Graph-Flashback Network for Next Location Recommendation

Xuan Rao*
University of Electronic Science and Technology of China
Chengdu, China
raoxuanzzz@gmail.com

Lisi Chen*
University of Electronic Science and Technology of China
Chengdu, China
lchen012@e.ntu.edu.sg

Yong Liu
Nanyang Technological University
Singapore
stephenliu@ntu.edu.sg

Shuo Shang†
University of Electronic Science and Technology of China
Chengdu, China
jedi.shang@gmail.com

Bin Yao
Shanghai Jiao Tong University
Shanghai, China
yaobin@cs.sjtu.edu.cn

Peng Han†
Aalborg University, Denmark
Aalborg, Denmark
pengh@cs.aau.dk

ABSTRACT

Next Point-of Interest (POI) recommendation plays an important role in location-based applications, which aims to recommend the next POIs to users that they are most likely to visit based on their historical trajectories. Existing methods usually use rich side information, or customized POI graphs to capture the sequential patterns among POIs. However, the graphs only focus on connectivity between POIs. Few studies propose to explicitly learn a weighted POI graph, which could reflect the transition patterns among POIs and show the importance of its different neighbors for each POI. In addition, these approaches simply utilize the user characteristics for personalized POI recommendation without sufficient consideration. To this end, we construct a novel User-POI Knowledge Graph with strong representation ability, called Spatial-Temporal Knowledge Graph (STKG). STKG is used to learn the representations of each node (i.e., user, POI) and each edge. Then, we design a similarity function to construct our POI transition graph based on the learned representations. To incorporate the learned graph into sequential model, we propose a novel network **Graph-Flashback** for recommendation. Graph-Flashback applies a simplified Graph Convolution Network (GCN) on the POI transition graph to enrich the representation of each POI. Further, we define a similarity function to consider both spatiotemporal information and user preference in modelling sequential regularity. Experimental results on two real-world datasets show that our proposed method achieves the state-of-the-art performance and significantly outperforms all existing solutions.

*Equal Contribution
†Corresponding Author

CCS CONCEPTS

• **Information systems → Location based services**; **Data mining**; • **Computing methodologies → Machine learning**.

KEYWORDS

Point-of-Interest; recommendation; knowledge graph

ACM Reference Format:
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 (KDD '22), August 14–18, 2022, Washington, DC, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3534678.3539383

1 INTRODUCTION

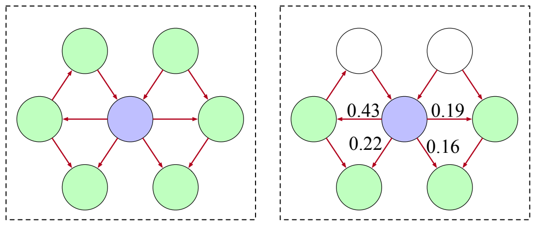
With the development of location based social media, people are becoming increasingly willing to record and share their life status along with geographical updates through mobile devices. As such, it is of great importance to make use of these geographical updates (e.g., check-in records) for understanding users' preference regarding their next movements. For the purpose, many studies focus on the problem of next POI recommendation [10, 26, 34]. As a fundamental functionality in smart city applications, next POI recommendation is generally defined as predicting the next POI a particular user will be most likely to visit based on his/her historical trajectory [2, 15, 19], aiming to help users plan their trips and explore more locations that may appeal to them based on their past preferences [4, 7, 22, 29, 32, 36].

Existing solutions to the next POI recommendation problem can be classified into two categories: sequence-based solutions and graph-based solutions. Sequence-based solutions include Markov chains, RNNs-based models, and attention-based models. Early studies use Markov chains to model sequential transitions patterns [2]. RNNs-based methods aims to combine the classic RNN architectures such as Long Short-Term Memory (LSTM) with temporal and spatial contexts for improving the model ability on capturing sequential regularity [15, 36]. Besides, attention-based models have been proposed for recommendation due to the great success of Transformer [24] in natural language processing (NLP) field [3, 5, 12, 20]. Recently, graph-based models have been proposed to enrich the POI representation by considering the property that similar users are likely to visit similar POIs [11, 13]. They propose different methods to construct POI graphs based on the spatial proximities among POIs or historical trajectories of different users.

Although existing approaches to next POI recommendation have gained much attention in their lines of research, they face the following limitations that significantly affect their effectiveness:

- **Graph construction**. Previous approaches construct customized homogeneous POI graphs, which only have one type of nodes and edges [8, 11, 13]. Moreover, the graphs only focus on the connectivity between POIs, neglecting the weight of edges. Few of researches propose to use heterogeneous graph that contains multiple types of nodes and edges to obtain a learning-based homogeneous graph.
- **POI representation**. Existing graph-based models apply Graph Neural Networks (GNNs) directly on the customized graphs to refine the representation of each POI through its neighbors based on neighbor sampling strategy without justification. As such, it is difficult to avoid some useless operations (e.g., feature transformation) in GNNs [9]. Besides, their learning process on the importance (i.e., weight coefficient) of its neighbors for each POI is non-transparent, which may degrade model effectiveness.
- **Personalized recommendation**. Most existing methods simply concatenate user feature vector (i.e., embedding) and model output to perform personalized POI recommendation. However, using such a simple operation fail take into account users' general preferences towards different POIs. Some other studies concatenate user and POI representations and regard them as the input of model to reflect the personalized user preferences. However, user preferences may be influenced by many factors (e.g., time, location, POI categories, etc.). It is hard to acquire accurate personalized user preferences simply by concatenating user and POI representations.

To this end, we propose the Graph-Flashback model to address the aforementioned three challenges. For addressing the first limitation, we first construct a Spatial-Temporal Knowledge Graph (STKG), which has strong representation ability. Based on our STKG, we use Knowledge Graph Embedding (KGE) algorithms to learn representations of each node and each edge. Next, we use the learned representations to construct POI transition graph, which is a weighted and learning-based graph. Our learned POI transition graph is derived from the STKG and the weight coefficients are constant in the next learning process. Figure 1 shows the difference between our learned graph and customized graphs. The learned POI graph explicitly shows the importance of its different neighbors (excluding itself) for each POI. However, previous graph-based methods apply GNN on constructed connectivity graph to learn the weight coefficients of its neighbors for each POI, which are time-varying and non-transparent [11, 13]. For addressing the second limitation, we apply a simplified Graph Convolution Network (GCN) [9] on the POI transition graph to enrich the representation of each POI, which could be then fed into RNN-based models to provide better recommendation service for users. Note that our POI transition graph is capable of reflecting the transition patterns among POIs. Meanwhile, it can be seamlessly integrated into other sequential models to enhance the ability of capturing sequential transition regularity. For addressing the third limitation, we design a similarity function to measure the preferences of different users based on current location and time to perform personalized POI recommendation.



(a) Customized POI graph (b) Learned POI graph

Figure 1: A simple example that illustrates the difference between customized POI graph and our learned POI transition graph. We use purple circle and green circle to indicate current POI and neighbor POI, respectively.

In summary, we make the following contributions to next-POI recommendation:

- We propose a novel Spatial-Temporal Knowledge Graph (STKG), which can be directly used to learn POI graph that reflects transition patterns between POIs. To our best knowledge, this is the first method that learns homogeneous weighted POI graph by using heterogeneous knowledge graph.
- We propose a novel RNN-based model, Graph-Flashback, to incorporate the learned POI transition graph into RNN-based models seamlessly for better capturing the sequential transition patterns. In addition, to further improve the effectiveness of personalized POI recommendation, we augment our model with a novel carefully-designed similarity function that measures preferences of different users.
- We conduct extensive experiments on two real-world datasets to evaluate the performance of Graph-Flashback. The results show that Graph-Flashback significantly outperforms the existing solutions in terms of accuracy.

The remaining parts of this paper are organized as follows. Section 2 investigates existing studies of next POI recommendation. Section 3 presents the problem formulation with relevant preliminaries. Section 4 presents our transition graph learning method. Next, Section 5 presents the overview of Graph-Flashback model. Section 6 reports the experimental results. Finally, Section 7 concludes the paper.

2 RELATED WORK

In this section, we will briefly review some works on next POI recommendation and knowledge graph embedding.

2.1 Next POI Recommendation

General POI recommendation mainly utilizes the collaborative filtering technique to mine intrinsic correlations among users, POIs, and context features, which takes into account geographical information and temporal effects [7, 16–19, 25]. Differently, next POI recommendation focus more on recommending POIs that are most likely to be visited next based on a user's most recent check-ins [23, 31]. It is challenging for those approaches used in general recommendation to capture the transition patterns between users' check-ins. Consequently, plenty of approaches have been proposed to solve the task, which can be classified into two categories: sequence-based methods and graph-based methods.

Sequence-based methods usually focus on individual user sequence, which models the transition patterns between POIs based on the sequence. On the early phase, Markov Chains (MC) models have been widely used for recommending next POI given a user's most recent check-ins [2, 4, 21]. FPMC [2] incorporates the Markov chain into matrix factorization model to learn a personalized transition matrix. PRME [4] proposes to utilize two different embedding spaces to capture the user preferences on POIs and sequential transitions between POIs, respectively. Recently, Recurrent Neural Networks (RNNs) and Attention mechanism have been extensively employed in next POI recommendation task due to the superior performance in modeling sequential information [12, 35]. DeepMove [3] combines an attention layer with gated recurrent units (GRU) to learn long-term periodicity and short-term sequential patterns, respectively. STGN [36] extends the LSTM by introducing spatial and temporal gates for learning the users' long-term and short-term preferences. CatDM [33] proposes to use POI categories and spatial distance to reduce candidate POIs. STAN [20] designs two attention layers to explicitly extract relative spatiotemporal information between consecutive and non-consecutive POIs. Graph-based methods mainly models sequential regularity from a global view, which refines the representation of POI within the sequence through those POIs outside the sequence. STP-UDGAT [13] uses GAT to learn correlations between POIs from both local and global views based on their customized graphs. ARNN [5] directly use meta-path based method to find related neighbor POIs in their knowledge graph. SGRec [11] combines graph-augmented POI sequence with transition patterns between POI categories to enrich POI representations and enhance model performance. However, none of the above methods consider to explicitly learn a weighted POI transition graph from a heterogeneous graph, which reflects transition pattern to some extent.

2.2 Knowledge Graph Embedding

A Knowledge Graph (KG) is a heterogeneous network, which contains multiple types of entities (nodes) and relations (edges) in the graph [6]. In this light, multiple attributes of an entity could be obtained based on different edges in the graph. In addition, high-order correlations between entities could also be found through these different relations. In a nutshell, a KG has strong representation ability. Knowledge Graph Embedding (KGE) is to embed a KG into a low dimensional space to learn the representations of each entity and each relation. The learned embedding could still preserve the inherent property of the graph. KGE algorithms can be divided into two categories: translation distance models, such as TransE [1], TransH [27], TransR [14], etc., and semantic matching models, such as DistMult [28]. In this work, we mainly focus on translation distance models, which regard relations as translation operation and calculate the distances (scores) between different entities based on different relations.

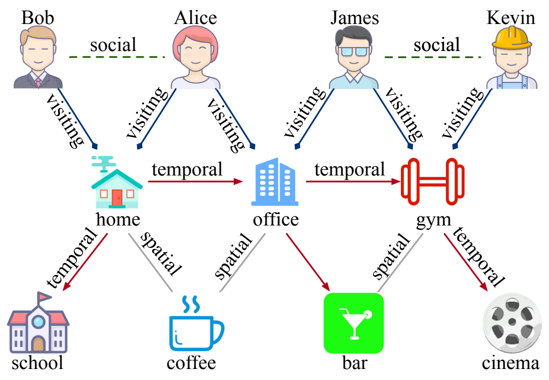


Figure 2: A simple illustration of our Spatial-Temporal Knowledge Graph.

3 PROBLEM STATEMENT

This section presents our formulation of next POI recommendation problem and relevant preliminary concepts. Here, we let $\mathcal{U} = \{u_1, u_2, \ldots u_{|\mathcal{U}|}\}$ be a set of users and $\mathcal{L} = \{l_1, l_2, \ldots, l_{|\mathcal{L}|}\}$ be a set of locations (POIs). Here, each location l_i is associated with a geographical coordinates (lat, lon) denoting its latitude and longitude, respectively.

Definition 1: (Check-in) A check-in is represented by $c = (u, l, t)$, which denotes that user u visits location l at time t. □

Definition 2: (User trajectory) A user trajectory is defined by a sequence of temporally ordered check-in records for a particular user. We use $T_{u_i} = \{c_1, c_2, \ldots, c_m\}$ to represent the historical trajectory of user u_i, where c_m denotes the most recent check-in record of u_i. □

Definition 3: (Knowledge graph) A knowledge graph, denoted by $\mathcal{G} = (V, E, \mathcal{A}, \mathcal{B}, \phi, \psi, \mathcal{R})$, is a directed graph with entity type function $\phi : V \longrightarrow \mathcal{A}$ and relation type function $\psi : E \longrightarrow \mathcal{B}$. Each entity $v \in V$ belongs to an entity type $\phi(v) \in \mathcal{A}$ and each edge $e \in E$ belongs to a relation type $\psi(e) \in \mathcal{B}$. $\mathcal{R}$ denotes a set of subject-property-object triples where each triple $r = (h, p, l)$ denotes a subject-property-object relation. Specifically, (h, p, l) denotes a relationship of p from head entity h to tail entity l. □

For example, (`Bob`, `visit`, `aquarium`) indicates that Bob visits the aquarium. The problem of next POI recommendation is defined as follow.

Definition 4: (Next POI recommendation) Given knowledge graph $\mathcal{G}$, user u_i, and the trajectory T_{u_i} of user u_i. The next POI recommendation problem aims to output the top-k POIs that user u_i will be most likely to visit next. □

4 TRANSITION GRAPH LEARNING

In this section, we present our approach to explicitly learning the transition patterns between POIs. The approach works with a knowledge graph to generate the representations for each entity and each relation in the graph. By properly defining a neighborhood based on the learned representations, we could obtain a graph representing the transition patterns between POIs, which can then be used for next POI recommendation task.

4.1 Spatial-Temporal Knowledge Graph

Unlike traditional knowledge graph, we design a novel knowledge graph called Spatial-Temporal Knowledge Graph (STKG) as shown in Figure 2. STKG integrates the tradition User-POI interaction graph with spatiotemporal correlations among POIs and friend relationships between users. Formally, STKG is $\mathcal{G} = (V, E, \mathcal{A}, \mathcal{B}, \phi, \psi, \mathcal{R})$, where V is the union of user set and location set ($\mathcal{U} \bigcup \mathcal{L}$), and E denotes the edges that belong to one of the following four types of relation: *visiting*, *temporal*, *spatial*, and *social*. Note that the *visiting* and *temporal* relations are directed. Here, our objective is to make full use of the check-ins recorded by all users to learn the transition patterns between POIs. We proceed to present how to construct the aforementioned four relations respectively.

Construction of *visiting* relation. We first construct the *visiting* (i.e., r_v) relation, which is represented by triplet (u, r_v, l), denoting that user u visits location l.

Construction of *temporal* relation. Next, we construct *temporal* (i.e., r_t) relation, which is represented by triplet (l_1, r_t, l_2), denoting that l_1 is visited just before l_2 based on a user's historical trajectory.

Construction of *spatial* relation. Given a POI l, we construct its *spatial* relations in two methods, threshold-based method and rank-based method. For threshold-based method, we set a pre-defined distance threshold Δ (e.g., 0.5km) and construct spatial relations between l and all of the other POIs whose distances to l are smaller than Δ. As for rank-based method, we construct spatial relations between l and l's top-k (e.g., 50) nearest neighbors. Note that given POIs l and l', l and l' form a *spatial* relation if l' is a spatial neighbor of l or l is a spatial neighbor of l'.

Construction of *social* relation. Finally, we construct *social* (i.e., r_f) relation, which is represented by triplet (u_1, r_f, u_2), denoting that user u_1 and u_2 are friends. The rationale of taking friend relation into consideration is that a POI visited by u_1 is likely to attract u_2 if u_1 and u_2 are friends.

4.2 Objective function

Given STKG $\mathcal{G}$, our next goal is to find an embedding function $f : V(E) \rightarrow \mathbb{R}^m$ that maps each entity $v \in \mathcal{G}.V$ or each relation $e \in \mathcal{G}.E$ into an m-dimensional feature vector (i.e., embedding). The embedding function $\boldsymbol{f}(\cdot)$ is expected to preserve the inherent property of $\mathcal{G}$. For the purpose, we use Knowledge Graph Embedding (KGE) algorithms based on translation distance models: TransE [1], TransH [27], and TransR [14] to learn the representation of each entity and each relation in the knowledge graph. For ease of understanding, we use boldface to indicate the learned embedding of an entity or relation. For example, given an entity h, $\boldsymbol{h} = \boldsymbol{f}(h)$ represents the corresponding embedding. To illustrate our embedding scheme, we take TransH [27] as an example. The basic idea of TransH is to use different hyperplanes to represent different relation spaces and regard relations as translation operation on hyperplanes. Specifically, for a triplet (h, r, t), the corresponding entity embeddings $\boldsymbol{h}$ and $\boldsymbol{t}$ are first projected to the hyperplane $\boldsymbol{w_r}$ with constraint $||\boldsymbol{w_r}||_2 = 1$, which is detailed by Equation 1.

$$\begin{aligned} \boldsymbol{h_\perp} &= \boldsymbol{h} - \boldsymbol{w_r^T h w_r}, \\ \boldsymbol{t_\perp} &= \boldsymbol{t} - \boldsymbol{w_r^T t w_r}, \end{aligned} \tag{1}$$

where $\boldsymbol{h_\perp}$ and $\boldsymbol{t_\perp}$ denote the projection embeddings of $\boldsymbol{h}$ and $\boldsymbol{t}$, respectively. Next, we use a scoring function $\boldsymbol{g}(\cdot)$ to measure the plausibility that the triplet is incorrect, which is presented by Equation 2.

$$\boldsymbol{g_r}(\boldsymbol{h}, \boldsymbol{t}) = ||\boldsymbol{h_\perp} + \boldsymbol{d_r} - \boldsymbol{t_\perp}||_2^2, \tag{2}$$

where $\boldsymbol{d_r}$ denotes the translation embedding on the hyperplane. Here, a lower value of $\boldsymbol{g_r}(\boldsymbol{h}, \boldsymbol{t})$ indicates that the triplet is more likely to be correct, while a higher value denotes that the triplet is less likely to be correct.

So far, the representations of each entity and each relation have been learned. The next goal is to design a function $\boldsymbol{s}(\cdot)$ for capturing the transition patterns among POIs. Inspired by the KGE algorithms, we compute the similarity between POI l_1 and l_2 by defining a temporal similarity function $\boldsymbol{s}(\cdot)$, which is expected to reflect the *temporal* relation between l_1 and l_2. We take TransE [1] as an example, the similarity function is presented by Equation 3:

$$\begin{aligned} \boldsymbol{s}(\boldsymbol{l_1}, \boldsymbol{l_2}) &= e^{-\boldsymbol{d}(\boldsymbol{l_1}, \boldsymbol{l_2})}, \\ \boldsymbol{d}(\boldsymbol{l_1}, \boldsymbol{l_2}) &= ||\boldsymbol{l_1} + \boldsymbol{r_t} - \boldsymbol{l_2}||, \end{aligned} \tag{3}$$

where e denotes the exponential function, $\boldsymbol{d}(\cdot)$ is the distance function, $\boldsymbol{l_1}$ and $\boldsymbol{l_2}$ denote the learned embeddings of POI l_1 and l_2, respectively, $\boldsymbol{r_t}$ is the temporal relation embedding, and $||\cdot||$ represents the L_1 or L_2 norm. Next, we construct the POI transition matrix $\boldsymbol{M} \in \mathbb{R}^{|\mathcal{L}| \times |\mathcal{L}|}$ based on Equation 3. However, maintaining $\boldsymbol{M}$ can be memory consuming when the number of POIs (i.e., $|\mathcal{L}|$) is large, which is common in recommendation scenarios. To reduce the space cost, we construct a sparse transition graph, which is denoted by $\boldsymbol{G}$. Specifically, we calculate k-nearest neighbor set $\mathcal{N}_k(l)$ for each POI l. Then, we have:

$$\boldsymbol{G}(i, j) = \begin{cases} \boldsymbol{M}(i, j), & \text{if POI } l_j \in \mathcal{N}_k(l_i), \\ 0, & \text{otherwise}, \end{cases} \tag{4}$$

where $i, j = 1, 2, \cdots, |\mathcal{L}|$. Finally, in order to normalize the sparse graph $\boldsymbol{G}$ between 0 and 1, we pick the maximum value in each row of $\boldsymbol{G}$ to construct a diagonal matrix $\boldsymbol{D}$. We obtain the learned POI transition graph $\boldsymbol{A}$ by Equation 5.

$$\boldsymbol{A} = \boldsymbol{D}^{-1}\boldsymbol{G}. \tag{5}$$

It is worthy of noting that our experiment results show that the model performance when we use both *spatial* and *temporal* relations [1] is not better than the performance when we use *temporal* relation only. The reason may be explained by the fact that *temporal* relation has contained the property implicitly that users usually prefer to visit those POIs with shorter distance. Further considering *spatial* relation may bring no benefits or even degrades model performance. Thus, we choose to purely consider the *temporal* relation to construct POI transition graph.

[1] $\boldsymbol{s}(\boldsymbol{l_1}, \boldsymbol{l_2}) = e^{-\boldsymbol{d_t}(\boldsymbol{l_1}, \boldsymbol{l_2})} e^{-\boldsymbol{d_s}(\boldsymbol{l_1}, \boldsymbol{l_2})}$, where $\boldsymbol{d_t}$ and $\boldsymbol{d_s}$ denote the temporal and spatial similarity function, respectively.

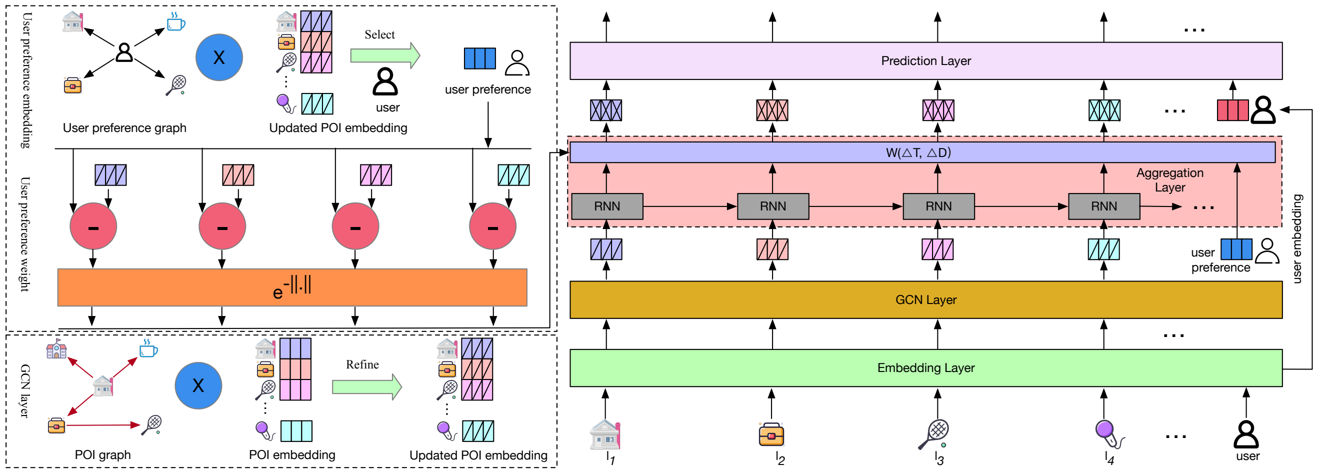


Figure 3: The overview of Graph-Flashback.

5 MODEL FRAMEWORK

This section presents the framework of our Graph-Flashback network, which consists of: (i) Embedding layer that learns the dense representations of users and POIs, (ii) GCN layer that enriches the representations of POIs by our learned POI transition graph, (iii) Aggregation layer that learns the aggregated hidden states by a weighted spatiotemporal and user preference effect as the output, and (iv) Prediction layer that recommends next POI by the aggregated output and user embedding. Figure 3 shows the network architecture of our proposed Graph-Flashback model.

5.1 Embedding layer

We design a multi-modal embedding layer to jointly learn the representations of the users and POIs. The representations encode user and POI information, which will later be combined with other modules for recommendation. In this light, it is important to learn the vector representations effectively. User and POI information contained in each check-in record is initially represented as one-hot vectors. However, it is difficult for a model to capture user preferences by using one-hot vector due to its sparsity. To this end, we propose to learn low dimensional dense representation for each user and each POI. Specifically, we first split the user historical trajectory T_{u_i} into multiple length-equally sub-sequences for each user u_i. Next, each sub-sequence are fed into the embedding layer as shown in the bottom right of Figure 3. Each POI is represented by a $|\mathcal{L}|$-dimensional one-hot vector. Besides, different users have different preferences, we represent each user as a $|\mathcal{U}|$-dimensional one-hot vector. The embedding layer will transform the user and POIs one-hot vectors into the corresponding low-dimensional dense representations, denoted by $\boldsymbol{e^u} \in \mathbb{R}^d$ and $\boldsymbol{e^l} \in \mathbb{R}^d$, respectively.

5.2 GCN Layer

To accurately reflect the characteristics of each POI, we find it insufficient to exclusively rely on the learned low-dimensional dense representation. As such, we use Graph Convolution Network (GCN) to refine the representation. Motivated by the success of LightGCN [9], we only focus on the core function of GCN: neighbor aggregation in our GCN layer. Note that transition graph $\boldsymbol{A}$ cannot reflect the impact of itself for each POI. To address the issue, we add identity matrix $\boldsymbol{I}$ into $\boldsymbol{A}$ (i.e., self-connection) to obtain a new transition graph $\hat{\boldsymbol{A}}$:

$$\hat{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I}. \tag{6}$$

Next, we normalize the new transition graph by Equation 7

$$\hat{\boldsymbol{A}} = \hat{\boldsymbol{D}}^{-1}\hat{\boldsymbol{A}}, \tag{7}$$

where $\hat{\boldsymbol{D}} = \textbf{diag}\left(\hat{\boldsymbol{A}}\right)$ denotes the out-degree diagonal matrix derived from $\hat{\boldsymbol{A}}$. We employ the transition graph $\hat{\boldsymbol{A}}$ to enrich the representation for each POI as shown in the bottom left of Figure 3. Finally, we have

$$\hat{\boldsymbol{X}} = \hat{\boldsymbol{A}}\boldsymbol{X}, \tag{8}$$

where $\boldsymbol{X} \in \mathbb{R}^{|\mathcal{L}|\times d}$ denotes the previous embeddings of all POIs, and $\hat{\boldsymbol{X}} \in \mathbb{R}^{|\mathcal{L}|\times d}$ is the updated POI embeddings.

5.3 Aggregation Layer

The aggregation layer consists of a recurrent module that captures the sequential patterns and an aggregation module that considers the impact of historical hidden states on current hidden state. The output of GCN layer, namely updated POIs embeddings, and user preference embedding are fed into the aggregation layer. Specifically, we use the basic RNNs as recurrent module to obtain all hidden states. However, using these hidden states directly for recommendation is incapable of making full use of the temporal periodicity and spatial context contained in user check-in trajectory. In particular, users are most likely to visit some POIs (e.g., home, office, etc.) regularly and those POIs with short distance. Inspired by Flashback [29], we explicitly utilize the spatiotemporal contexts to design a similarity function $\boldsymbol{w}(\cdot)$, which reflects the correlation between historical hidden state $\boldsymbol{h_j}$ and current one $\boldsymbol{h_i}$, $j < i$. Then, we have

$$\boldsymbol{w}(\Delta T_{i,j}, \Delta D_{i,j}) = hvc(2\pi\Delta T_{i,j})e^{-\alpha\Delta T_{i,j}}e^{-\beta\Delta D_{i,j}}, \tag{9}$$

where $hvc(x) = \frac{1+\cos x}{2}$ reflects the temporal periodicity, $\Delta D_{i,j}$ and $\Delta T_{i,j}$ denote the spatial and temporal interval between two POIs l_i

and l_j, and α and β represent temporal and spatial decay weight, respectively. Note that the similarity function $\boldsymbol{w}(\cdot)$ only focuses on the similarity correlation between POIs, neglecting the user general preference on POIs. Besides, the output of function $\boldsymbol{w}(\cdot)$ is limited in [0, 1]. To this end, we propose to integrate user preference into the function $\boldsymbol{w}(\cdot)$. As shown in the upper left of Figure 3, the *visiting* relation is used to construct a sparse User-POI preference graph $\mathcal{G}_p \in \mathbb{R}^{|\mathcal{U}|\times|\mathcal{L}|}$ in the same way as we introduced in Section 4 [2]. Then, we could obtain the general preference on POIs of each user by Equation 10.

$$\boldsymbol{P} = \mathcal{G}_p \hat{\boldsymbol{X}}, \tag{10}$$

where $\boldsymbol{P} \in \mathbb{R}^{|\mathcal{U}|\times d}$ is the user preference matrix. Finally, for each user u, we obtain new similarity function $\hat{\boldsymbol{w}}(\cdot)$ (Equation 11) that takes both user preference weight and spatiotemporal weight into consideration.

$$\hat{\boldsymbol{w}}(\Delta T_{i,j}, \Delta D_{i,j}) = \boldsymbol{w}(\Delta T_{i,j}, \Delta D_{i,j}) e^{-||\boldsymbol{P_u} - \boldsymbol{e}^{\boldsymbol{l_j}}||}, \tag{11}$$

where $\boldsymbol{P_u}$ denotes the preference embedding of user u, $||\cdot||$ is the L_2 distance. The aggregation module incorporates similarity function $\hat{\boldsymbol{w}}(\cdot)$ and historical hidden states into current one at each time step i. Thus, we have

$$\hat{\boldsymbol{h}}_{\boldsymbol{i}} = \frac{\sum_{j=0}^{i} \hat{w}_j * \boldsymbol{h_j}}{\sum_{j=0}^{i} \hat{w}_j}, \tag{12}$$

where $\hat{w}_j$ denotes the similarity $\hat{\boldsymbol{w}}(\Delta T_{i,j}, \Delta D_{i,j})$.

5.4 Prediction Layer

The output $\hat{\boldsymbol{h}}_{\boldsymbol{t}}$ of aggregation layer at each time step t and user embedding $\boldsymbol{e^u}$ is concatenated into a new vector, which is then fed into a fully connected layer to generate the final output through Equation 13:

$$\hat{\boldsymbol{y}}_{\boldsymbol{t}}^{\boldsymbol{u}} = \boldsymbol{W_f}[\hat{\boldsymbol{h}}_{\boldsymbol{t}} || \boldsymbol{e^u}], \tag{13}$$

where $\boldsymbol{W_f} \in \mathbb{R}^{|\mathcal{L}|\times 2d}$ is the learnable weight matrix, $||$ denotes the concatenation operation. We use the cross-entropy function, denoted by Equation 14, as our loss function.

$$-\sum_{u=1}^{|\mathcal{U}|}\sum_{i=1}^{m}\left(\log \sigma(\boldsymbol{y}_{\boldsymbol{k}}^{\boldsymbol{u}}) + \sum_{j=1, j\neq k}^{|\mathcal{L}|} \log(1 - \sigma(\hat{\boldsymbol{y}}_{\boldsymbol{j}}^{\boldsymbol{u}}))\right), \tag{14}$$

where m denotes the length of check-in sequence of each user, $\boldsymbol{y}_{\boldsymbol{k}}^{\boldsymbol{u}} \in \hat{\boldsymbol{y}}_{\boldsymbol{i}}^{\boldsymbol{u}}$ and $\hat{\boldsymbol{y}}_{\boldsymbol{j}}^{\boldsymbol{u}} \in \hat{\boldsymbol{y}}_{\boldsymbol{i}}^{\boldsymbol{u}}$ represent the corresponding prediction value of label l_k and other POIs $l_j \neq l_k$ for current location l_i of user u, respectively, and σ is the softmax function.

6 EXPERIMENTS

This section presents our experiments that demonstrate the effectiveness of our approach. We first present the datasets and baseline methods. Next, we present our experimental results and analysis by comparing our proposal against the state-of-the-art baselines. We also conduct ablation experiments to show the effect of each component in our framework, contrast experiments to show the effect of our learned transition graph, and parameter-tuning experiments to perform parameter sensitivity analysis.

[2]$\boldsymbol{d}(\boldsymbol{u}, \boldsymbol{l}) = ||\boldsymbol{u} + \boldsymbol{r_v} - \boldsymbol{l}||$, where $\boldsymbol{r_v}$ denotes *visiting* relation in Equation (3).

Table 1: Datasets Statistics

Dataset	Gowalla	Foursquare
#Users	7,768	45,343
#POIs	106,994	68,879
#Check-ins	1,823,598	9,361,228
#Entities	114,762	114,222
#Relations	4	4
#Triplets	6,420,914	7,200,989

6.1 Datasets

We evaluate our Graph-Flashback model on two widely used real-world datasets: Gowalla [3] and Foursquare [4]. Gowalla contains the check-ins from February 2009 to October 2010. Foursquare [30] is collected from April 2012 to January 2014. Each check-in consists of userID, POIID, latitude, longitude, and timestamp. Following the pre-processing technique in [29], we discard inactive users who have less than 100 check-ins and sort each user's check-ins in ascending order of timestamp. The first 80% check-ins of each user are split into multiple length-equally (e.g., 20) sequences, which are chosen as training set. Likewise, the remaining 20% are regarded as testing set. In addition, the training set is also used to construct our Spatial-Temporal knowledge graph. Table 1 shows the statistics of the two datasets used in our experiments.

6.2 Baselines

We consider the following state-of-the-art methods as the baselines in the experiments.

- PRME [4]: A metric embedding method, which captures the personalized sequential transition patterns by learning user and POI embeddings.
- STRNN [15]: Extends RNNs by customized spatiotemporal transition matrices.
- DeepMove [3]: Combines an attention layer with gated recurrent units (GRU) to learn long-term periodicity and short-term sequential patterns.
- LBSN2Vec [30]: A hypergraph embedding method, which ranks POIs based on their similarity with user and time embeddings.
- STGN [36]: Extends the LSTM by introducing spatial and temporal gates for learning the users' long-term and short-term preferences.
- LightGCN [9]: Simplifies Graph Convolution Netwotk (GCN) to learns user's preferences on POIs with pairwise ranking loss.
- Flashback [29]: An RNN-based model, which is good at dealing with past historical check-in records.
- STAN [20]: An attention-based model, which explicitly uses relative spatiotemporal information between POIs within the user trajectory.

[3]http://snap.stanford.edu/data/loc-gowalla.html
[4]https://sites.google.com/site/yangdingqi/home

Table 2: Performance comparison against baselines on Gowalla and Foursquare datasets. In each column, we use boldface and underline to indicate the best and second-best results, respectively. (*STAN use a part of users (100) to train model at a time and test performance on these users. Therefore, it requires running many experiments to test model performance on all users and regards the average performance of these experiments as the final performance. In this light, we use all users and first 2000 users to test model performance on Gowalla and Foursquare datasets due to the large number of users on Foursquare dataset.)

Methods	Gowalla				Foursquare			
	Acc@1	Acc@5	Acc@10	MRR	Acc@1	Acc@5	Acc@10	MRR
PRME	0.0740	0.2146	0.2899	0.1503	0.0982	0.3167	0.4064	0.2040
STRNN	0.0900	0.2120	0.2730	0.1508	0.2290	0.4310	0.5050	0.3248
DeepMove	0.0625	0.1304	0.1594	0.0982	0.2400	0.4319	0.4742	0.3270
LBSN2Vec	0.0864	0.1186	0.1390	0.1032	0.2190	0.3955	0.4621	0.2781
STGN	0.0624	0.1586	0.2104	0.1125	0.2094	0.4734	0.5470	0.3283
LightGCN	0.0428	0.1439	0.2115	0.1224	0.0540	0.1790	0.2710	0.1574
Flashback	0.1158	0.2754	0.3479	0.1925	0.2496	0.5399	0.6236	0.3805
STAN*	0.0891	0.2096	0.2763	0.1523	0.2265	0.4515	0.5310	0.3420
Graph-Flashback	**0.1512**	**0.3425**	**0.4256**	**0.2422**	**0.2805**	**0.5757**	**0.6514**	**0.4136**
Improvement (%)	30.57%	24.36%	22.33%	25.82%	12.38%	6.63%	4.46%	8.70%

6.3 Evaluation Metrics

We use two widely used evaluation metrics: average Accuracy@K (Acc@K), and Mean Reciprocal Rank (MRR) following [29] to evaluate the performance of Graph-Flashback. Acc@K is the rate of true positive samples in the predicted top-K positive samples, In our experiment, we adopt $K = \{1, 5, 10\}$ to evaluate recommendation performance. The Acc@K is computed as follows:

$$Acc@K = \frac{1}{|U|} \sum_{u \in U} \frac{|S_u^K \bigcap S_u^L|}{|S_u^L|}, \tag{15}$$

where S_u^K is the set of predicted top-K POIs for user u, S_u^L denotes the ground truth label of user u. Unlike Acc@K, which focuses on top-K, MRR aims to measure the overall recommendation performance of the model. The formulation of MRR is presented as follows:

$$MRR = \frac{1}{|U|} \sum_{u \in U} \frac{1}{rank_u}, \tag{16}$$

where $rank_u$ denotes the rank of S_u^L in S_u^K for user u.

6.4 Settings

We use the second scheme (i.e., rank-based) to construct knowledge graph and TransE [1] is chosen to construct our POI transition graph. When constructing the POI transition graph $\boldsymbol{A}$ and User-POI preference graph $\boldsymbol{\mathcal{G}_p}$, we consider k_t and $k_p = \{10, 20, 30, 50, 100\}$ nearest neighbors for each POI (user). In addition, the dimension of hidden states and user (POI) embedding is set to 10 empirically. And the temporal decay factor α and spatial decay factor β follow the default setting in [29]. We evaluate the model performance every 5 epochs. Our implementation is available in Pytorch [5].

6.5 Results and Analysis

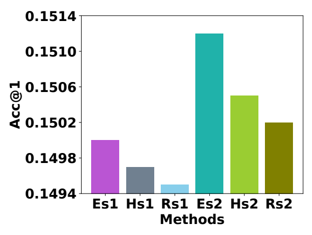
Table 2 shows the experiment results on Gowalla and Foursquare datasets. Based on these results, we have the following observations and corresponding analysis:

[5]https://github.com/kevin-xuan/Graph-Flashback

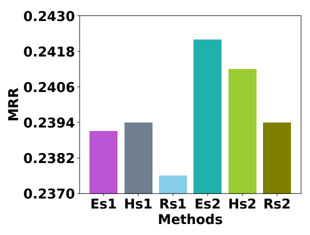
Table 3: Ablation experiments on Gowalla dataset. Graph-Flashback *w/o* GCN denotes disabling the GCN layer, and Graph-Flashback *w/o* Preference represents disabling the User-POI preference graph.

Methods	Gowalla			
	Acc@1	Acc@5	Acc@10	MRR
Flashback	0.1158	0.2754	0.3479	0.1925
Graph-Flashback *w/o* GCN	0.1356	0.3055	0.3762	0.2163
Graph-Flashback *w/o* Preference	0.1506	0.3419	0.4253	0.2419
Graph-Flashback	**0.1512**	**0.3425**	**0.4256**	**0.2422**

- The results on Gowalla and Foursquare datasets show that our proposed Graph-Flashback significantly outperforms all other state-of-the-art baseline methods under all evaluation metrics. Specifically, Graph-Flashback outperforms the second-best method Flashback with ratios 30.57%, 24.36%, 22.33%, and 25.82% on Acc@1, Acc@5, Acc@10, and MRR respectively on Gowalla dataset. In addition, Graph-Flashback gains 8.04% average improvement compared with Flashback on Foursquare dataset. In a nutshell, the aforementioned findings clearly show the superiority of Graph-Flashback.
- Compared with Flashback, our Graph-Flashback exhibits substantial improvement in general. The reason is that the POI transition graph $\boldsymbol{A}$ may help to better capture the transition patterns among POIs by refining POI embeddings. In addition, personalized POI recommendation may significantly benefit from tackling user preference with sufficient consideration properly. Similarly, STAN is an attention-based latest model considering the effect of past historical check-ins. However, it performs poorly based on our experimental results. The reason is that its training scheme may make model performance rely on the selected users.
- Comparing the performance improvement on Foursquare datasets, we see that Graph-Flashback performs better on the Gowalla datasets. Meanwhile, we observe that the density

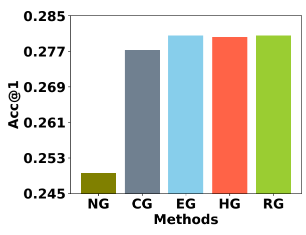


(a) Acc@1 comparison on Gowalla

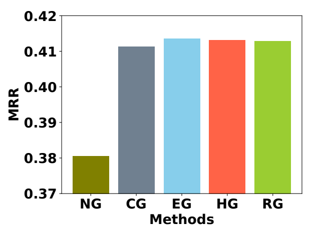


(b) MRR comparison on Gowalla

Figure 4: The performance comparison among different KGE algorithms used in different knowledge graphs constructed by different schemes on Gowalla dataset. E is TransE, $s1$ denotes that knowledge graph is constructed by first scheme.

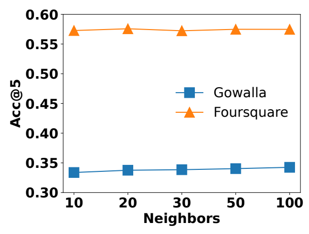


(a) Acc@1 comparison on Foursquare

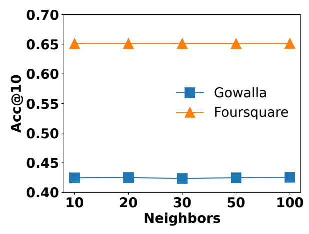


(b) MRR comparison on Foursquare

Figure 5: The performance comparison among learned POI graphs and customized POI graph. NG denotes not using any graph, CG represents the customized graph, and EG denotes the POI graph learned by TransE algorithm.



(a) The impact of neighbor number k_t



(b) The impact of neighbor number k_p

Figure 6: The performance comparison about the number of nearest neighbors k_t and k_p .

(0.002) of check-in records on Gowalla dataset is lower than that (0.003) on Foursquare dataset. It is obvious that our proposed Graph-Flashback model has a strong ability to handle sparse data.

6.6 Ablation Study

There are two main components in our framework: (i) the GCN layer, and (ii) the aggregation layer. To show the effects of these components, we conduct an ablation experiment on Graph-Flashback and the results on Gowalla dataset are shown in Table 3. Based on the statistics from Table 3, we have the following findings:

- We observe that the POI transition graph used in GCN layer help to improve the model performance. The learned graph could enrich the representation of each POI, which further helps sequential models capture the transition patterns among POIs.
- We find that using user preference properly can improve the model performance. When we integrate the learned User-POI preference graph into spatiotemporal contexts, the performance regarding personalized POI recommendation can be boosted.
- We notice that Graph-Flashback performs slightly better than Graph-Flashback w/o Preference. The reason is that the impact of user preference has been partially reflected by the GCN.

6.7 Contrast Study

We conduct different contrast experiments to show the effects of our learned graphs. To this end, we construct customized graph by calculating POI similarity based on whether the two POIs are visited by the same user on the same day. Specifically, the similarity between POI pair indicates the number of days when both POIs are visited by same user. Next, the POI similarity graph is normalized by Equation 4 and Equation 5. From the results shown in Figure 4 and Figure 5, we have the following observations:

- Figure 4 shows that the POI transition graph learned from the Spatial-Temporal knowledge graph constructed by the second scheme (i.e., rank-based) and TransE algorithm could obtain better model performance on Gowalla dataset. The reason might be that the Spatial-Temporal knowledge graph constructed by the second scheme contains more *spatial* edges, which could help to learn better representation of each POI implicitly. In addition, simpler algorithm TransE might be more suitable for our knowledge graph.
- As shown in Figure 5, the results are significantly better with POI graphs than without any graph on Foursquare dataset. Moreover, we could find that our POI transition graphs learned by different KGE algorithms outperform the customized graph. This shows that our learned POI graph could reflect the correlation among POIs more accurately.

6.8 Hyperparameter Sensitivity Analysis

We conduct parameter sensitivity experiment and then introduce our analyses. As shown in Figure 6, we have the following two findings:

- Figure 6 (a) shows that $k_t = 100$ and $k_t = 20$ are the best number of nearest neighbors for POI transition graph on Gowalla and Foursquare datasets respectively. On Gowalla dataset, the model performance improves with the increasement of k_t. However, as k_t increases, we could find that the performance improves firstly and then degrades on Foursquare dataset. The reason might be that the check-ins data of Gowalla is more sparse than that of Foursquare. Therefore, each POI needs more neighbors and proper neighbors to enrich the representation of itself on Gowalla and Foursquare dataset, respectively.

- We find that $k_p = 100$ obtains better model performance on both Gowalla and Foursquare datasets from the results in Figure 6 (b). And the model performance remains almost stable on Acc@10 metric. In a nutshell, the performance of our Graph-Flashback method is insensitive to the hyperparameter k_p.

7 CONCLUSION AND FUTURE WORK

To the end, we present a learning-based method to learn a POI graph, which reflects the transition patterns among POIs. We design a novel model called Graph-Flashback that integrates the learned graph into existing sequential models seamlessly for better capturing the transition patterns. In addition, the property that the same user may have different preferences on different spatiotemporal contexts is sufficiently considered by a carefully-designed similarity function. Experimental results on Gowalla and Foursquare datasets show the superiority of Graph-Flashback in term of accuracy. Moreover, we give the ablation experiments and analysis of different components in our framework, which confirm the effectiveness of each component.

In the future, we will consider other side information (e.g., POI categories, user information) in the construction of Spatial-Temporal knowledge graph. Besides, we will consider the similarity correlation between users to further improve the model performance. Moreover, personalized POI transition graph is a promising direction as well.

ACKNOWLEDGEMENTS

This work was supported by the NSFC (U2001212, U21B2046, 62032001, and 61932004). Bin Yao was supported by the NSFC (61922054, 61872235, 61832017), the National Key Research and Development Program of China (2020YFB1710200), the Science and Technology Commission of Shanghai Municipality (STCSM) AI under Project 19511120300.

REFERENCES

[1] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating Embeddings for Modeling Multi-relational Data. In *NIPS*.
[2] Chen Cheng, Haiqin Yang, Michael R. Lyu, and Irwin King. 2013. Where You Like to Go Next: Successive Point-of-Interest Recommendation. In *IJCAI*.
[3] Jie Feng, Yong Li, Chao Zhang, Funing Sun, Fanchao Meng, Ang Guo, and Depeng Jin. 2018. DeepMove: Predicting Human Mobility with Attentional Recurrent Networks. In *WWW*.
[4] Shanshan Feng, Xutao Li, Yifeng Zeng, Gao Cong, Yeow Meng Chee, and Quan Yuan. 2015. Personalized Ranking Metric Embedding for Next New POI Recommendation. In *IJCAI*.
[5] Qing Guo, Zhu Sun, Jie Zhang, and Yin-Leng Theng. 2020. An Attentional Recurrent Neural Network for Personalized Next Location Recommendation. In *AAAI*.
[6] Qingyu Guo, Fuzhen Zhuang, Chuan Qin, Hengshu Zhu, Xing Xie, Hui Xiong, and Qing He. 2020. A Survey on Knowledge Graph-Based Recommender Systems. *CoRR* (2020).
[7] Peng Han, Zhongxiao Li, Yong Liu, Peilin Zhao, Jing Li, Hao Wang, and Shuo Shang. 2020. Contextualized Point-of-Interest Recommendation. In *IJCAI*.
[8] Peng Han, Jin Wang, Di Yao, Shuo Shang, and Xiangliang Zhang. 2021. A Graph-based Approach for Trajectory Similarity Computation in Spatial Networks. In *ACM SIGKDD*. https://doi.org/10.1145/3447548.3467337
[9] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yong-Dong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. In *SIGIR*.
[10] Dejiang Kong and Fei Wu. 2018. HST-LSTM: A Hierarchical Spatial-Temporal Long-Short Term Memory Network for Location Prediction. In *IJCAI*.
[11] Yang Li, Tong Chen, Yadan Luo, Hongzhi Yin, and Zi Huang. 2021. Discovering Collaborative Signals for Next POI Recommendation with Iterative Seq2Graph Augmentation. In *IJCAI*.
[12] Defu Lian, Yongji Wu, Yong Ge, Xing Xie, and Enhong Chen. 2020. Geography-Aware Sequential Location Recommendation. In *SIGKDD*.
[13] Nicholas Lim, Bryan Hooi, See-Kiong Ng, Xueou Wang, Yong Liang Goh, Renrong Weng, and Jagannadan Varadarajan. 2020. STP-UDGAT: Spatial-Temporal-Preference User Dimensional Graph Attention Network for Next POI Recommendation. In *CIKM*.
[14] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning Entity and Relation Embeddings for Knowledge Graph Completion. In *AAAI*.
[15] Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. 2016. Predicting the Next Location: A Recurrent Model with Spatial and Temporal Contexts. In *AAAI*.
[16] Xin Liu, Yong Liu, Karl Aberer, and Chunyan Miao. 2013. Personalized point-of-interest recommendation by mining users' preference transition. In *CIKM*.
[17] Xin Liu, Yong Liu, and Xiaoli Li. 2016. Exploring the Context of Locations for Personalized Location Recommendations. In *IJCAI*.
[18] Yong Liu, Wei Wei, Aixin Sun, and Chunyan Miao. 2014. Exploiting Geographical Neighborhood Characteristics for Location Recommendation. In *CIKM*.
[19] Yong Liu, Lifan Zhao, Guimei Liu, Xinyan Lu, Peng Gao, Xiao-Li Li, and Zhihui Jin. 2018. Dynamic Bayesian Logistic Matrix Factorization for Recommendation with Implicit Feedback. In *IJCAI*.
[20] Yingtao Luo, Qiang Liu, and Zhaocheng Liu. 2021. STAN: Spatio-Temporal Attention Network for Next Location Recommendation. In *WWW*.
[21] Wesley Mathew, Ruben Raposo, and Bruno Martins. 2012. Predicting future locations with hidden Markov models. In *UbiComp*.
[22] 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*.
[23] Haining Tan, Di Yao, Tao Huang, Baoli Wang, Quanliang Jing, and Jingping Bi. 2021. Meta-Learning Enhanced Neural ODE for Citywide Next POI Recommendation. In *MDM*.
[24] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *NIPS*.
[25] Hao Wang, Huawei Shen, Wentao Ouyang, and Xueqi Cheng. 2018. Exploiting POI-Specific Geographical Influence for Point-of-Interest Recommendation. In *IJCAI*.
[26] Xiao Wang, Zheng Wang, Toshihiko Yamasaki, and Wenjun Zeng. 2021. Very Important Person Localization in Unconstrained Conditions: A New Benchmark. In *AAAI*.
[27] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge Graph Embedding by Translating on Hyperplanes. In *AAAI*.
[28] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. In *ICLR*.
[29] Dingqi Yang, Benjamin Fankhauser, Paolo Rosso, and Philippe Cudré-Mauroux. 2020. Location Prediction over Sparse User Mobility Traces Using RNNs: Flashback in Hidden States!. In *IJCAI*.
[30] Dingqi Yang, Bingqing Qu, Jie Yang, and Philippe Cudré-Mauroux. 2019. Revisiting User Mobility and Social Relationships in LBSNs: A Hypergraph Embedding Approach. In *WWW*.
[31] Di Yao, Chao Zhang, Jian-Hui Huang, and Jingping Bi. 2017. SERM: A Recurrent Model for Next Location Prediction in Semantic Trajectories. In *CIKM*.
[32] Mao Ye, Peifeng Yin, Wang-Chien Lee, and Dik Lun Lee. 2011. Exploiting geographical influence for collaborative point-of-interest recommendation. In *SIGIR*.
[33] 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*.
[34] Zeping Yu, Jianxun Lian, Ahmad Mahmoody, Gongshen Liu, and Xing Xie. 2019. Adaptive User Modeling with Long and Short-Term Preferences for Personalized Recommendation. In *IJCAI*.
[35] Yuyu Zhang, Hanjun Dai, Chang Xu, Jun Feng, Taifeng Wang, Jiang Bian, Bin Wang, and Tie-Yan Liu. 2014. Sequential Click Prediction for Sponsored Search with Recurrent Neural Networks. In *AAAI*.
[36] Pengpeng Zhao, Haifeng Zhu, Yanchi Liu, Jiajie Xu, Zhixu Li, Fuzhen Zhuang, Victor S. Sheng, and Xiaofang Zhou. 2019. Where to Go Next: A Spatio-Temporal Gated Network for Next POI Recommendation. In *AAAI*.