Complex & Intelligent Systems (2025) 11:12
https://doi.org/10.1007/s40747-024-01611-z

ORIGINAL ARTICLE

Travel route recommendation with a trajectory learning model

Xiangping Wu[1,2] · Zheng Zhang[1] · Wangjun Wan[3]

Received: 29 March 2024 / Accepted: 14 September 2024 / Published online: 9 November 2024
© The Author(s) 2024

Abstract
This study addresses a critical issue in location-based services: travel route recommendation. It leverages historical trajectory data to predict the actual route on a road network from a starting point to a destination, given a specific departure time. However, capturing the latent patterns in complex trajectory data for accurate route planning presents a significant challenge. Existing route recommendation methods commonly face two major problems: first, inadequate integration of multi-source data, which fails to fully consider the potential factors affecting route choice; and second, limited capability to capture road network characteristics, which restricts the effective application of node features and negatively impacts recommendation accuracy. To address these issues, this research introduces a Trajectory Learning Model for Route Recommendation (TLMR) based on deep learning techniques. TLMR enhances the understanding of user route choice behavior in complex environments by integrating multi-source data. Moreover, by incorporating road network features, TLMR more effectively captures and utilizes the structural and dynamic information of the road network. Specifically, TLMR first employs a Position-aware Graph Neural Network to learn features of intersections from the road network, incorporating context features like weather and traffic conditions. Then, it integrates this information through neural networks to predict the next intersection. Finally, a beam search algorithm is applied to generate and recommend multiple candidate routes. Extensive experiments on four large real-world datasets demonstrate that TLMR outperforms existing methods in four key performance metrics. These results prove the effectiveness and superiority of TLMR in route recommendation.

Keywords Route recommendation · Trajectory data mining · Graph neural network · Deep learning

Introduction

As GPS-enabled mobile devices become increasingly common, trajectory data is accumulating at an unprecedented rate, becoming more open and accessible. Currently, this vast amount of trajectory data is being used to optimize various applications in the real world [1–3]. Among these applications, route recommendation is particularly important, playing a central role in many location-based online services [4, 5]. This study focuses on non-personalized route recommendations, aiming to predict and provide the most probable route from a starting point to a destination on a given road network, based on historical trajectory data and contextual information, thereby achieving efficient and reasonable navigation.

Travel route planning methods can generally be divided into two categories. The first category, early heuristic-based route recommendations, is dedicated to finding the shortest geographical distance or the least time-consuming path in road networks, with typical methods including Dijkstra's algorithm and A* algorithm, among others [6, 7]. However, some studies have shown that people often do not choose just the shortest or fastest path when traveling [8], but are influenced by a variety of latent factors such as weather, traffic conditions, and public preferences. Therefore, an effective

Zheng Zhang and Wangjun Wan contributed equally to this work.

✉ Xiangping Wu
xiangpingwu@cjlu.edu.cn

Zheng Zhang
p21030854076@cjlu.edu.cn

Wangjun Wan
181892505@qq.com

[1] College of Information Engineering, China Jiliang University, Hangzhou 310018, China

[2] Key Laboratory of Electromagnetic Wave Information Technology and Metrology of Zhejiang Province, Jiliang University, Hangzhou 310018, China

[3] Technology Center of Hangzhou Customs District, Hangzhou 311202, China

route recommendation method should integrate various influencing factors along with rich contextual information. The second category views route recommendation as a modeling task, aimed at mining latent patterns in users' historical trajectories and then generating candidate routes. Early research commonly employed mathematical models or machine learning, such as the Markov chain model [9, 10]. However, effective path modeling in large road networks with millions of trajectories is a challenging task, especially in capturing transition patterns in complex network topologies. Incorrect predictions may lead to significant deviations between the predicted and actual routes, especially when the starting point and destination are far apart.

In recent years, deep learning technology has demonstrated its potential in handling complex data relationships by automatically learning to capture effective features from raw data [11]. Some existing studies have introduced deep learning techniques on top of traditional heuristic algorithms, creating cost functions that integrate various factors based on deep learning and then searching for the least costly path in road networks [12, 13]. Moreover, many studies utilize deep learning models, particularly Recurrent Neural Networks (RNNs), to plan routes by learning transition patterns from historical trajectories [14–16]. Despite the progress made by existing methods in travel route recommendation tasks, they still face two main limitations. First, various influencing factors such as weather, traffic conditions, node features, and historical trajectory data play a key role in travel route planning. Integrating these multidimensional data can lead to more effective travel route planning models. However, most existing methods consider only a single data source, failing to effectively integrate multi-source data and delve into their deep nonlinear relationships. Second, to accurately learn from historical trajectories, it's crucial to effectively capture road network features. The road network, composed of nodes and segments, can be understood by learning effective representations of nodes, thus effectively mining historical trajectory data. However, many existing studies use simple encoding or ordinary GCNs (like GCN [17] and GAT [18]) to learn node features, which do not accurately represent node position information in the graph network, limiting cross-node information transmission and feature updates during model training, affecting recommendation accuracy.

To address these challenges, this study proposes a novel Trajectory Learning Model for Route Recommendation, aimed at effectively integrating contextual features, node features, and historical trajectory data. This study treats the route planning problem as a sequence generation process, which is well-suited for solution using deep neural networks. The TLMR model includes three main stages: First, the feature extraction stage uses Position-aware Graph Neural Networks (P-GNNs) [19] to learn node features, transforming historical trajectories into patterns of nodes and their successors, while also extracting contextual features such as weather and traffic conditions. Second, in the model learning stage, a deep learning framework is introduced to learn the probability of visiting the next node under a given input pattern. This framework combines P-GNNs, Gated Recurrent Units (GRUs) [20], and Multi-Layer Perceptrons (MLPs), thus capable of learning the probability distribution of the next node. Finally, a beam search algorithm is used to generate multiple effective candidate routes.

The main contributions of this study can be summarized as follows:

1. Innovative Trajectory Learning Model: This study proposes an innovative Trajectory Learning Model for Route Recommendation, which decomposes the problem into two sub-problems: transition probability prediction and route search. By combining Recurrent Neural Networks with Graph Neural Network embeddings, the model learns the probability distribution of the next node based on the current route. A beam search algorithm is then employed to generate candidate routes based on this probability distribution.
2. Integration of Multi-Source Features: The proposed model integrates multi-source data and effectively utilizes contextual information to characterize complex trajectory features, enhancing the understanding of user route choice behavior in complex environments. It adopts a location-aware graph neural network that considers the topological structure and spatial relationships of road networks, effectively extracting and representing the characteristics of intersections. Through a message passing mechanism, the model can propagate information learned during training to rare or unseen nodes, significantly enhancing its generalization capability. Furthermore, recurrent neural networks capture long-term dependencies and dynamic changes, improving prediction accuracy.
3. Empirical Validation with Real-World Datasets: This study introduces four real-world datasets and evaluates the model using four different performance metrics. Extensive experiments demonstrate the effectiveness and superiority of TLMR in route recommendation tasks.

Related work

In recent years, travel route planning tasks have attracted the attention of many researchers. Existing relevant work can be broadly categorized into two types. The first type is heuristic algorithm-based path recommendation [21], which aims to find the path with the minimum cost in the road network, such as the shortest geographical distance or the minimum time-

consuming path. Typical methods in this category include Dijkstra's algorithm [22] and A* algorithm. Zhang et al. [23] improved the Dijkstra's algorithm to provide the vehicle path with the minimum fuel consumption. Wang et al. [24] proposed a dynamic path planning method based on the A* algorithm, capable of efficiently searching for the shortest time path. Wang et al. [25] utilized an enhanced A* algorithm to reduce both the time and road costs of touring paths within scenic areas. Qin et al. [26] introduced an improved heuristic algorithm based on Dijkstra and the k-shortest path algorithm to address the reliable energy consumption pathfinding problem. However, some studies suggest that people often do not choose the shortest or fastest path when traveling [27]. Instead, their choices are influenced by various potential factors such as weather, traffic conditions, and public preferences.

The second strategy attempts to treat route planning as a route modeling task, aiming to generate candidate routes based on extensive historical trajectory data. Early researchers primarily utilized probabilistic graph models, particularly variants of Markov Chain (MC) models [28], which have been widely employed for mining latent patterns in mobile trajectories. Simmons et al. [9] constructed a Hidden Markov Model (HMM) for learning and predicting drivers' routes and destinations from trajectories. Akabane et al. [10] proposed an approach that utilizes Hidden Markov Models and the Viterbi algorithm. The method involves analyzing historical path data of vehicles to establish model parameters and predict the future paths of vehicles. However, these methods rely on one or two previous positions for trajectory prediction, failing to consider long-term temporal dependencies. They exhibit poor performance in long-range trajectory prediction and lack flexibility in capturing environmental and other external factors. With the explosive growth of user-generated trajectory data and the success of deep learning in modeling complex data relationships or features, especially RNNs and their variants like Long Short-Term Memory (LSTM) [29] and Gated Recurrent Unit (GRU) [20] are widely applied in the modeling of trajectory data [30, 31].

Wu et al. [14] proposed two novel models designed to fully leverage the capabilities of recurrent neural networks in capturing variable-length sequences, while simultaneously addressing topological constraints in trajectory modeling. Feng et al. [32] designed a model named DeepMove, which combines attention mechanisms and recurrent neural networks to capture the periodicity of human mobility and relationships between historical trajectories. Wang et al. [12] introduced a neural network-based A* search algorithm for route recommendation. The approach involves utilizing Recurrent Neural Networks (RNN) to compute observable costs, employing a graph attention network for estimating costs, and additionally incorporating attention mechanisms to enhance the learning of mobile states. Li et al. [15] proposed a novel deep probabilistic model, DeepST, for learning spatial transfer patterns on road networks. DeepST comprehensively takes into account sequential characteristics, the influence of destinations, and real-time traffic factors. The model proposed by Liang et al. [16] employs directional representations and spatiotemporal attention mechanisms, utilizing an LSTM encoder-decoder network with a sliding temporal attention mechanism to capture both long-term and short-term temporal dependencies. Yan et al. [33] proposed a pre-trained contrastive learning network for vehicle trajectory prediction, introducing a pre-trained model to extract advanced features of trajectories to enhance the expressive capability of trajectory representations. Furthermore, Xu et al. [34] introduced the ASNN-FRR model, which integrates the A* algorithm with an adaptive graph convolutional recurrent network to accurately estimate travel times for different paths. The model is designed to offer rapid and efficient route recommendations based on real-time traffic conditions. Xue et al. [13] introduced the DGCN, which establishes a weighted network model based on road network structure using the Grivan–Newman algorithm for road network partitioning. The model comprehensively considers traffic factors for travel route recommendations.

Preliminaries

Definition 1 (*Road network*) A road network is represented as a directed graph $G = (V, E, l)$, where V is the set of road intersections, also referred to as the set of nodes; E is the set of road segments, also referred to as the set of edges. In this study, an edge $e_i = (v_i, v_{i+1})$ represents a road segment from node v_i to node v_{i+1}, and $l(e_i)$ denotes the length of the edge e_i.

Definition 2 (*Trajectory*) A trajectory is a sequence composed of GPS records emitted by vehicles, where each GPS record is represented as a tuple (latitude, longitude, time), indicating the vehicle's location at a specific time.

Definition 3 (*Route*) A route is defined as $R(s, d) = \{v_1, v_2, \ldots, v_{|R|}\}$, where the starting point is $s = v_1$ and the destination is $d = v_{|R|}$. It represents a non-cyclic path in the road network G from s to d.

Definition 4 (*Query*) A user's query is defined as $q : (s, d, t)$, where s and d are the starting point and destination, respectively, and t is the departure time.

Definition 5 (*Route planning*) Given a road network G, and a historical trajectory dataset D, our goal is to infer the most likely route R^* in response to a query $q : (s, d, t)$.

The key to effective route recommendation is obtaining the true transition probability distribution for each node

in the road network. Based on these probabilities, the correct next node is sequentially chosen until the destination is reached. However, the true transition probability distribution is not directly observable. Therefore, we model the historical trajectory dataset D to obtain an approximate transition probability distribution $\hat{P}(v_i \mid v_1 \rightarrow v_{i-1}, s, d, t)$, since D is a sample drawn from the true distribution. Using these learned probabilities, an ideal candidate route can then be generated.

Methodology

As illustrated in Fig. 1, the TLMR framework proposed in this study includes three key stages. In the feature extraction phase, a Position-aware Graph Neural Network is employed to effectively learn node features. Additionally, the model comprehensively incorporates the impact of weather and traffic conditions, encoding these factors into latent vectors that are concatenated with other features to enhance the model's predictive capability. Historical trajectory data is transformed into a sequence of nodes, where the subsequent node in each sequence is used as the prediction label, with the last node serving as the trajectory endpoint.

In the model learning phase, deep learning methods are employed to predict the probability of the next node visit within a given input sequence. This phase consists of four core components: an Embedding layer, GRU layer, Concatenation layer, and MLP Layer. The Embedding layer transforms each node into a corresponding feature vector, while the GRU layer encodes the input node sequence into a low-dimensional feature vector, capturing rich sequential semantics. In the Concatenation layer, the context, destination, and sequence feature vectors are combined, and an MLP model is used to predict the next node's probability distribution. Finally, in the route generation phase, an enhanced beam search strategy is employed to generate multiple candidate routes based on the predicted probability distribution.

Feature extraction stage

Node feature extraction

The structure of a road network is fundamentally a typical graph network. A key step before performing graph prediction tasks is to map nodes into an embedding space, effectively capturing their features. This study aims to ensure that node features accurately reflect distance information within the road network. Although GPS coordinates can be used as node features, they do not fully and accurately express a node's position in the graph network because vehicle movement is strictly constrained by the road network structure.

To address this issue, this study opts to use a position-aware Graph Neural Network to learn more effective node features. Position-aware Graph Neural Networks have certain advantages; compared to traditional methods, they not only preserve the structural characteristics of the graph but also update the features of nodes not present in the training set through their message-passing mechanism, thereby enhancing the model's robustness. The node update formula for the Graph Neural Network is as follows:

$$n_i^{(l+1)} = \mathrm{GNN}\left(n_i^{(l)}\right) \tag{1}$$

Here, $n_i^{(l)}$ represents the feature vector of node v_i after the l-th iteration, with the vector dimension in this study denoted by $|n|$.

In graph embedding tasks, traditional graph neural network models such as GCN and GAT face inherent limitations. The node features learned by these methods mainly rely on the node's local structure, making it difficult for them to capture broader graph structural features. This limitation is evident when two nodes in a graph have similar local structures; they are assigned the same feature representation, making it challenging to accurately differentiate their unique positions in the entire graph network. Inspired by recent research [19], we adopt P-GNNs in this study. This approach overcomes the limitations of traditional methods by selecting a set of anchor points in the road network and effectively passing messages between these anchor points and target nodes. With P-GNNs, the learned node features can better reflect global network position information. The core of P-GNN includes two main parts: firstly, the construction of an anchor point set, and secondly, the learning of node features.

Construction of the anchor point set This study first introduces the concept of an anchor point set, a selected group of nodes in the graph network, used to compute the position-aware features of other nodes. To generate features with position-aware capabilities, we select some representative nodes as anchor points. Based on the relationship between the anchor point set and the target node, we can map the target node into the embedding space. Following the methods in [19], we use the k-means algorithm to cluster K anchor point sets from all nodes, each represented as S_k, where $k = 1, \ldots, K$.

Learning node features In P-GNN, the learning of node features is divided into four steps.

1. Message Passing Function Definition: We define a message passing function $F(v_i, v_j)$ to describe the positional relationship between nodes v_i and v_j, represented as:

$$F(v_i, v_j) = \mathrm{Dist}(v_i, v_j) \cdot (h_i \parallel h_j) \cdot W_1, \tag{2}$$

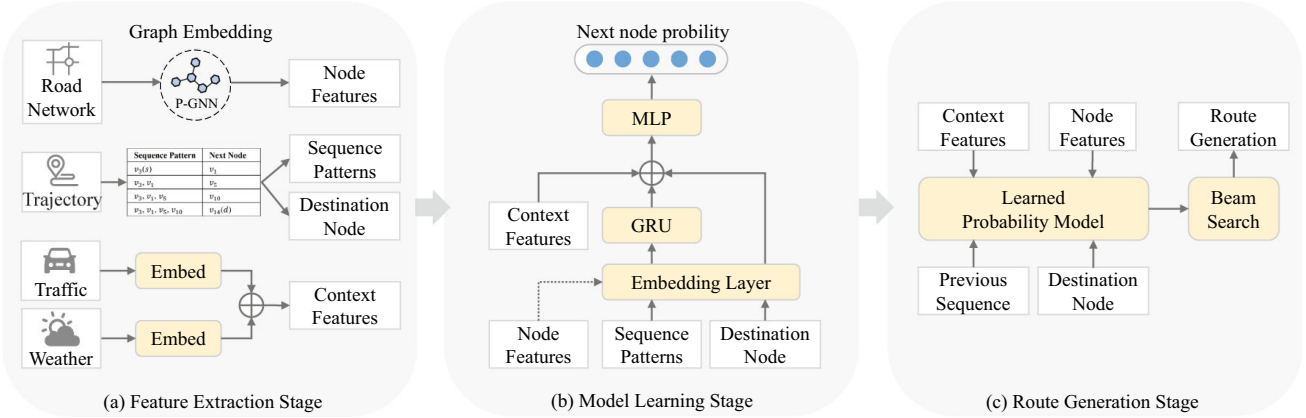


Fig. 1 The framework of TLMR for route recommendation

where Dist() calculates the shortest distance between two nodes, implemented using Dijkstra's algorithm; h_i is the message passing vector; $\|$ denotes concatenation operation; $W_1 \in \mathbb{R}^{2r \times r}$ is a learnable parameter matrix.

2. Generating Representation Vectors Based on Anchor Point Sets: Next, we use the message passing function to generate the representation vector of node v_i based on each anchor point set. For each anchor point set S_k, the representation vector of node v_i, $m_i^{(k)} \in \mathbb{R}^r$, is calculated as:

$$m_i^{(k)} \leftarrow \mathrm{AP}\left(\{F(v_i, v_j) \mid \forall v_j \in S_k\}\right), \tag{3}$$

where AP() is the average pooling function, combining all outputs of $F()$ into a single representation vector.

3. Average Pooling of Anchor Point Set Representation Vectors: Then, we perform an average pooling operation on the representation vectors of all anchor point sets:

$$h_i \leftarrow \mathrm{AP}\left(\{m_i^{(k)} \mid k = 1, \ldots, K\}\right), \tag{4}$$

The generated h_i is substituted into equation (14) to involve the positional information of v_i with the anchor points in the GNN updating process.

4. Computing Position-Aware Features: Finally, compute the position-aware feature of node v_i:

$$n_i = \sigma(M_i \cdot w_{\text{out}}), \tag{5}$$

where $M_i \in \mathbb{R}^{K \times r}$ is a matrix composed of $m_i^{(k)}$ as rows, $w_{\text{out}} \in \mathbb{R}^r$ is a learnable weight vector, and σ is a nonlinear mapping function.

Sequence pattern extraction

To implement model training, this study designed a method to convert given complete historical trajectories into a set of node sequences and their corresponding next nodes. This process is illustrated in Fig. 2: First, a series of consecutive nodes from the historical trajectories are merged into a sequence pattern. In this sequence, the node immediately following the sequence is considered the prediction node and is used as the label during the training process. Furthermore, to preserve the integrity of the trajectory, the last node in the sequence is also retained as the end point of the trajectory.

(a) A sample of input trajectory.

Trajectory	$\{v_3, v_1, v_5, v_{10}, v_{14}\}$

(b) Sequence pattern extraction.

Sequence Pattern	**Next Node**
$v_3(s)$	v_1
v_3, v_1	v_5
v_3, v_1, v_5	v_{10}
v_3, v_1, v_5, v_{10}	$v_{14}(d)$

Fig. 2 An example of sequence pattern extraction based on trajectories

By inputting this sequence pattern into a deep learning model, we can effectively generate a probability distribution that represents the likelihood of transitioning from the current node to the next node. This method leverages the temporal information in historical trajectory data, thereby enhancing the model's understanding and predictive ability regarding the dynamics of trajectories.

Context feature extraction

Considering the potential impact of external factors such as weather and traffic conditions on route selection, this study designed a method to integrate these factors into a comprehensive contextual feature. First, to incorporate the impact of weather, this study used an open API (https://www.

timeanddate.com/) to obtain the weather conditions for each time period of each day in a specific city. The study categorized the weather into six types: clear, cloudy, fog, light rain, heavy rain, and snow, and represented each category using one-hot encoding.

Next, to capture the traffic conditions of the road network at any given moment, the study selected the top 1% of the most frequently traversed road segments $e \subset E$ and integrated their average speeds into a $|e|$-dimensional vector. Due to the similarity in speeds of adjacent road segments, the study faced the problem of information redundancy. To address this issue and simultaneously reduce the dimensionality of features, the study applied Principal Component Analysis (PCA) to the traffic data for dimensionality reduction. PCA, as an effective dimensionality reduction technique, can map high-dimensional feature vectors into a lower-dimensional space while retaining as much of the original data's structural information as possible. In this way, the study mapped the original $|e|$-dimensional feature vector into a lower-dimensional d-dimensional ($d < |e|$) space, using this d-dimensional vector as the principal component of traffic conditions. In this study, these dimensionally reduced feature vectors were used to represent traffic conditions. This not only effectively highlighted the key features of the traffic information but also reduced the consumption of storage and computational resources. Finally, the feature vectors of weather and traffic conditions were concatenated to form a comprehensive contextual feature, providing more comprehensive data support for model training.

Model learning stage

Through the model learning phase, this study obtains a trained model. This model can receive inputs and compute the probability distribution of the next node, laying a solid foundation for the route generation phase. The study employs deep learning methods to predict the probability distribution of visiting the next node following a given input sequence. This learning process consists of four core parts: Embedding layer, GRU layer, Concatenation layer, and MLP layer. The model in this study is based on the sequence pattern $x = \{v_1, v_2, \ldots, v_{|x|}\}$ and learns the following probability distribution: $\{P(v_i \mid x), \forall v_i \in V_x^{\text{next}}\}$, where V_x^{next} represents the set of all possible next nodes to transition to.

As shown in Fig. 3, the model's input includes the sequence pattern and the destination node. To effectively handle input sequences of arbitrary lengths and learn long-term dependencies, this study employs GRU to encode the sequence pattern into a sequence feature vector. In the final stage of the model, we concatenate the sequence feature, destination node feature, and contextual feature, and output the probability distribution of the next node through the MLP. It is noteworthy that the contextual feature is determined based on the departure time, i.e., the vehicle's time at the first node, and is considered the time of user query during the training phase.

Input The input includes a sequence pattern $x = \{v_1, v_2, \ldots, v_{|x|}\}$ and a destination node d.

Embedding layer In the embedding layer, each node is mapped to its corresponding feature vector $n_i \in \mathbb{R}^K$. Simultaneously, the corresponding context feature c_t is obtained based on the departure time t. After embedding, the sequence pattern becomes $\{n_1, n_2, \ldots, n_{|x|}\}$, and the feature of the destination node is n_d.

GRU layer The GRU layer is responsible for encoding the input sequence pattern into a vector as a sequence feature. GRU selectively retains or forgets key information through update gates, reset gates, and nonlinear transformations, generating the output for the next time step h_{i+1}. The computation process in GRU is as follows:

$$h_i = GRU(n_i, h_{i-1}) \tag{6}$$

where n_i is the i-th input to GRU, $\tilde{h}_i$ is the i-th candidate hidden state, h_i is the i-th hidden output. The output of the last GRU layer $h_{|x|}$ is the extracted sequence feature.

Concatenation layer In the concatenation layer, the sequence feature $h_{|x|}$, destination node feature n_d, and contextual feature c_t are concatenated, represented as $[h_x; n_d; c_t]$.

MLP The MLP consists of three fully connected layers, with the input $[h_x; n_d; c_t]$. The output $y' \in \mathbb{R}^{|V_x^{\text{next}}|}$ contains the probability distribution of all possible next nodes:

$$O = \text{MLP}([h_x; n_d; c_t]), \tag{7}$$

$$y_i' = \frac{e^{O_i}}{\sum_{k=1}^{|V_x^{\text{next}}|} e^{O_k}}, \tag{8}$$

This study uses a cross-entropy loss function to compute the model's loss and an Adam optimizer to update neural network parameters. The loss function is defined as follows:

$$\text{Loss} = -\sum_{i=1}^{|V_x^{\text{next}}|} y_i \cdot \log(y_i'), \tag{9}$$

Where $|V_x^{\text{next}}|$ represents the number of all possible next nodes; y is the real probability distribution, and y_i is the ground truth label during the training process.

Route generation stage

This section provides a detailed introduction to the route generation strategy. Inspired by [35], this study adopts a beam search method to generate candidate routes based on the learned probabilistic model. Beam search is a heuristic search

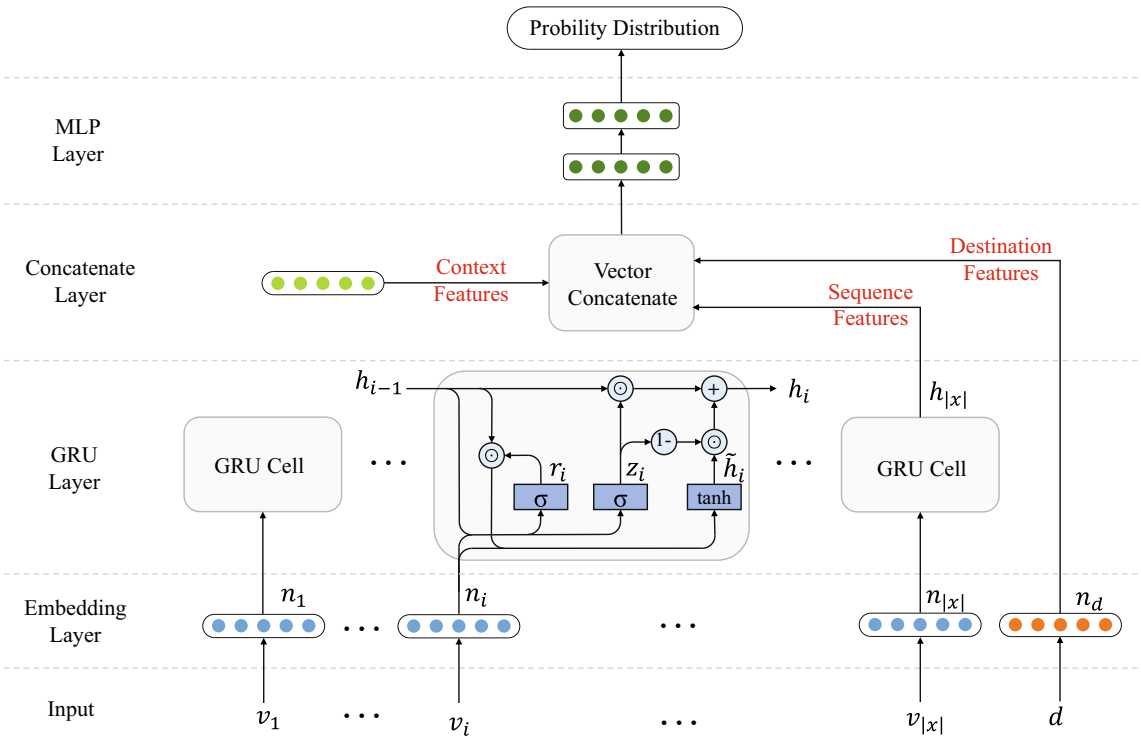


Fig. 3 Internal composition of the model learning stage

algorithm whose core idea is to maintain a finite set of candidate solutions (referred to as the "beam") and retain only the most promising solutions during the search process. The pseudocode of the route generation algorithm is shown in Algorithm 1, and its steps are as follows:

1. Input and Output: The input of the algorithm includes the starting point s, the destination d, the departure time t, and the beam size n. The output is a set of candidate routes and their corresponding probabilities. Here, the function $p(T_n)$ is defined as the probability score of the route T_n.
2. Initialization: After initializing the candidate route set A and the temporary route set T, the initial temporary route $\{s\}$ is inserted into T.
3. Iterative Process: In each iteration, for each temporary route T' in T, use the learned model to calculate the probability distribution of its next node. For the n nodes with the highest probability in the next node set V^{next}, add them to T' to form a new temporary route T_n, and calculate the probability of the route T_n.
4. Conditional Judgment: If the last node of T_n is the destination d, then T_n is considered an appropriate candidate route and is added to the set A. If the number of nodes in T_n exceeds the limit max, but does not include the destination d, select the node with the smallest Haversine distance to the destination as the end of the route. If the number of nodes in T_n is less than the limit max and does not include the destination d, insert T_n back into T for further processing.
5. Iteration End: At the end of each iteration, remove the route T' from T. Repeat this process until there are no more routes in T.
6. Final Selection: Choose the candidate route with the highest probability from the candidate route set A as the final result.

Experiment

Experimental setup

This study utilized four public taxi GPS trajectory datasets, originating from the cities of Beijing [36], Porto [37], Chengdu and Xi'an [38]. In this study, the dataset was cleaned by removing trajectories shorter than 1 km and those with fewer than 3 nodes. Table 1 summarizes the key statistical information of these datasets. To construct the road networks of each city, we employed map data provided by OpenStreetMap [39]. Furthermore, the research also adopted an open-source map matching algorithm [40] to accurately map GPS trajectories onto the corresponding road networks. During experiments, we used the first 80% of trajectories in each

Algorithm 1 The Algorithm for Route Generation

Input: Starting node s, Destination node d, Departure time t, Beam size n
Output: Candidate route set A

Initialize $A = \emptyset$, temp route set $T = \emptyset$
Insert initial route $\{s\}$ into T
repeat
  for each $T' \in T$ **do**
    $y' = \text{Next_Prediction}(T', d, t)$
    $V^{\text{next}} \leftarrow$ Get the top-n nodes from y'
    for each $v \in V^{\text{next}}$ **do**
      $T_n = T'.\text{append}(v)$
      $p(T_n) = p(T') \times y'_v$
      if $v = d$ **then**
        Insert T_n into A
      end if
      if $\text{length}(T_n) = \max$ and $v \neq d$ **then**
        $T_n = \text{Truncate}(T_n)$
        Insert T_n into A
      end if
      if $\text{length}(T_n) < \max$ and $v \neq d$ **then**
        Insert T_n into T
      end if
    end for
    Remove T' from T
  end for
until $T = \emptyset$
return A

dataset as the training set, while the remaining 20% served as the test set.

In this study, we employed four metrics to evaluate the performance of the model: Precision, Recall, F1-score, and Reachability. Precision and Recall primarily measure the overlap ratio between the actual routes and the model-predicted routes. The F1-score, being the harmonic mean of Precision and Recall, provides a comprehensive performance assessment. Additionally, we introduced the Reachability metric to assess whether the routes recommended by the model successfully reach the destination. The following are the specific calculation methods for these four evaluation metrics:

$$Precision = \frac{\sum_{e \in (R \cap R^*)} l(e)}{\sum_{e \in R^*} l(e)}, \tag{10}$$

$$Recall = \frac{\sum_{e \in (R \cap R^*)} l(e)}{\sum_{e \in R} l(e)}, \tag{11}$$

$$F1 - score = \frac{2 \times precision \times recall}{precision + recall}, \tag{12}$$

$$Reachability = \begin{cases} 1 & \text{if } d = d^* \\ 0 & \text{otherwise} \end{cases}, \tag{13}$$

Where R is the actual route, and R^* is the predicted route. $l(e)$ represents the length of edge e, and d^* denotes the last node in R^*.

To verify the model's effectiveness, we compared it with several classic methods and mainstream deep learning models. These comparison models were chosen for their fundamental and widely recognized approaches, established benchmarks, and extensive validation in numerous studies, providing a stable basis for comparison.

- Shortest Path: This method employs Dijkstra's algorithm to calculate the shortest path from the starting point to the destination. It considers only distance factors and does not involve other complex variables.
- RNN: This method utilizes the basic RNN model, which learns only the sequence of past nodes without integrating P-GNN, destination node and context information.
- DeepMove [32]: This model leverages attention mechanisms and recurrent neural networks to capture the periodicity of human mobility and the correlations between historical trajectories.
- CSSRNN [14]: Aimed at fully utilizing the capabilities of recurrent neural networks to capture variable-length sequences, this model addresses long-term dependencies in road segments while simultaneously tackling topological structure constraints in trajectory modeling.
- NASR [12]: It learns the cost function in the A* algorithm through deep learning. Combining RNNs for observable costs and graph attention networks for estimating costs, it uses the A* algorithm for route planning.

The experimental code was written in Python and executed on a computer running the Windows 10 operating system, equipped with a 3.30 GHz Intel Core i5 processor and 16GB of memory. For the deep learning framework, this study implemented the Pytorch library. During the feature extraction phase, node features were embedded in 128 dimensions. In the model learning phase, the number of hidden neurons in GRU was set to 256, and MLP was configured as a two-layer fully connected network, activated respectively by tanh and softmax, with a learning rate of 1e-3. In the route generation phase. For comparative methods, this study adjusted the models according to the parameters provided in their papers and trained all models until convergence.

To validate the significance of our experimental results, we performed 10 independent training runs for each method. For subsequent analysis, we selected the best-performing result from each run. This approach minimizes the impact of random initialization and stochastic processes, providing a more reliable assessment of the model's capabilities.

Comparative experiment

Table 2 shows the comparative performance of different algorithms in four cities: Chengdu, Beijing, Xi'an, and Porto.

Table 1 The statistics of datasets

Dataset	Trajectories	Intersections	Segments	Avg length (km)
Chengdu	3,886,537	46,008	107,655	4.29
Beijing	3,098,645	125,558	311,769	4.03
Xi'an	2,418,736	26,979	59,927	4.75
Porto	1,257,963	78,054	185,074	6.19

Table 2 Performance metrics of different algorithms in various cities. The best-performing item for each metric is highlighted in bold

Algorithm	Precision	Recall	F1-score	Reachability
Chengdu				
Short	0.702	0.683	0.692	–
RNN	0.726	0.732	0.729	0.917
CSSRNN	0.834	0.817	0.825	0.987
DeepMove	0.737	0.761	0.749	0.968
NASR	0.765	0.743	0.754	–
TLMR	**0.856**	**0.829**	**0.842**	**0.993**
Beijing				
Short	0.762	0.717	0.739	–
RNN	0.413	0.528	0.463	0.437
CSSRNN	0.757	0.778	0.767	0.916
DeepMove	0.424	0.559	0.482	0.463
NASR	0.711	0.643	0.675	–
TLMR	**0.826**	**0.817**	**0.821**	**0.989**
Xi'an				
Short	0.667	0.635	0.651	–
RNN	0.679	0.653	0.666	0.912
CSSRNN	0.804	0.798	0.801	0.991
DeepMove	0.712	0.744	0.728	0.979
NASR	0.737	0.715	0.726	–
TLMR	**0.821**	**0.812**	**0.816**	**0.995**
Porto				
Short	0.572	0.513	0.541	–
RNN	0.583	0.591	0.587	0.784
CSSRNN	0.709	0.713	0.711	0.939
DeepMove	0.618	0.635	0.626	0.816
NASR	0.708	0.606	0.653	–
TLMR	**0.752**	**0.715**	**0.733**	**0.986**

The performance indicators include Precision, Recall, F1-score, and Reachability. The analysis results indicate that the TLMR algorithm outperforms baseline methods in all four indicators, particularly in terms of reachability. This finding underscores TLMR's efficiency in simulating travel route planning. The CSSRNN algorithm closely follows, especially in terms of precision, recall, and F1-score. Although other algorithms perform well in specific cities and indicators, overall, they are inferior to TLMR and CSSRNN.

Taking the Chengdu dataset as an example, TLMR's precision is 0.856, which is 2.6% higher than CSSRNN's precision of 0.834, demonstrating its significant advantage. In terms of recall, TLMR scores 0.829, surpassing CSSRNN's 0.817 by 1.5%. In the F1-score aspect, TLMR leads with a score of 0.842, compared to CSSRNN's 0.825, an increase of 2.1%. In the Beijing dataset, TLMR's precision reaches 0.826, significantly exceeding CSSRNN's 0.757 (a 9.1% improvement), showing a notable difference in effectiveness. TLMR's recall rate is 0.817, 5% higher than CSSRNN's 0.778, and in F1-score, TLMR scores 0.821, surpassing CSSRNN's 0.767 by 7%. Notably, although almost all methods perform poorly on the Porto dataset, TLMR still shows the best performance. Across all datasets, TLMR consistently outperforms other existing methods, further confirming its exceptional effectiveness. Moreover, it is worth mentioning that the Short algorithm generally performs poorly across three datasets. For example, in the Porto dataset, its F1-score is only 0.541. This result suggests that when choosing mobility routes, humans do not always follow the shortest path and may be influenced by various potential factors such as traffic conditions and weather.

TLMR demonstrates near-perfect reachability across all datasets. For instance, in the Beijing, Xi'an, and Porto datasets, TLMR's reachability scores are 0.989, 0.995, and 0.986, respectively, outperforming other algorithms. This showcases its consistency and reliability across various datasets, indicating its ability to almost always successfully predict routes to destinations. This aspect can likely be attributed to TLMR's destination-driven and strategy of learning from historical sequence features. As for the Short algorithm, due to its fixed characteristic of always predicting the shortest path, its reachability is always 1. However, this does not mean it is always the most effective route choice, as it does not consider other factors that may influence route selection. TLMR's high reachability across all datasets highlights its superiority in predicting effective routes, especially in complex urban environments and variable routing situations. The lower reachability of other methods in certain datasets may be due to their models not being fully capable of learning and adapting to complex route prediction scenarios.

Table 3 F1 scores for different variants in different cities. The best-performing item for each metric is highlighted in bold

Variant	Chengdu	Beijing	Xi'an	Porto
TLMR	**0.842**	**0.821**	**0.816**	**0.733**
TLMR-Context	0.827	0.775	0.796	0.702
TLMR-PGNN	0.809	0.738	0.747	0.691
TLMR-All	0.769	0.625	0.705	0.672

Ablation experiment

In this study, we incorporated context information and P-GNN embeddings. To explore the effectiveness of this combination, we designed three different model configurations for comparison: TLMR-Context (without context information), TLMR-PGNN (without P-GNN embeddings), and TLMR-All (neither used). As shown in Table 3, comparative analysis reveals that in the absence of context information or P-GNN embeddings, variant models experience varying degrees of performance degradation. Conversely, models that simultaneously employ both context information and P-GNN embeddings demonstrate significant improvements in prediction accuracy. Additionally, even the isolated application of either context information or P-GNN embedding techniques can effectively enhance model performance, further confirming their important role in improving model efficacy.

Parameter exploration

To comprehensively assess the robustness of the model, this study considered different Euclidean distances between query start points and destinations, testing the model's performance across these varying distances. The experimental results are shown in Fig. 4. As the travel distance increases, the number of possible routes multiplies, making the modeling task more complex. We categorized the test trajectories based on the distance between the start point and destination into three groups: short distance (less than 2 km), medium distance (2 to 5 km), and long distance (more than 5 km). Figure 1 displays the F1-scores of different algorithms at these varied distances. A key observation from Fig. 1 is that all methods generally perform better on short-distance trajectories. Notably, the TLMR algorithm significantly outperforms other competing methods across short, medium, and long distances, robustly demonstrating TLMR's effectiveness and superiority.

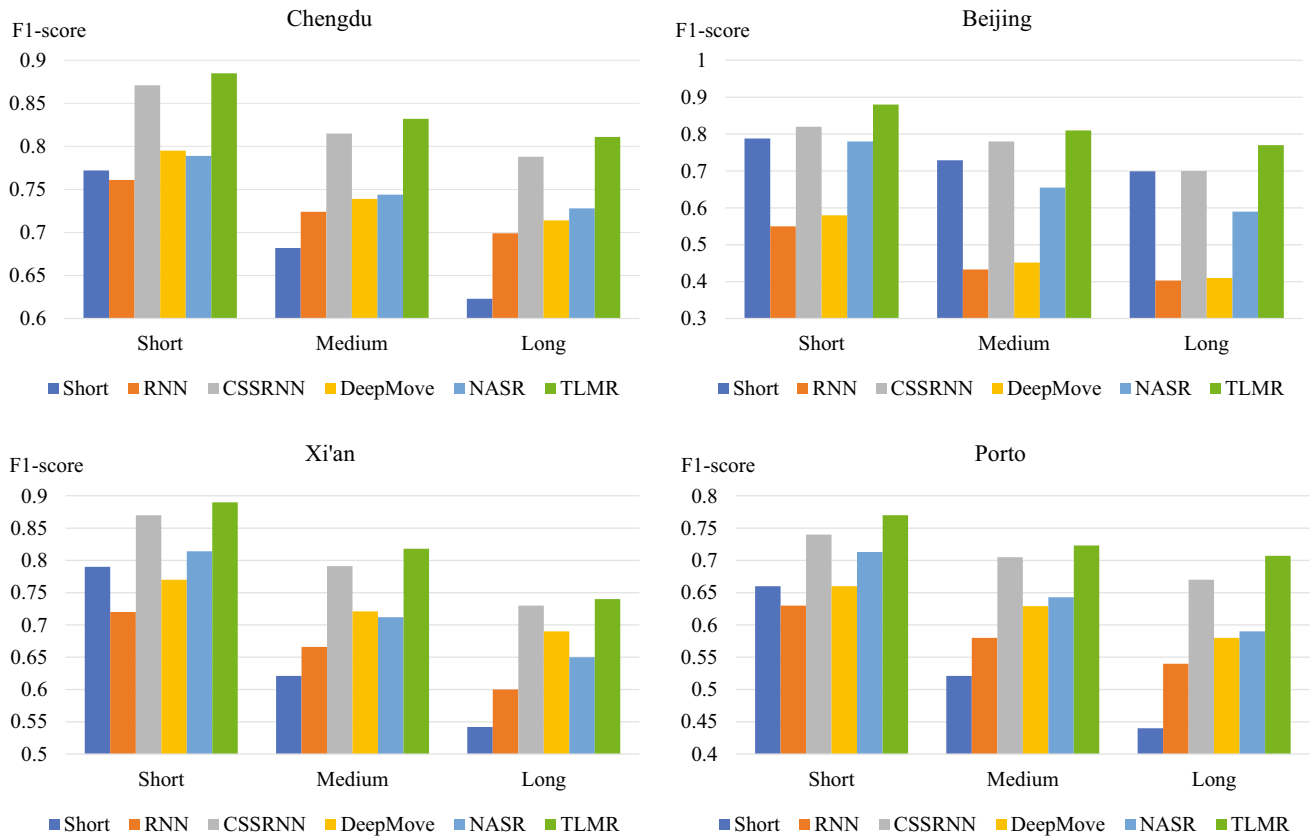


Fig. 4 Explore the F1-score of various models under different route lengths

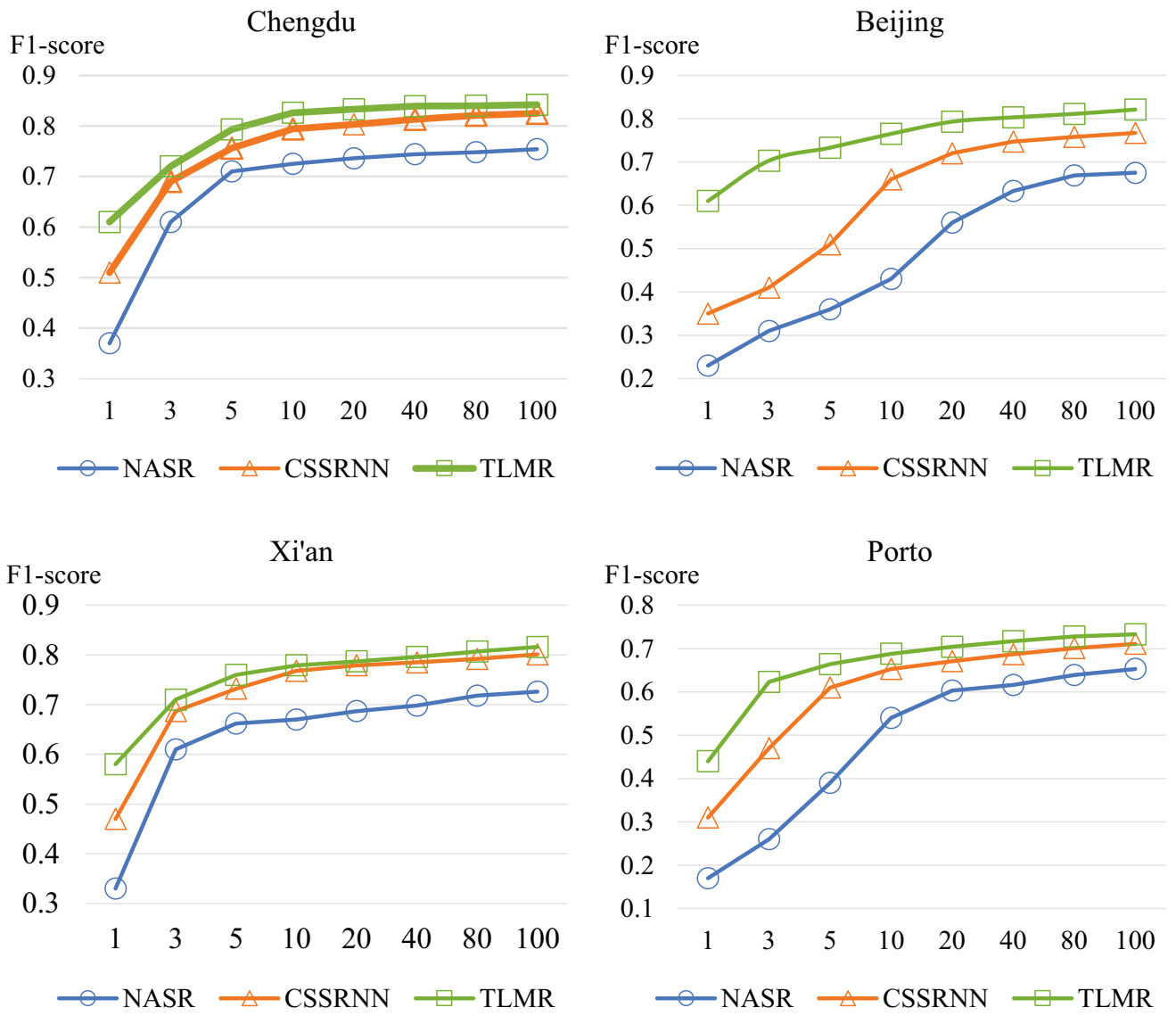


Fig. 5 Explore the F1-score of various models across different training data sizes

To comprehensively evaluate the model's performance across different scales of training data, this study conducted a series of experiments, as shown in Fig. 5. We systematically varied the amount of training data for the TLMR model, covering different proportions from 1% to 100% of the total training data. This experimental design aimed to explore the trend of model performance as the scale of training data changes. The comparative analysis included two high-performing algorithms, CSSRNN and NASR, as benchmarks. Through comparing the F1 scores, we observed that the performance of all models decreases with the reduction in the amount of training data. Notably, the performance gap between TLMR and CSSRNN, NASR widens in some cases. Particularly in scenarios with extremely limited data, TLMR demonstrated robustness. This result suggests that the TLMR algorithm not only excels in data-rich environments but also maintains a high level of performance in data-limited situations. This could be attributed to TLMR's use of P-GNN embeddings, which possess an efficient inductive capability to share information between nodes, enabling it to better generalize and adapt to unseen test data.

Conclusion

This study introduces a novel route recommendation model, TLMR. This model aims to effectively recommend routes by analyzing historical trajectory data to learn the underlying route choice patterns. The essence of TLMR lies in integrating multiple sources of features, significantly enhancing

the model's ability to understand user route choice behavior in complex environments. The model combines a location-aware graph neural network with a recurrent neural network, enabling TLMR to deeply understand and effectively utilize the structural features of the road network and its dynamic changes, thereby demonstrating strong generalization performance. Moreover, the model applies a beam search algorithm to generate candidate routes, further enhancing the accuracy and practicality of the recommendations. Extensive experimental results on four large real-world datasets show that TLMR significantly outperforms comparative methods in key performance indicators such as precision, recall rate, F1 score, and accessibility. These results not only validate the effectiveness of the TLMR model in the field of route recommendation but also highlight its advanced nature.

However, a limitation of this study is that it does not consider the need for personalized recommendations, meaning the current model has yet to offer customized travel suggestions for different users. Given the importance of personalized recommendations in practical applications, future research will focus on developing a personalized route recommendation system that meets individual user needs. Additionally, we plan to explore the model's transferability across different cities to verify and enhance TLMR's applicability and effectiveness in various urban environments.

Acknowledgements This study was supported by the "Pioneer" and "Leading Goose" R&D Program of Zhejiang (Grant number 2023C01231).

Data availability No applicable to this article.

Declarations

Conflict of interest The authors declare that they have no Conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License, which permits any non-commercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if you modified the licensed material. You do not have permission under this licence to share adapted material derived from this article or parts of it. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by-nc-nd/4.0/.

References

1. Mondal S, Gupta A (2023) Evaluation of driver acceleration/deceleration behavior at signalized intersections using vehicle trajectory data. Transportation Letters 15(4):350–362
2. Fu C, Liu H (2023) Investigating distance halo effect of fixed automated speed camera based on taxi gps trajectory data. Journal of Traffic and Transportation Engineering (English Edition) 10(1):70–85. https://doi.org/10.1016/j.jtte.2021.05.005
3. Huang Tang YY, Shen H (2022) A model for vessel trajectory prediction based on long short-term memory neural network. Journal of Marine Engineering & Technology 21(3):136–145
4. Joshi M, Singh A, Ranu S et al (2022) Foodmatch: Batching and matching for food delivery in dynamic road networks. ACM Trans Spatial Algorithms Syst 8(1):1–2. https://doi.org/10.1145/3494530
5. Emir Žunic SD, Donko D (2022) Adaptive multi-phase approach for solving the realistic vehicle routing problems in logistics with innovative comparison method for evaluation based on real gps data. Transportation Letters 14(2):143–156
6. Risald Mirino AE, Suyoto (2017) Best routes selection using dijkstra and floyd-warshall algorithm. In: 2017 11th International Conference on Information & Communication Technology and System (ICTS), pp 155–158,https://doi.org/10.1109/ICTS.2017.8265662
7. Cao W, Shi H, Zhu S, et al. (2009) Application of an improved a* algorithm in route planning. In: 2009 WRI Global Congress on Intelligent Systems, pp 253–25https://doi.org/10.1109/GCIS.2009.76
8. Quercia D, Schifanella R, Aiello LM (2014) The shortest path to happiness: Recommending beautiful, quiet, and happy routes in the city. In: Proceedings of the 25th ACM Conference on Hypertext and Social Media. Association for Computing Machinery, HT '14, pp 116–125
9. Simmons R, Browning B, Zhang Y, et al. (2006) Learning to predict driver route and destination intent. In: 2006 IEEE Intelligent Transportation Systems Conference, pp 127–13https://doi.org/10.1109/ITSC.2006.1706730
10. Akabane AT, Pazzi RW, Madeira ERM, et al. (2017) Modeling and prediction of vehicle routes based on hidden markov model. In: 2017 IEEE 86th Vehicular Technology Conference (VTC-Fall), pp 1–https://doi.org/10.1109/VTCFall.2017.8288309
11. Najafabadi MM, Villanustre F, Khoshgoftaar TM et al (2015) Deep learning applications and challenges in big data analytics. Journal of big data 2(1):1–21
12. Wang J, Wu N, Zhao WX, et al. (2019) Empowering a* search algorithms with neural networks for personalized route recommendation. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. Association for Computing Machinery, KDD '19, pp 539–547,https://doi.org/10.1145/3292500.3330824
13. Xing X, Li X (2023) Recommendation of urban vehicle driving routes under traffic congestion: A traffic congestion regulation method considering road network equilibrium. Computers and Electrical Engineering 110:108863. https://doi.org/10.1016/j.compeleceng.2023.108863
14. Wu H, Chen Z, Sun W, et al. (2017) Modeling trajectories with recurrent neural networks. In: IJCAI, pp 3083–3090
15. Li X, Cong G, Cheng Y (2020) Spatial transition learning on road networks with deep probabilistic models. In: 2020 IEEE 36th International Conference on Data Engineering (ICDE), pp 349–36https://doi.org/10.1109/ICDE48307.2020.00037
16. Liang Y, Zhao Z (2022) Nettraj: A network-based vehicle trajectory prediction model with directional representation and spatiotemporal attention mechanisms. IEEE Transactions on Intelligent Transportation Systems 23(9):14470–1448. https://doi.org/10.1109/TITS.2021.3129588
17. Kipf TN, Welling M (2016) Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907
18. Velickovic P, Cucurull G, Casanova A et al (2017) Graph attention networks. stat 1050(20):10–48550

19. You J, Ying R, Leskovec J (2019) Position-aware graph neural networks. In: Chaudhuri K, Salakhutdinov R (eds) Proceedings of the 36th International Conference on Machine Learning, Proceedings of Machine Learning Research, vol 97. PMLR, pp 7134–7143
20. Chung J, Gulcehre C, Cho K, et al. (2014) Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555
21. Fu L, Sun D, Rilett L (2006) Heuristic shortest path algorithms for transportation applications: State of the art. Computers & Operations Research 33(11):3324–3343. Part Special Issue: Operations Research and Data Mining
22. Fan D, Shi P (2010) Improvement of dijkstra's algorithm and its application in route planning. In: 2010 Seventh International Conference on Fuzzy Systems and Knowledge Discovery, pp 1901–190https://doi.org/10.1109/FSKD.2010.5569452
23. Zhang Jd, Yj Feng, Ff Shi et al (2016) Vehicle routing in urban areas based on the oil consumption weight-dijkstra algorithm. IET Intelligent Transport Systems 10(7):495–502
24. Wang C, Wang L, Qin J, et al. (2015) Path planning of automated guided vehicles based on improved a-star algorithm. In: 2015 IEEE International Conference on Information and Automation, pp 2071–207https://doi.org/10.1109/ICInfA.2015.7279630
25. Wang X, Zhang H, Liu S et al (2022) Path planning of scenic spots based on improved a* algorithm. Scientific Reports 12(1):1320
26. Qin Z, Shao H, Wang F et al (2022) A reliable energy consumption path finding algorithm for electric vehicles considering the correlated link travel speeds and waiting times at signalized intersections. Sustainable Energy, Grids and Networks 32:100877
27. Quercia D, Schifanella R, Aiello LM (2014) The shortest path to happiness: Recommending beautiful, quiet, and happy routes in the city. In: Proceedings of the 25th ACM Conference on Hypertext and Social Media. Association for Computing Machinery, HT '14, pp 116–12https://doi.org/10.1145/2631775.2631799
28. Eddy SR (2004) What is a hidden markov model? Nature biotechnology 22(10):1315–1316
29. Yu Y, Si X, Hu C et al (2019) A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures. Neural Computation 31(7):1235–1270. https://doi.org/10.1162/neco_a_01199
30. Li H, Xing W, Jiao H et al (2024) Deep bi-directional information-empowered ship trajectory prediction for maritime autonomous surface ships. Transportation Research Part E: Logistics and Transportation Review 181:103. https://doi.org/10.1016/j.tre.2023.103367
31. Hu W, Li M, Kwan MP et al (2024) Real-time taxi spatial anomaly detection based on vehicle trajectory prediction. Travel Behaviour and Society 34:10069. https://doi.org/10.1016/j.tbs.2023.100698
32. Feng J, Li Y, Zhang C, et al. (2018) Deepmove: Predicting human mobility with attentional recurrent networks. In: Proceedings of the 2018 World Wide Web Conference. International World Wide Web Conferences Steering Committee, WWW '18, pp 1459–146https://doi.org/10.1145/3178876.3186058
33. Yan B, Zhao G, Song L et al (2023) Precln: Pretrained-based contrastive learning network for vehicle trajectory prediction. World Wide Web 26(4):1853–1875
34. Wang C, Li C, Huang H et al (2023) Asnn-frr: A traffic-aware neural network for fastest route recommendation. Geoinformatica 27:39–60
35. Chen X, Li J, Wang H (2019) Keyphrase enhanced diverse beam search: A content-introducing approach to neural text generation. IEEE Access 7:72716–7272. https://doi.org/10.1109/ACCESS.2019.2919974
36. Shang Z, Li G, Bao Z (2018) Dita: Distributed in-memory trajectory analytics. In: Proceedings of the 2018 International Conference on Management of Data. Association for Computing Machinery, SIGMOD '18, pp 725–740,https://doi.org/10.1145/3183713.3183743
37. kaggle (2022) Porto dataset. https://www.kaggle.com/datasets/
38. Gaia (2022) Didi chuxing open data. https://gaia.didichuxing.com/
39. OpenStreetMap (2022) Map. https://www.openstreetmap.org/
40. Yang C, Gidófalvi G (2018) Fast map matching, an algorithm integrating hidden markov model with precomputation. International Journal of Geographical Information Science 32(3):547–57https://doi.org/10.1080/13658816.2017.1400548

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.