Incremental Spatio-Temporal Graph Learning for Online Query-POI Matching

Zixuan Yuan[1], Hao Liu[2*], Junming Liu[3], Yanchi Liu[1], Yang Yang[4], Renjun Hu[5], Hui Xiong[1*]
[1]Rutgers University, [2]Business Intelligence Lab, Baidu Research, [3]City University of Hong Kong,
[4]Nanjing University of Science and Technology, [5]Beihang University.
{zy101,yanchi.liu,hxiong}@rutgers.edu,liuhao30@baidu.com,
junmiliu@cityu.edu.hk,yyang@njust.edu.cn,hurenjun@buaa.edu.cn

ABSTRACT

Query and Point-of-Interest (POI) matching, aiming at recommending the most relevant POIs from partial query keywords, has become one of the most essential functions in online navigation and ride-hailing applications. Existing methods for query-POI matching, such as Google Maps and Uber, have a natural focus on measuring the static semantic similarity between contextual information of queries and geographical information of POIs. However, it remains challenging for dynamic and personalized online query-POI matching because of the non-stationary and situational context-dependent query-POI relevance. Moreover, the large volume of online queries requires an adaptive and incremental model training strategy that is efficient and scalable in the online scenario. To this end, in this paper, we propose an *Incremental Spatio-Temporal Graph Learning* (IncreSTGL) framework for intelligent online query-POI matching. Specifically, we first model dynamic query-POI interactions as microscopic and macroscopic graphs. Then, we propose an *incremental graph representation learning* module to refine and update query-POI interaction graphs in an online incremental fashion, which includes: (i) a contextual graph attention operation quantifying query-POI correlation based on historical queries under dynamic situational context, (ii) a graph discrimination operation capturing the sequential query-POI relevance drift from a holistic view of personalized preference and social homophily, and (iii) a multi-level temporal attention operation summarizing the temporal variations of query-POI interaction graphs for subsequent query-POI matching. Finally, we introduce a lightweight semantic matching module for online query-POI similarity measurement. To demonstrate the effectiveness and efficiency of the proposed algorithm, we conduct extensive experiments on two real-world datasets collected from a leading online navigation and map service provider in China.

KEYWORDS

Query-POI Matching, Spatio-Temporal Analysis, Incremental Graph Learning, User Modeling

* Hao Liu and Hui Xiong are corresponding authors.

ACM Reference Format:
Zixuan Yuan[1], Hao Liu[2*], Junming Liu[3], Yanchi Liu[1], Yang Yang[4], Renjun Hu[5], Hui Xiong[1*]. 2021. Incremental Spatio-Temporal Graph Learning for Online Query-POI Matching. In *Proceedings of the Web Conference 2021 (WWW '21), April 19–23, 2021, Ljubljana, Slovenia.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3442381.3449810

1 INTRODUCTION

Recent years have witnessed a worldwide prevalence of online navigation (e.g., Google Maps, Baidu Maps) and ride-hailing applications (e.g., Uber and Lyft) [24, 33]. For a specific user, the query-POI matching is developed to retrieve the most relevant destination POI from a list of candidates based on the (partial) query keywords, as demonstrated in Figure 1 (a). As one of the most essential map service functions, query-POI matching is playing a crucial role in helping users explore new places, improving customer service experiences, and ultimately boosting the commercial benefits.

The query-POI matching problem has been partially addressed in previous literature by building semantic relevance between contextual information of queries and static geographical locations [33]. Such method successfully incorporated multi-field semantic features of a query-POI pair for semantic similarity measurement, but ignored the impacts of historical matching records for dynamic and personalized query-POI matching. To resolve this issue, the recent advances of graph learning in recommender systems model the general user-item interactions as a bipartite user-item graph to capture latent user-item relevance [18, 20]. However, all the above methods typically require periodic re-training based on all historical data to capture the gradually adapted user preference, and thus are difficult to be scaled in an online environment. For instance, STDGAT [31], the state-of-the-art query-POI matching method, requires 195.9 hours of training time on a NVIDIA GeForce RTX 2080 Ti GPU to incorporate users' evolving preferences for the real-world dataset for Beijing (refer to dataset details in Section 9.1). In fact, the large-scale online stream data have placed high demand for both matching effectiveness and learning efficiency, which motivates us to improve the learning efficiency in an incremental way without losing matching accuracy, as training on all historical data from scratch.

However, three major challenges arise to develop such an intelligent online query-POI matching system. (i) **Dynamic spatio-temporal correlation**. The query-POI relevance is dependent on the sophisticated situational spatio-temporal context. Based on real-world large-scale analysis of user queries, Figure 1 (b) summarizes

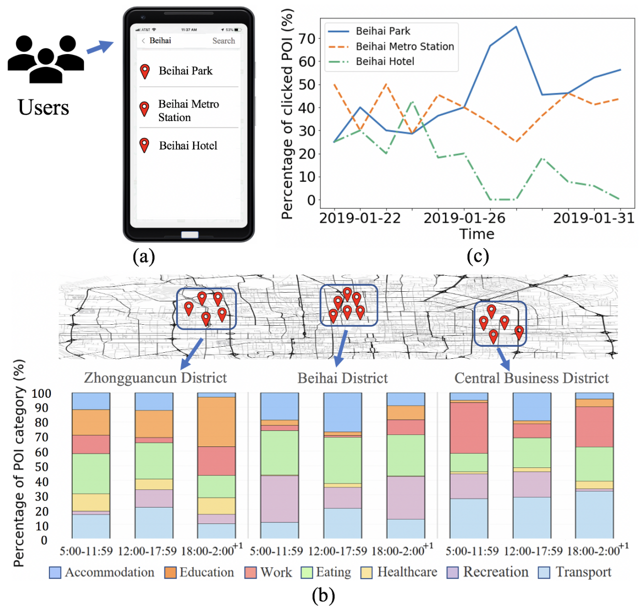


Figure 1: Illustrative example and challenges of online query-POI matching. (a) An online query-POI matching interface. (b) POI category distribution under different spatio-temporal contexts. (c) Non-stationary temporal variations in POI click distribution of query keyword "Beihai".

the category distribution of user-chosen POIs in three representative places in Beijing, China, under different time slots from Jan 1, 2019 to Jan 31, 2019. As can be seen, the chosen POI category distribution differs significantly under different spatio-temporal contexts, which renders it challenging to integrate such dynamic information during online query-POI matching. (ii) **Non-stationary relevance drift**. The relevances between queries and POIs are non-stationary and drifting over time (i.e., the relevance distribution is not i.i.d. in different time periods). For illustration, we extract historical records of the exemplar query keyword "Beihai" in two consecutive weeks, and report the distribution of user-chosen POIs in Figure 1 (c). As can be seen, the user preference on POI candidates of the corresponding query keyword is changing over time without significant recurrent pattern. How to model such non-stationary and drifting user preference is another challenge. Recently, the behavioral tendency for people to have ties with others, termed as social homophily, has been successfully applied for user modeling [23]. However, it remains unexplored how to integrate social homophily while preserving individual preference in online query-POI matching. (iii) **Online learning efficiency**. Although it is plausible to incorporate the most recent query history to overcome the relevance drift, it poses a challenge to capture the time-shifting user preference and the dynamic contextual information from the numerous streaming query logs in an online fashion. Most conventional methods regarding streaming data as a sequence of time windows [18] require periodic model reconstruction based on all historical data, which are not scalable for online query-POI interaction modeling. Therefore, a more efficient online query-POI relevance learning framework is desired for large-scale matching services with billions of query events per day.

To address the aforementioned challenges, in this paper, we propose an *incremental spatio-temporal graph learning* (IncreSTGL) framework for intelligent online query-POI matching. Our key insight is to continuously encode time-varying query-POI correlations into a *unitary graph structure* to facilitate online query-POI matching. Instead of periodically reconstructing the matching model based on all historical data, our approach directly refines the unitary graph structure to derive query and POI representations for query-POI matching, which is therefore more efficient. Specifically, we first model historical user query records into query-POI interaction graphs from both macroscopic and microscopic perspectives, where the macroscopic graph reflects social homophily of the general query-POI correlation, and the microscopic graph retains user-specific preferences. Then, to incrementally capture query-POI relevance shifts based on new coming user query records, we propose the *incremental graph representation learning* module, where (i) a contextual graph attention is devised to quantify the query-POI correlation under dynamic situational context in each time step, (ii) a graph discrimination operation is introduced to capture the temporally adapted query-POI relevance distributions among temporally-consecutive graph snapshots, and (iii) a multi-level temporal attention operation is proposed to integrate the preference variations among past query-POI graphs and output the unitary query-POI graph. The incrementally aggregated query-POI interaction graphs incorporate both past query-POI correlations and most recent user preferences, which are informative enough for query-POI relevance learning. Finally, we devise a lightweight semantic matching module for online POI recommendation solely based on the most recent query-POI interaction graphs.

Contributions. To the best of our knowledge, this is the first study of the query-POI matching problem from an online and incremental graph learning perspective. The constructed macroscopic and microscopic query-POI interaction graphs respectively reflect social homophily and personalized preference. Moreover, our approach is capable of continuously integrating past query-POI correlations and most recent user preferences into unitary query-POI interaction graphs, which significantly improves model update efficiency without losing matching accuracy. Extensive experimental results on two real-world map search query datasets demonstrate the efficiency and effectiveness of the proposed framework for online query-POI matching.

2 PRELIMINARIES AND PROBLEM FORMULATION

In this section, we first present some definitions and notations, then formally formulate the query-POI matching problem. The frequently used notations are listed in Table 1.

2.1 Preliminaries

Definition 1. ***Map query event*** *is defined as a 4-tuple:* $mqe = \{u, \tau, q, p\}$*, which records the clicked POI* p *when user* u *inputs a query* q *at timestamp* τ*. A POI* p *includes 3 elements: its location* l_p*, name* pw*, and category* pc*, while a query* q *is comprised of 2 elements: its*

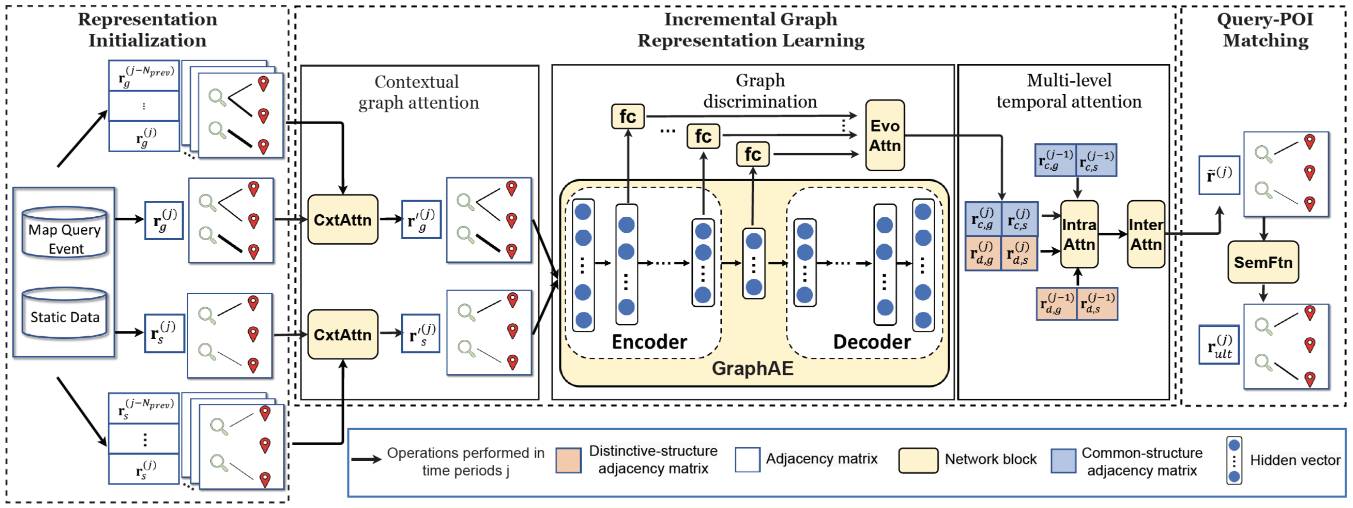
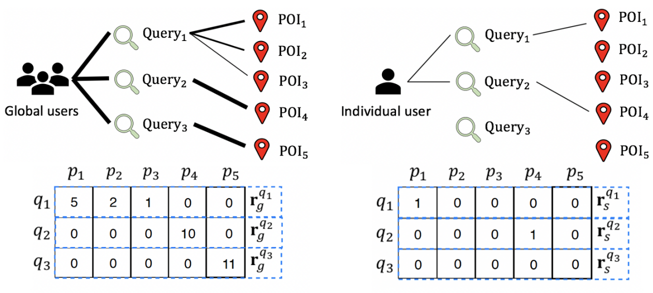

Representation Initialization
Incremental Graph Representation Learning
Query-POI Matching
Map Query Event
Static Data
Contextual graph attention
CxtAttn
CxtAttn
Graph discrimination
fc
fc
fc
Evo Attn
Encoder
Decoder
GraphAE
Multi-level temporal attention
Intra Attn
Inter Attn
SemFtn
Operations performed in time periods j
Distinctive-structure adjacency matrix
Adjacency matrix
Network block
Common-structure adjacency matrix
Hidden vector

Figure 2: Framework overview of IncreSTGL.



$$\begin{array}{c|ccccc|c} & p_1 & p_2 & p_3 & p_4 & p_5 & \\ \hline q_1 & 5 & 2 & 1 & 0 & 0 & \mathbf{r}_g^{q_1} \\ q_2 & 0 & 0 & 0 & 10 & 0 & \mathbf{r}_g^{q_2} \\ q_3 & 0 & 0 & 0 & 0 & 11 & \mathbf{r}_g^{q_3} \end{array} \qquad \begin{array}{c|ccccc|c} & p_1 & p_2 & p_3 & p_4 & p_5 & \\ \hline q_1 & 1 & 0 & 0 & 0 & 0 & \mathbf{r}_s^{q_1} \\ q_2 & 0 & 0 & 0 & 1 & 0 & \mathbf{r}_s^{q_2} \\ q_3 & 0 & 0 & 0 & 0 & 0 & \mathbf{r}_s^{q_3} \end{array}$$

(a) Macroscopic query-POI graph and adjacency matrix $\mathbf{r}_g$.

(b) Microscopic query-POI graph and adjacency matrix $\mathbf{r}_s$.

Figure 3: Illustrative example of macroscopic and microscopic graphs, and the graph adjacency matrices. Bold lines indicate greater edge weights.

location l_q and keyword qw. The location (e.g., l_q and l_p) is defined as geographical coordinates (longitude and latitude).

A map query event records a specific POI click action of a map query, which indicates a query-POI interaction. Based on historical interactions, we further construct two types of query-POI interaction graphs as follows.

Definition 2. ***Macroscopic query-POI interaction graph**, or the macroscopic graph, in the j-th time period is defined as $G_g^{(j)} = (V_q^{(j)} \cup V_p^{(j)}, E^{(j)})$, where $V_q^{(j)}, V_p^{(j)}$ are vertices of queries and POIs, and $E^{(j)}$ is the set of edges connecting pairs of vertices. The weight of $e_{qp}^{(j)} \in E^{(j)}$ is initialized by the frequency of map query events from all users that include both q and p. As depicted in Figure 3 (a), we discretize its weighted adjacency matrix $\mathbf{r}_g$ as a set of query vectors to quantify the 1-hop neighbour effects of POIs of each query, denoted by $\mathbf{r}_g = \{\mathbf{r}_g^{q_1}, \mathbf{r}_g^{q_2}, \ldots, \mathbf{r}_g^{q_L}\}$.*

Definition 3. ***Microscopic query-POI interaction graph**, or the microscopic graph, in the j-th time period is defined as $G_s^{u,(j)} = (V_q^{(j)} \cup V_p^{(j)}, E^{(j)})$, where the edge $e_{qp}^{(j)}$ is weighted by the frequency of map query events from the specific user u that include both query q and POI p. Similarly, as shown in Figure 3 (b), the set of query vectors is derived from its corresponding weighted adjacency matrix, i.e., $\mathbf{r}_s = \{\mathbf{r}_s^{q_1}, \mathbf{r}_s^{q_2}, \ldots, \mathbf{r}_s^{q_L}\}$.*

Table 1: Table of notations.

Notations	Description
$\mathcal{U} = \{u_1, u_2, \ldots, u_M\}$	The set of all users. $\lvert\mathcal{U}\rvert = M$.
$\mathcal{Q} = \{q_1, q_2, \ldots, q_L\}$	The set of all queries. $\lvert\mathcal{Q}\rvert = L$.
$\mathcal{P} = \{p_1, p_2, \ldots, p_N\}$	The set of all POIs. $\lvert\mathcal{P}\rvert = N$.
$q = (l_q, qw), q \in \mathcal{Q}$	A query is a 2-tuple: query location l_q and query word qw.
$p = (l_p, pw, pc), p \in \mathcal{P}$	A POI is a 3-tuple: POI location l_p, POI name pw, and POI category pc.
$l_q = (lng, lat)_q$	The location of query q is represented by longitude and latitude.
$l_p = (lng, lat)_p$	The location of POI p is represented by longitude and latitude.
$mqe = \{u, \tau, q, p\}$	A map search query event.
Δt	The time period interval.
$G_g^{(j)}$	The macroscopic query-POI interaction graph in the j-th time period.
$G_s^{u,(j)}$	The microscopic query-POI interaction graph in the j-th time period for user u.
$\mathbf{r}_g^{(j)}$	The macroscopic adjacency matrix in the j-th time period.
$\mathbf{r}_s^{(j)}$	The microscopic adjacency matrix in the j-th time period.
u, τ, geo	The symbols for user, time slot, and geographical location separately.
$\mathbf{x}$	The notation for each embedding vector.
$\mathbf{r}$	The notation for adjacency matrix.
d	The dimension of all representation vectors.

The macroscopic graph is designed to encode all users' query behaviors to reflect social homophily, whereas the microscopic graph is introduced to track individual user's query records to preserve personalized preference. Note that, in both graphs, we fix the dimension of each query vector to the number $\lvert V_p\rvert$ of possible

POI candidates. The time period is defined as the time range across two dates, e.g., 01/21/2019 – 01/25/2019.

2.2 Problem Formulation

In this paper, we formulate the problem of online query-POI matching as follows:

Input: A user u, timestamp τ, user location l, input query q, the t-th time period, the macroscopic graph $G_g^{(t)}$, and the microscopic graph $G_s^{u,(t)}$.

Output: The personalized ranking function $\mathcal{F}_{t+1}$ in the $t+1$-th time period that calculates the estimated probability of clicking a POI candidate p:

$$\hat{y} \leftarrow \mathcal{F}_{t+1}(p|u, \tau, l, q, G_g^{(t)}, G_s^{u,(t)}).$$

3 FRAMEWORK OVERVIEW

Figure 2 presents an overview of the proposed Incremental Spatio-Temporal Graph Learning (**IncreSTGL**) framework, which consists of three major parts: (i) the *Representation initialization* module, (ii) the *Incremental graph representation learning* module, and (iii) the *Query-POI matching* module. Specifically, the *Representation initialization* module first projects five types of input features (i.e., tokenized queries, tokenized POIs, users, timestamps, and geographical locations) to low-dimensional representation vectors. Then, in the *Incremental graph representation learning* module, for each past time period, a contextual graph attention (CxtAttn) operation incorporates multi-factor contextual information to update the graph structures for historical macroscopic and microscopic graphs. After that, a graph discrimination operation (GraphAE) generates a set of common-structure graphs and distinctive-structure graphs that separately represent the commonalities and differences of preference shifts in each time period. Later, a multi-level temporal attention (i.e., the combination of IntraAttn and InterAttn) periodically aggregates the above graphs along the timeline, and outputs the latest unitary query-POI graph. Finally, in the *Query-POI matching* module, a semantic function (SemFtn) is proposed to update the query-POI graph with query-POI semantic correlations, and calculate the query-POI relevance score for the matching task.

4 REPRESENTATION INITIALIZATION

We first develop the *Representation initialization* module to obtain low-dimensional feature vectors and adjacency matrices. During the j-th time period, where $j \in \{1, 2, \ldots, k\}$, we first extract two types of inputs: (i) map query events, which include query keywords, POI names, geographical locations, timestamps, and users; (ii) static data, which include POI names, POI locations, and POI categories [33]. Then, we initialize the macroscopic adjacency matrix $\mathbf{r}_g^{(j)}$ and the microscopic adjacency matrix $\mathbf{r}_s^{(j)}$ based on users' historical map query events as described in *Definitions 2 and 3*. After that, we project raw features into d-dimensional dense representation vectors through a convolutional neural network (CNN) and feed-forward neural networks (NNs), as detailedly illustrated below.

Queries and POIs are converted into tokenized words and characters. We initialize each word and character by random vectors, and embed them through CNN.

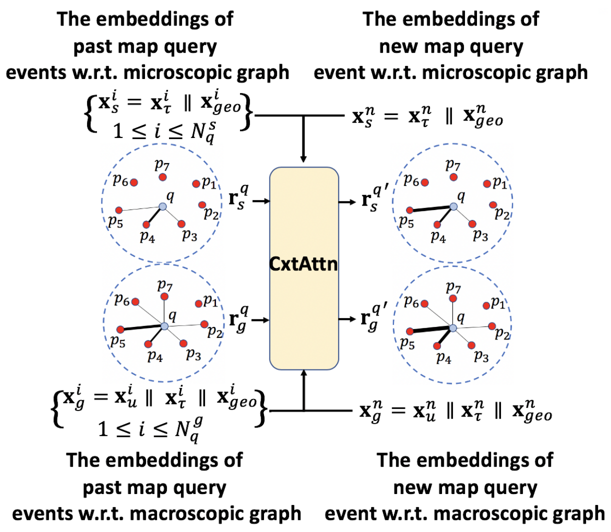

The embeddings of past map query events w.r.t. microscopic graph
The embeddings of new map query event w.r.t. microscopic graph
$\{\mathbf{x}_s^i = \mathbf{x}_\tau^i \parallel \mathbf{x}_{geo}^i, 1 \le i \le N_q^s\}$
$\mathbf{x}_s^n = \mathbf{x}_\tau^n \parallel \mathbf{x}_{geo}^n$
p_1 p_2 p_3 p_4 p_5 p_6 p_7 q
$\mathbf{r}_s^q$
$\mathbf{r}_s^{q'}$
CxtAttn
$\mathbf{r}_g^q$
$\mathbf{r}_g^{q'}$
$\{\mathbf{x}_g^i = \mathbf{x}_u^i \parallel \mathbf{x}_\tau^i \parallel \mathbf{x}_{geo}^i, 1 \le i \le N_q^g\}$
$\mathbf{x}_g^n = \mathbf{x}_u^n \parallel \mathbf{x}_\tau^n \parallel \mathbf{x}_{geo}^n$
The embeddings of past map query events w.r.t. macroscopic graph
The embeddings of new map query event w.r.t. macroscopic graph

Figure 4: The contextual graph attention.

Geographical locations are comprised of query location l_q and POI location l_p. We feed longitudes and latitudes into separate NNs, and finalize with the representation vectors of corresponding longitudes and latitudes.

Timestamps are mapped to the pre-defined time periods, and discretized into one-hot embeddings. Due to the temporal periodicity [30], we split the day into 24 non-overlapping time slots.

Users are represented by their previous query behaviors. Specifically, we first partition POIs into h pre-defined POI categories (e.g., resident, transport, education, etc.), and generate a $h \times 24$-dimensional vector for each user, where 24 is the number of time slots in each day, the $i \times j$-th dimension is the portion of the user's clicks on the i-th POI category in the j-th time slot. We further project the user vector to d-dimensional embedding through NNs. Note that user embeddings of the same user in different map query events are identical.

5 INCREMENTAL GRAPH REPRESENTATION LEARNING

Then, we present the *Incremental graph representation learning* module to capture drifting query-POI relevance incrementally, including the contextual graph attention, the graph discrimination, and the multi-level temporal attention operation.

5.1 Contextual Graph Attention

Existing studies have uncovered the repetitive patterns of human mobility [19]. For a specific user, his current query is highly correlated with his historical query records under various situational contexts. To capture such influence of historical query events on the new coming queries, one straightforward method is to construct unified representation vectors that entail situational information such as timestamps, geographical locations, and users. However, the dynamic dependencies among these exogenous factors significantly challenge the representation effectiveness of unified representations. We therefore introduce the contextual graph attention (CxtAttn) operation. Figure 4 illustrates the detailed architecture of CxtAttn within the specific j-th time period, $j \in \{1, 2, \dots, k\}$. Specifically, for each new query event mqe_n, we first extract map query events of N_{prev} previous days, and then update its corresponding query vectors in the macroscopic graph $G_g^{(j)}$ and microscopic graph $G_s^{u,(j)}$. Note that the N_{prev}-day query history is relatively small (less than one Gigabyte when $N_{prev} \le 30$), and separately stored in the corresponding graphs on a daily basis. For query q, N_q^g and N_q^s represent the total number of relevant query events in the macroscopic graph and microscopic graph, respectively. The embeddings of past map query events $\{\mathbf{x}_g^i\}_{1 \le i \le N_q^g}$, $\{\mathbf{x}_s^i\}_{1 \le i \le N_q^s}$ and new query event $\mathbf{x}_g^n$, $\mathbf{x}_s^n$ w.r.t. macroscopic and microscopic graphs are defined as:

$$\mathbf{x}_g^i = \mathbf{x}_u^i \parallel \mathbf{x}_\tau^i \parallel \mathbf{x}_{geo}^i; \quad \mathbf{x}_s^i = \mathbf{x}_\tau^i \parallel \mathbf{x}_{geo}^i, \tag{1}$$

$$\mathbf{x}_g^n = \mathbf{x}_u^n \parallel \mathbf{x}_\tau^n \parallel \mathbf{x}_{geo}^n; \quad \mathbf{x}_s^n = \mathbf{x}_\tau^n \parallel \mathbf{x}_{geo}^n, \tag{2}$$

where $\mathbf{x}_u^i$, $\mathbf{x}_\tau^i$, $\mathbf{x}_{geo}^i$ are the embeddings of users, timestamps, and geographical locations in the i-th query event. $\parallel$ denotes vector concatenation operation. By considering the edge weights in the macroscopic graph and the microscopic graph, the update operation of query vectors is:

$$\mathbf{r}_g^{q\prime} = \sum_{i=1}^{N_q^g} \frac{\exp(\alpha(\mathbf{x}_g^i, \mathbf{x}_g^n))}{\sum_{v_p \in N(v_q)} \exp(\alpha(\mathbf{x}_g^i, \mathbf{x}_g^n))} \mathbf{r}_g^q, \tag{3}$$

$$\mathbf{r}_s^{q\prime} = \sum_{i=1}^{N_q^s} \frac{\exp(\alpha(\mathbf{x}_s^i, \mathbf{x}_s^n))}{\sum_{v_p \in N(v_q)} \exp(\alpha(\mathbf{x}_s^i, \mathbf{x}_s^n))} \mathbf{r}_s^q, \tag{4}$$

where the updated query vectors $\mathbf{r}_g^{q\prime}$, $\mathbf{r}_s^{q\prime}$ are derived from an attention mechanism based on their original query vectors $\mathbf{r}_g^q$, $\mathbf{r}_s^q$ in corresponding macroscopic graph and microscopic graph, respectively. $v_p \in N(v_q)$ represents the 1-hop neighbours of node query q. $\alpha(\cdot)$ is the unnormalized attention weight, defined as:

$$\alpha(x, y) = (\mathbf{W}_x x + \mathbf{b}_x) \cdot (\mathbf{W}_y y + \mathbf{b}_y)^\top, \tag{5}$$

where $\mathbf{W}_{(\cdot)}$ and $\mathbf{b}_{(\cdot)}$ are learnable weight matrices and bias terms that measure the dynamic correlations among situational features. By applying the contextual attention operation, we can update the structures of all macroscopic graphs and microscopic graphs in each time step by re-evaluating the importance of raw query-POI retrieval records. For instance, for a new coming query, the matched POIs of historical queries that possess a similar situational context tend to be recommended by allocating a higher query-POI edge weight through CxtAttn.

5.2 Graph Discrimination

In a query-POI graph, the edge e_{qp} indicates the proximity between each query-POI pair (q, p), where $q \in Q$ and $p \in \mathcal{P}$. Based on the updated macroscopic and microscopic graphs in each time step, we propose the graph discrimination operation (GraphAE) to incorporate the social effects and personalized preference by capturing the non-stationary and drifting user preference. Specifically, GraphAE incrementally distills the similarity and difference of both general and user-specific preference shifts between two consecutive time periods and respectively embed them into the common-structure and distinctive-structure graph adjacency matrices. As demonstrated

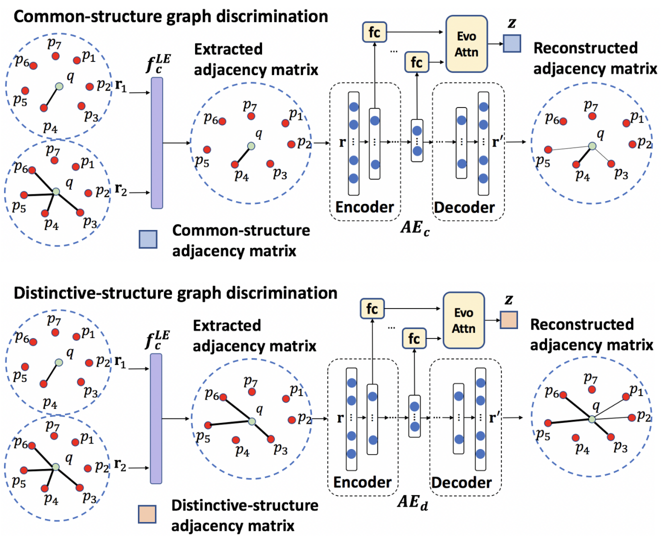

Common-structure graph discrimination
p_7
p_6
p_1
q
p_2
p_5
p_3
p_4
$\mathbf{r}_1$
$\mathbf{r}_2$
f_c^{LE}
Extracted adjacency matrix
Common-structure adjacency matrix
$\mathbf{r}$
$\mathbf{r}'$
fc
Evo Attn
z
Encoder
Decoder
AE_c
Reconstructed adjacency matrix
Distinctive-structure graph discrimination
Distinctive-structure adjacency matrix
AE_d

Figure 5: The auto-encoders for common-structure and distinctive-structure learning.

in Figure 5, the common-structure graph discrimination stores the shared query-POI edges, while the distinctive-structure graph discrimination keeps the differential query-POI edges.

Although the straightforward linear-dependent approach [1] is applicable for graph discrimination operation, it suffers from two weaknesses: (i) it aggravates the sparsity issue of extracted graphs, and (ii) extremely high-dimensional adjacency matrix produces tremendous unnecessary parameters, thus burdening the model's training efficiency. To this end, beyond the linear model, we employ two independent auto-encoders to learn latent representations of common-structure graph and distinctive-structure graph, respectively. Specifically, the auto-encoder is comprised of an encoding phase and a decoding phase. We describe the common-structure auto-encoder, and the distinctive-structure auto-encoder is in the same architecture. In the training stage, the encoding phase takes the common-structure linear extraction $f_c^{LE}(\mathbf{r}_1, \mathbf{r}_2)$ of two adjacency matrices $\mathbf{r}_1$, $\mathbf{r}_2$ as input, and projects them into latent space:

$$\begin{cases} \mathbf{r} & = f_c^{LE}(\mathbf{r}_1, \mathbf{r}_2), \\ \mathbf{h}_1 & = \sigma(\mathbf{W}_1 \mathbf{r} + \mathbf{b}_1), \\ \mathbf{h}_H & = \phi_H(\dots \phi_2(\mathbf{h}_1)), \\ \phi_j(\mathbf{h}_{j-1}) & = \sigma(\mathbf{W}_j \mathbf{h}_{j-1} + \mathbf{b}_j), \\ \mathbf{h}_{H+1} & = \sigma(\mathbf{W}_{H+1} \mathbf{h}_H + \mathbf{b}_H). \end{cases} \tag{6}$$

The decoding step then reconstructs the common-structure adjacency matrix $\hat{\mathbf{r}}$ based on latent vectors as output:

$$\begin{cases} \hat{\mathbf{h}}_1 & = \sigma(\hat{\mathbf{W}}_1 \mathbf{h}_{H+1} + \hat{\mathbf{b}}_1), \\ \hat{\mathbf{h}}_H & = \hat{\phi}_H(\dots \hat{\phi}_2(\hat{\mathbf{h}}_1)), \\ \hat{\phi}_j(\hat{\mathbf{h}}_{j-1}) & = \sigma(\hat{\mathbf{W}}_j \hat{\mathbf{h}}_{j-1} + \hat{\mathbf{b}}_j), \\ \hat{\mathbf{r}} & = \sigma(\hat{\mathbf{W}}_{H+1} \hat{\mathbf{h}}_H + \hat{\mathbf{b}}_H), \end{cases} \tag{7}$$

where $\mathbf{W}$s, $\mathbf{b}$s are the learnable weights and biases. $j \in \{2, \dots, H\}$. $\{\mathbf{h}_i\}_{1 \le i \le H+1}$, $\{\hat{\mathbf{h}}_i\}_{1 \le i \le H}$ are hidden vectors at the i-th hidden layer in the encoding phase and decoding phase correspondingly. $\sigma(\cdot)$ is the non-linear activation function.

The loss function between the common-structure adjacency matrix $\mathbf{r}$ generated by the linear model and the reconstructed adjacency matrix $\hat{\mathbf{r}}$ generated by the auto-encoder model is defined as:

$$\mathcal{L}_{AE} = \frac{1}{2}\left\|(\mathbf{r} - \hat{\mathbf{r}})\right\|_2^2. \tag{8}$$

The abstracted representation by each layer in the auto-encoder usually contains information in different granularity [26]. We further introduce $H + 1$ fully-connected layers to transform the latent feature representations $\{\mathbf{h}_i\}_{1 \le i \le H+1}$ in each layer and aggregate them via an attention operation (EvoAttn) to derive the final representation $\mathbf{z}$ of the common-structure adjacency matrix:

$$\begin{cases} \mathbf{z} & = Attn(fc_1(\mathbf{h}_1), \ldots, fc_{H+1}(\mathbf{h}_{H+1})) \\ fc_j(\mathbf{h}_j) & = \tilde{\mathbf{W}}_j \mathbf{h}_j + \tilde{\mathbf{b}}_j \end{cases}, \tag{9}$$

where $\tilde{\mathbf{W}}_j$ and $\tilde{\mathbf{b}}_j$s are the trainable weight matrices and bias terms, $Attn(\cdot)$ is the general form of attention used throughout the paper:

$$\begin{cases} \mathbf{v}_j & = \sigma(\mathbf{W}\mathbf{x}_j + \mathbf{b}) \\ \alpha_j & = \frac{\exp(\mathbf{v}_j \cdot \mathbf{v}_j^\top)}{\sum_{i \in \mathcal{I}} \exp(\mathbf{v}_i \cdot \mathbf{v}_i^\top)} \\ \mathbf{y} & = \sum_{j \in \mathcal{I}} \alpha_j \mathbf{v}_j \end{cases}, \tag{10}$$

in which the vector $\mathbf{y}$ is derived from the whole sequence $\{\mathbf{x}_j\}_{j \in \mathcal{I}}$, and $\mathcal{I}$ is the index set of input variables.

We use the GraphAE to generate a series of common-structure and distinctive-structure adjacency matrices for each time step, as depicted in Figure 2. Take common-structure graph discrimination (AE_c) for instance. In the first time period, it outputs the common-structure adjacency matrix $\mathbf{r}_c^{(1)} = AE_c(\mathbf{r}_g^{(1)}, \mathbf{r}_s^{(1)})$. In the second time period, it computes the macroscopic one $\mathbf{r}_{c,g}^{(2)} = AE_c(\mathbf{r}_g^{(2)}, \mathbf{r}_c^{(1)})$ and microscopic one $\mathbf{r}_{c,s}^{(2)} = AE_c(\mathbf{r}_s^{(2)}, \mathbf{r}_c^{(1)})$. In the remaining time periods, it generates the macroscopic one $\mathbf{r}_{c,g}^{(j)} = AE_c(\mathbf{r}_g^{(j)}, \mathbf{r}_{c,g}^{(j-1)})$ and microscopic one $\mathbf{r}_{c,s}^{(j)} = AE_c(\mathbf{r}_s^{(j)}, \mathbf{r}_{c,s}^{(j-1)})$, $j \ge 3$. The distinctive-structure graph generation process is defined in the same way.

Overall, GraphAE has three major advantages: (i) During the consecutive time periods, GraphAE preserves the shared information that indicates users' invariant map query patterns as well as the distinctive properties that convey user's preference drifts. (ii) Compared to the linear method, GraphAE can extract higher-order representations and alleviate the data sparsity issue by recovering the missing data from the latent space. (iii) For efficiency concern, GraphAE distills structural information of previous graphs and avoids subsequent repetitive, redundant computations. Note that the replacement of such auto-encoder architecture with other unsupervised graph approaches (e.g., ABCGraph-Adv [6]) is also compatible with our framework.

5.3 Multi-Level Temporal Attention

The users' preferences are changing gradually over time [27]. GraphAE preserves the intermediate preference shift by learning common-structure graphs and distinctive-structure graphs. Now we integrate the distilled preferences from the long-term view via a multi-level temporal attention network. As illustrated in Figure 2, the multi-level temporal attention is comprised of two components: (i) the intra temporal attention (IntraAttn), which aggregates user preference variation in both common-structure and distinctive-structure adjacency matrices within each time period, and (ii) the inter temporal attention (InterAttn), which integrates user preference variation patterns along the timeline and output the unitary graph structure.

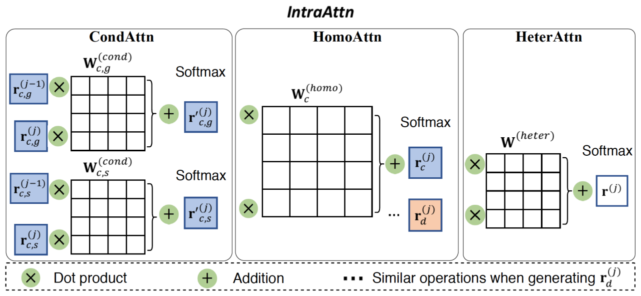


Figure 6: The workflow of intra temporal attention.

Intra temporal attention. We use IntraAttn to summarize the variations of user preference for each time period. As depicted in Figure 6, IntraAttn is comprised of three layers: (i) conditional layer (CondAttn) emphasizes the exact influence of same typed graphs in consecutive time steps, (ii) homogeneous layer (HomoAttn) combines all common-structure graphs and distinctive-structure graphs respectively in each time step, and (iii) heterogeneous layer (HeterAttn) aggregates the combined common-structure graphs and distinctive-structure graphs to form a unified graph feature representation. Since the process on common-structure graphs and distinctive-structure graphs are identical, we use common-structure graphs for the explanation of IntraAttn. Formally, in the initial time period, we skip the conditional layer and homogeneous layer due to the single number of common-structure and distinctive-structure adjacency matrices from the start, and jump to the HeterAttn for adjacency matrix fusion. In the j-th time period, $j \ge 2$, CondAttn incorporates the attention model to calculate the conditional effect of the current adjacency matrix over attentive embeddings of 1-step backward one and the current one. Such conditional effect captures the exact influence of the current adjacency matrix, which is derived from the previous one. Then, HomoAttn attentively aggregates a set of common-structure adjacency matrices and outputs the combined common-structure adjacency matrix $\mathbf{r}_c^{(j)}$. After that, in each time period, HeterAttn takes these combined common-structure and distinctive-structure adjacency matrices $\mathbf{r}_c^{(j)}$, $\mathbf{r}_d^{(j)}$ as input, and generates the unified adjacency matrix $\mathbf{r}^{(j)}$ in an attentive manner.

Inter temporal attention. Different from IntraAttn, InterAttn is introduced to capture the users' time-shifting preference. While there exist other temporal network alternatives (e.g., LSTM [7]), for the sake of computational efficiency, we reuse the attention mechanism defined in Equation (10) to incorporate the sequential variations of user preference. That is, based on the unified adjacency matrix $\mathbf{r}^{(j)}$ in past time periods, $j \ge 1$, we leverage the attention mechanism to generate the temporal graph structural embedding $\tilde{\mathbf{r}}^{(j)}$ along the timeline.

6 QUERY-POI MATCHING

Finally, we present the semantic function (SemFtn) to refine the aforementioned adjacency matrices with semantic information. SemFtn is designed to leverage the adjacency matrix of a bipartite query-POI graph as input, where the edge e_{qp} is weighted by the semantic similarity between query q and POI p via *Gestalt pattern matching algorithm* [17]. Note that other pre-training techniques (e.g., Bert [4]) can also be applied here for query-POI semantic similarity learning.

Formally, SemFtn derives the ultimate graph structural representation $\mathbf{r}_{ult}^{(j)}$ based on the semantic adjacency matrix $\mathbf{r}_{sem}$ and temporal adjacency matrix $\tilde{\mathbf{r}}^{(j)}$ as:

$$\mathbf{r}_{ult}^{(j)} = \mathbf{W}_{U_1}(\sigma(\mathbf{W}_{U_2}\tilde{\mathbf{r}}^{(j)} + \mathbf{b}_{U_2}) \parallel \mathbf{r}_{sem}) + \mathbf{b}_{U_1}, \tag{11}$$

where $\mathbf{W}_s$ and $\mathbf{b}_s$ are the trainable weight matrices and bias terms. The purpose of $\mathbf{W}_{U_2}$ and $\mathbf{b}_{U_2}$ is to project the low-dimensional adjacency matrix back to the adjacency matrix with original dimension $|V_p|$ (i.e., the dimension of query vector). The motivation of SemFtn is two-fold. First, it can integrate semantic information to facilitate query-POI matching. Second, it can alleviate the cold start problem even if there is no historical check-in for new users, POIs, or queries. In this case, the macroscopic graph will dominate the query-POI matching, since the microscopic graph is a set of nodes without edge connection (i.e., a zero matrix) and contributes less to query-POI relevance calculation.

Since the learned graph structural representation of corresponding queries and POIs preserves the query-POI correlation in the unitary graph structure, we adopt the following simple yet effective function to calculate the relevance between the query q and the POI candidate p:

$$\hat{y}(q,p) = \frac{\exp(\mathbf{r}_{ult}^{(j)}\{e_{qp}\})}{\sum_{i=1}^{|V_p|}\exp(\mathbf{r}_{ult}^{(j)}\{e_{qp_i}\})}, \tag{12}$$

where $\mathbf{r}_{ult}^{(j)}\{e_{qp}\}$ represents the edge weight of e_{qp} in the ultimate adjacency matrix $\mathbf{r}_{ult}^{(j)}$.

7 TRAINING AND OPTIMIZATION

Overall, IncreSTGL aims to minimize the error of estimated similarity score [33]:

$$\mathcal{L} = \hat{y}(q,p) = \frac{\exp(\mathbf{r}_{ult}^{(j)}\{e_{qp}\})}{\sum_{i=1}^{|V_p|}\exp(\mathbf{r}_{ult}^{(j)}\{e_{qp_i}\})}. \tag{13}$$

To further improve the discriminative power, we employ negative sampling [2] for training set augmentation, such that for each query-POI pair, we randomly select four unclicked, semantically similar POIs $\{p_j^-\}$ as negative samples, and move the embeddings of queries away from the ones of POIs. Then, given a query q, the probability of a POI p to be clicked is calculated by a softmax function of its similarity score:

$$\mathcal{L}_1 = -\log \prod_{q,p^+} Pr(p^+|q), \tag{14}$$

$$Pr(p|q) = \frac{\exp(\hat{y}(q,p))}{\sum_{p' \in \{p^+\}\cup\{p_j^-\}}\exp(\hat{y}(q,p'))}. \tag{15}$$

8 COMPLEXITY ANALYSIS

Computational efficiency is the first-class consideration of the online learning process. We argue the drifting user preference is mainly reflected by the time-varying query-POI interaction graphs, while the content feature representations and semantic matching function are relatively stable. Therefore, we directly adopt the initialized feature representations and SemFtn function obtained in early stages of training, which avoids $O(|V_q^{(j)}||V_p|^2)$ representation initialization cost and $O(|V_q^{(j)}||V_p|d)$ semantic function learning cost in online learning. Moreover, the incremental graph representation learning module in IncreSTGL provides an online incremental infrastructure that focuses on the incremental discrepancies of user's query preferences. Concretely, as illustrated at the bottom of Figure 2, we enable online incremental graph representation learning based on the pre-learned temporal query-POI adjacency matrices. In the current time period $t = j$, where $j \geq 2$, temporal adjacency matrix $\tilde{\mathbf{r}}^{(j-1)}$ that encodes the past user preference variations are directly loaded from memory, and fused with the newly-learned adjacency matrix $\tilde{\mathbf{r}}^{(t)}$ for subsequent online query-POI matching. The incremental graph representation learning module is computationally efficient from three aspects: (i) CxtAttn only considers the vectorized situational information, macroscopic and microscopic graphs in the most recent period, of which the computational complexity is $O(|V_q^{(t)}||E^{(t)}|d^2)$. (ii) GraphAE simply produces the latest common-structure and distinctive-structure adjacency matrices $[\mathbf{r}_{c,g}^{(t)}, \mathbf{r}_{c,s}^{(t)}, \mathbf{r}_{d,g}^{(t)}, \mathbf{r}_{d,s}^{(t)}]$ without re-training, of which the computational cost is $O(|V_q^{(t)}||V_p|d + H|V_q^{(t)}|d^2)$. (iii) Multi-level temporal attention further computes the newest temporal adjacency matrix $\tilde{\mathbf{r}}^{(t)}$, of which the learning cost is $O(|V_q^{(t)}|d^2 + |V_q^{(t)}||E^{(t)}|d)$. Note that $V_q^{(t)}$ and $E^{(t)}$ are the union of query vertices and edges in current time period, respectively.

9 EXPERIMENTS

In this section, we evaluate the proposed framework on two real-world datasets with respect to: (i) the query-POI matching accuracy of IncreSTGL, (ii) the effectiveness of each component in IncreSTGL, (iii) the influence of hyper-parameters, (iv) learning and matching efficiency, (v) matching robustness of IncreSTGL, (vi) the analysis of user satisfaction, and (vii) qualitative study.

9.1 Data Description

We use two real-world large-scale datasets, Beijing and Shanghai, to evaluate our model. All data are randomly sampled from 60 consecutive days in 2019. We split the dataset into three parts: 80% for training, 10% for validation, and 10% for testing.

9.2 Baselines

In order to verify the prediction accuracy, we compare the proposed predictor with two dynamic graph neural network methods, four deep learning based semantic models, and two spatio-temporal recommendation models:

- **DySAT** [18] utilizes graph structure features and temporal patterns to learn node representations in dynamic graphs.

Table 2: Overall performance.

Algorithm	Beijing							Shanghai						
	Hits@1	Hits@3	Hits@5	NDCG@3	NDCG@5	NDCG@10	p-value	Hits@1	Hits@3	Hits@5	NDCG@3	NDCG@5	NDCG@10	p-value
DySAT	0.5837	0.6270	0.6893	0.5984	0.6539	0.7148	5.30e-16	0.6025	0.6416	0.7125	0.6157	0.6583	0.7221	2.03e-15
CTDNE	0.5662	0.6292	0.6741	0.5843	0.6488	0.6902	3.38e-11	0.5930	0.6530	0.7304	0.6218	0.6892	0.7315	1.07e-14
DPSM	0.5497	0.6221	0.6819	0.5945	0.6424	0.6804	7.32e-17	0.5901	0.6178	0.6537	0.5984	0.6127	0.6689	4.54e-17
LSTM-DSSM	0.6045	0.6439	0.7303	0.6312	0.6996	0.7483	8.58e-12	0.5826	0.6197	0.7382	0.6292	0.7088	0.7576	7.49e-12
PALM	0.6139	0.6755	0.7390	0.6628	0.7053	0.7671	6.43e-9	0.6153	0.6579	0.7550	0.6344	0.7306	0.7694	8.37e-14
STDGAT	0.6382	0.7213	0.8027	0.7022	0.7730	0.8035	6.21e-5	0.6487	0.7356	0.8191	0.7254	0.7690	0.7992	3.59e-6
STGN	0.5772	0.6142	0.6793	0.5871	0.6320	0.6891	7.02e-10	0.5543	0.6038	0.6528	0.5731	0.6319	0.6932	3.92e-11
LSTPM	0.5538	0.6011	0.6591	0.5721	0.6067	0.6523	2.58e-15	0.6021	0.6204	0.6731	0.6154	0.6518	0.7013	8.16e-14
IncreSTGL	**0.6945**	**0.7531**	**0.8175**	**0.7240**	**0.7954**	**0.8237**	-	**0.7039**	**0.7734**	**0.8320**	**0.7428**	**0.7806**	**0.8114**	-

In the experiments, we input both macroscopic and microscopic graphs for dynamic graph embedding learning via additional graph embedding attention network during each time window.

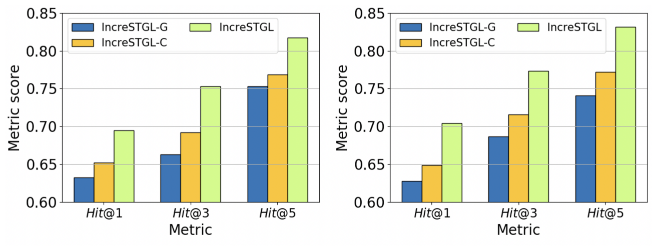
- **CTDNE** [15] leverages random walk based methods to learn the time-evolving network embeddings among dynamic graphs. In the experiments, we adopt the identical operations in DySAT.
- **DPSM** [34] proposes a deep semantic based model to effectively extract query and POI information features for the query-POI matching.
- **LSTM-DSSM** [16] is an extended version of the Deep Structured Semantic Model (DSSM) [8], which applies LSTM [7] to capture the temporal effects for semantic matching.
- **PALM** [33] proposes an attention-based neural network to incorporate semantic similarity and geographical correlation to quantify the query-POI relevance. For fair evaluation, we merge the timestamp vector with the geographical representations as a new spatio-temporal feature.
- **STDGAT** [31] proposes a spatio-temporal dual graph attention network to capture the evolving query-POI matching preference based on specific-user and global-user records.
- **STGN** [35] introduces multiple neural gates to capture the spatio-temporal relationship among consecutive query events for POI recommendation. For fair competition, we fuse the semantic representations of POI with original input vector during each time window.
- **LSTPM** [21] leverages non-local contextual network and geo-dilated LSTM to model long-term and short-term preferences, respectively. For fair comparison, we introduce the similar process in STGN.

9.3 Evaluation Metrics

We adopt *Hits@K* [25] and *NDCG@K* [9] for evaluation, where K=1, 3, 5 for *Hits* and K=3, 5, 10 for *NDCG*. For *Hits@K*, it computes what percentage of POIs among the top-K recommended POIs based on queries has been selected by a given user. For *NDCG@K*, it takes both relevance score and the orders of all potential POI destinations into account, and demonstrates the ranking quality of prediction list. The formal definition is shown as below:

$$Hits@K = \frac{P_{u,q} \cap R_{u,q}(K)}{K}, \tag{16}$$

$$NDCG@K = \frac{1}{IDCG} \sum_{i=1}^{C} \frac{2^{rel_i} - 1}{\log(1+i)}, \tag{17}$$



(a) Beijing Dataset (b) Shanghai Dataset

Figure 7: Ablation study.

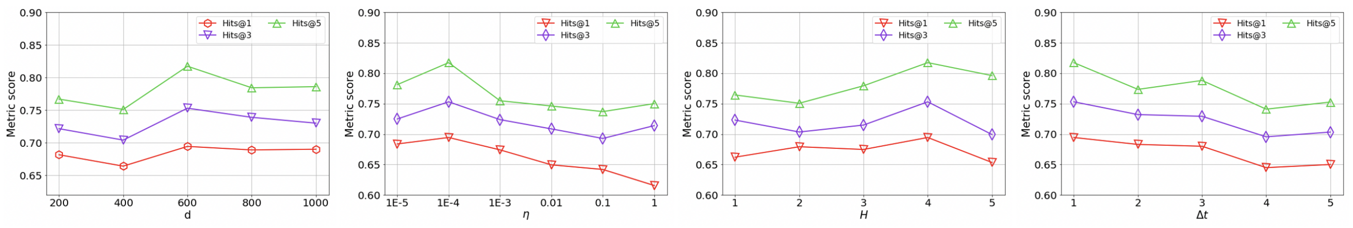
where $P_{u,q}$ is a set of selected POIs based on query q for a user u. $R_{u,q}(K)$ records the top-K recommended POIs based on query q for user u. *IDCG* stands for the maximum possible *DCG* for a given top-K POI recommendation list, and we set rel_i as 1 if the POI at position i is clicked and 0 otherwise. The parameter C is the number of correctly recommended POIs.

9.4 Implementation

Our model IncreSTGL is implemented by using the PaddlePaddle platform[1]. IncreSTGL is optimized to have its dimension d of all representation vectors set to be 600, and we use 1-layer CNN and 1-layer neural network in representation block. We set the learning rate η to 0.0001, the time period interval Δt to 1 (day), and the number of hidden layers H in auto-encoder to 4. The activation function $\sigma(\cdot)$ is LeakyReLU activation function with slope 0.2. The number of training epochs is set to 40. The number N_{prev} of previous days considered in CxtAttn is set to 21, since we found that for over 70% users, the average time span between recent frequent usage and the previous one is 3 weeks. The tokenized words and characters are obtained with jieba[2]. For fair comparison, we fine-tune the model parameters and set the number of training epochs to be 40 for all considered baselines.

Since most queries are uncorrelated with a large portion of POIs, we further downscale the dimension of query vector in both macroscopic and microscopic graphs. Specifically, for each query vector $\mathbf{r}_{\omega}^{q}$, $\omega \in \{g, s\}$ in the j-th time period, $j \geq 1$, we use the Gestalt pattern matching algorithm [17] to extract the top-S semantically similar POIs based on query q, reducing the dimension of query vector $|V_p|$ from N to S, where N is the number of POIs and $N \gg S$. Here, S is set to $5,000$ such that all possible POI candidates have been considered based on a given query.

[1]https://github.com/PaddlePaddle
[2]https://github.com/fxsjy/jieba



(a) Dimension of latent vector d (*Hits@top-K*)

(b) Learning rate η (*Hits@top-K*)

(c) The number H of layers in GraphAE (*Hits@top-K*)

(d) The time period interval Δt (*Hits@top-K*)

Figure 8: Results of hyper-parameters.

9.5 Overall Performance

The overall performance comparison is presented in Table 2. As can be seen, IncreSTGL achieves the best performance compared with all baselines on both datasets using all metrics, where all the p-values between our IncreSTGL and each baseline are much smaller than 0.05, demonstrating the statistical significance of improvements. Specifically, IncreSTGL outperforms the state-of-the-art baseline, STDGAT on both datasets, by (5.63%, 3.18%, 1.48%, 2.18%, 2.24%, 2.02%), and (5.52%, 3.78%, 1.29%, 1.74%, 1.16%, 1.22%) in terms of six metrics respectively. In addition, both dynamic graph based models perform relatively better than DPSM, STGN, and LSTPM, but slightly worse than PALM, LSTM-DSSM and STDGAT, which indicates that dynamic graphs are competitive to deep neural networks in learning the representations of queries and POIs, and unifying them together may achieve better representation vectors. Finally, our full approach IncreSTGL outperforms all baselines by (10.86%, 11.13%, 11.05%, 10.74%, 12.52%, 10.55%) in average on Beijing dataset, and (10.53%, 12.97%, 11.52%, 11.61%, 9.91%, 8.10%) in average on Shanghai dataset. Indeed, the introduction of dynamic situational context and user preference variations does exert positive influence on query-POI matching.

9.6 Ablation Study

In this subsection, we study the superiority of IncreSTGL by evaluating the impact of two factors, i.e., various situational context and dynamic user preference. In particular, we compare with two variants of IncreSTGL: (i) IncreSTGL-C is a basic variant without considering CnxAttn during the incremental graph representation learning. (ii) IncreSTGL-G is another basic variant that removes the graph discrimination step. Note that in each time period, macroscopic and microscopic graphs are attentively aggregated in the temporal layer.

Justification of various situational context. Figure 7 shows that IncreSTGL outperforms IncreSTGL-C by (4.24%, 6.08%, 4.86%) on Beijing dataset, and (5.50%, 5.79%, 6.01%) on Shanghai dataset, demonstrating positive effect of the situational context. Largely, the situational context that consists of temporal and geographical factors delineates the user's real-time status. However, existing semantic approaches neglect such information, and therefore generate relatively static recommendations, which is not practical in our scenarios. Hence, by weighing the past influences based on contextual information, contextual graph attention enables dynamic recommendations for new queries.

Justification of dynamic user preference. According to Figure 7, IncreSTGL dominates IncreSTGL-G by (6.23%, 9.03%, 6.44%) on Beijing dataset, and (7.65%, 8.67%, 9.12%) on Shanghai dataset. One possible reason is that our proposed *graph discrimination* helps to quantify the dynamicity of user preference, an essential role in predicting user behaviors, including online destination inquiry. Specifically, it can explore the time-shifting correlations between social homophily and user-specific preference in a latent vector space, where the low-dimensional vectors keep track of their temporal variations, and reduce the computational cost along the timeline.

9.7 Impact of Hyper-Parameters

We report the experimental results of hyper-parameters in IncreSTGL based on Beijing dataset, including the latent vector dimension d, the learning rate η, the number H of layers in GraphAE, and the time period interval Δt. The results on Shanghai are similar, and we omit them due to space limit. Due to the stability of IncreSTGL, once these hyper-parameters are optimized, there is no need for additional tweaks in future online incremental learning.

First, we vary the latent vector dimension d from 200 to 1000. Figure 8 (a) reports the performance improvements when we increase d from 200 to 600 and performance degrades when we further increase d from 600 to 1000. These results illustrate that 600-dimensional latent vector is powerful enough to capture the semantic information.

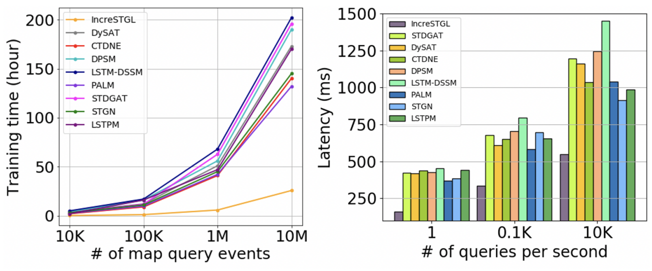
Second, we vary the learning rate η from 10^{-5} to 1. Figure 8 (b) shows that the performance rises when η increases from 10^{-5} to 10^{-4}, and gradually falls when η increases from 10^{-4} to 1, probably because large learning rate results in divergent weight updates, oscillating the model performance.

Third, we vary the number H of layers in GraphAE from 1 to 5. Figure 8 (c) demonstrates the optimal performance when $H = 4$, and the performance degrades when H is either too small or too large. One possible reason is that an inappropriate H value is unable to precisely extract useful feature representations for quantifying the behavioral patterns.

Fourth, we evaluate the impact of time period interval Δt. As shown in Figure 8 (d), we vary Δt from 1 to 5. Generally, the performance reaches optimal when we set $\Delta t = 1$. We observe remarkable performance degradation when we increase Δt.

9.8 Efficiency Test

We further compare the training efficiency and latency of IncreSTGL with all baselines. As reported in Figure 9, for the Beijing dataset



(a) Model efficiency comparison (b) Model latency comparison

Figure 9: Model efficiency and latency test.

of one million records with a hundred thousand different POIs, IncreSTGL took the shortest training time and matching latency, compared with all baselines. For the training time and matching latency, IncreSTGL is 11.6× and 2.2× faster than its closest competitor, DySAT, respectively on average. According to the comparison of model structures, DySAT trains structural attention and temporal attention layers, but IncreSTGL only preserves the graph structure in each time period, thereby saving running time by avoiding repeated graph representation learning. Moreover, the running time and matching latency of IncreSTGL are comparable to other deep learning based models (i.e., DPSM, LSTM-DSSM, PALM, and STDGAT) that contain sophisticated neural layers, indicating the effectiveness of the concise design in our approach. For the upward trend, compared to all the baselines, both training efficiency and matching latency of IncreSTGL witness milder increases with the number of query events.

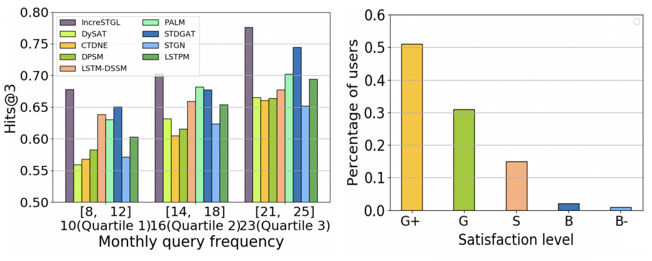
Justification of incremental learning efficiency. Online deployment requires efficient training speed for the arrival of new instances. Unlike traditional methods that repeatedly train the entire data, IncreSTGL supports incremental learning. Furthermore, since graph structural information can explicitly model node correlations, we use graph-based structure to directly measure query-POI relevance, rather than introducing additional graph neural networks, and thus significantly improve the computational efficiency.

9.9 Robustness Check

To test the robustness of our approach, we evaluate the performances of IncreSTGL and all baselines on different user groups clustered by different monthly query frequency. As reported in Figure 10 (a), it is notable that higher query frequency leads to better performances for all models. Compared with all baselines, IncreSTGL is more stable and has outstanding performance with sparse data. Furthermore, with the increase of user's monthly quests, IncreSTGL's performance can be improved from 0.6781 to 0.7764, indicating its potentials in modeling user's dynamic preference.

9.10 User Experience

In order to evaluate the user satisfaction of model recommendations, we invite ten domain experts to evaluate the matching results. To demonstrate the effectiveness of our framework, we compare IncreSTGL with the existing online query-POI matching service. We set five levels of satisfaction for performance comparisons: G+, G, S, B, B-, where G+ represents significantly better, S represents comparable, and B- represents worsened. Overall, we receive 814



(a) Robustness check (b) User experience

Figure 10: Results of online services.

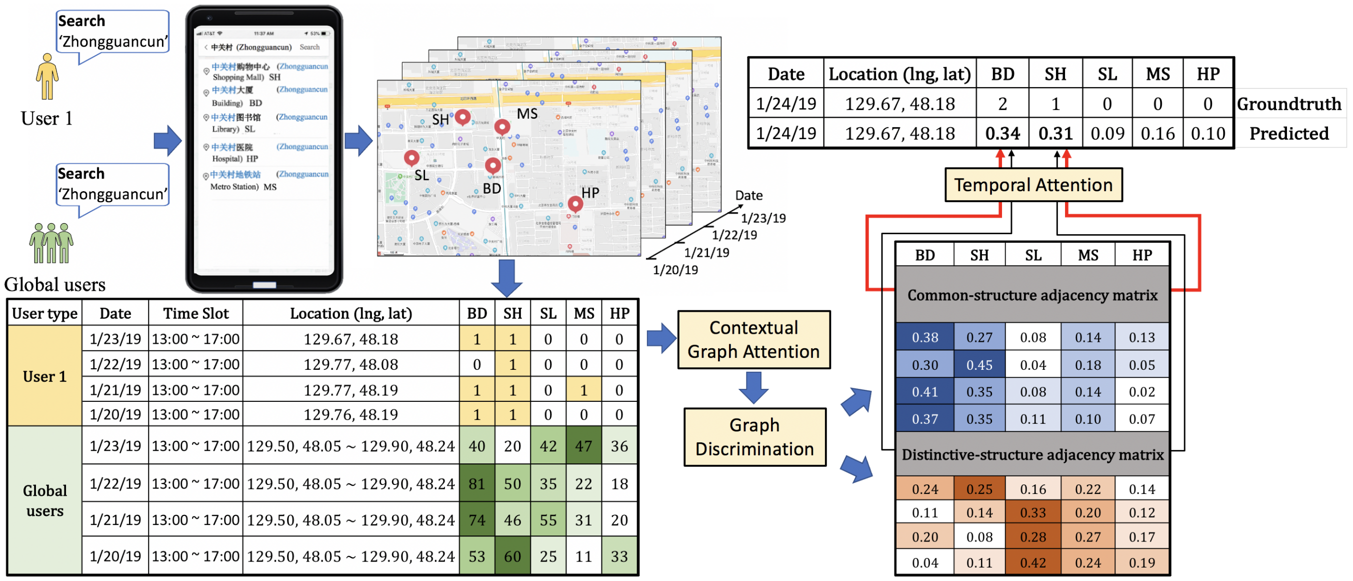
valid scored recommendations. As depicted in Figure 10 (b), more than 82.9% users consider the recommendation quality is above G, while only 3% users complain the worse quality of recommendation result (B or B-). These results demonstrate that our proposed model provides better recommendations in terms of user experience.

9.11 Case Study

Finally, we conduct qualitative analysis to analyze how our framework utilizes preference variations of the user communities and individual users. Specifically, we select exemplar query keyword "Zhongguancun", a landmark in Beijing, and its five possible POI candidates, i.e., "Zhongguancun Building" (BD), "Zhongguancun Shopping Mall" (SH), "Zhongguancun Library" (SL), "Zhongguancun Metro Station" (MS), "Zhongguancun Hospital" (HP). We extract users who have ever searched the keyword as the user community, and visualize their historical query events that happened within 13:00 — 17:00 from Jan 20, 2019 to Jan 23, 2019. As shown in Figure 11, IncreSTGL transforms the query-POI matching records of the target user and user community into adjacency matrices. After the process of contextual graph attention and graph discrimination, common-structure and distinctive-structure adjacency matrices are generated for each day accordingly. As can be seen, the target user used to click BD and SH, similar to the query preferences possessed by user community. Therefore, common-structure adjacency matrices out-win the distinctive-structure ones in capturing such taste similarity by allocating larger weights to BD and SH. After that, the multi-level temporal attention operation assigns larger importance (thick, red line) to the common-structure adjacency matrix compared to the distinctive-structure one (thin, black line). Finally, the predicted possibilities of POIs to be clicked in the near future exactly match the real-life events in this scenario, demonstrating the predictive power of IncreSTGL.

10 DISCUSSIONS AND LIMITATIONS

IncreSTGL is capable of incrementally capturing dynamic preference shifts between global users and individual user based on query semantics and spatio-temporal information. Specifically, given a query, the proposed incremental graph representation learning module and semantic function measure the sequential query-POI relevance variations from a joint perspective of personalized preference and social homophily. We argue that such incremental design will be a crucial ingredient in making high-capacity scalable models of existing recommender systems.



User type	Date	Time Slot	Location (lng, lat)	BD	SH	SL	MS	HP
User 1	1/23/19	13:00 ~ 17:00	129.67, 48.18	1	1	0	0	0
	1/22/19	13:00 ~ 17:00	129.77, 48.08	0	1	0	0	0
	1/21/19	13:00 ~ 17:00	129.77, 48.19	1	1	0	1	0
	1/20/19	13:00 ~ 17:00	129.76, 48.19	1	1	0	0	0
Global users	1/23/19	13:00 ~ 17:00	129.50, 48.05 ~ 129.90, 48.24	40	20	42	47	36
	1/22/19	13:00 ~ 17:00	129.50, 48.05 ~ 129.90, 48.24	81	50	35	22	18
	1/21/19	13:00 ~ 17:00	129.50, 48.05 ~ 129.90, 48.24	74	46	55	31	20
	1/20/19	13:00 ~ 17:00	129.50, 48.05 ~ 129.90, 48.24	53	60	25	11	33

Date	Location (lng, lat)	BD	SH	SL	MS	HP	
1/24/19	129.67, 48.18	2	1	0	0	0	Groundtruth
1/24/19	129.67, 48.18	**0.34**	**0.31**	0.09	0.16	0.10	Predicted

BD	SH	SL	MS	HP
Common-structure adjacency matrix				
0.38	0.27	0.08	0.14	0.13
0.30	0.45	0.04	0.18	0.05
0.41	0.35	0.08	0.14	0.02
0.37	0.35	0.11	0.10	0.07
Distinctive-structure adjacency matrix				
0.24	0.25	0.16	0.22	0.14
0.11	0.14	0.33	0.20	0.12
0.20	0.08	0.28	0.27	0.17
0.04	0.11	0.42	0.24	0.19

Figure 11: Case study.

The current incremental learning framework mainly employs user click actions as feedback, which remains other behaviors such as in-app zoom in/out and slide actions, queries from highway road network, real-world check-in histories unexplored. Besides, the model only considers typed Chinese characters in query keywords, which may exclude the information in the Chinese pinyin during input, despite its ability of handling query keywords in any position of the given POI name (e.g., keyword in the middle of the POI name). We left the above limitations as future research.

11 RELATED WORK

In this section, we discuss existing research related to our work, including dynamic graph construction, deep neural network in semantic matching, and POI recommendation.

Dynamic graph learning. Dynamic graph learning technique has gradually received attention from multiple domains, such as spectral clustering [11], and social recommendation [14]. In [11], a dynamic affinity graph construction approach provides a robust affinity matrix for feature partitioning, which is critical in computer vision tasks. Liu et al. [14] applied random walk with restart policy for a sequence of user-event bipartite graphs for personalized upcoming event recommendation. In a sense, one of its amazing advantages is to efficiently capture the pattern of newly-constructed graphs by yielding an incremental graph based on the difference between previous graph and new graph [12, 13, 32]. To our best knowledge, we are the first of learning graph structure for online query-POI matching task, and our new architecture, IncreSTGL, can efficiently capture dynamic situational context and recommend appropriate POIs for users.

Deep neural network in semantic matching. In recent years, deep neural networks have demonstrated its effectiveness in the semantic matching task [16, 31, 33, 34]. LSTM-DSSM [16], the extension of DSSM [8], considered the long-term contextual influence within the queries or documents, and used a deep neural network architecture to map a bag of letter-trigram words from search queries and documents to low-dimensional semantic embeddings. DPSM [34] fully exploited semantic information from queries and POIs and achieved multi-field matches for query-POI matching by modeling POI's name and detailed address, respectively. Most recently, in order to compensate for the limitations brought by one-sided semantic source, PALM [33] introduced external geographical information with semantic similarity for measuring query-POI relevance. However, the above approaches have two major limitations: (i) they only capture static representations and structures of queries and documents, and take the risk of losing important user information for relevance analysis, and (ii) they ignore the effects of time-evolving user preference onto the query-POI relevance. To handle the above problems, STDGAT [31] explored the evolving query-POI matching preference through the spatio-temporal dual graph attention network. However, unlike our approach, because of the expensive graph computation overhead, STDGAT is hard to be scaled in an online scenario.

POI recommendation. As an emerging LBS service, POI recommendation [22, 28, 36] is characterized by sophisticated contexts such as temporal information [29], geographical correlation[10], social effects[5], and successive check-ins between POIs [3]. To name a few, Yuan et al. [29] focused on incorporating temporal information in POI recommendations, Kurashima et al. [10] emphasized the importance of geographical relevance for user interest modeling. To improve the recommendation performance, Feng et al. [5] decoded the triadic closure in social networks, which may influence personalized preference. Cheng et al. [3] addressed the successive personalized POI recommendation task using Markov process and localized features. From a more general perspective, query-POI matching can be viewed as an extended problem of POI recommendation with query text as the additional hint, which only limited attention has been paid.

12 CONCLUSION

In this paper, we proposed an incremental spatio-temporal graph learning (IncreSTGL) framework for online query-POI matching. Different from the existing industry approach that considers textual semantic similarity and simple spatial relationship, we jointly modeled dynamic situational context and users' historical sequential behaviors in an online paradigm. Specifically, we model the sophisticated query-POI interactions from both macroscopic and microscopic perspectives. Then, we proposed the incremental graph representation learning module to quantify the drifting query-POI relevance variations from a combined view of personalized preference and social homophily. After that, we propose a lightweight semantic fusion layer for online query-POI matching. To tackle the computational intractability incurred by large query volumes in the online environment, we enable the graph representation learning module to continuously encode the latest user preferences, which avoids re-training the model from scratch. Finally, extensive experiments using two real-world map search query datasets demonstrated the efficiency and effectiveness of the proposed IncreSTGL framework for online query-POI matching.

This work is one step towards incremental learning for user's query preferences based on user click actions. Future research in this area may include the following: (i) modeling other behaviors as feedback, such as in-app zoom in/out and slide actions, and (ii) considering the highly incomplete query inputs for real-time query-POI matching.

ACKNOWLEDGEMENT

This research was partially supported by the National Science Foundation (NSF) via two grant numbers IIS-1814510 and IIS-2040799.

REFERENCES

[1] HA Almohamad and Salih O Duffuaa. 1993. A Linear Programming Approach for the Weighted Graph Matching Problem. *IEEE Trans. Pattern Anal. Mach. Intell.* 15, 5, 522–525.

[2] Buru Chang, Yonggyu Park, Donghyeon Park, Seongsoon Kim, and Jaewoo Kang. 2018. Content-Aware Hierarchical Point-of-Interest Embedding Model for Successive POI Recommendation. In *IJCAI.* ijcai.org, 3301–3307.

[3] Chen Cheng, Haiqin Yang, Michael R Lyu, and Irwin King. 2013. Where You Like to Go Next: Successive Point-of-Interest Recommendation. In *IJCAI.* IJCAI/AAAI, 2605–2611.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT (1).* Association for Computational Linguistics, 4171–4186.

[5] Shanshan Feng, Xutao Li, Yifeng Zeng, Gao Cong, and Yeow Meng Chee. 2015. Personalized Ranking Metric Embedding for Next New POI Recommendation. In *IJCAI.* AAAI Press, 2069–2075.

[6] Chaoyang He, Tian Xie, Yu Rong, Wenbing Huang, Yanfang Li, Junzhou Huang, Xiang Ren, and Cyrus Shahabi. 2019. Bipartite Graph Neural Networks for Efficient Node Representation Learning. *arXiv preprint arXiv:1906.11994* (2019).

[7] Sepp Hochreiter and Jürgen Schmidhuber. 1996. LSTM Can Solve Hard Long Time Lag Problems. In *NIPS.* MIT Press, 473–479.

[8] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning Deep Structured Semantic Models for Web Search using Clickthrough Data. In *CIKM.* ACM, 2333–2338.

[9] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated Gain-Based Evaluation of IR Techniques. *ACM Trans. Inf. Syst.* 20, 4, 422–446.

[10] Takeshi Kurashima, Tomoharu Iwata, Takahide Hoshide, Noriko Takaya, and Ko Fujimura. 2013. Geo Topic Model: Joint Modeling of User's Activity Area and Interests for Location Recommendation. In *WSDM.* ACM, 375–384.

[11] Zhihui Li, Feiping Nie, Xiaojun Chang, Yi Yang, Chengqi Zhang, and Nicu Sebe. 2018. Dynamic Affinity Graph Construction for Spectral Clustering using Multiple Features. *IEEE Trans. Neural Networks Learn. Syst.* 29, 12, 6323–6332.

[12] Hao Liu, Jindong Han, Yanjie Fu, Jingbo Zhou, Xinjiang Lu, and Hui Xiong. 2020. Multi-Modal Transportation Recommendation with Unified Route Representation Learning. *VLDB Endowment* 14, 3 (2020), 342–350.

[13] Hao Liu, Yongxin Tong, Jindong Han, Panpan Zhang, Xinjiang Lu, and Hui Xiong. 2020. Incorporating Multi-Source Urban Data for Personalized and Context-Aware Multi-Modal Transportation Recommendation. *IEEE TKDE* (2020).

[14] Shenghao Liu, Bang Wang, Minghua Xu, and Laurence T Yang. 2019. Evolving Graph Construction for Successive Recommendation in Event-Based Social Networks. *Future Gener. Comput. Syst.* 96, 502–514.

[15] Giang Hoang Nguyen, John Boaz Lee, Ryan A Rossi, Nesreen K Ahmed, Eunyee Koh, and Sungchul Kim. 2018. Continuous-Time Dynamic Network Embeddings. In *WWW (Companion Volume).* ACM, 969–976.

[16] Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and R Ward. 2014. Semantic Modelling with Long-Short-Term Memory for Information Retrieval. *arXiv preprint arXiv:1412.6629.*

[17] John W Ratcliff and David E Metzener. 1988. Pattern-Matching-the Gestalt Approach. *Dr Dobbs Journal* 13, 7, 46.

[18] Aravind Sankar, Yanhong Wu, Liang Gou, Wei Zhang, and Hao Yang. 2020. DySAT: Deep Neural Representation Learning on Dynamic Graphs via Self-Attention Networks. In *WSDM.* ACM, 519–527.

[19] Chaoming Song, Tal Koren, Pu Wang, and Albert-László Barabási. 2010. Modelling the scaling properties of human mobility. *Nature Physics* 6, 10 (2010), 818.

[20] Weiping Song, Zhiping Xiao, Yifan Wang, Laurent Charlin, Ming Zhang, and Jian Tang. 2019. Session-Based Social Recommendation via Dynamic Graph Attention Networks. In *WSDM.* ACM, 555–563.

[21] Ke Sun, Tieyun Qian, Tong Chen, Yile Liang, Quoc Viet Hung Nguyen, and Hongzhi Yin. 2020. Where to Go Next: Modeling Long-and Short-Term User Preferences for Point-of-Interest Recommendation. In *AAAI.* AAAI Press, 214–221.

[22] Qinyong Wang, Hongzhi Yin, Tong Chen, Zi Huang, Hao Wang, Yanchang Zhao, and Nguyen Quoc Viet Hung. 2020. Next Point-of-Interest Recommendation on Resource-Constrained Mobile Devices. In *WWW.* ACM / IW3C2, 906–916.

[23] Qitian Wu, Hengrui Zhang, Xiaofeng Gao, Peng He, Paul Weng, Han Gao, and Guihai Chen. 2019. Dual Graph Attention Networks for Deep Latent Representation of Multifaceted Social Effects in Recommender Systems. In *WWW.* ACM, 2091–2102.

[24] Xiangye Xiao, Qiong Luo, Zhisheng Li, Xing Xie, and Wei-Ying Ma. 2010. A Large-Scale Study on Map Search Logs. *ACM Trans. Web* 4, 3, 8:1–8:33.

[25] Xiwang Yang, Harald Steck, Yang Guo, and Yong Liu. 2012. On Top-k Recommendation using Social Networks. In *RecSys.* ACM, 67–74.

[26] Yang Yang, Da-Wei Zhou, De-Chuan Zhan, Hui Xiong, and Yuan Jiang. 2019. Adaptive Deep Models for Incremental Learning: Considering Capacity Scalability and Sustainability. In *KDD.* ACM, 74–82.

[27] Haochao Ying, Fuzhen Zhuang, Fuzheng Zhang, Yanchi Liu, Guandong Xu, Xing Xie, Hui Xiong, and Jian Wu. 2018. Sequential Recommender System based on Hierarchical Attention Network. In *IJCAI.* ijcai.org, 3926–3932.

[28] Fuqiang Yu, Lizhen Cui, Wei Guo, Xudong Lu, Qingzhong Li, and Hua Lu. 2020. A Category-Aware Deep Model for Successive POI Recommendation on Sparse Check-in Data. In *WWW.* ACM / IW3C2, 1264–1274.

[29] Quan Yuan, Gao Cong, Zongyang Ma, Aixin Sun, and Nadia Magnenat Thalmann. 2013. Time-Aware Point-of-Interest Recommendation. In *SIGIR.* ACM, 363–372.

[30] Quan Yuan, Gao Cong, and Aixin Sun. 2014. Graph-Based Point-of-Interest Recommendation with Geographical and Temporal Influences. In *CIKM.* ACM, 659–668.

[31] Zixuan Yuan, Hao Liu, Yanchi Liu, Denghui Zhang, Fei Yi, Nengjun Zhu, and Hui Xiong. 2020. Spatio-Temporal Dual Graph Attention Network for Query-POI Matching. In *SIGIR.* ACM, 629–638.

[32] Weijia Zhang, Hao Liu, Yanchi Liu, Jingbo Zhou, and Hui Xiong. 2020. Semi-Supervised Hierarchical Recurrent Graph Neural Network for City-Wide Parking Availability Prediction. In *AAAI*, Vol. 34. 1186–1193.

[33] Ji Zhao, Dan Peng, Chuhan Wu, Huan Chen, Meiyu Yu, Wanji Zheng, Li Ma, Hua Chai, Jieping Ye, and Xiaohu Qie. 2019. Incorporating Semantic Similarity with Geographic Correlation for Query-POI Relevance Learning. In *AAAI.* AAAI Press, 1270–1277.

[34] Ji Zhao, Meiyu Yu, Huan Chen, Boning Li, Lingyu Zhang, Qi Song, Li Ma, Hua Chai, and Jieping Ye. 2019. POI Semantic Model with a Deep Convolutional Structure. *arXiv preprint arXiv:1903.07279.*

[35] Pengpeng Zhao, Anjing Luo, Yanchi Liu, Fuzhen Zhuang, Jiajie Xu, Zhixu Li, Victor S Sheng, and Xiaofang Zhou. 2019. Where to Go Next: A Spatio-Temporal Gated Network for Next POI Recommendation. In *AAAI.* AAAI Press, 5877–5884.

[36] Fan Zhou, Ruiyang Yin, Kunpeng Zhang, Goce Trajcevski, Ting Zhong, and Jin Wu. 2019. Adversarial Point-of-Interest Recommendation. In *WWW.* ACM, 3462–34618.