Waypoint POI Recommendation for Vehicle Navigation Services using Hierarchical Graphs and Contrastive Learning

Jongsoo Lee
Hanyang University
Seoul, South Korea
leejongsoo@hanyang.ac.kr

Heejun Shin
Hanyang University
Seoul, South Korea
shin970421@hanyang.ac.kr

Namhyuk Kim
Hyundai Motor Company
Seoul, South Korea
namhyuk.kim@hyundai.com

Dong-Kyu Chae*
Hanyang University
Seoul, South Korea
dongkyu@hanyang.ac.kr

Abstract

Modern vehicle navigation systems can greatly benefit from *waypoint point-of-interest* (POI) *recommendation*, which suggests personalized intermediate stops along a driving route. This paper defines the novel waypoint POI recommendation problem: given a starting point and a destination, recommend one or more personalized POIs to visit en route. This scenario (e.g., suggesting a lunch stop during a road trip) differs from the conventional "next POI" recommendation in that it infers waypoint POIs from only two (origin and destination) inputs and predicts multiple intermediate stops rather than a single next location. To solve this problem, we propose **WayPOI**, a novel recommender model for Waypoint POI suggestion based on *hierarchical graph based contrastive learning* (*WayPOI*). WayPOI constructs a hierarchical graph that captures both individual and group-level behavioral patterns of users and POIs, and it employs a contrastive learning strategy to learn effective user and POI representations from sparse data. Through experiments on real-world driving data provided by Hyundai as well as on three public datasets, we demonstrate that WayPOI significantly outperforms several recent POI recommendation models, even though these baselines were carefully re-formed and retrained to perform waypoint recommendation for a fair comparison. Our ablation study confirms the benefit of each proposed component.

CCS Concepts

• **Information systems** → **Recommender systems**.

Keywords

Waypoint POI recommendation, contrastive learning, graph neural networks, hierarchical graphs

ACM Reference Format:
Jongsoo Lee, Heejun Shin, Namhyuk Kim, and Dong-Kyu Chae. 2025. Waypoint POI Recommendation for Vehicle Navigation Services using Hierarchical Graphs and Contrastive Learning. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management (CIKM '25), November 10–14, 2025, Seoul, Republic of Korea.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3746252.3761519

*Corresponding author.

CIKM '25, Seoul, Republic of Korea

ACM ISBN 979-8-4007-2040-6/2025/11
https://doi.org/10.1145/3746252.3761519

1 Introduction

Personalized vehicle navigation has become increasingly important as more drivers rely on in-car GPS navigation systems [9]. Traditional navigation systems primarily compute the shortest or fastest routes from a starting point to a destination, based on Global Positioning System (GPS) data [22]. However, recent advances in sensor technology and connectivity have enabled the collection of large-scale user driving data, sparking interest in more intelligent navigation services that leverage data-driven insights [8, 19]. Automakers such as Hyundai Motor Company are now exploring data-powered smart services beyond basic route guidance.

One such service is *waypoint recommendation*, which suggests personalized places to stop along a route to enhance the driver's journey. For example, consider a typical navigation scenario where a user enters a destination into the car's navigation. In addition to the primary route guidance, the system could recommend a set of personalized waypoints (e.g., restaurants, scenic spots, or attractions) to visit along the way. These suggestions can be personalized based on the user's preferences (learned from their historical check-in data) and current trip context (start and end points). By providing such intermediate Point of Interest (POI) recommendations, the navigation system can enrich the travel experience (e.g., discovering a popular local eatery for lunch) and increase user satisfaction. To enable this functionality, this paper formally defines the *waypoint POI recommendation problem*: given a starting location and a final destination (and access to the user's historical POI check-ins), determine one or more intermediate POIs that the user would be interested in visiting en route.

This problem is distinct from conventional next POI recommendation [6, 18, 20, 31] in several ways. First, a waypoint recommender system only takes the trip's start and end points as input (plus historical profiles) to infer the user's intermediate stops. Second, instead of predicting a single next location, the waypoint recommendation system suggests multiple POIs as potential waypoints. As a result, the waypoint recommendation poses a challenge as the model has to produce multi-label prediction with less contextual input compared to the traditional next-POI task. Unfortunately, existing next-POI recommenders [17, 23, 29, 32, 34, 35] highly focus on sequential modeling based on a rich sequence of recent check-ins, thus they do not naturally fit the waypoint recommendation task.

Considering the typical navigation scenarios where the system only knows a start and end point as immediate input, it is important to design a powerful approach to learning high-quality representations of users and POIs. Herein, we propose a novel framework called **WayPOI** for personalized waypoint recommendation. In

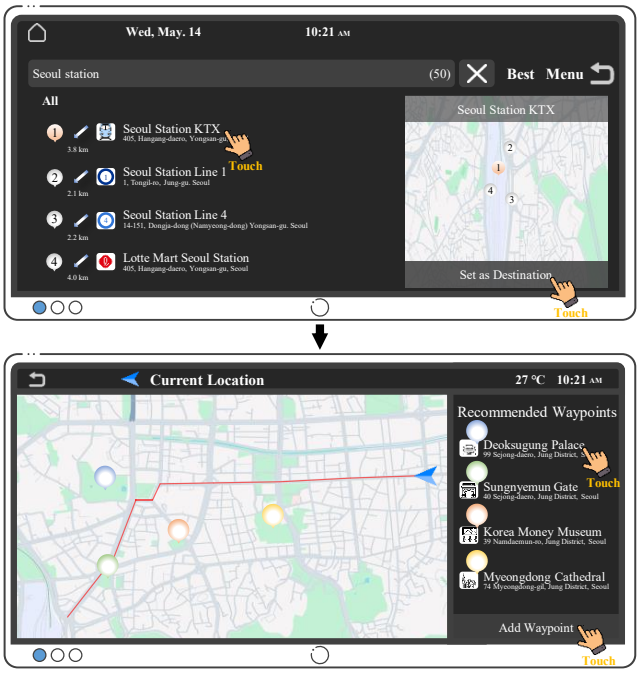

Figure 1: An example of waypoint recommendation scenario in an in-car navigation system.

order to strengthen the representational power of each individual (user/POI) embedding, our WayPOI builds hierarchical graph representation and performs contrastive learning based on it.

(1) *Hierarchical graph*: Basically, we construct a user graph and a POI graph that represent relationships between objects. Together with this, our method focuses on higher-level patterns shared across similar users and POIs, which we term *group-level information*. In reality, users often fall into latent groups or clusters (e.g., 'pet owner', 'museum lovers', 'family with kids') that share similar preferences, and POIs can be grouped by categories or popular visit patterns. Based on this insight, we construct a *hierarchical graph* that organizes user/POI nodes into multiple levels, linking individual users to *group nodes* representing clusters of users with similar behavior, and likewise for POIs. This graph allows the model to learn embeddings that blend rich relationships at both the individual level and the group level. We then perform representation learning on this graph using *graph neural networks* (GNNs) [15, 26, 30] that aggregates information through the hierarchy.

(2) *Contrastive learning*: We conduct contrastive learning [3, 7, 14] based on the hierarchical graph. We classify the group nodes into a positive pair and a negative pair based on whether they include the given target node. Contrastive learning with these positive/negative pairs enables representation learning that effectively captures group-level characteristics. However, there is a risk that some group nodes sharing similar characteristics with the target node might be incorrectly categorized into the negative pair. To alleviate this issue, we design an improved contrastive learning by allowing a *neutral* pair that can be excluded from contrasting. To this end, we measure the similarity between the target node and candidate negative group nodes based on a predefined feature list, and if the similarity exceeds a certain threshold, the nodes are classified as a neutral pair. This ensures that only clearly negative pairs are considered in the contrastive learning process.

Additionally, we employ a multi-head decoder [34] to incorporate *time and category* information into the waypoint recommendation process. We performed extensive experiments using (anonymized) user driving data provided by Hyundai and three public datasets. The results confirm that WayPOI outperforms the existing POI recommenders which were carefully re-formed and retrained to perform waypoint recommendation for a fair comparison.

2 Related Work

Next-POI recommendation aims to predict a user's personalized subsequent location based on her/his historical check-in data. Due to its sequential nature, early studies attempted to address this problem using **recurrent neural networks** (RNNs) or Long Short-Term Memory (LSTM) [18, 32, 35]. For instance, ST-RNN [18] incorporated spatio-temporal transition information into the RNN layer. STGN [35] enhanced the LSTM units with time gates and distance gates to capture user preferences in short-term and long-term check-in sequences. Flashback [32] encoded the visiting sequence by using a weighted average of historical hidden states based on spatio-temporal contexts.

Recently, motivated by **Transformers** [28], a number of works have applied the self-attention mechanism to POI recommendation [5, 17, 20]. GeoSAN [17] applies self-attention to the encoding of hierarchical grid map information as well as path information. STAN [20] aims to learn the relationships between discontinuous POIs in the same trajectory by applying self-attention to a multi-modal embedding that combines user trajectory embeddings and spatio-temporal embeddings. MTNet [5] encodes tree-structured trajectories (built from check-in context) using a Transformer-based self-attention encoder and a TreeLSTM [27].

On the other hand, graph-based approaches have also been explored to incorporate relational information beyond just sequential paths [23, 34]. These works model a graph that represents relationships between various entities (e.g., POIs, users) and apply **Graph neural network** to learn better representations for POI recommendation. For instance, GETNext [34] proposed a method that learns POI embeddings for recommendation by leveraging a trajectory flow POI graph and a transition attention map. Graph Flashback [23] leveraged a knowledge graph incorporating social relationships between users, spatio-temporal relationships between POIs, and user–POI visit information to learn enriched representations.

3 Problem Definition

This section defines a novel *waypoint recommendation problem* in the context of vehicle navigation scenarios. This setting aligns closely with real-world navigation use cases and indeed holds high potential for deployment as a real service in the automotive industry. It can assist drivers in quickly and easily deciding intermediate destinations, such as selecting a restaurant for dining—even while driving. It ultimately contributes to improved customer satisfaction.

Formally, let $U = \{u_1, u_2, \ldots, u_N\}$ be a set of N users, $P = \{p_1, p_2, \ldots, p_M\}$ be a set of M POIs, and $T = \{t_1, t_2, \ldots, t_K\}$ be the set of timestamps. Each $u_i \in U$ and each POI $p_j \in P$ are represented by feature vectors (e.g., age, gender for users; category,

location for POIs; see Table 1 for more details), where user feature vectors are of dimension d_{user} and POI feature vectors are of dimension d_{POI}. We define a trajectory $R_u^j = \{r_u^0, r_u^1, r_u^2, \ldots\}$, which represents user u's check-in sequence for day j[1]. In our waypoint recommendation problem, each trajectory R_u^j is defined as a single instance: For a trajectory consisting of D check-ins[2], we treat the first check-in r_u^0's POI as the starting point and the last check-in r_u^D's POI as the destination. These two POIs are used as input to the recommender model; then, it aims to provide a top-k list of candidate waypoints that u is likely to visit.

4 Proposed Method

We propose a hierarchical graph-based contrastive learning framework for waypoint recommendation, referred to as WayPOI. Figure 2 illustrates an overview of WayPOI. It consists of the contrastive learning module using the hierarchical graph and the multi-head decoder for recommendation using the time-category encoder.

4.1 User/POI Graph Construction

To effectively learn the relationships within users and within POIs, we construct a user graph and a POI graph.

4.1.1 POI Graph. Formally, a POI graph $G_P = (V_P, E_P)$ where each node represents a POI and has a feature vector of the POI's attributes (e.g., longitude, latitude, category). We connect POIs which frequently appear in the same trajectory, which would allow the model to well-capture POI visitation patterns and relationships between POIs. Additionally, such co-occurring POIs are often geographically close, so constructing the graph in this way helps prevent the model from recommending waypoints that deviate significantly from the intended route. To incorporate this, we build a POI graph where an edge between two POIs is given a weight equal to the number of times those POIs appear together on a trajectory.

To update the POI embeddings by aggregating information from neighboring nodes, we employ a graph convolutional network (GCN) [13] as an encoder. Using G_p's adjacency matrix $A_P \in \mathbb{R}^{M\times M}$, its degree matrix D_p, and the identity matrix I_p, we first compute the normalized Laplacian matrix $\tilde{L}_p$ by:

$$\tilde{L}_p = (D_p + I_p)^{-1}(A_p + I_p). \tag{1}$$

Then, the initial POI feature matrix $X_p \in \mathbb{R}^{M\times d_{POI}}$ is transformed into a node embedding matrix $H_p \in \mathbb{R}^{M\times d'_{POI}}$ via a learnable weight matrix W_p, where d'_{POI} is the POI embedding dimension. We use the GCN to update these embeddings. At the l-th GCN layer, the node embedding update is given by:

$$H_p^{(l)} = \sigma\left(\tilde{L}_p H_p^{(l-1)} W_p^{(l)} + b_p^{(l)}\right) \tag{2}$$

where l denotes the layer index, $W_p^{(l)}$ and $b_p^{(l)}$ are the learnable weight matrix and bias at the l-th layer, and $\sigma()$ is the non-linear activation function (we use LeakyReLU).

[1]We consider any check-in between 00:00 and 03:00 as an extension of the previous day's trajectory R_u^{j-1} and is included in that previous trajectory.

[2]The number of check-ins in each trajectory can vary.

4.1.2 User Graph. Next, in a similar manner, we construct a user graph $G_u = (V_u, E_u)$. we construct a user graph where an edge between two users has a weight equal to the frequency of their co-visits to the same POI. Each node in G_u represents a user and has a feature vector of user attributes (e.g., gender, age, hobby). As done for the POI graph, we again use a separate GCN encoder to update user node embeddings by:

$$\tilde{L}_u = (D_u + I_u)^{-1}(A_u + I_u) \tag{3}$$

$$H_u^{(l)} = \sigma\left(\tilde{L}_u H_u^{(l-1)} W_u^{(l)} + b_u^{(l)}\right) \tag{4}$$

4.2 Hierarchical Graph

The employed GNN-based model aggregates information from neighbor nodes based on pairwise relationships to update a target (user/POI) node's embedding. In addition to such individual information, we further consider group-level information to improve the quality of node representations. Identifying a group of related nodes can be seen as a graph partitioning or subgraph retrieval. We construct a hierarchical graph by recursively partitioning G_u and G_p according to certain features, such that each subgraph represents a related node group.

For the POI graph, we first define a list of partitioning features Q_p. This feature list can be heuristically defined by human experts (in our case, domain experts working at Hyundai Motor Company selected the features for Q_p as well as Q_u, see Table 1). Then, using the first feature q_1 in $Q_p = \{q_1, q_2, \ldots, q_n\}$, we partition the POI graph G_p into a set of subgraphs $C_{q_1} = \{c_{q_1}^1, c_{q_1}^2, \ldots, c_{q_1}^o\}$, where n is the number of partitioning features and o is the number of unique values of feature q_1. Each subgraph retains the edges from the original graph G_p (e.g., $c_{q_1}^1 = (V_{q_1}^1, E_{q_1}^1)$), except that edges between nodes in different subgraphs are removed (i.e., only connections among POIs sharing the same q_1 value remain).

Next, each subgraph is processed by a GCN to aggregate information from its neighboring nodes and update the node embeddings. We then apply mean pooling to the node embeddings of each subgraph to obtain a group node representation $e_{q_1}^1$ by:

$$e_{q_1}^i = Mean(f_{GCN}(c_{q_1}^i)) \tag{5}$$

where $f_{GCN}(c_{q_1}^i)$ outputs the set of node embeddings in subgraph $c_{q_1}^i$ after the GCN update.

Finally, to learn relationships among the groups, we connect all group nodes obtained from the above step (within the same level) to form a fully-connected group graph $z_{q_1}^0$. We then apply another GCN (denoted $f_{GCN_{clus}}$) on this group graph to learn the inter-group relationships, producing a new set of higher-level node embeddings $E_{q_1} = \{e_{q_1}^0, e_{q_1}^1, \ldots, e_{q_1}^o\}$:

$$E_{q_1} = f_{GCN_{clus}}(z_{q_1}^0) \tag{6}$$

After the first split using q_1, the same partitioning process is repeated for the remaining features $\{q_2, \ldots, q_n\}$ in Q_p, recursively dividing each subgraph further to build deeper layers of the hierarchical graph. At each subsequent split, edges are maintained only among nodes that originated from the same parent group in order to encourage the model to focus only on within-group information and reduce the influence of unrelated nodes.

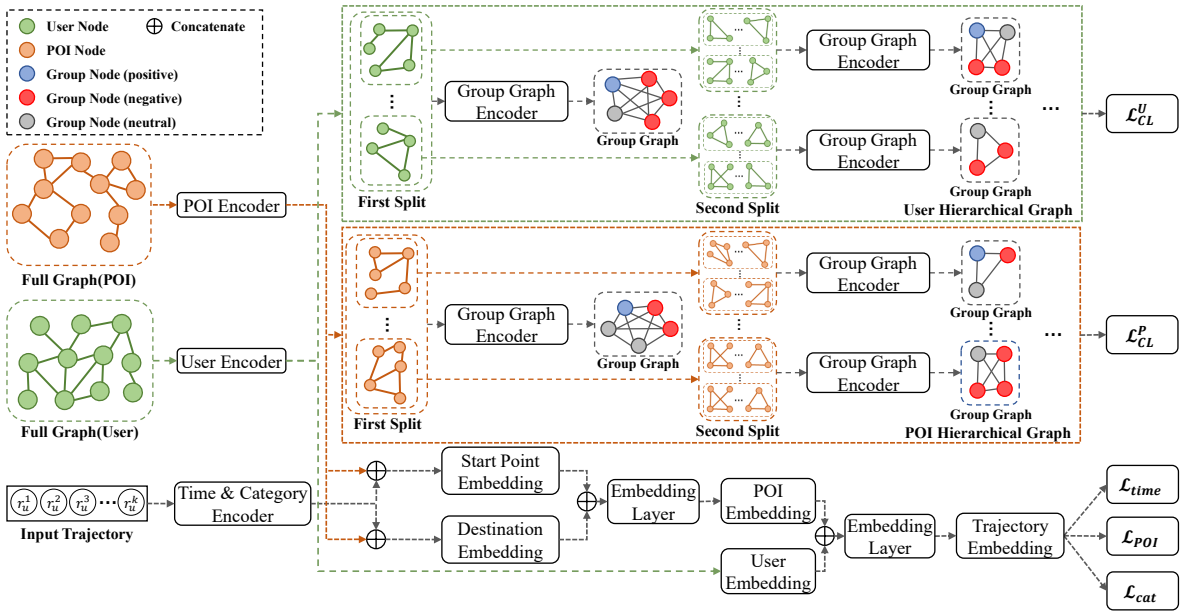

Figure 2: The overview of our WayPOI framework.

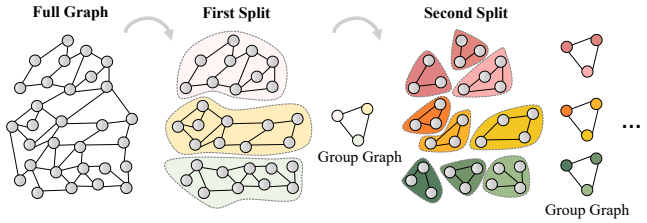

Figure 3: Hierarchical graph construction process.

Figure 3 illustrates this hierarchical graph construction process. We perform an analogous hierarchical partitioning on the user graph G_u using a feature list

However, the limited availability of user and POI attributes in *public* trajectory datasets would make it difficult to pre-define partitioning features. For this case, we propose an alternative graph partitioning method that does not require a predefined feature list. Given the adjacency matrix of the graph (user or POI), we first compute the normalized Laplacian matrix ($\tilde{L}_p$ or $\tilde{L}_u$), and then calculate the Fiedler vector of the Laplacian (the eigenvector corresponding to the second-smallest eigenvalue). We finally sort the nodes based on the values in the Fiedler vector and split the graph into two subgraphs by minimizing the Normalized Cut (NCut) score:

$$\mathcal{L}_{\text{NCut}} = \frac{\text{cut}(V_1, V_2)}{\text{vol}(V_1)} + \frac{\text{cut}(V_1, V_2)}{\text{vol}(V_2)} \tag{7}$$

where $\text{cut}(V_1, V_2)$ is the sum of edge weights between the two subgraphs V_1 and V_2, and $\text{vol}(V_i)$ is the sum of weights of all edges incident to nodes in V_i.

If the two resulting subgraphs are highly imbalanced in size, it can hinder further hierarchical splitting. To mitigate this, we introduce a balance term to penalize large size disparity between the subgraphs. The final graph partitioning can be defined as:

$$\mathcal{L}_{\text{NCut-bal}} = \mathcal{L}_{\text{NCut}} + \lambda \left(\frac{|V_1| - |V_2|}{|V_1| + |V_2|} \right)^2 \tag{8}$$

where $|V_1|$ and $|V_2|$ denote the number of nodes in each subgraph and λ is a hyperparameter controlling the strength of the balance term. This partitioning approach allows us to construct a hierarchical graph using only the graph connectivity, which is particularly useful when node attributes are limited as in public datasets.

4.3 Contrastive Learning

In order to inject group-level information into the node representation learning process, we employ contrastive learning that encourages embeddings of nodes from the same group to be close to each other, while pushing apart those from different groups.

A naive way to apply contrastive learning here would be to treat the group nodes containing a target node as the positive pair and all other groups as negative pairs. However, due to the initial partitioning feature, such approach would sometimes result in semantically similar group nodes being considered negative pairs. To address this issue, we improve the contrastive learning method by filtering out negative pairs which are actually quite similar within each other.

Given a target node, for each hierarchy level, we compute a similarity of features between the positive (containing the target) group node and each negative group node. If the similarity exceeds a threshold γ, we regard that pair as *neutral* (not truly dissimilar and do not include it as a negative pair in contrastive learning. We denote the set of positive pairs as S^+ and the set of negative pairs as S^-, where S^- is obtained by removing neutral pairs from the set of negative pairs. The following contrastive learning loss function then excludes the neutral pairs from training:

$$\mathcal{L}_{\text{PCL}} = -\log \frac{\exp\left(\frac{1}{|S^+|} \sum_{i \in S^+} \frac{\text{sim}_{\text{pos},i}}{\tau} \right)}{\exp\left(\frac{1}{|S^+|} \sum_{i \in S^+} \frac{\text{sim}_{\text{pos},i}}{\tau} \right) + \sum_{j \in S^-} \exp\left(\frac{\text{sim}_{\text{neg},j}}{\tau} \right)} \tag{9}$$

where $|S^+|$ and $|S^-|$ are the number of positive and negative pairs respectively and τ is the temperature hyperparameter.

Again, in the case of using public datasets that do not involve partitioning features, we propose the following alterative way to compute similarity based on the structure of our hierarchical graph:

$$\text{sim}(\mathcal{C}_{\text{pos}}, C_j) = \frac{d_{\text{common}}}{d_{\text{max}}} \tag{10}$$

where $\mathcal{C}_{\text{pos}}$ denotes the set of positive groups that include the target node and C_j represents a negative group. d_{common} is the depth of the closest common ancestor of groups $\mathcal{C}_{\text{pos}}$ and C_j in the hierarchy, and d_{max} is the depth of C_j. This metric yields a higher similarity value for groups that are nearer in the hierarchy.

4.4 Multi-Head Decoder

Typically, the departure time and the category information of the start/destination points are crucial factors that influence POI selection [18, 20, 32, 34]. We thus design a time-category encoder to incorporate such additional information into our framework.

We discretize the 24-hour day into 30-minute intervals, and apply Time2Vec [10] to encode teach departure time slot t into a time embedding e_t:

$$e_t[i] = \begin{cases} \omega_i t + \lambda_i, & i = 0 \\ \sigma(\omega_i t + \lambda_i), & 1 \le i < d_{time} \end{cases} \tag{11}$$

where ω_i and λ_i are learnable parameters, i is the index of the time embedding dimension and σ is an activation function (we use the sine function following Time2Vec [10]). d_{time} is a dimension of time embedding vectors.

We embed the POI categories of the starting/destination points using an embedding layer f_{cat}, resulting in a vector $e_{cat} \in \mathbb{R}^{d_{cat}}$. We then concatenate time embedding e_t and category embedding e_{cat}, and pass the result through a dense layer to obtain the time-category embedding e_{tc}:

$$e_{tc} = \text{ReLU}\left(W_{tc}[e_t, e_{cat}] + b_{tc}\right) \tag{12}$$

where $[\quad \| \quad]$ indicates the concatenation operation.

The time-category embeddings of the starting point e^s_{tc} and destination e^d_{tc} are each combined with their corresponding POI embeddings e^s_p and e^d_p. The concatenated embeddings are then passed through a dense layer to produce the final source-destination (sd) pair embedding e_{sd}:

$$e_{sd} = \text{ReLU}\left(W_{sd}[[e^s_{tc}, e^s_p], [e^d_{tc}, e^d_p]] + b_{sd}\right) \tag{13}$$

Next, we concatenate the derived e_{sd} with the target user embedding e_u, and pass this through another dense layer to obtain the trajectory embedding e_r that captures the context of *who is traveling, from where to where, and when*:

$$e_r = \text{ReLU}\left(W_r[e_{sd}, e_u] + b_r\right) \tag{14}$$

Finally, we design a multi-head decoder that jointly predicts the waypoint POI, visit time, and category. All decoder heads share the same trajectory embedding e_r as input and each head predicts its respective objective:

$$\hat{Y}_{POI} = e_r W_{POI} + b_{POI} \tag{15}$$

$$\hat{Y}_{time} = e_r W_{time} + b_{time} \tag{16}$$

$$\hat{Y}_{cat} = e_r W_{cat} + b_{cat} \tag{17}$$

where $W_{POI} \in \mathbb{R}^{d \times M}$, $W_{time} \in \mathbb{R}^{d \times 1}$, and $W_{cat} \in \mathbb{R}^{d \times \Omega}$ are learnable weight matrices (with biases b_{POI}, b_{time}, b_{cat} for each head). Here M is the total number of POIs and Ω is the number of POI categories. During training, we optimize all decoder heads jointly. At inference time, however, we use only the POI decoder's output $\hat{Y}_{POI}$ to produce the ranked list of waypoint recommendations.

4.5 Loss Function

The waypoint recommendation problem is formulated as a multi-label classification task, since a user's trajectory may include multiple waypoint POIs. This setting is highly imbalanced: the number of possible POIs is very large, and for any given trajectory only a few POIs are actually visited (positive labels) while the rest are not (negative labels). Due to this extreme class imbalance, a model could trivially achieve a low error by predicting every POI as "not visited." To address this problem, we adopt an asymmetric loss function [25] that was designed for multi-label classification with many negatives by down-weighting them:

$$\mathcal{L}_{POI} = \begin{cases} \mathcal{L}_+ = (1-p)^{\eta+} \log(p) \\ \mathcal{L}_- = p^{\eta-} \log(1-p) \end{cases} \tag{18}$$

where $p = \sigma(\hat{Y}^k_{POI})$ is the predicted probability for POI k based on the given waypoint POI prediction logit $\hat{Y}^k_{POI}$. $\mathcal{L}_+$ is the loss calculated for positive samples (i.e., ground-truth POIs) and $\mathcal{L}_-$ is the loss for negative samples. $\eta+$ and $\eta-$ control (reduce) the influence of positive and negative samples, respectively. We set $\eta- > \eta+$ to give more emphasis to learning from the scarce positive examples. We apply the same form of loss for the category prediction head as well, treating the true waypoint category as a multi-label target (one-hot encoding over Ω categories). For the time prediction head, we use mean squared error (MSE) between the predicted visit time and the actual average visit time for the ground-truth waypoint.

Finally, we adopt a joint training approach where the contrastive learning loss and the multi-head decoder loss are optimized simultaneously to enable effective model learning:

$$\mathcal{L}_{joint} = \mathcal{L}_{POI} + \mathcal{L}_{cat} + \alpha \cdot \mathcal{L}_{time} + \beta \cdot (\mathcal{L}^U_{CL} + \mathcal{L}^P_{CL}), \tag{19}$$

where $\mathcal{L}^U_{CL}$ and $\mathcal{L}^P_{CL}$ denotes the loss for contrastive learning on the user and POI hierarchical graph, respectively. α and β adjust the scales of the time loss and contrastive loss respectively.

5 Experimental Results

5.1 Settings

5.1.1 Dataset. We conducted experiments on four real-world datasets. The statistics of each dataset are summarized in Table 2.

- **Hyundai:** This dataset is constructed from user location traces and search records collected via Hyundai's navigation system. It includes various attributes of users and POIs (e.g., gender, age, vehicle type, POI category) as well as POI visit records. Throughout the entire data collection and processing pipeline, *strict anonymity has been maintained at all times.*
- **Foursquare-NYC[33], Foursquare-TKY[33], Gowalla-CA [1]:** These are public datasets collected from location-based social media platforms. These datasets consist of check-in records

Table 1: User/POI features in the Hyundai and public datasets. The underlined features are the partitioning features chosen by human experts.

Feature type	Feature name	Hyundai	Public (NYC, TKY, CA)
User features	**Vehicle type (1st)**	✓	
	Age (2nd)	✓	
	Gender (3rd)	✓	
	With Pet	✓	
	With children	✓	
	Corporate/Individual	✓	
	Total check-ins	✓	✓
	Most Category	✓	✓
POI features	**Season (most freq.) (1st)**	✓	
	Daytype (most freq.) (2nd)	✓	
	Timezone (most freq.) (3rd)	✓	
	Total check-ins	✓	✓
	Category	✓	✓
	Longitude	✓	✓
	Longitude	✓	✓
	Latitude	✓	✓

from New York City (NYC), Tokyo (TKY), and the California & Nevada (CA) region, respectively. Each record contains user ID, POI ID, POI category, longitude, latitude, and timestamp.

Following prior studies, we removed infrequently visited POIs (those with fewer than 5 check-ins) and users with fewer than 5 trajectories from each dataset. For each user, the first 80% of their trajectories were used as the training set, the middle 10% as the validation set, and the last 10% as the test set.

5.1.2 Compared Models. We compare our method with the following baselines and next-POI recommendation systems:

- **Baselines:** We implemented three simple popularity-based baselines. **Popularity (Global)** recommends POIs that are most frequently visited across all users; **Popularity (Route)** recommends POIs that frequently appear in historical trajectories with the same starting point and destination; **Popularity (Distance)** recommends the most popular POI within a radius of 3km around the midpoint between the starting point and destination.
- **Traditional recommenders:** We employed several recommendation models such as **LightGCN** [4], **FPMC** [24] and **PRME** [2]. These methods focus on modeling user-item interactions without incorporating route-specific information.
- **Next-POI recommenders:** We employed the following models, covering a comprehensive range from traditional RNN-based recommenders to recent Transformer and GNN-based ones: **Flashback** [32], **STRNN** [18], **STAN** [20], **GETNext** [34] and **MTNet** [5]. They typically perform next-POI prediction based on a sliding window of previous visits, which is not well-suited to our waypoint recommendation task. For a fair comparison, we modified their input to use only the start/end POIs; their prediction head were also modified to use the same asymmetric loss function [25] as our model. We then trained them using the training datasets so that they can learn to recommend waypoints given only the starting point and destination.

5.1.3 Implementation Details. Our model was trained for 100 epochs using the Adam optimizer [12] with a learning rate of 1e-3, with early stopping applied (patience of 15 epochs). The embedding dimension for d^l_{POI} and d^l_{user} was set to 64, and for d_{time} and d_{cat} to 32. For the asymmetric loss function, the hyperparameters $\eta+$ and $\eta-$ were set to 4 and 1, respectively. The scaling factors α and β for the time and contrastive losses were both set to 10. The attribute similarity threshold for selecting neutral pairs in the contrastive learning was 60%. For performance evaluation, we used Recall@K, MAP@K, and NDCG@K [16, 21].

Table 2: Dataset statistics.

Dataset	# user	# POI	# category	# check-in	# trajectory
Hyundai	1,001	8,897	37	293,305	146,783
NYC	1,075	5,099	318	104,074	14,160
TKY	2,281	7,844	291	361,430	44,692
CA	4,318	9,923	301	250,780	32,920

5.2 Performance Comparisons

The comparison results on each dataset are shown in Tables 3 and 4. Overall, the proposed method consistently and significantly outperforms all baselines across all datasets and evaluation metrics. In particular, on the Hyundai dataset, our model achieved an average performance improvement of 10.12% over the best baseline. This demonstrates the effectiveness of our model in the waypoint recommendation scenario and confirms its practical applicability in real-world navigation systems. On the public LBSN datasets (NYC, CA, TKY), WayPOI also achieved significant improvements over the best baseline (49.32%, 38.22%, and 10.8%, respectively).

We observed that the models specifically designed for next-POI recommendation (e.g., STRNN, STAN, GETNext) showed lower performance compared to the traditional recommendation models such as FPMC and PRME. As expected, the next-POI models do not naturally fit the waypoint recommendation problem because their design typically relies on a user's rich sequence of prior check-in records, so they would fail the scenario where only a starting point and destination are given for predicting waypoint POIs.

On the other hand, MTNet [5], despite being designed for the next-POI recommendation, achieved relatively better performance among the baselines. MTNet uses a tree-structured representation of user routes and captures user temporal patterns via multi-granularity time slots. These strategies appear well-suited to our waypoint recommendation scenario.

5.3 Ablation Study

We performed ablation experiments to assess the impact of each component of our WayPOI on recommendation performance using the Hyundai dataset. We evaluated the following four variants: (1) '*w/o graph*': removes all graph-based learning (i.e., both the full graph and hierarchical graph); the model learns node embeddings using only the embedding layer. (2) '*w/o WayPOI*': removes the hierarchical graph and contrastive learning based on it; the model learns node embeddings using only the full graph. (3) '*w/o Neutral*': removes our improved contrastive learning strategy that allows neutral pairs; it just employs a vanilla contrastive learning strategy. (4) '*w/o MHD*': removes the multi-head decoder for time-category information; it is trained using only the POI prediction loss.

The results reported in Table 5 show that each component of our WayPOI contributes to improving waypoint recommendation accuracy. Especially, our WayPOI without the hierarchical graph and contrastive learning (w/o WayPOI) shows 23.6% performance drop. These results validate that all components of our model are helpful, and in particular, the combination of full graph and hierarchical

Table 3: Performance comparison on Hyundai and TKY datasets. The best (second-best) results are in bold (underlined).

Method	Hyundai					TKY				
	Recall@1	Recall@5	Recall@20	MAP@20	NDCG@20	Recall@1	Recall@5	Recall@20	MAP@20	NDCG@20
Popularity (Global)	0.0041	0.0063	0.0201	0.0089	0.0130	0.0277	0.0940	0.1454	0.1565	0.2338
Popularity (Route)	0.0668	0.1243	0.1318	0.1101	0.1272	0.0451	0.0897	0.1086	0.1227	0.1649
Popularity (Distance)	0.0776	0.1913	0.3014	0.1610	0.2144	0.0420	0.0898	0.1488	0.1650	0.2267
FPMC (*WWW'10*)	0.0474	0.0732	0.0996	0.0860	0.0995	0.0403	0.1004	0.1749	0.1747	0.2592
PRME (*IJCAI'15*)	0.2008	0.3241	0.3252	0.3628	0.3888	0.0815	0.1204	0.1299	0.2918	0.3296
STRNN (*AAAI'16*)	0.0002	0.0011	0.0052	0.0112	0.0128	0.0007	0.0016	0.0035	0.0133	0.0184
LightGCN (*SIGIR'20*)	0.0005	0.0010	0.0029	0.0011	0.0018	0.0004	0.0010	0.0032	0.0033	0.0054
Flashback (*IJCAI'20*)	0.0009	0.0039	0.0115	0.0037	0.0066	0.0788	0.2129	0.3431	0.2698	0.4326
STAN (*WWW'21*)	0.0002	0.0008	0.0026	0.0008	0.0015	0.0001	0.0007	0.0024	0.0020	0.0037
GETNext (*SIGIR'22*)	0.0035	0.0101	0.0250	0.0101	0.0162	0.0275	0.0935	0.1444	0.1552	0.2313
MTNet (*AAAI'24*)	0.1990	0.4114	0.5255	0.3508	0.4339	0.0924	0.2429	0.3846	0.2913	0.4659
WayPOI (*Ours*)	**0.2338**	**0.4591**	**0.5635**	**0.3869**	**0.4720**	**0.1031**	**0.2803**	**0.4388**	**0.3083**	**0.4996**
Improvement (%)	**16.4%**	**11.6%**	**7.2%**	**6.6%**	**8.8%**	**11.6%**	**15.4%**	**14.1%**	**5.7%**	**7.2%**

Table 4: Performance comparison on NYC and CA datasets. The best (second-best) results are in bold (underlined).

Method	NYC					CA				
	Recall@1	Recall@5	Recall@20	MAP@20	NDCG@20	Recall@1	Recall@5	Recall@20	MAP@20	NDCG@20
Popularity (Global)	0.0071	0.0260	0.0388	0.0402	0.0585	0.0108	0.0265	0.0493	0.0524	0.0804
Popularity (Route)	0.0310	0.0618	0.0679	0.0710	0.0925	0.0138	0.0215	0.0257	0.0341	0.0431
Popularity (Distance)	0.0146	0.0429	0.0980	0.0601	0.1033	0.0214	0.0587	0.1185	0.0813	0.1295
FPMC (*WWW'10*)	0.0961	0.2014	0.2491	0.2730	0.3605	0.0265	0.0444	0.0593	0.0851	0.1027
PRME (*IJCAI'15*)	0.0932	0.1528	0.1619	0.2961	0.3410	0.0667	0.1012	0.1027	0.2250	0.2474
STRNN (*AAAI'16*)	0.0002	0.0011	0.0054	0.0059	0.0118	0.0002	0.0007	0.0031	0.0027	0.0063
LightGCN (*SIGIR'20*)	0.0004	0.0014	0.0039	0.0026	0.0049	0.0002	0.0009	0.0029	0.0023	0.0048
Flashback (*IJCAI'20*)	0.0703	0.1726	0.3050	0.2029	0.3264	0.0469	0.1232	0.2144	0.1556	0.2460
STAN (*WWW'21*)	0.0004	0.0009	0.0033	0.0026	0.0051	0.0002	0.0006	0.0022	0.0013	0.0028
GETNext (*SIGIR'22*)	0.0085	0.0306	0.0442	0.0432	0.0625	0.0121	0.0275	0.0519	0.0471	0.0717
MTNet (*AAAI'24*)	0.0290	0.0942	0.1858	0.1226	0.2166	0.0158	0.0551	0.1021	0.0772	0.1263
WayPOI (*Ours*)	**0.1373**	**0.3477**	**0.5271**	**0.3364**	**0.5215**	**0.0889**	**0.1989**	**0.3133**	**0.2399**	**0.3554**
Improvement (%)	**42.9%**	**72.6%**	**72.8%**	**13.6%**	**44.7%**	**33.3%**	**61.4%**	**46.1%**	**6.6%**	**43.7%**

Table 5: Ablation study results.

Method	Recall@1	Recall@5	Recall@20	MAP@20	NDCG@20
w/o Graph	0.1248	0.2758	0.4142	0.2392	0.3118
w/o WayPOI	0.1743	0.3379	0.4419	0.2961	0.3655
w/o Neutral	0.2249	0.4362	0.5393	0.3727	0.4532
w/o MHD	0.2333	0.4451	0.5518	0.3867	0.4681
ours	**0.2338**	**0.4591**	**0.5635**	**0.3869**	**0.4720**

Table 6: Performance according to hierarchical graph depth.

Method	Recall@1	Recall@5	Recall@20	MAP@20	NDCG@20
Depth-0	0.0598	0.1439	0.2349	0.1229	0.1685
Depth-1	0.2248	0.4473	0.5433	0.3755	0.4575
Depth-2	**0.2349**	0.4529	0.5521	0.3862	0.4684
Depth-3	0.2338	**0.4591**	**0.5635**	**0.3869**	**0.4720**

graph (to account for both individual and group-level information) is key to achieving high-quality waypoint recommendation results.

Table 6 reports our performance with varying hierarchical graph depth. As the number of hierarchy levels increases, finer-grained group information is incorporated into user and POI representations, and accordingly we observe continuous improvement in most metrics. This result demonstrates that the hierarchical graph structure, which leverages diverse group-level information, contributes to further enhancing waypoint recommendation performance. However, we observed that levels deeper than 3 incur a high computational cost due to the large number of unique values in POI features. We thus restrict the hierarchical depth to 3.

6 Conclusion and Future Work

In this work, we formally defined a novel *waypoint POI recommendation* problem for smart vehicle navigation systems, and proposed a novel framework called **WayPOI** to effectively perform this task. We constructed a hierarchical graph to capture both individual and group-level behavioral patterns of users and POIs in the representation learning process, and applied an improved contrastive learning strategy to alleviate the false negative pair issue. Through extensive experiments on real-world driving data collected from Hyundai and three public LBSN datasets, we demonstrated the superiority of WayPOI. Ablation studies further verified the effectiveness of each component of our method. For future work, we are integrating real-time traffic information (one of the most important features in navigation services) into our recommendation framework, in order to further improve performance and enhance user satisfaction [11].

Acknowledgement

This work was supported **Hyundai Motor Company**. This work was also supported by (1) the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT)(**RS-2024-00345398**) and (2) the Institute of Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT)(**RS-2020-II201373**, Artificial Intelligence Graduate School Program (Hanyang University)).

GenAI Usage Disclosure

No GenAI tools were used in any stage of the research, nor in the writing.

References

[1] Eunjoon Cho, Seth A Myers, and Jure Leskovec. 2011. Friendship and mobility: user movement in location-based social networks. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. 1082–1090.

[2] Shanshan Feng, Xutao Li, Yifeng Zeng, Gao Cong, and Yeow Meng Chee. 2015. Personalized ranking metric embedding for next new poi recommendation. In *IJCAI'15 Proceedings of the 24th International Conference on Artificial Intelligence*. ACM, 2069–2075.

[3] Raia Hadsell, Sumit Chopra, and Yann LeCun. 2006. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*, Vol. 2. IEEE, 1735–1742.

[4] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 639–648.

[5] Tianhao Huang, Xuan Pan, Xiangrui Cai, Ying Zhang, and Xiaojie Yuan. 2024. Learning time slot preferences via mobility tree for next poi recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 8535–8543.

[6] Md Ashraful Islam, Mir Mahathir Mohammad, Sarkar Snigdha Sarathi Das, and Mohammed Eunus Ali. 2022. A survey on deep learning based Point-of-Interest (POI) recommendations. *Neurocomputing* 472 (2022), 306–325.

[7] Yangqin Jiang, Chao Huang, and Lianghao Huang. 2023. Adaptive graph contrastive learning for recommendation. In *Proceedings of the 29th ACM SIGKDD conference on knowledge discovery and data mining*. 4252–4261.

[8] Dah-Jing Jwo, Amita Biswal, and Ilayat Ali Mir. 2023. Artificial neural networks for navigation systems: A review of recent research. *Applied Sciences* 13, 7 (2023), 4475.

[9] Hitoshi Kanoh. 2007. Dynamic route planning for car navigation systems using virus genetic algorithms. *International Journal of Knowledge-based and Intelligent Engineering Systems* 11, 1 (2007), 65–78.

[10] Seyed Mehran Kazemi, Rishab Goel, Sepehr Eghbali, Janahan Ramanan, Jaspreet Sahota, Sanjay Thakur, Stella Wu, Cathal Smyth, Pascal Poupart, and Marcus Brubaker. 2019. Time2vec: Learning a vector representation of time. *arXiv preprint arXiv:1907.05321* (2019).

[11] Namhyuk Kim, Dong-Kyu Chae, Jung Ah Shin, Sang-Wook Kim, Duen Horng Chau, and Sunghwan Park. 2022. Context-aware Traffic Flow Forecasting in New Roads. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 4133–4137.

[12] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[13] Thomas N Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations*.

[14] Jongsoo Lee and Dong-Kyu Chae. 2024. Multi-view mixed attention for contrastive learning on hypergraphs. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2543–2547.

[15] Jongsoo Lee, Byeongtae Park, and Dong-Kyu Chae. 2023. DuoGAT: Dual Time-oriented Graph Attention Networks for Accurate, Efficient and Explainable Anomaly Detection on Time-series. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. 1188–1197.

[16] Jongsoo Lee, Jung Ah Shin, Dong-Kyu Chae, and Sang-Chul Lee. 2022. Personalized Tour Recommendation via Analyzing User Tastes for Travel Distance, Diversity and Popularity. *Electronics* 11, 7 (2022), 1120.

[17] Defu Lian, Yongji Wu, Yong Ge, Xing Xie, and Enhong Chen. 2020. Geography-aware sequential location recommendation. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*. 2009–2019.

[18] Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. 2016. Predicting the next location: A recurrent model with spatial and temporal contexts. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 30.

[19] Zhejun Liu, Yanbin Gao, Yunlong Sun, and Ye Wang. 2022. Application of LSTM neural network in RISS/GNSS integrated vehicle navigation system. In *China Satellite Navigation Conference*. Springer, 355–365.

[20] Yingtao Luo, Qiang Liu, and Zhaocheng Liu. 2021. Stan: Spatio-temporal attention network for next location recommendation. In *Proceedings of the web conference 2021*. 2177–2185.

[21] Jaewan Moon, Yoonki Jeong, Dong-Kyu Chae, Jaeho Choi, Hyunjung Shim, and Jongwuk Lee. 2023. CoMix: Collaborative filtering with mixup for implicit datasets. *Information sciences* 628 (2023), 254–268.

[22] Vi Tran Ngoc Nha, Soufiene Djahel, and John Murphy. 2012. A comparative study of vehicles' routing algorithms for route planning in smart cities. In *2012 First international workshop on vehicular traffic management for smart cities (VTM)*. IEEE, 1–6.

[23] Xuan Rao, Lisi Chen, Yong Liu, Shuo Shang, Bin Yao, and Peng Han. 2022. Graph-flashback network for next location recommendation. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*. 1463–1471.

[24] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th international conference on World wide web*. 811–820.

[25] Tal Ridnik, Emanuel Ben-Baruch, Nadav Zamir, Asaf Noy, Itamar Friedman, Matan Protter, and Lihi Zelnik-Manor. 2021. Asymmetric loss for multi-label classification. In *Proceedings of the IEEE/CVF international conference on computer vision*. 82–91.

[26] Eunseon Seong, Harim Lee, and Dong-Kyu Chae. 2024. Self-Supervised Framework based on Subject-wise Clustering for Human Subject Time Series Data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 22341–22349.

[27] Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075* (2015).

[28] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[29] Zhaobo Wang, Yanmin Zhu, Chunyang Wang, Wenze Ma, Bo Li, and Jiadi Yu. 2023. Adaptive graph representation learning for next POI recommendation. In *Proceedings of the 46th international ACM SIGIR conference on research and development in information retrieval*. 393–402.

[30] Azmine Toushik Wasi, Taki Hasan Rafi, Raima Islam, and Dong-Kyu Chae. 2024. BanglaAutoKG: Automatic Bangla Knowledge Graph Construction with Semantic Neural Graph Filtering. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation*.

[31] Dongbo Xi, Fuzhen Zhuang, Yanchi Liu, Jingjing Gu, Hui Xiong, and Qing He. 2019. Modelling of bi-directional spatio-temporal dependence and users' dynamic preferences for missing poi check-in identification. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 5458–5465.

[32] Dingqi Yang, Benjamin Fankhauser, Paolo Rosso, and Philippe Cudre-Mauroux. 2020. Location prediction over sparse user mobility traces using rnns. In *Proceedings of the twenty-ninth international joint conference on artificial intelligence*. 2184–2190.

[33] Dingqi Yang, Daqing Zhang, Vincent W Zheng, and Zhiyong Yu. 2014. Modeling user activity preference by leveraging user spatial temporal characteristics in LBSNs. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 45, 1 (2014), 129–142.

[34] Song Yang, Jiamou Liu, and Kaiqi Zhao. 2022. GETNext: trajectory flow map enhanced transformer for next POI recommendation. In *Proceedings of the 45th International ACM SIGIR Conference on research and development in information retrieval*. 1144–1153.

[35] Pengpeng Zhao, Anjing Luo, Yanchi Liu, Jiajie Xu, Zhixu Li, Fuzhen Zhuang, Victor S Sheng, and Xiaofang Zhou. 2020. Where to go next: A spatio-temporal gated network for next poi recommendation. *IEEE Transactions on Knowledge and Data Engineering* 34, 5 (2020), 2512–2524.