6.1}$$

- Macro-P: In TUL each trajectory is assigned to a user or label, and the goal of the linking algorithm is to correctly identify the user for each trajectory. Macro-precision, in this case, is the average precision across all the users or labels in the dataset, where precision is the ratio of true positives to the sum of true positives and false positives for each user or label.

  To calculate macro-precision in the context of TUL, we first calculate the precision for every user(label) using the formula mentioned earlier:

$$\text{Precision of a User} = \frac{TP}{(TP + FP)} \tag{6.2}$$

Here, TP refers to the number of true positive trajectories (trajectories correctly linked to the user), while FP is the number of false positive trajectories (trajectories incorrectly linked to the user). Once the precision for each user or label is calculated, the macro-precision is obtained by taking the average of all the precision values:

$$Macro - P = \frac{\sum_{i=1}^{\mathcal{U}} \text{Precision of i}}{|\mathcal{U}|} \tag{6.3}$$

A high macro-precision score indicates that the linking algorithm is performing well in correctly identifying the user for each trajectory, while a low macro-precision score implies that the linking algorithm is struggling to identify the correct user, leading to a high number of false positives.

- Macro-Recall: is the average of recall for all users or labels in the datasets, where recall is the ratio of true positives to the sum of true positives and false negatives for each user or label. To calculate macro-Recall in the context of TUL, we first calculate the Recall for every user(label) using the formula mentioned earlier:

$$\text{Recall of a User} = \frac{TP}{(TP + FN)} \tag{6.4}$$

Where FN is the number of false negative trajectories (trajectories belonging to the user but incorrectly linked to other users). The Macro-Recall is obtained by taking the average of all the Recall values:

$$Macro - R = \frac{\sum_{i=1}^{\mathcal{U}} \text{Recall of i}}{|\mathcal{U}|} \tag{6.5}$$

A high macro-recall score implies that the linking algorithm is correctly identifying the majority of the trajectories that belong to a particular user or label, with a relatively low number of false negatives. This means that the linking algorithm is able to effectively distinguish between different users and accurately identify their trajectories.

- F1-Score:is an evaluation metric that combines both Macro-Recall and Macro-P and is computed as the harmonic mean of the Macro-P and Macro-Recall scores. A high F1 score indicates that the model achieves high precision and recall simultaneously. It accurately identifies positive instances while minimizing false positives(trajectories incorrectly linked to a user or label) and negatives (trajectories belonging to a user or label but incorrectly linked to another user or label). The formula for F1-score is:

$$F1 = 2 * \frac{Macro - P * Macro - R}{Macro - P + Macro - R} \tag{6.6}$$

6.4 Experimental Results

6.4.1 Accuracy 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 6.2 and 6.3, 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 2.53%, 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.

Table 6.2: 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														
	$\|\mathcal{U}\| = 108$					$\|\mathcal{U}\| = 209$					$\|\mathcal{U}\| = 234$				
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
POLY-SVM	0.640	0.712	0.916	0.556	0.657	0.478	0.586	0.678	0.440	0.509	0.495	0.584	0.634	0.415	0.480
RBF-SVM	0.885	0.932	0.949	0.850	0.885	0.763	0.842	0.760	0.673	0.702	0.724	0.797	0.697	0.600	0.632
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%

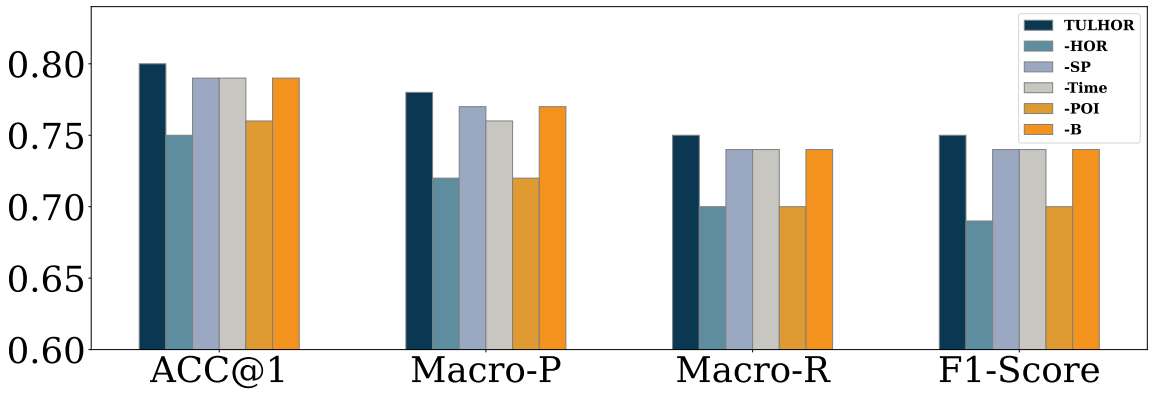
Table 6.3: Results on FOURSQUARE-TKY mobility dataset. The highest performance is indicated in bold and the second best performance has been underlined. 'Improvement' denotes the improvement of TULHOR model over the strongest baseline.

	Foursquare-TKY														
	$\|\mathcal{U}\| = 108$					$\|\mathcal{U}\| = 209$					$\|\mathcal{U}\| = 451$				
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
POLY-SVM	0.716	0.791	**0.954**	0.602	0.706	0.581	0.686	0.765	0.483	0.564	0.432	0.539	0.528	0.321	0.375
RBF-SVM	0.890	0.948	0.914	0.873	0.885	0.772	0.872	0.787	0.713	0.732	0.598	0.738	0.584	0.487	0.504
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%	-1.78%	0.71%	0.37%	2.7%	1.77%	1.33%	2.53%	1.30%	7.86%	7.47%	9.52%	9.53%	8.11%

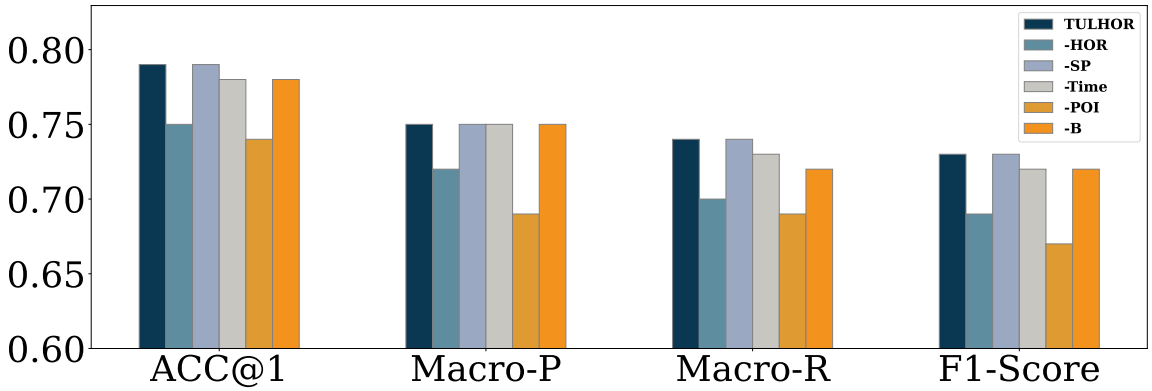
6.4.2 Ablation Study

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 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 ablation study is conducted across different tessellation levels to evaluate each component's impact as grid cell size increases. We discuss the ablation study for each tessellation separately before extrapolating how each component behaves as the grid changes in size. The results of the ablation study are shown in Figure 6.2.

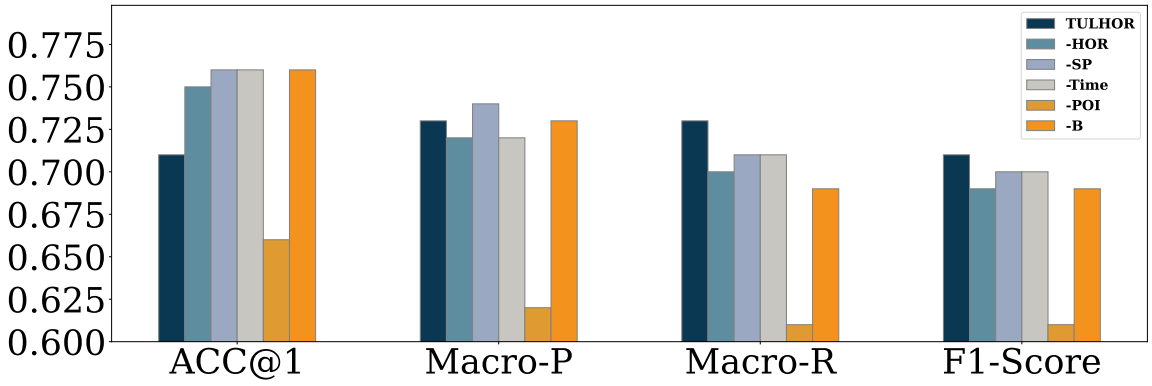
Hex@9: Observing Figure 6.2a, where we notice that the removal of any of the components results in a decrease in performance of the model, with varying degrees. The results indicate that the inclusion of higher-order information in Hex@9 is crucial for the model's performance, as evidenced by the fact 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



(a) Hex@9 tessellation



(b) Hex@8 tessellation



(c) Hex@7 tessellation

Figure 6.2: Results of ablation experiments of TULHOR model for FOURSQUARE-TKY, $|\mathcal{U}| = 451$ on different tessellation levels

when dealing with imbalanced datasets, which is characteristic of the TUL datasets.

Hex@8: Upon examination of the Figure 6.2b, it is apparent that removing any component, excluding **-SP**, leads to a reduction in performance. Notably, including or excluding spatial embedding (**-SP**) produces equivalent performance levels across all metrics because as the cell size increases, it becomes increasingly challenging for the grid to capture the physical characteristics of the map. Specifically, in Hex@8, the cell can encompass an entire block, which may comprise numerous roads. The presence of multiple physical features within a single cell may hinder the model's ability to differentiate between them, thereby impeding its trajectory classification capabilities. This may explain why the inclusion or exclusion of **-SP** from TULHOR in Hex@8 does not impact the model's performance. We notice that **-Time**, **-POI**, and **-B** result in similar performance declines as in the previous tessellation.

Hex@7:The results of the ablation study conducted on Hex@7 are presented in Figure 6.2c. The findings indicate that TULHOR performs best in terms of Macro-Recall and F1-score. However, compared to **-HOR**, **-Time**, **-SP**, and **-B**, TULHOR falls short in Acc@1. This is because as trajectories are represented on Higher-order (Hex@7) where each cell covers a significant spatial size, all the trajectories will look similar, making it harder for the model to classify the trajectory accurately. We notice a significant decrease in performance for **-POI** variation since POIs can be used to distinguish different trajectories and patterns. Removing them will make the linking task harder, explaining the noticeable decrease in performance for **-POI** variation.

When we examine various tessellation options, we notice that the **-POI** variation's performance worsens as the size of the grid cell increases. This is not surprising, as POIs are useful for differentiating between visits to the same cell. The decline in performance is most noticeable in the lower levels of tessellation. In contrast, we observe that **-HOR** maintains consistent performance across all tessellation levels because it eliminates the higher-order generation step. With **-SP**, we observe a decline in performance, particularly in the Hex@9 settings. However, there seems to

be little reason to use spatial embedding at lower tessellation levels such as Hex@7 and Hex@8, as the grid cell becomes too large to capture the finer details of the map's spatial properties. Finally, removing time and balancing (**-Time** and **-B**) leads to reduced performance across all tessellation levels.

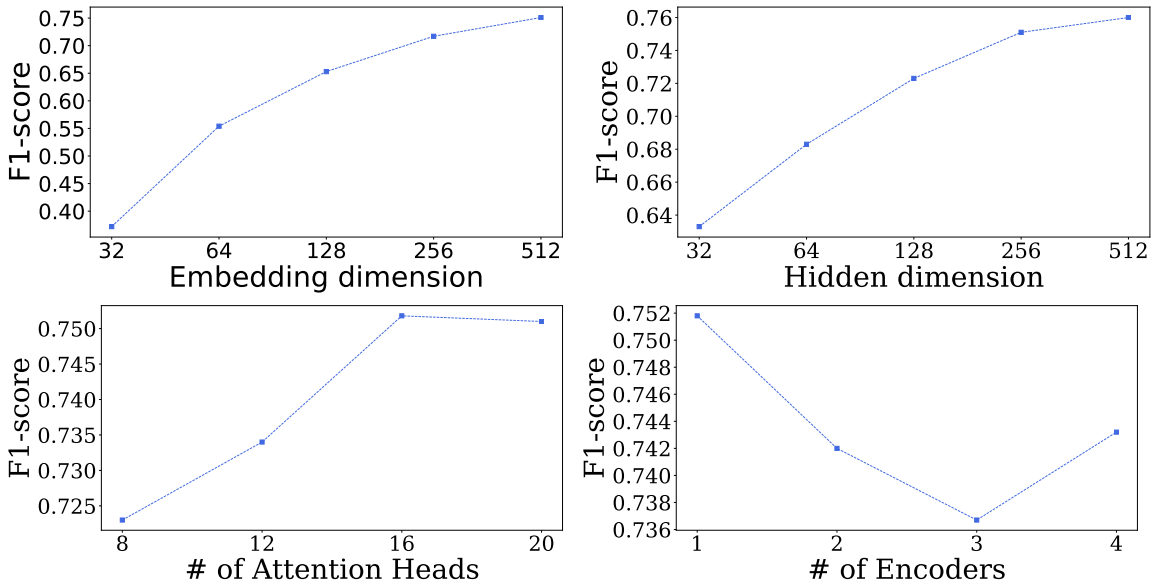


Figure 6.3: Results of varying hyperparameters on TULHOR model for FOURSQUARE-TKY, $|\mathcal{U}| = 451$.

6.4.3 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 6.3. 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.

Table 6.4: Statistics about different tessellations with the # of cells and cell size for each resolution for FOURSQUARE-TKY

Resolution	**# of cells**	**cell size** (km^2)
Hex@7	334	5.160
Hex@8	2,003	0.730
Hex@9	11,036	0.015

Table 6.5: 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	0.943	0.832	0.817	0.815	0.711	0.883	0.734	0.734	0.711
Hex@8	0.926	**0.977**	0.925	0.917	0.917	0.868	0.940	0.862	0.849	0.849	0.790	0.884	0.753	0.740	0.733
Hex@9	**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**

Table 6.6: 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	379	5.160
Hex@8	2,181	0.730
Hex@9	11,343	0.015

Table 6.7: Results of impact of different grid sizes on the performance of TULHOR on FOURSQUARE-NYC dataset.

	Foursquare-NYC														
	#users = 108					**#users = 209**					**#users = 234**				
METHOD	ACC@1	ACC@5	P	R	F1	ACC@1	ACC@5	P	R	F1	ACC@1	ACC@5	P	R	F1
Hex@7	0.785	0.883	0.795	0.755	0.760	0.681	0.789	0.649	0.630	0.621	0.609	0.742	0.558	0.530	0.523
Hex@8	0.808	0.886	0.814	0.782	0.787	0.692	0.808	0.678	0.6489	0.647	0.642	0.765	0.628	0.597	0.592
Hex@9	**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**

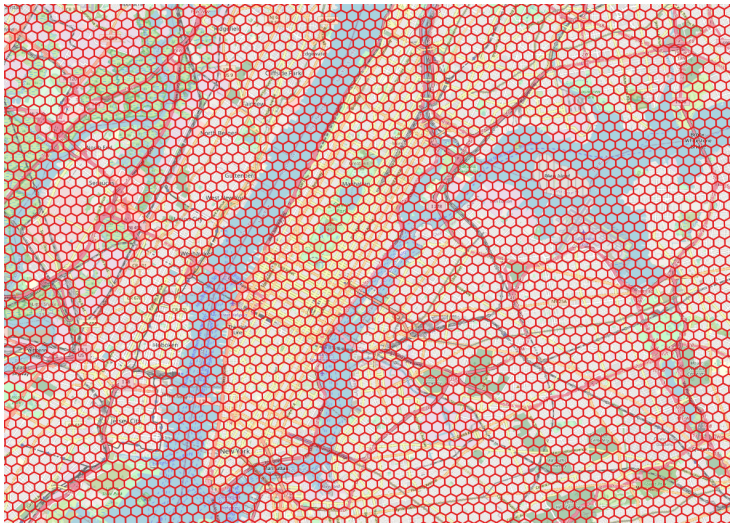
6.4.4 Impact of Grid Cell Size

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 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 6.4 and 6.6 provides the statistics about the different tessellations. The results of this experiment are presented in Table 6.5 and 6.7 .

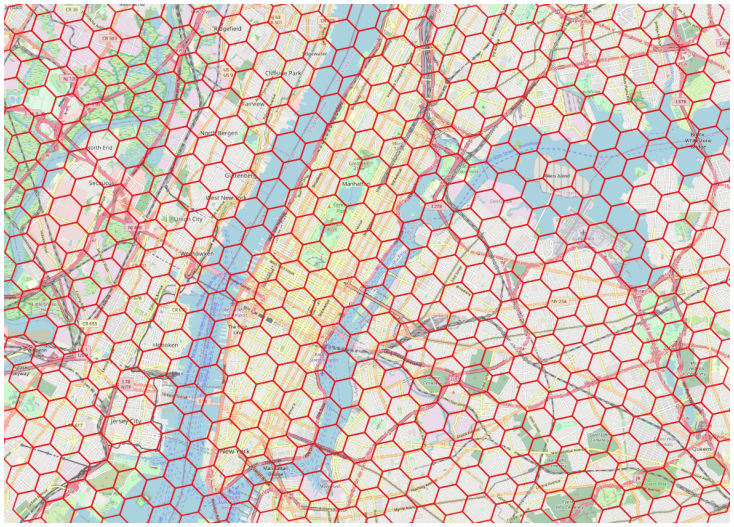
Foursquare-TKY:For the 108 user setting, a 2% decrease in macro F1 is observed for Hex@8 and Hex@7, with slightly higher accuracy for Hex@9. For 209 users, the gap in performance between Hex@9 and other tessellations grows, reaching a 6% difference in macro F1 for Hex@7 and a 3% difference for Hex@8, 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.

Foursquare-NYC:Hex@9 achieves the best performance across different user settings. The gap in performance between Hex@9 and other tessellation, i.e., Hex@8 and Hex@7, widen as the number of users grows. For example, Hex@9 outperforms Hex@8 in the 108 users settings by 14.5% in F1, while for the 209 users, the difference in F1 is 22.9%, and finally, the difference grows to 26.8% in the 234 user settings. These results are consistent across other performance metrics as well.

To conclude, as the cell size increases, capturing user movement patterns becomes increasingly challenging, as seen in the results.



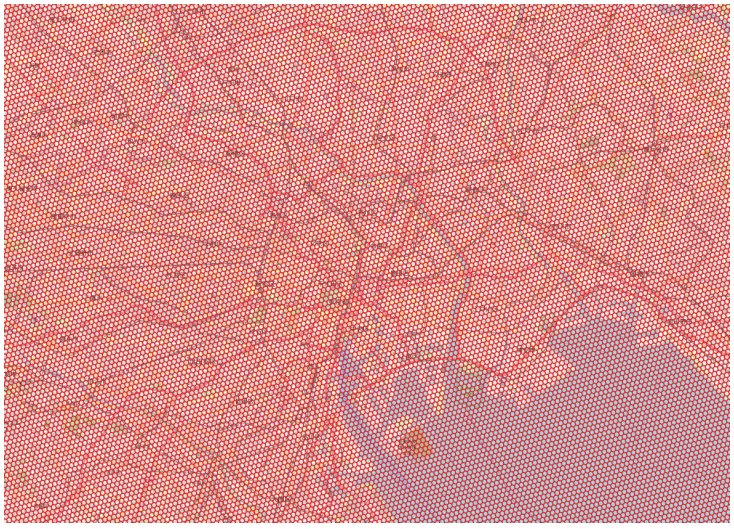
(a) Hex@9 tessellation



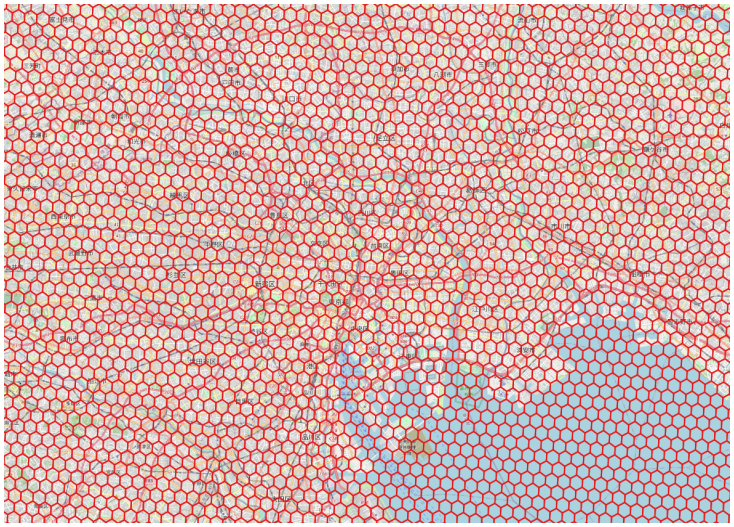
(b) Hex@8 tessellation

(c) Hex@7 tessellation

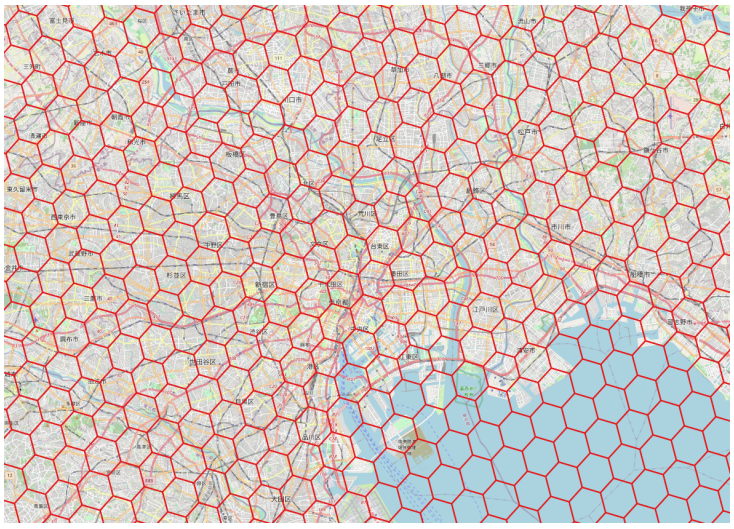
Figure 6.4: Different tessellations levels for FOURSQUARE-NYC dataset.



(a) Hex@9 tessellation



(b) Hex@8 tessellation



(c) Hex@7 tessellation

Figure 6.5: Different tessellations levels for FOURSQUARE-TKY dataset.

6.4.5 Impact of Sparsity

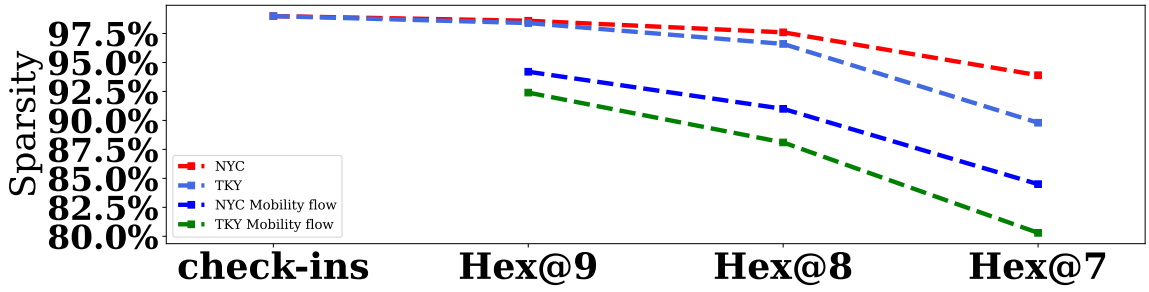


Figure 6.6: The sparsity of check-ins, High-order check-ins, and High-order mobility flow across different tessellations For Foursquare-NYC & Foursquare-TKY

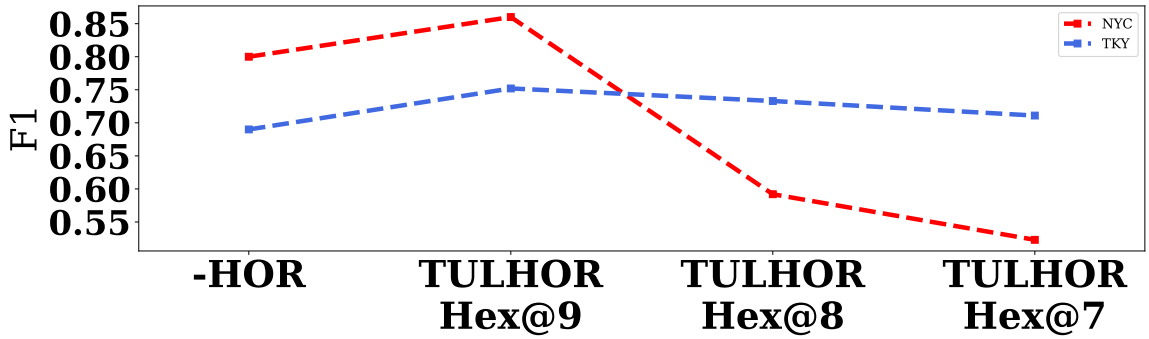


Figure 6.7: The F1 score of -HOR and TULHOR across different tessellations. The -HOR variation utilizes raw check-ins and acts as an indicator of the impact of sparsity.

We carried out an additional study to investigate how sparsity affects TULHOR and how to determine the appropriate tessellation resolution. We plotted the sparsity of high-order check-ins and mobility flow in NYC and TKY across various tessellations as shown in Figure 6.6. As grid cells become larger, the sparsity decreases because each cell covers a larger spatial area. To understand how changes in sparsity affect model performance, we also plotted the sparsity of raw check-ins and the F1 score of TULHOR across different tessellations.To provide a comparison metric, we also plotted the performance of the -HOR variation in Figure 6.7, which removes the higher-order generation step and reduces the model to a BERT architecture that processes trajectory data without high-order information. This allows us to compare the approach to other baseline models, where the input is raw check-ins.

We observed that a small reduction in sparsity can lead to a substantial improvement in performance. For instance, a 1% reduction in sparsity can increase the F1 score by more than 5%. However, further reductions in sparsity have a negative impact on model performance. For example, a 2% reduction in sparsity due to the switch from Hex@9 to Hex@8 results in a 2% decrease in F1 for TKY and a 20% decrease in F1 for NYC. The decline in performance is more pronounced in NYC since most of the data is concentrated in Manhattan, which requires only a few Hex@8 grid cells to cover.

A decrease in sparsity implies an increase in the density of the interaction matrix, making it challenging for the model to capture individual patterns. Therefore, choosing an appropriate resolution requires selecting a resolution that reduces sparsity without significantly increasing density and confusing the model.

Chapter 7

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. We proposed a method for generating higher-order check-ins and mobility flow data to address some of the data quality and sparsity challenges. The method creates a hexagonal tessellation of a map. Then it projects both check-ins and mobility flow data to the higher spatial order to obtain high-order check-ins and high-order mobility flow. The high-order data serves as an input to TULHOR, where we use high-order mobility flow data to learn the spatial properties of grid cells and capture the physical properties of the map, while the higher-order check-ins serve as input to the embedding layer in TULHOR. We adopted a non-invasive attention approach in the encoding layer to combine the spatial and temporal features. The results of extensive experiments over two real-world datasets demonstrated the effectiveness of the proposed model, which consistently outperforms several strong baselines. We achieved a 7.19% improvement over SOTA in the F1 metric on NYC and 8.11% on TKY. TULHOR outperformed other baselines across all metrics, achieving a 6.61% improvement in Macro-P on NYC and a 9.52% improvement on TKY. We conducted an ablation study of different tessellation levels to evaluate the impact of each component of our model at each tesselation level. Finally, we ran a parameter study to explore the impact of hyperparameters and grid cell size.

7.1 Future Work

7.1.1 Multi-trajectory User Linking

The TUL task involves processing a single trajectory to identify the user it belongs to. In contrast, the Multi-trajectory user linking approach involves handling multiple trajectories, all attributable to an unknown user, to identify the said user. The primary obstacle in Multi-trajectory user linking lies in effectively capturing the extended spatial-temporal interdependencies spanning the multiple trajectories and associating them with their respective user. However, the adaptable nature of TULHOR, which leverages Transformer-based architectures known for their proficiency in handling lengthy sequences, renders it a viable candidate for addressing Multi-trajectory user linking.

7.1.2 Trajectory-user Linking on the Roads Network

In the context of Trajectory-user Linking on Road Networks, the trajectory under consideration is composed of road segments that describe the path traversed by the user on the underlying road network, as opposed to the more conventional trajectories that are composed of check-ins. This departure from the norm has the potential to introduce novel insights into the analysis of user behavior and mobility patterns. Using road segments in trajectory representation enables a more granular and detailed account of the user's movements, thereby affording greater accuracy in user identification tasks. An intriguing avenue for future research would be to investigate the efficacy of incorporating high-order trajectory information in the context of road segments.

7.1.3 Point-of-Interest Recommendation

The point-of-interest recommendation is the task of predicting the next place that a user will visit. It is a widely known and researched problem with various applications. TULHOR can be extended with ease to address the POI recommendation task, as all

it requires is to change how training and fine-tuning are done. It will be interesting to explore the impact of incorporating higher-order mobility flow data and how it can facilitate the recommendation task.

7.1.4 Mixed Tessellations

All tessellation techniques divide the spatial map into an equally-sized polygon. A new avenue for future research is exploring the impact of mixed-sized tessellation. The advantage of using mixed-sized tesselation is the ability to control what is included in each polygon. For example: in our case, some polygons have more POI than others, while other polygons might only have one. The proposed solution is to divide the polygons with a high concentration of POI and merge those with fewer POI. This approach should create a balanced distribution of POI across polygons, leading to improved performance.

Bibliography

[1] D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series.," in *KDD workshop*, Seattle, WA, USA: vol. 10, 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] M. Alsaeed, A. Agrawal, and M. Papagelis, "Trajectory-user linking using higher-order mobility flow representations," in *2023 24th IEEE International Conference on Mobile Data Management (MDM)*, IEEE, 2023, pp. 1–10.

[9] F. Arasteh, S. SheikhGarGar, and M. Papagelis, "Network-aware multi-agent reinforcement learning for the vehicle navigation problem," in *Proceedings of the 30th International Conference on Advances in Geographic Information Systems*, 2022, pp. 1–4.

[10] A. Nematichari, T. Pechlivanoglou, and M. Papagelis, "Evaluating and forecasting the operational performance of road intersections," in *Proceedings of the 30th International Conference on Advances in Geographic Information Systems*, 2022, pp. 1–12.

[11] T. Pechlivanoglou, G. Alix, N. Yanin, J. Li, F. Heidari, and M. Papagelis, "Microscopic modeling of spatiotemporal epidemic dynamics," in *Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Spatial Computing for Epidemiology*, 2022, pp. 11–21.

[12] T. Pechlivanoglou, J. Li, J. Sun, F. Heidari, and M. Papagelis, "Epidemic spreading in trajectory networks," *Big Data Research*, vol. 27, p. 100 275, 2022.

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

[14] 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.

[15] 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, 419–34, 2009.

[16] 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.

[17] A. Sawas, A. Abuolaim, M. Afifi, and M. Papagelis, "Tensor methods for group pattern discovery of pedestrian trajectories," in *19th IEEE Intl. Conf. on Mobile Data Management (MDM)*, 2018, pp. 76–85.

[18] A. Sawas, A. Abuolaim, M. Afifi, and M. Papagelis, "Trajectolizer: Interactive analysis and exploration of trajectory group dynamics," in *2018 19th IEEE International Conference on Mobile Data Management (MDM)*, IEEE, 2018, pp. 286–287.

[19] A. Sawas, A. Abuolaim, M. Afifi, and M. Papagelis, "A versatile computational framework for group pattern mining of pedestrian trajectories," *GeoInformatica*, vol. 23, pp. 501–531, 2019.

[20] Y. Zheng, "Trajectory data mining: An overview," *ACM Trans. Intell. Syst. Technol.*, vol. 6, no. 3, 2015, ISSN: 2157-6904.

[21] 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, ISSN: 0360-0300.

[22] 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," en, *Artif. Intel. Review*, vol. 55, no. 2, 1441–1488, 2022, ISSN: 0269-2821, 1573-7462.

[23] 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, ISSN: 1558-2191.

[24] J.-G. Lee, J. Han, X. Li, and H. Gonzalez, "Traclass: Trajectory classification using hierarchical region-based and trajectory-based clustering," *Proceedings of the VLDB Endowment*, vol. 1, no. 1, pp. 1081–1094, 2008.

[25] C. A. Ferrero, L. M. Petry, L. O. Alvares, W. Zalewski, and V. Bogorny, "Discovering heterogeneous subsequences for trajectory classification," *arXiv preprint arXiv:1903.07722*, 2019.

[26] X. Yang, K. Stewart, L. Tang, Z. Xie, and Q. Li, "A review of gps trajectories classification based on transportation mode," *Sensors*, vol. 18, no. 11, p. 3741, 2018.

[27] Y. Zheng, Q. Li, Y. Chen, X. Xie, and W.-Y. Ma, "Understanding mobility based on gps data," in *Proceedings of the 10th international conference on Ubiquitous computing*, 2008, pp. 312–321.

[28] S. Tragopoulou, I. Varlamis, and M. Eirinaki, "Classification of movement data concerning user's activity recognition via mobile phones," in *Proceedings of the 4th International Conference on Web Intelligence, Mining and Semantics (WIMS14)*, 2014, pp. 1–6.

[29] A. Bolbol, T. Cheng, I. Tsapakis, and J. Haworth, "Inferring hybrid transportation modes from sparse gps data using a moving window svm classification," *Computers, Environment and Urban Systems*, vol. 36, no. 6, pp. 526–537, 2012.

[30] M. Etemad, A. Soares Júnior, and S. Matwin, "Predicting transportation modes of gps trajectories using feature engineering and noise removal," in *Advances in Artificial Intelligence: 31st Canadian Conference on Artificial Intelligence, Canadian AI 2018, Toronto, ON, Canada, May 8–11, 2018, Proceedings 31*, Springer, 2018, pp. 259–264.

[31] J.-G. Lee, J. Han, X. Li, and H. Cheng, "Mining discriminative patterns for classifying trajectories on road networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 5, pp. 713–726, 2011. DOI: 10.1109/TKDE.2010.153.

[32] D. S. Gaikwad and U. A. Jogalekar, "Classifying trajectories on road network using neural network," *International Journal of Computer Applications*, vol. 975, p. 8887, 2013.

[33] 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.

[34] 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.

[35] Y. Yu *et al.*, "Tulsn: Siamese network for trajectory-user linking," in *2020 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2020, pp. 1–8.

[36] H. Huang and G. Gartner, "Using trajectories for collaborative filtering-based poi recommendation," *International Journal of Data Mining, Modelling and Management*, vol. 6, no. 4, pp. 333–346, 2014.

[37] H. Yin, W. Wang, H. Wang, L. Chen, and X. Zhou, "Spatial-aware hierarchical collaborative deep learning for poi recommendation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 11, pp. 2537–2551, 2017.

[38] X. Ren, M. Song, E Haihong, and J. Song, "Context-aware probabilistic matrix factorization modeling for point-of-interest recommendation," *Neurocomputing*, vol. 241, pp. 38–55, 2017.

[39] J.-B. Griesner, T. Abdessalem, and H. Naacke, "Poi recommendation: Towards fused matrix factorization with geographical and temporal influences," in *Proceedings of the 9th ACM Conference on Recommender Systems*, 2015, pp. 301–304.

[40] R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization," in *NIPS*, 2007.

[41] S. Kabbur, X. Ning, and G. Karypis, "Fism: Factored item similarity models for top-n recommender systems," *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2013.

[42] Y. Koren, "Factorization meets the neighborhood: A multifaceted collaborative filtering model," in *KDD*, 2008.

[43] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *WWW '01*, 2001.

[44] G. Shani, D. Heckerman, and R. Brafman, "An mdp-based recommender system," in *J. Mach. Learn. Res.*, 2002.

[45] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme, "Factorizing personalized markov chains for next-basket recommendation," in *WWW '10*, 2010.

[46] R. He, W.-C. Kang, and J. McAuley, "Translation-based recommendation," *Proceedings of the Eleventh ACM Conference on Recommender Systems*, 2017.

[47] R. He and J. McAuley, "Fusing similarity models with markov chains for sparse sequential recommendation," *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pp. 191–200, 2016.

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

[49] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

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

[51] 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.

[52] X. Chen *et al.*, "Sequential recommendation with user memory networks," *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, 2018.

[53] W.-C. Kang and J. McAuley, "Self-attentive sequential recommendation," *2018 IEEE International Conference on Data Mining (ICDM)*, pp. 197–206, 2018.

[54] Q. Liu, Y. Zeng, R. Mokhosi, and H. Zhang, "Stamp: Short-term attention/memory priority model for session-based recommendation," *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*, 2018.

[55] J. Tang and K. Wang, "Personalized top-n sequential recommendation via convolutional sequence embedding," *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, 2018.

[56] A. Vaswani *et al.*, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[57] 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.

[58] Q. Guo, Z. Sun, J. Zhang, and Y.-L. Theng, "An attentional recurrent neural network for personalized next location recommendation," in *AAAI*, vol. 34, 2020, pp. 83–90.

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

[60] 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.

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

[62] 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.

[63] D. Yao *et al.*, "Learning deep representation for trajectory clustering," *Expert Systems*, vol. 35, no. 2, e12252, 2018.

[64] 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.

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

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

[67] 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.

[68] R. Salakhutdinov, A. Mnih, and G. E. Hinton, "Restricted boltzmann machines for collaborative filtering," in *ICML '07*, 2007.

[69] D. Kim, C. Park, J. Oh, S. Lee, and H. Yu, "Convolutional matrix factorization for document context-aware recommendation," *Proceedings of the 10th ACM Conference on Recommender Systems*, 2016.

[70] H. Wang, N. Wang, and D. Yeung, "Collaborative deep learning for recommender systems," *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015.

[71] W.-C. Kang, C. Fang, Z. Wang, and J. McAuley, "Visually-aware fashion recommendation and design with generative image models," *2017 IEEE International Conference on Data Mining (ICDM)*, pp. 207–216, 2017.

[72] S. Wang, Y. Wang, J. Tang, K. Shu, S. Ranganath, and H. Liu, "What your images reveal: Exploiting visual contents for point-of-interest recommendation," *Proceedings of the 26th International Conference on World Wide Web*, 2017.

[73] 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.

[74] A. Grover and J. Leskovec, "Node2vec: Scalable feature learning for networks," in *ACM SIGKDD*, 2016, pp. 855–864.

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

[76] 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, 2021, pp. 4249–4256.

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

[78] 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.