Sequential-Knowledge-Aware Next POI Recommendation: A Meta-Learning Approach

YUE CUI and HAO SUN, University of Electronic Science and Technology of China, China
YAN ZHAO, Aalborg University, Denmark
HONGZHI YIN, The University of Queensland, Australia
KAI ZHENG, University of Electronic Science and Technology of China, China

Accurately recommending the next point of interest (POI) has become a fundamental problem with the rapid growth of location-based social networks. However, sparse, imbalanced check-in data and diverse user check-in patterns pose severe challenges for POI recommendation tasks. Knowledge-aware models are known to be primary in leveraging these problems. However, as most knowledge graphs are constructed statically, sequential information is yet integrated. In this work, we propose a meta-learned sequential-knowledge-aware recommender (Meta-SKR), which utilizes sequential, spatio-temporal, and social knowledge to recommend the next POI for a location-based social network user. The framework mainly contains four modules. First, in the graph construction module, a novel type of knowledge graph—the sequential knowledge graph, which is sensitive to the check-in order of POIs—is built to model users' check-in patterns. To deal with the problem of data sparsity, a meta-learning module based on latent embedding optimization is then introduced to generate user-conditioned parameters of the subsequent sequential-knowledge-aware embedding module, where representation vectors of entities (nodes) and relations (edges) are learned. In this embedding module, gated recurrent units are adapted to distill intra- and inter-sequential knowledge graph information. We also design a novel knowledge-aware attention mechanism to capture information surrounding a given node. Finally, POI recommendation is provided by inferring potential links of knowledge graphs in the prediction module. Evaluations on three real-world check-in datasets show that Meta-SKR can achieve high recommendation accuracy even with sparse data.

CCS Concepts: • **Information systems** → **Recommender systems**; *Location based services;*

Additional Key Words and Phrases: POI recommendation, knowledge graphs, meta-learning

This work was supported by NSFC (61972069, 61836007, 61832017) and Sichuan Science and Technology Program under grant 2020JDTD0007.

Authors' addresses: Y. Cui and H. Sun, University of Electronic Science and Technology of China, 2006 Xiyuan Ave., Chengdu, Sichuan, China, 611731; emails: cuiyue@uestc.edu.cn, sunhao@std.uestc.edu.cn; Y. Zhao, Office 3.2.57, Selma Lagerlöfs Vej 300, Department of Computer Science, Aalborg University, Aalborg, Denmark, DK-9220; email: yanz@cs.aau.dk; H. Yin, St Lucia Campus, The University of Queensland, Brisbane, Queensland, Australia, QLD 4072; email: h.yin1@uq.edu.au; K. Zheng (corresponding author), 2006 Xiyuan Ave., University of Electronic Science and Technology of China, Chengdu, Sichuan, China, 611731; email: zhengkai@uestc.edu.cn.

1046-8188/2021/09-ART23 $15.00
https://doi.org/10.1145/3460198

ACM Reference format:
Yue Cui, Hao Sun, Yan Zhao, Hongzhi Yin, and Kai Zheng. 2021. Sequential-Knowledge-Aware Next POI Recommendation: A Meta-Learning Approach. *ACM Trans. Inf. Syst.* 40, 2, Article 23 (September 2021), 22 pages. https://doi.org/10.1145/3460198

1 INTRODUCTION

The proliferation of GPS-enabled devices and **location-based social networks (LBSNs)** has promoted users to check-in at **points of interest (POIs)** so as to keep mementos and share life experience with friends. One of the most fundamental techniques maintaining LBSNs is POI recommendation, which assists users' decision making when they face huge volumes of information. For example, a travel recommendation system can help a user arrange trips and discover next places to visit based on her own preference.

As an important feature, people's mobile behavioral patterns can be significantly influenced by their own historical experiences [7, 20, 29, 44, 48, 50]. As a consequence, POI recommendation based on historical spatio-temporal data has proven to be effective. Traditionally, LRT [10] incorporates temporal feature as one of the properties of user behavior in LBSNs. It models users' check-in temporal patterns *weekly*/*monthly*/*yearly* with check-in matrix $C^{(t)}$ and then implements matrix factorization to learn user and POI representation vectors, in which user vectors are of time slots. More recently, machine learning methods have greatly improved state-of-the-art performance on the task of POI recommendation. To capture temporal patterns of user preference, some works [17, 19, 44, 50] jointly learn users' long- and short-term preferences via the attention mechanism for next POI recommendation with sequential and contextual information considered. ST-RNN [18] models local temporal and spatial contexts with **recurrent neural networks (RNNs)**, where there are time-specific transition matrices for different time intervals and distance-specific transition matrices for different geographical distances in each layer. DeepMove [7] learns user preference using RNNs for historical sequence and short-term current sequence. Yao [48] profiles the temporal popularity of POIs and then incorporates the degree of temporal matching between users and POIs into personalized POI recommendations. Zhao et al. [54] hierarchically learn user's check-in patterns with state-based stacked RNNs. Despite superior results those models get, massive training data is required to obtain such performance, which is probably inaccessible for the majority of LBSNs in real practice.

The recent advances in meta-learning has made it prevalent in instantiating a model with limited training data [3, 9, 23, 30, 47]. Meta-learned predictions based on sequential data has proven to be effective in domains including traffic forecasting [28, 47], taxi demand prediction [28], resource quality prediction [47], online item recommendation of E-commerce [6], and so forth. However, there have been several differences between the preceding tasks and POI recommendation, which make it hard to implement those approaches directly. With regard to temporally imbalanced data, in problem settings such as traffic prediction, it is easier to obtain continuous temporal sequences, whereas in LBSNs, people's check-ins are discrete and unevenly distributed, which makes it rather unpredictable. For example, a user may create extensive check-in records during national holidays while rarely visiting POIs on work days. With regard to complex intereactions, sequential relation is the primarily consideration of those tasks. However, given the explicit and potential interactions between users and/or POIs, in the task of POI recommendation, side information in LBSNs is also of great significance.

In this article, to tackle the preceding challenges, we take two special considerations:

- *Sequential knowledge*: Interactions between users and/or POIs can be utilized to give accurate recommendations [24]. These interactions are hidden in the spatial, temporal, and social

context of LBSNs. There have been works taking advantage of knowledge graphs to build a context-aware recommender [27, 29, 32, 35, 51]. However, most of them ignore sequential information, which might lead to the problem of either forgetting past experience or "over-recommendation." For example, a well-trained model that has accurately learned the preference of a user might continuously recommend the same type of POI to her, which is definitely an unpleasant experience. Thus, it is desirable to incorporate a user's sequential check-in behaviors when building such a knowledge-aware recommender.

- *Sparse data with large number of null intervals*: Predicting user behavior from limited and sparse data is one of the fundamental research problems in POI recommendation. To remedy the problem of data sparsity, some works ignore the exact physical time of check-in and deal merely with a sequential pattern. However, this practice spoils the check-in patterns hidden in "null intervals," which refers to the intervals in a check-in graph when a user does not make any check-in. Consider a scenario where two users have visited the same POIs during a certain period of time, in which the first has regular check-in patterns, e.g., every day, whereas the other finished all the check-ins in a few days but stays offline for the rest of the time. Apparently, the recommender should not make the same recommendations to them even though they share the same check-in sequence. A recommender should be able to capture the difference between the two users. Therefore, a model that is adaptive to various check-in habits on sparse data is required.

In this work, we propose a meta-learning framework, named ***Meta*-Learned *S*equential-*K*nowledge-Aware *R*ecommender (Meta-SKR)** for next POI recommendation. We deal with the first consideration by constructing **sequential knowledge graphs (SKGs)** from users' check-in sequences, which is a novel type of knowledge graph that jointly models sequential, geographical, temporal, and social information. Specifically, we take users and POIs as entities (nodes) and link them with five types of relations (edges) that capture check-in features hidden in POI visiting orders and null check-in intervals. Each SKG is constructed on the basis of the check-in order a user made, and thus sequential information can be integrated and users' dynamic and evolving preferences can be distilled. Regarding the second consideration, to fast learn from sparse training data, we introduce a meta-learner that can generate user-specific parameters based on the input check-in sequence, which are then used in a subsequent embedding network to learn representation vectors of entities and edges of SKGs. Finally, the task of POI recommendation can be accomplished by predicting links between the user and POI nodes.

Our contributions are summarized as follows:

- We propose a novel meta-learned framework that leverages both fine-granularity sequential patterns and spatial-temporal-social context information for POI recommendation. To the best of our knowledge, this is the first work to jointly model these factors into a shared knowledge graph and adopt meta-learning to address the challenges of user behavior modeling and data sparsity in POI recommendation tasks.
- To learn representation vectors of users and POIs from SKGs, we develop an attention-based graph embedding method that first extracts intra- and inter-SKG information for a node and then propagates the information to the node with edge directions considered.
- We conduct extensive experiments on three public real-world check-in datasets to evaluate the performance of our model. The favorable results verify our expectation that the proposed framework can reach a high accuracy even with sparse data.

2 PRELIMINARIES

In this section, we briefly introduce a set of preliminary concepts in the context of a spatio-temporal social-based knowledge graph, then give an overview of our framework.

Table 1. Summary of Notations

Notation	Description
c	A check-in
u, U	A user and a set of users
p, P	A POI and a set of POIs
$s, \mathcal{S}$	A check-in sequence and a set of sequences
$e, \mathcal{E}$	An entity and a set of entities
$r, \mathcal{R}$	A relation and a set of relations
$G, \mathcal{G}$	An SKG and a set of SKGs
$t = (e_i, r_k, e_j)$	A triple with head entity e_i, tail entity e_j, and relation r_k of a knowledge graph
$\mathcal{M}^{tr}$,$\mathcal{M}^{val}$,$\mathcal{M}^{ts}$	The meta-training, meta-validation, meta-test set
($\mathcal{D}^{tr}$, $\mathcal{D}^{ts}$)	A meta-learning task, composed of a training set and test set
τ	A time instance
l_p	The location of POI p
$d_i^{+/-}$	The out/in-degree of node i
$\mathcal{N}_i$	The set of neighbors of node i
λ	The length of a check-in sequence s
T	The number of SKGs in $\mathcal{D}^{tr}$
$\mathbf{h}$	The representation vector of an entity
$\mathbf{g}$	The representation vector of a relation
$\mathbf{x}$	The representation vector of a triple
Θ	General notation for parameters in the SKR network
$\hat{y}_t$	The estimated probability of triple t being valid in a knowledge graph

2.1 Definitions and Notations

Definition 1 (Check-in Sequence). Each check-in c includes user ID u, POI ID p_c, check-in time τ_c, and check-in location in terms of (latitude, longitude). Given the overall check-in sequence of user u, i.e., $C_u = [c_1, c_2, \ldots, c_n]$, where the check-ins are ordered in time, a set of subsequences, $\mathcal{S}_u = \{s_1, \ldots, s_m\}$, can be built by continuously sampling λ-length subsequences from C_u with stride r, i.e., $s_i = [c_{i1}, c_{i2}, \ldots, c_{i\lambda}]$.

Definition 2 (Sequential Knowledge Graph (SKG)). Given a sequence s_i, an SKG denoted as $G_i = (\mathcal{E}, \mathcal{R})$ can be constructed by generating triples that retrieve certain check-in patterns in s_i, where $\mathcal{E}$ is the set of entities (nodes) and $\mathcal{R}$ is the set of relations (edges). A triple in a SKG can be denoted as $t = (e_i, r_k, e_j)$, which describes that entities e_i and e_j are connected by relation r_k. The set containing all SKGs constructed from $\mathcal{S}_u$ is denoted as $\mathcal{G}_u = \{G_1, \ldots, G_m\}$.

Table 1 summarizes the major notations of this article.

2.2 Problem Statement

Before mathematically defining the problem, we first introduce the meta-training paradigm used in this article, which is based on the mainstream episodic formulation [40]. Suppose we have a group of users and their check-in history, and we split each user's check-in data into meta-training set $\mathcal{M}_u^{tr}$, meta-validation set $\mathcal{M}_u^{val}$, and meta-test set $\mathcal{M}_u^{ts}$ for model training, selection, and final

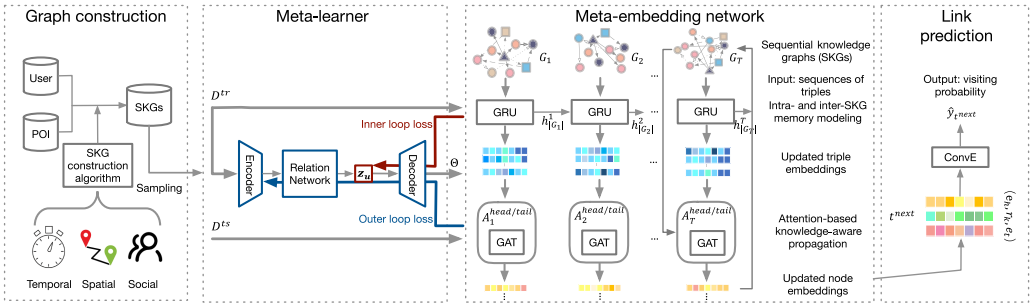


Fig. 1. Overview of Meta-SKR. In the meta-embedding network, for limited space, only dataflow of G_T is fully depicted, whereas the same flow exists in all SKGs.

test, respectively. Each task sampled from the meta set has two components: $\mathcal{D}^{tr}$ and $\mathcal{D}^{ts}$. $\mathcal{D}^{tr}$ is composed of T successive check-in sequences and is used for user-specific parameter generation, i.e., $\mathcal{D}^{tr} = \{s_{i1}, \ldots, s_{iT}\}$. $\mathcal{D}^{ts}$, a randomly chosen sequence with $\mathcal{D}^{tr} \cap \mathcal{D}^{ts} = \emptyset$, is used for evaluating the generated parameters and making predictions.

Given a set of users $U = \{u_1, \ldots, u_n\}$ and their check-in sequences $\{\mathcal{S}_1, \ldots, \mathcal{S}_n\}$, we aim to recommend the next POI that might be visited by u_i after her most recent check-in sequence $s^r_{u_i}$ in the meta-test set, i.e.,

$$t^{next}_{u_i} = \arg\max_{t^{next}_{u_i}} f\left(t^{next}_{u_i} | s^r_{u_i}, g_\Theta\right),$$

where $t^{next}_{u_i}$ is the next check-in of u_i in the form of triple, which consists of two entities (the user and a POI that is going to be visited) and a relation (a specific check-in pattern), f denotes meta-SKR that computes the probability of whether $t^{next}_{u_i}$ is a valid triple, and g_Θ denotes the the initialized model with parameters adapted from $\mathcal{D}^{tr}$ of the task.

2.3 Framework Overview

As illustrated in Figure 1, the proposed Meta-SKR contains four modules: the graph construction module, the meta-learner, the meta-embedding module, and the link prediction module. In *the graph construction module*, to distill sequential, geographical, temporal, and social information from user check-in history and model user check-in patterns, we design SKGs, which are a novel type of knowledge graph. The triples of SKGs are construed sequentially based on check-in sequences. Five types of relations are built to extract certain check-in patterns. With regard to the *meta-learner*, we sample tasks in the form of $(\mathcal{D}^{tr}, \mathcal{D}^{ts})$ from SKGs of each user, which are then taken as inputs of the meta-learner, where user-dependent latent representations of parameters in subsequent meta-embedding module are learned. Gradient-based adaptation is performed in the latent space with training loss (the red flow). The meta-learner is optimized based on test loss (the blue flow). In *the meta-embedding module*, nodes and edges embeddings are obtained. Intra- and inter-SKG sequential correlations are captured by adapting **gated recurrent units (GRUs)**, and then **graph attention networks (GATs)** that use triple head/tail-dependent connection matrix to incorporate context-aware edges in the nodes' neighborhood. At last, in the *link prediction module*, we make recommendations on users' next POIs by computing the probability of whether potential triples are valid using ConvE [5], where a multi-layer convolutional network is adopted. The four modules can be categorized into two parts: the base model, SKR, which is composed of

the graph construction module, the meta-embedding module, and the link prediction module, and the meta-learner.

3 PROPOSED METHOD

In this section, we describe the framework of Meta-SKR for POI recommendation. The section is organized as follows. In Section 3.1, we introduce the algorithm for graph triple generation, where sequential, spatio-temporal, and social relations are explored to generate triples from check-in sequence set $\mathcal{S}$. In Section 3.2, an embedding network is designed to learn representations of users and POIs, followed by a discussion on how to make POI recommendations by inferring potential links in Section 3.3. In Section 3.4, we provide a description of the meta-learning strategy for advanced adaption of user-specific model parameters of the embedding network discussed in Section 3.2.

3.1 SKG Construction

For a given user u and one of her check-in sequences $s = \{c_1, \ldots, c_\lambda\}$, an SKG can be constructed with the following entities and relations.

3.1.1 Entities of an SKG. There are three kinds of entities in the SKG:

- User: This represents the user who creates the current check-in sequence, denoted as u.
- POI: This represents a specific POI, denoted as p_i.
- Friend: This represents a friend of u, denoted as f_i .

There will be an entity of FRIEND type in the SKG, if existing check-in c_{f_i} in one of the friend's check-in sequences s_{f_i} such that:

i. $p_{c_{f_i}} = p_{c_u}$,
ii. $\tau_{c_{f_i}} \prec \tau_{c_u}$,

where $p_{c_{f_i}}$ denotes POI ID of check-in c_{f_i}. The intuition of the preceding friend-based entity conditions is that the user and her friend check in at the same location but the check-in time of her friend is earlier.

The reasons for leveraging this kind of interaction are twofold. First, people, especially friends, who check in at similar POIs have greater similarities. Second, there is potential possibility that the user visited the location is because a friend, who had been there before, explicitly or implicitly recommended it to her. In this way, we could make sure that the social effect imposed by friends sharing similar check-in behavior is incorporated in SKGs.

3.1.2 Relations of the SKG. Five types of relations can be defined as follows. Note that all edge weights are normalized within their edge types:

- Temporal Sequential-check-in: If the time interval between two successive check-ins in s_u is no longer than Δ_τ, then nodes of the two corresponding POIs can be linked by a temporal sequential-check-in edge, which is denoted as r_t, directed from the earlier check-in POI to the later one. The weight of r_t is defined as the time interval between the two check-ins.
- Spatial Sequential-check-in: If the distance of two successive check-in records in s_u is no farther than Δ_d, then nodes of the two corresponding POIs can be linked by a spatial sequential-check-in edge, which is denoted as r_s, directed from the temporally checked-in earlier POI to the later one. The weight of r_s is defined as the Euclidean distance between the two POIs.

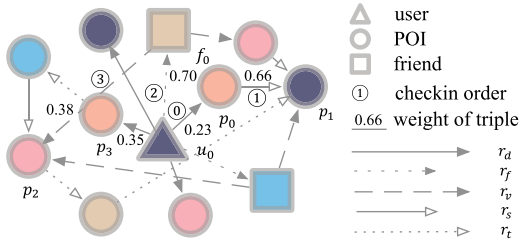


Fig. 2. SKG: a running example (best viewed in color). In this example, the user u_0 (triangle) checked in at $p_0 \rightarrow p_1 \rightarrow p_2 \rightarrow \cdots$ sequentially, whereas her friend f_0 (brown rectangle) checked in at p_2 ahead of her. For simplicity, only edges related to the toy check-in sequence sample $p_0 \rightarrow p_1 \rightarrow p_2$ are labeled with weights, whereas for others, only direction and edge type are illustrated. Weights of edges need to be normalized within edges of the same type.

- Friend Visited: If a friend visited a POI earlier than the user with friend-based entity conditions satisfied, then the friend node and the POI node can be linked by a friend visited edge, denoted as r_v, directed from the friend node to the POI node. The weight of r_v is defined as the frequency the friend visited the POI in her check-in history.
- Friendship. The friendship relation exists between the user and her friend, denoted as r_f, if friend-based entity conditions can be satisfied by certain check-in(s) of the friend. The edge is directed from the user to the friend. Weight of the edge is defined as

$$\gamma \frac{|F_u \cap F_{f_i}|}{|F_u \cup F_{f_i}|} + (1-\gamma)\frac{|P_u \cap P_{f_i}|}{|P_u \cup P_{f_i}|},$$

  where F_u, P_u and F_{f_i}, P_{f_i} are the friend set and POI set of user u and her friend f_i, respectively, and γ is a manually set parameter for the trade-off between the two factors.

  Friend visited and friendship relations (edges) always co-exist. In other words, if there is a POI, user, and friend satisfying friend-based entity conditions, a friendship edge and a friend visited edge will be added to connect the three entities (nodes).
- Direct Check-in. For a given check-in, if all preceding relations cannot be established, then a direct check-in edge will be built between the user and the POI, denoted as r_d, directed from the user to the POI. It represents an independent check-in. Edge weight is defined as the frequency the user visiting the POI in sequence s.

With the preceding defined entities and relations, we can now construct an SKG for the sequence s by sequentially traversing s, as described in Algorithm 1. For each check-in, we first try to construct the first four relations, and if none of the four relations' conditions could be satisfied, a direct check-in relation will be added to the graph.

An example describing elements in a SKG is demonstrated in Figure 2. Note that all edges are directed, from head entity e_h to tail entity e_t.

There are various advantages for constructing such SKGs. First, there can be multiple edges containing comprehensive information between two nodes since that triples are constructed sequentially in the order of check-ins. Second, information for why a triple is constructed (the relation) and how strong the connection is (the weight) are included in SKGs as well. Moreover, by defining r_t, r_s, r_v, r_f, r_d, null values in the check-in data can be translated into relations of SKGs and is thus expected to provide more insights to user check-in behaviors.

ALGORITHM 1: SKG Construction

Input: Check-in sequence $s = \{c_1, \ldots, c_\lambda\}$ of user u, F, P, Δ_s, Δ_t.
Output: A SKG in the form of ordered triples: $\{t_1, \ldots, t_m\}$.
Initialize SKG set G_s={ };
$p^{prev} \leftarrow$ the POI in check-in c_1;
for c *in* s **do**
  $Size_{G_s} = |G_s|$;
  for r *in* $\{r_t, r_s\}$ **do**
    if *corresponding conditions could be satisfied* **then**
      $G_s \leftarrow ((p^{prev}, r, p^c), weight) \cup G_s$
    end
  end
  if *friend-visited check-in condition could be satisfied* **then**
    find the closest satisfied check-in record from friend f;
    $G_s \leftarrow ((u, r_f, f), weight) \cup G_s$;
    $G_s \leftarrow ((f, r_v, p^c), weight) \cup G_s$
  end
  if $Size_{G_s} == |G_s|$ **then**
    $G_s \leftarrow ((u, r_d, p^c), weight) \cup G_s$
  end
end

3.2 Sequential-Knowledge-Aware Embedding

With SKGs obtained, we now discuss how to learn representation vectors for entities and relations.

GATs [39] are extensions of graph convolutional networks [15], which learn to assign varying importances to a given node's neighbors rather than treat all neighboring nodes equally.

However, in knowledge graphs, entities play different roles depending on the relations they are associated with [25]. Although advanced in correlation modeling, GATs ignore context information in edges. To this end, Nathani et al. [25] proposed an extension of GAT that incorporates relation and neighboring node features with the attention mechanism. This idea forms the basis of the embedding network we propose for the sequential-knowledge-aware recommender.

3.2.1 Intra- and Inter-SKG Memory Modeling. Inspired by Nathani et al. [25], the representation of a triple $t_{ij}^k = (e_i, r_k, e_j)$, denoted as $\mathbf{x}^{ijk}$, can be calculated as linear transformation of the concatenation of its components' representation vectors, i.e.,

$$\mathbf{x}^{ijk} = \boldsymbol{W}_1[\mathbf{h}_i \| \mathbf{h}_j \| \mathbf{g}_k], \tag{1}$$

where $\mathbf{h}_i$ and $\mathbf{h}_j$ are embeddings of e_i and e_j, respectively, and $\mathbf{g}_k$ denotes the representation vector of r_k, $\mathbf{h}_i, \mathbf{h}_j, \mathbf{g}_k \in \mathbb{R}^\kappa$, and $\boldsymbol{W}_1 \in \mathbb{R}^{\kappa \times 3\kappa}$. Thus, with constructed SKGs, which can be treated as series of triples, we take the transformed representation vectors of triples as inputs of the embedding network.

Temporal correlation is an important factor affecting people's check-in behaviors. The order of check-in implies a user's check-in pattern, and more recent items in a sequence may have larger impact on the next item [37]. However, to the best of our knowledge, very few knowledge-aware POI recommendation models take temporal correlations into account. To leverage information hidden in check-in orders, we model intra-SKG (sequence) and inter-SKG (sequence of sequences) memories with GRUs [4].

GRUs have proven to be effective in processing sequential data because of the structure of update gates, denoted as a, and reset gates, denoted as r, which can effectively control information

pipelines. Here we adopt a standard GRU. Formally, for each current evaluated SKG G_i, the embedding update formulas can be written as follows:

$$\begin{aligned}
\mathbf{a}_\tau &= \sigma(\mathbf{W}_a \mathbf{v}_{\tau-1} + \mathbf{U}_a \mathbf{x}_\tau + \mathbf{b}_a), \\
\mathbf{r}_\tau &= \sigma(\mathbf{W}_r \mathbf{v}_{\tau-1} + \mathbf{U}_r \mathbf{x}_\tau + \mathbf{b}_r), \\
\hat{\mathbf{v}}_\tau &= \tanh(\mathbf{W}_v(\mathbf{r}_\tau \odot \mathbf{v}_{\tau-1}) + \mathbf{U}_v \mathbf{x}_\tau + \mathbf{b}_v), \\
\mathbf{v}_\tau &= (1 - \mathbf{a}_\tau) \odot \mathbf{v}_{\tau-1} + \mathbf{a}_\tau \odot \hat{\mathbf{v}}_\tau,
\end{aligned} \tag{2}$$

where $\mathbf{x}_\tau$ is the initial embedding of the τth triple in G_i, $\tau \in [0, |\mathcal{R}|]$, $\mathbf{W}_\epsilon$, $\mathbf{U}_\epsilon$, $\epsilon \in \{a, r, v\}$ are learnable parameters, $\mathbf{b}_\epsilon$, $\epsilon \in \{a, r, v\}$ are bias of the function, $\sigma(\cdot)$ is the sigmoid function, and $\odot$ is the Hadamard product.

As discussed in Section 2, in each training set there are T SKGs, and it is desirable and doable to make use of past experience for embedding learning of the current SKG. To capture long-term correlations between SKGs and eliminate the problem of catastrophic forgetting in continual meta-learning tasks [23], we feed the embedding network with the hidden state embedded in historical graphs. Therefore, the specific description of $\mathbf{v}_\tau$ can be written as follows (note that the notation i has been omit in previous equations, which denotes the current graph (the i-th) we modeled):

$$\mathbf{v}_\tau^i = \begin{cases} \mathbf{v}_{|G_{i-1}|}^{i-1} & i \neq 1, \tau = 1, \\ \mathbf{0}, & i = 1, \tau = 1, \\ \textit{calculated recursively}, & \text{otherwise}, \end{cases} \tag{3}$$

where $\mathbf{v}_{|G_{i-1}|}^{i-1}$ denotes the last hidden state of previous SKG. Then the hidden state v_τ is taken as the updated embedding of x_τ.

3.2.2 Entity Embeddings Update. We propose an attention-based approach with graph neural networks to propagate information from triples to nodes, considering the role (head/tail) the node plays in triples. It is worth mentioning that our approach shares some similarities with Wu et al. [43], where the in- and out-flowing features of a node's neighbors are modeled. However, connections in the work of Wu et al. [43] are between homogeneous nodes, whereas in our problem setting, triple is the minimum operable unit. Therefore, their approach cannot be directly adopted. To fit the natural instinct of SKGs, we adapt the incidence matrix to model the connection between entities and triples. Moreover, in SKGs, an entity can be of different meanings given if it is the head or tail in a triple. For example, a temporal sequential-check-in from POI A, a hot-pot restaurant in Chengdu, China, to POI B, a bubble-tea shop, would be less unusual if inversed. Thus, it is problematic to use a global representation of triples, Wu et al. [43] deal with nodes when propagating information from a node's neighbors. To deal with this problem, we apply a linear transformation layer to expand the representation vector of a triple into head and tail partitions. Mathematically, the approach can be described with the following formula:

$$\mathbf{x}_\tau^{head/tail} = \mathbf{W}_2^{head/tail} \mathbf{x}_\tau^T, \tag{4}$$

where $\mathbf{W}_2^{head}, \mathbf{W}_2^{tail} \in \mathbb{R}^{\kappa \times \kappa}$ are the head- and tail-related triple transformation matrices and the calculated $\mathbf{x}_\tau^{head}$, $\mathbf{x}_\tau^{tail}$ are taken as outgoing and incoming representation vectors of the τth triple, respectively.

Obtaining outgoing and incoming latent vectors of each triple, we get the absolute value of attention of a triple from the following formula:

$$\alpha_\tau^{head/tail} = LeakyReLU(\mathbf{W}_3 \mathbf{x}_\tau^{head/tail}), \tag{5}$$

	head					tail				
	t_0	t_1	t_2	t_3	...	t_0	t_1	t_2	t_3	...
u_0	0.23/6	0.66/6	0.70/6	0	...	0	0	0	0	...
f_0	0	0	0	0.38/2	...	0	0	0.70/1	0	...
p_0	0	0.66/1	0	0	...	0.23/1	0	0	0	...
p_1	0	0	0	0	...	0	0.66/3	0	0	...
p_2	0	0	0	0	...	0	0	0	0.38/3	...
...	...	...	...	...	...	...	...	...	...	...

Fig. 3. The connection matrix of SKG in Figure 2. The connection matrix describes the role that each node plays in all triples. Taking entry (u_0, t_0) as an example, its value 0.23/6 is resulted from the following: (1) u_0 is the head node of triple t_0, and (2) the weight of the corresponding edge is 0.23 and the out-degree of u_0 is 6.

where $\mathbf{W}_3 \in \mathbb{R}^{1\times\kappa}$ is the shared weight matrix. Thus, the absolute graph attention for entity e_i over all triples can be described as

$$\beta'_{i:} = A_{s,i:} \odot \left[\left(\alpha_1^{head}, \ldots \alpha_\tau^{head}, \ldots, \alpha_{|\mathcal{R}|}^{head}\right) || \left(\alpha_1^{tail}, \ldots \alpha_\tau^{tail}, \ldots, \alpha_{|\mathcal{R}|}^{tail}\right)\right], \tag{6}$$

where $A_s \in \mathbb{R}^{|\mathcal{E}|\times 2|\mathcal{R}|}$ is the connection matrix that describes incoming and outgoing neighbors of the unique entity set $\mathcal{E}$ of the evaluated SKG, $|2\mathcal{R}|$ is two times of the number of triples in the SKG, and $A_{s,i:}$ is the specific row corresponding to entity e_i. A_s is defined as the concatenation of two weighted incidence matrices A_s^{head} and A_s^{tail}, which represent connections of outgoing and incoming edges in the SKG, respectively, and are calculated as

$$A_{s,ij}^{head/tail} = w_{ij} \frac{1}{d_i^{+/-}}, \tag{7}$$

where w_{ij} is the weight of edge between entity e_i and e_j, and $d_i^{+/-}$ denotes the out/in-degree of entity e_i. For example, in the running example of Figure 2, A_s can be calculated as shown in Figure 3, where t_0, t_1, t_2, t_3 are the first four generated triples as marked in Figure 2.

Softmax is then applied over $\beta'_{i:}$ to get the relative attention values between a head (tail) entity and its tail (head) neighbors, i.e.,

$$\beta_{ij} = softmax_j(\beta'_{ij}) = \frac{exp(\beta'_{ij})}{\sum_{k=1}^{2|\mathcal{R}|} exp(\beta'_{ik})}. \tag{8}$$

Multi-head attention was first implemented by Velickovic et al. [39] to stabilize the learning process and encapsulate more information of the neighbor. Inspired by other works [25, 39], we also adopt M independent attention layers to calculate the embeddings and then they are concatenated by the following formula:

$$\mathbf{h}'_i = ||_{m=1}^{M} \sigma\left(\sum_{k=1}^{2|\mathcal{R}|} \beta_{ik}^m \mathbf{x}_k^m\right), \tag{9}$$

where || denotes concatenation operation.

Multiple layers can be stacked to propagate information from a node's higher-level neighbors. In the final layer, instead of concatenating output of different layers, we take the average of M layers to get the final embedding.

3.2.3 Edge Embeddings Update. Relations are far less complex than entities. For simplicity, linear transformation is then performed on relation embedding matrix $\mathbf{G}_R$, where

$\mathbf{G}_R = [\mathbf{g}_1||\mathbf{g}_2||...||\mathbf{g}_n]$ and n is the number of relation types, to get further representation of relations in a similar fashion to the work of Nathani et al. [25], i.e.,

$$\mathbf{G}_R^{'} = \mathbf{G}_R\mathbf{W}_R, \tag{10}$$

where $\mathbf{W_R}$ is a parameterized transformation matrix.

3.3 Recommendation Based on Knowledge Inference

Thus far, we have obtained the embeddings of entities and relations. We then make recommendations by scoring potential triples using a link predictor. Given a triple $t = (e_i, r_k, e_j)$, where r_k is one of the possible check-in relations that the head entity e_i can generate, we predict the probability that a user would take a specific POI as her next destination using the state-of-the-art relation inference method ConvE [5]. The scoring function is defined as

$$\hat{y}_{t,next} = f(t) = f(vec(f([\overline{\mathbf{h}_i}; \overline{\mathbf{g}_k}] * \Omega)\mathbf{W}_4) \circ \mathbf{h}_j, \tag{11}$$

where $\overline{\mathbf{h}_i}$ and $\overline{\mathbf{g}_k}$ are 2D reshaping of $\mathbf{h}_i$ and $\mathbf{g}_k$, and * represents convolution operator. According to Dettmers et al. [5], the model first reshapes $\mathbf{h}_i$ and $\mathbf{g}_k \in \mathbb{R}^{\kappa}$ into $\overline{\mathbf{h}_i}, \overline{\mathbf{g}_k} \in \mathbb{R}^{\kappa_w \times \kappa_h}$, where $\kappa_w \times \kappa_h = \kappa$. Then the reshaped matrices are taken as inputs for 2D convolutional layer with filter Ω. The output tensors are then subsequently reshaped into a vector $vec()$ and linearly transformed by $\mathbf{W}_4 \in \mathbb{R}^{|vec| \times \kappa}$. The returned vector of κ dimension is then used to compute the dot product, which is denoted by $\circ$, with tail entity embedding vector $\mathbf{h}_j$.

Parameters can be learned by minimizing the following binary cross-entropy loss:

$$\mathcal{L}_{rec} = -\sum_{i}^{n} y^i_{t,next} log(\sigma(\hat{y}^i_{t,next})) + (1 - y^i_{t,next}) log(1 - \sigma(\hat{y}^i_{t,next})), \tag{12}$$

where $y^i_{t,next}$ is the ground-truth value of whether triple t is valid, which is 1 if t is in the generated triple set of the user's next check-in, 0 otherwise, and $\sigma(\cdot)$ denotes the sigmoid function.

The link predictor serves as a decoder of the encoded embeddings. Besides ConvE [5], there can be other implementations for this module, such as ConvKB [26] and so forth.

3.4 The Meta-Learner

In the scenario of POI recommendation, it usually does not have massive data of users that could be used for training due to check-in sparsity. It is desirable to design a model that could fast learn with limited data. Additionally, people having different sequential check-in patterns may require different parameterized models for next check-in prediction. To this end, in this section, we present a meta-learner adapted from LEO [30], which automatically adjusts model weights of the embedding network of SKR by users instead of making predictions of all users' next check-ins with unified model weights.

As one of the state-of-the-art works on optimization-based meta-learning, LEO [30] proves that latent embedding optimization is beneficial to decouple optimization-based meta-learning techniques from high-dimensional space of model parameters. We extend LEO for the problem of POI recommendation because it is straightforward to feed embeddings of SKR to LEO and LEO can be extended to fit the graph structured of SKGs.

The meta-learning approach is developed as a two-stage process. In the first stage, named the *inner loop*, the meta-learner generates parameterized weights to its subsequent embedding network of SKR and then updates the weights based on the returned loss after training data passing through the initialized embedding network. After several iterations in the inner loop, the optimized SKR is believed to be fitted to the training data. In the subsequent stage, which is called the *outer loop*, we

then use test data to evaluate the generated embedding network and then make predictions, and the returned loss will be used to update parameters of the meta-learner itself.

3.4.1 Generate Parameters—Optimization on the Training Set (the inner loop). Let u represent a specific user, and we encode user-specific latent vector $\mathbf{z_u}$ from embeddings of SKGs as

$$\begin{aligned}\mu_{\mathbf{u}}^{\mathbf{e}}, \sigma_{\mathbf{u}}^{\mathbf{e}} = \frac{1}{T} \sum_{G_k \in \mathcal{D}_u^{tr}} \frac{1}{\sum_{i \in \mathcal{E}_{G_k}} d_i} \sum_{i \in \mathcal{E}_{G_k}} \sum_{j \in \mathcal{N}_i} \sum_{k \in \mathcal{R}_{ij}} g_{\phi_r}(g_{\phi_e}(\mathbf{h_i}), g_{\phi_e}(\mathbf{h_j}), g_{\phi_e}(\mathbf{g_k})),\\ \mathbf{z_u} \sim q(\mathbf{z_u}|\mathcal{D}_u^{tr}) = \mathcal{N}(\mu_u^e, diag(\sigma_u^{e2})),\end{aligned} \tag{13}$$

where T is the number of SKGs in $\mathcal{D}_u^{tr}$, d_i denotes the degree of node i, $\mathcal{N}_i$ denotes the neighborhood of entity e_i, $\mathcal{R}_i$ denotes the set of relations connecting entity e_i and e_j, and g_{ϕ_e}, g_{ϕ_r} are the encoder network and relation network, respectively, that are used to obtain data-dependent multivariate Gaussian distribution with a diagonal covariance $\mathcal{N} \sim (\mu_{\mathbf{u}}^{\mathbf{e}}, \sigma_{\mathbf{u}}^{\mathbf{e}})$, where e denotes encoder. $\mathbf{z_u}$ can then be sampled from the obtained distribution. As a common practice, we adopt a re-parameterization trick to deal with the undifferentiable problem of sampling operation.

As the embedding network has multiple parameters to learn, it is unreasonable to define a single decoder and expect that it could fit all parameters. Thus, in a similar fashion to Pan et al. [28], after obtaining the latent vector $\mathbf{z_u}$ of a user, we then generate Gaussian distribution of parameters via parameter-specific decoders. For a given parameter θ, the decoding formulas are

$$\begin{aligned}\mu_{\mathbf{n},\theta}^{\mathbf{d}}, \sigma_{\mathbf{n},\theta}^{\mathbf{d}} = g_{\phi_d}^{\theta}(\mathbf{z}_n),\\ \omega_n^{\theta} \sim p(\omega^{\theta}|\mathbf{z}_n) = \mathcal{N}\left(\mu_{n,\theta}^{d}, diag({\sigma_{n,\theta}^{d}}^2)\right),\end{aligned} \tag{14}$$

where $g_{\phi_d}^{\theta}$ is the decoder, ω_n^{θ} represents sampled user-specific weights of parameter θ, e.g., $\mathbf{W}_a$ in Equation (2), and d denotes the decoder.

The decoded parameters are taken as user-specific temporal model weights of SKR, using which we update triples, node, and edge embeddings according to Equations (1) through (10). We define the inner loop training objective function using hinge loss:

$$L_u^{tr}(g_{\Theta_u}) = \sum_{t_{ij} \in \mathcal{T}} \sum_{t'_{ij} \in \mathcal{T}'} max\{l_{t_{ij}} - l_{t'_{ij}} + \zeta, 0\}, \tag{15}$$

where $l_{t_{ij}}$ is the L1-norm dissimilarity measure given by $l_{t_{ij}} = ||\mathbf{h}_i + \mathbf{g}_k - \mathbf{h}_j||$, $\zeta > 0$ is a marginal hyper-parameter, $\mathcal{T}$ is the valid triple set, and $\mathcal{T}'$ is a set of randomly sampled invalid triples. Then, latent representation $\mathbf{z}_u$ can be updated with

$$\mathbf{z_u}' = \mathbf{z_u} - \eta \bigtriangledown_{\mathbf{z_u}} L_u^{tr}(g_{\Theta_u}), \tag{16}$$

where η is the learning rate. We adopt the architectures of encoder, decoder, and relation networks similar to Rusu et al. [30]. The inner loop runs multiple times and finally reaches a new embedding network of SKR, g'_{Θ_u}.

3.4.2 Meta-Learning Strategy—Optimization on the Test Set (the outer loop). Parameters of the meta-learner hold through the optimization process in the inner loop. We now discuss how to evaluate the parameter generating performance of the meta-learner. According to LEO [30], with a goal of fast adapting to new data, given a test set D^{tr} of user u, the optimization strategy of meta-learner can be defined as follows:

$$\min_{\phi_e, \phi_r, \phi_d^{\theta_1 \ldots, \theta_m}} L_u^{ts}(g'_{\Theta_u}) + \gamma_1 D_{KL}(q(\mathbf{z_u}||D^{tr}), p(\mathbf{z_u})) + \gamma_2 ||stopgrad(\mathbf{z_u}') - \mathbf{z_u}||_2^2 + R, \tag{17}$$

ALGORITHM 2: Framework of Meta-SKR

Input: SKGs of users $\mathcal{G}_u$, $u \in U$; hyper-parameters T, γ_1, γ_2, η, ζ.
Output: Top-k recommended POIs.
Randomly initialize ϕ_e, ϕ_r, ϕ_d, hidden state of GRU;
Compute connection matrix A;
while *not converged* **do**
  Sample a batch of users from U;
  for *user u in batch* **do**
    Sample task instances ($\mathcal{D}_u^{tr}$, $\mathcal{D}_u^{ts}$) from $\mathcal{M}_u^{tr}$;
    Encode $\mathcal{D}_u^{tr}$ to $\mathbf{z}_u$;
    Decode $\mathbf{z}_u$ to Θ_u;
    $\mathbf{z}_u^{'} \leftarrow \mathbf{z}_u$, $\Theta_u^{'} \leftarrow \Theta_u$;
    /*Compute user-specific parameters based on $\mathcal{D}_u^{tr}$*/;
    for *number of adaption iterations* **do**
      Feed the SKR with $\mathcal{D}_u^{tr}$;
      Update node and relation embeddings with Equation (9) and Equation (10);
      Compute training loss $L_u^{tr}(g_{\Theta_u})$;
      Update Θ_u performing gradient steps w.r.t. Equation (16) to obtain $\mathbf{z}_u^{'}$;
      Decode $\mathbf{z}_u^{'}$ to $\Theta_u^{'}$;
    end
    /*Make predictions based on $\mathcal{D}_u^{ts}$*/;
    Feed the network with $\mathcal{D}_u^{ts}$ and compute test loss L_u^{ts};
    Make predictions with $\mathcal{D}_u^{ts}$;
    Compute recommendation loss $\mathcal{L}_{rec}$;
  end
  Update ϕ_e, ϕ_r, ϕ_d and the link prediction module by gradient descend;
  /* We ignore the validation process in this pseudo code */;
end
for *user u in U* **do**
  Sample task instances ($\mathcal{D}_u^{tr}$, $\mathcal{D}_u^{ts}$) from $\mathcal{M}^{test}$;
  Encode $\mathcal{D}_u^{tr}$ to $\mathbf{z}_u$ and then decode $\mathbf{z}_u$ to Θ_u;
  Make predictions with $\mathcal{D}_u^{ts}$;
end

where $p(\mathbf{z_u}) = \mathcal{N}(0, \mathcal{I})$, $D_{KL}(\cdot)$ denotes the KL divergence, $stopgrad(\cdot)$ is the function used to stop gradient computation, γ_1 and γ_2 are term importance hyper-parameters, and R is a regularizer that calculated as L_2 regularization of all the weights in the encoder model with the layer-wise orthogonality constraint on decoder network weights. Similar to other works [12, 30], we use the second term to regularize the latent space and encourage the generative model to learn a disentangled embedding, and we use the third term to encourage the encoder and relation net to output a parameter initialization that is close to the adapted code, which are beneficial to simplify the adaptation procedure [30].

The overall algorithm for Meta-SKR is shown in Algorithm 2.

4 EXPERIMENTS AND RESULTS

All the algorithms are implemented on an Intel Xeon CPU E5-2650 v4 @ 2.20 GHz, two GeForce GTX 1080 GPUs, and two TITAN XP GPUs.

Table 2. Statistics of Dataset

Attribute	Gowalla	Weeplaces	Yelp
Check-ins (#)	1,922,604	1,819,632	473,439
POIs (#)	29,976	37,802	18,990
Users (#)	22,566	10,287	6,563

4.1 Datasets

We use three real public LBSN datasets collected from Gowalla[1] and Weeplaces[1] [22] and Yelp[2] [21], respectively. The Gowalla dataset spans from January 1, 2011, to May 31, 2011. It contains user profiles, user friendship, location profiles, and users' check-in history. Each check-in is associated with the user ID, POI ID, and check-in time. Weeplaces is an application mapping an individual's shared locations on other LBSNs such as Foursquare. In Weeplaces, users can login using their LBSN accounts and connect with their friends in the same LBSN who are also using this application [22]. The Weeplaces dataset, which contains user friendship and users' check-in history, spans from March 1, 2010, to November 30, 2010. For each check-in, the user ID, POI ID, check-in time, latitude, longitude, city, and category are recorded. The Yelp dataset spans over several metropolitan areas in 5 years: January 1, 2010, to August 1, 2015. Each POI is tagged with a category ID and location. Users' check-in history and social relations are also provided.

The statistics of preprocessed datasets are described in Table 2.

4.2 Implementation Details

Data preprocessing. We divide each dataset into 6:2:2 for training, validation, and test. We filter out users with less than 30 check-ins. The pre-trained embeddings of entities and relations are obtained via TransE [2], with random initialization and embedding size of $\kappa = 100$.

Hyper-parameter settings. The embedding layer described in Section 3.2 is set as 1. We optimize all models with the Adam optimizer, where the batch size is fixed at 64, and all learning rates are initialized as 1×10^{-3} and weight decay of 1×10^{-5}. The hyper-parameters Δ_τ and Δ_d in Section 3.1 are set as 2 h and 3 km separately for graph construction. Trade-off parameter γ is set as 0.5. We report the results with default parameter values $T = 4$ and $\lambda = 5$ to fit the size of our datasets. In all experiments, except when studying the effect of r in Section 4.5.2, sampling stride r is set as $r = \lambda$. Unless otherwise stated, we choose the best settings for other hyper-parameters regarding each dataset using random grid search.

4.3 Evaluation Metrics

Accuracy@k is commonly adopted to measure next POI recommendation results [13, 45]. Given a set of test users U, Accuracy@k can be described as

$$Accuracy@k = \frac{\sum_{i=0}^{|U|} Avg\_hit_i@k}{|U|}$$

where $Avg\_hit_i@k$ is the average of $hit@k$, which is calculated as

$$Avg\_hit_i@k = \frac{|Topk_i \cap Ground - truth_i|}{min\{k, |Ground - truth_i|\}},$$

[1] https://www.yongliu.org/datasets/index.html.
[2] https://www.yelp.com/dataset.

where $Topk_i$ is the set of top-k prediction results of user u_i and $Ground-truth_i$ is the ground-truth triple set of the user, which is generated in a similar strategy as described in Section 3.1, which means there might be multiple ones.

4.4 Performance Comparison

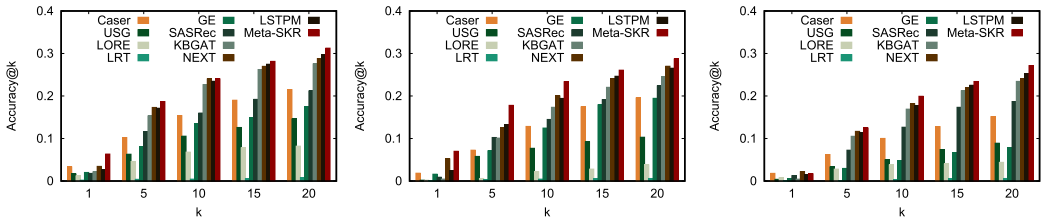
To evaluate the performance of our model, we first compare our Meta-SKR with the following models:

- *LRT* [10]: LRT is a variant of the matrix factorization model, which makes POI recommendation based on both user features and users' temporal check-in patterns..
- *USG* [49]: USG adopts the **Collaborative Filtering (CF)** framework for POI recommendation, which incorporates the geographical influence, social influence, and user preference.
- *LORE* [52]: As a hybrid model, LORE employs additive Markov chain), CF, and kernel density estimation to exploit the sequential, social, and geographical influence between users and POIs.
- *GE* [45]: GE is a graph-based embedding POI recommendation model that integrally captures sequential effect, geographical influence, temporal cyclic effect, and semantic effect into a shared low-dimensional space.
- *SASRec* [13]: SASRec is a self-attention based sequential model that allows to capture long-term semantics (like an RNN) but, using an attention mechanism, makes its predictions based on relatively few actions.
- *Caser* [38]: The convolutional sequence embedding recommendation model (Caser) proposes a convolutional network structure for capturing both general user preferences and sequential behavior patterns.
- *KBGAT* [25]: The recent proposed KBGAT learns graph attention based embeddings that cater to relation prediction on knowledge graphs.
- *NEXT* [53]: NEXT incorporates meta-data information and temporal contexts, i.e., time interval and visiting time, to predict a user's next move.
- *LSTPM* [34]: The method LSTPM models a user's long-term preference with a nonlocal network and short-term preference with a geo-dilated RNN.

4.4.1 Recommendation Effectiveness. We first present comparison results on the three datasets in terms of Accuracy@k in Figure 4, where $k \in \{1, 5, 10, 15, 20\}$, as a greater value of k is usually ignored in top-k recommendation tasks [45].

As presented in Figure 4, it can be observed that our proposed Meta-SKR outperforms other baselines in terms of Accuracy@k on all three datasets and nearly all cases. Some other interesting observations are the following:

(1) Meta-SKR achieves 84%, 34% relative improvements of Accuracy@1 on Gowalla, Weeplaces and 5.2%, 8.2%, 7.3% relative improvements of Accuracy@20 on Gowalla, Weeplaces, and Yelp, respectively, over the best results of the baselines, which demonstrates the superiority of our model.
(2) Another observation is that the evaluated models gain their best performances on Gowalla but perform the worst on Yelp. An explanation could be that although Gowalla and Yelp is of similar density in terms of the average number of check-ins per user, data collected from Yelp is of a much longer time span, which might lead to weak user behavior patterns.
(3) Although GE uses the same type of information as Meta-SKR, it performs significantly worse. This might be due to their different embedding strategies. To be more specific, different from Meta-SKR that jointly models context information, GE encodes sequential



(a) On Gowalla (b) On Weeplaces (c) On Yelp

Fig. 4. Recommendation effectiveness comparisons.

effect, geographical effect, and temporal effect separately in different graphs (models) and then optimizes the embedding model together by a fusing objective function.

(4) LRT performs significantly worse than other algorithms. This might be because LRT only considers temporal information and the explicit check-in behaviors but ignores geographical and other side information that is important as well in location recommendation.

4.4.2 Impact of Data Sparsity. In this section, we study the effect of data sparsity. To simulate datasets with different sparsity, we randomly reduce the training data by a ratio of 5%, 10%, 15%, and 20%. For limited space, we only report the recommendation accuracy when 20% data is reduced on the Gowalla and Yelp datasets, as listed in Table 3 and Table 4.

As expected, when more data is reduced, the recommendation accuracy of all methods drops. However, Meta-SKR still outperforms all the other methods in most of the cases. It could be observed that when evaluating Accuracy@1 on the Yelp dataset, Caser performs slightly better than Meta-SKR. However, when it comes to relative accuracy drop, Meta-SKR is 4.1% less than Caser. It also worth mentioning that the relative advantage of Meta-SKR over other methods becomes more significant. For example, compared with the state-of-the-art baseline model LSTPM, Meta-SKR achieves 317.6%, 35% relative improvements of Accuracy@1 and 18.2%, 16.3% relative improvements of Accuracy@20 on the two reduced datasets, which indicates that Meta-SKR is more robust on sparse data. This probabliy results from two reasons. First, accuracy on baseline models drops faster. On account of their data-driven nature, their advanced performances heavily rely on large datasets. Once the dataset becomes sparser and smaller, the model might fail to draw a complete picture of users' behavior patterns. Second, with the implementation of a meta-learning strategy, Meta-SKR is trained to leverage user-specific features from limited data, and thus it is less sensitive to the reduction of data size.

As for the experiments on other reduction values (5%, 10%, 15%) and on the Weeplaces dataset, we observe similar trends.

4.5 Study of Meta-SKR

In the preceding section, we compared our Meta-SKR model with other approaches. To further study the benefits of hyper-parameters and components of Meta-SKR, we design three groups of experiments to evaluate how they affect recommendation accuracy on datasets of Gowalla and Yelp.

4.5.1 Effect of Check-In Sequence Length λ. Parameter sequence length λ as defined in Definition 1 of the triple generator directly influences what knowledge graphs consist of, and thus eventually affects Accuracy@k. Four comparison methods are designed to evaluate the impact of λ on recommendation accuracy: Meta-SKR-λ-3, Meta-SKR-λ-5, Meta-SKR-λ-7, Meta-SKR-λ-9, and

Table 3. Effect of Data Sparsity on the Gowalla Dataset (−20%)

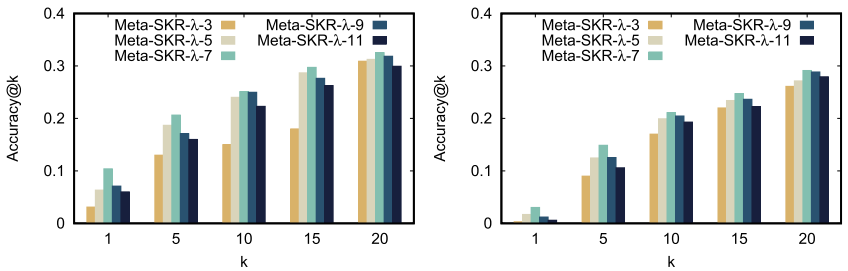
Acc \ M / k	LRT		USG		LORE		GE		Caser	
	LRT	LRT-	USG	USG-	LORE	LORE-	GE	GE-	Caser	Caser-
1	0.0011	0.0005	0.0165	0.0125	0.0119	0.0081	0.0201	0.0134	0.0330	0.0149
5	0.0030	0.0005	0.0630	0.0535	0.0462	0.0211	0.0810	0.0622	0.1023	0.0558
10	0.0035	0.0010	0.1065	0.0799	0.0681	0.0342	0.1346	0.1101	0.1539	0.0887
15	0.0041	0.0011	0.1261	0.1095	0.0782	0.0460	0.1493	0.1354	0.1901	0.1187
20	0.0072	0.0020	0.1461	0.1250	0.0819	0.0527	0.1745	0.1563	0.2151	0.1421
Acc \ M / k	SASRec		KBGAT		NEXT		LSTPM		Meta-SKR	
	SASRec	SASRec-	KBGAT	KBGAT-	NEXT	NEXT-	LSTPM	LSTPM-	Meta-SKR	Meta-SKR-
1	0.0179	0.0112	0.0214	0.0089	0.0344	0.0101	0.0263	0.0131	**0.0634**	**0.0547**
5	0.1163	0.0674	0.1531	0.0976	0.1731	0.1109	0.1712	0.1129	**0.1869**	**0.1255**
10	0.1599	0.1012	0.2268	0.1426	0.2400	0.1402	0.2346	0.1671	**0.2403**	**0.1831**
15	0.1921	0.1259	0.2623	0.1670	0.2698	0.1837	0.2747	0.1972	**0.2815**	**0.2239**
20	0.2119	0.1459	0.2754	0.1812	0.2882	0.2120	0.2973	0.2250	**0.3126**	**0.2660**

Note: A hyphen (-) denotes the method with reduced data.

Table 4. Effect of Data Sparsity on the Yelp Dataset (−20%)

Acc \ M / k	LRT		USG		LORE		GE		Caser	
	LRT	LRT-	USG	USG-	LORE	LORE-	GE	GE-	Caser	Caser-
1	0.0002	/	0.0040	0.0021	0.0088	0.0014	0.0059	0.0032	0.0180	**0.0164**
5	0.0012	0.0005	0.0333	0.0180	0.0285	0.0042	0.0297	0.0101	0.0624	0.0470
10	0.0027	0.0017	0.0506	0.0289	0.0380	0.0141	0.0474	0.0229	0.1002	0.0728
15	0.0054	0.0029	0.0739	0.0375	0.0411	0.0223	0.0665	0.0400	0.1280	0.0972
20	0.0060	0.0043	0.0892	0.0433	0.0447	0.0284	0.0782	0.0683	0.1514	0.0984
Acc \ M / k	SASRec		KBGAT		NEXT		LSTPM		Meta-SKR	
	SASRec	SASRec-	KBGAT	KBGAT-	NEXT	NEXT-	LSTPM	LSTPM-	Meta-SKR	Meta-SKR-
1	0.0126	0.0040	0.0044	0.0031	**0.0221**	0.0092	0.0152	0.0120	0.0170	0.0162
5	0.0725	0.0597	0.1053	0.0400	0.1169	0.0418	0.1142	0.0857	**0.1247**	**0.0979**
10	0.1266	0.0735	0.1691	0.0790	0.1818	0.1043	0.1776	0.1248	**0.1993**	**0.1536**
15	0.1735	0.1017	0.2121	0.1116	0.2197	0.1345	0.2253	0.1723	**0.2342**	**0.1992**
20	0.1871	0.1273	0.2344	0.1380	0.2411	0.1376	0.2522	0.1892	**0.2716**	**0.2201**

Note: A hyphen (-) denotes the method with reduced data.

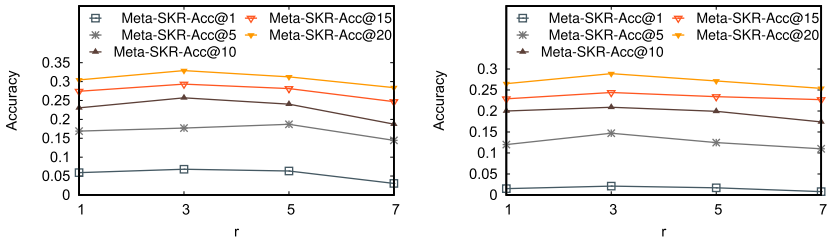


(a) On Gowalla (b) On Yelp

Fig. 5. Effect of check-in sequence length λ.

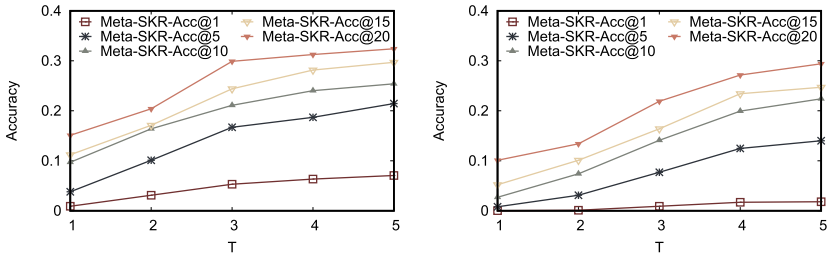
Meta-SKR-λ-11, which represents the Meta-SKR model with $\lambda = \{3, 5, 7, 9, 11\}$, respectively, and 5 is chosen to be the default value of λ in our Meta-SKR.

The results are depicted in Figure 5. It can be observed that both a super short sequence and a super long sequence contribute negatively to model performance. From the figures, we can tell that a relatively longer sequence benefits recommendation (see Meta-SKR-λ-5 and Meta-SKR-λ-7).



(a) On Gowalla (b) On Yelp

Fig. 6. Effect of sampling stride r.



(a) On Gowalla (b) On Yelp

Fig. 7. Effect of the SKG number in D^{tr}.

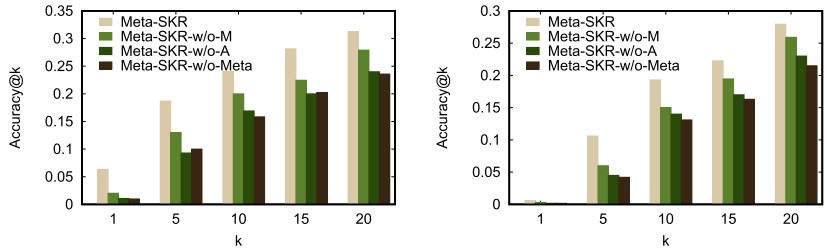
4.5.2 Effect of Sampling Stride r. Sampling stride r not only influences the size of training data but also has an effect on the granularity of the modeled user behavior patterns. With λ fixed as 5, we increase r from 1 to 7 with step size 2.

Some observations of Figure 6 are as follows. A middle-sized sampling stride benefits recommendation the most. This could result from that for the current λ, which is set as 5, $r = 3$ and $r = 5$ cover all source data and the observed user behavior patterns are dynamic and coherent enough for the model to learn. Stride 7 gives the lowest accuracy probably because not all data is observed by the model, and thus the prediction of a user's next move is based on incoherent patterns. Although the model with stride 1 performs slightly worse than $r = 5$, it is worth mentioning that generating a set of SKGs with $r = 1$ is rather time consuming, given the large number of previous SKGs that need to be considered for the construction of the later ones.

4.5.3 Effect of the SKG Number in D^{tr}. To study the impact of length of the SKG sequence in the meta-training process, we increase the number of SKGs of D^{tr} from 1 to 5 by step size 1, where 4 is taken as the default value. The experiment results are shown in Figure 7.

Overall, recommendation accuracy tends to raise as the number of SKGs increases. It can also be observed that with the SKG number increasing, recommendation accuracy first increases significantly, then shows signs of slow growth and finally becomes relatively stable when the SKG number is around 4. This might result from the following: first, when more SKGs are fed into the model, more data can be utilized to obtain a comprehensive user profile; moreover, long-term preference can also be extracted with more SKGs. This trend indicates that making use of inter-SKG information is likely to be beneficial for good performance.

4.5.4 Effect of Model Components. The major results of the ablation study regarding Meta-SKR with different components on Gowalla and Yelp datasets are shown in Figure 8. We design three variants of Meta-SKR: Meta-SKR-w/o-M, Meta-SKR-w/o-A, and Meta-SKR-w/o-Meta.



(a) On Gowalla (b) On Yelp

Fig. 8. Effect of model components.

Meta-SKR-w/o-M is a variant related to sequential memory extracting. To investigate the effect of retrieving memory from previous sequences, Meta-SKR-w/o-M reduces the hidden state initialization function (see Equation (3)) by replacing the initialization when $i \neq 1, \tau = 1$ with vector $\mathbf{0}$ as well. It can be seen from Figure 8 that the performance of Meta-SKR-w/o-M is significantly worse than Meta-SKR. Without inheriting status from prior SKGs, only intra-sequence sequential information can be extracted for temporal correlation modeling, which inevitably leads to poor performance.

Meta-SKR-w/o-A represents a variant of the attention-based knowledge propagation part. Instead of building a head/tail-specific triple connection matrix, Meta-SKR-w/o-A treats head and tail related triples symmetrically. Specifically, if an entity e_i is of the triple t_j, regardless of whether it is the head or tail, e_i is considered connected with t_j. Direction information is eliminated by this strategy. Accordingly, the recommendation accuracy suffers an even more significant drop.

Meta-SKR-w/o-Meta simply uses the sequential-knowledge-aware recommender defined in Sections 3.1 through 3.3 to recommend next POIs, with shared model weight for all users. Comparing Meta-SKR-w/o-Meta with baseline models in Figure 4, in most cases, the variant still outperforms most baseline models. Moreover, comparing Meta-SKR-w/o-Meta with other methods in Figure 8, models with parameter generating, i.e., Meta-SKR, Meta-SKR-w/o-M, and Meta-SKR-w/o-A, perform better in most cases, among which the performance improvement of Meta-SKR and Meta-SKR-w/o-M is more significant. These results indicate first that the proposed sequential-knowledge-aware recommendation network is effective and second that parameter adaptation significantly contributes to the model's performance.

5 RELATED WORK

5.1 POI Recommendation

POI recommendation can be deemed as one of the main enablers of LBSNs. As a special branch of context-aware recommendation [29], POI recommendation methods take (combinations of) geographical, temporal, social, sequential, and categorical information into account. According to the utilized techniques, they can be categorized into CF based [10, 11, 46], matrix factorization based [22, 49], Poisson factor models, hybrid models [8, 52], embedding-based models [29, 53], and deep models [7, 18, 34, 44].

The recent advances in knowledge graph embedding and completion [5, 25] have greatly helped with the problems of properly incorporating context-aware information and data sparsity [21, 29, 41, 45, 51]. For example, Qian et al. [29] propose a spatio-temporal context-aware and translation-based recommender framework (STA) to model the third-order relationship among users, POIs, and spatio-temporal contexts. Xie et al. [45] model sequential effect, geographical influence, temporal cyclic effect, and semantic effect into four corresponding relational graphs. Although these

methods have provided strong performance, most of them require rich types of side information and rely on massive training data, which are inaccessible in most real-world check-in datasets. Moreover, since the interactions are constructed in a statistic setting, they might not well model users' sequential check-in orders. Based on embedding-based models, our model takes the idea of meta-learning and sequential-knowledge-aware graph embeddings to solve the preceding problems.

5.2 Meta-Learning

Meta-learning studies how to distill the prior knowledge from past experiences and enable fast adaptation to novel tasks with only a limited amount of samples [23]. The main research in meta-learning include (1) learning a widely generalizable initialization or leveraging an optimizer as the meta-learner to adjust model weights [3, 9, 30], (2) learning an embedding function that can embed inputs into shared spaces where there are distance metrics serving as the meta-learner [14, 23, 33, 36, 40], and (3) learning with memories of a previous task obtained from RNNs [31].

Meta-learning has been proven to be promising in recommender systems, especially in the task of cold-start recommendation. Several works [1, 6, 16, 42] are relevant to our study. Compared to other works [1, 16, 42], which use MAML [9] for fast user adaptation, our work is adapted from LEO (latent embedding optimization) [30] since it well suits our problem where embeddings can be easily obtained and taken as inputs of the meta-learner and the original LEO model can be extended to adapt to our graph structured data. To make full use of sequential information, we also propose a memory transmission strategy to utilize inter-SKG interactions during meta-learning. From the perspective of problems, although a great number of algorithms have been proposed with meta-learning in recommender systems, e.g., some works [1, 16, 42] introduce meta-learning to the task of general recommendations and another work [6] is designed for E-commerce sequential recommendation, our work is the first to study the problem of meta-learning through knowledge-aware graph neural networks in the context of POI recommendation, which requires providing spatio-temporal-aware recommendations based on users and and items' spatio-temporal context.

6 CONCLUSION

In this work, we studied the problem of utilizing user sequential check-in and spatio-temporal-social information in SKGs for next POI recommendation. We proposed a meta-learning framework named *Meta-SKR*, which contains four modules. Our solution starts with graph construction where triples of SKGs are built, followed by the embedding network with a meta-learning paradigm, and at last we provide the next POI recommendation by scoring potential triples in the link prediction module. Extensive experiments based on real check-in datasets are conducted, and the favorable results confirm the superiority of our framework with sparse training data.

REFERENCES

[1] Homanga Bharadhwaj. 2019. Meta-learning for user cold-start recommendation. In *Proceedings of the 2019 International Joint Conference on Neural Networks*. IEEE, Los Alamitos, CA, 1–8.

[2] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Proceedings of the 26th International Conference on Neural Information Processing Systems*. 2787–2795.

[3] Junkun Chen, Xipeng Qiu, Pengfei Liu, and Xuanjing Huang. 2018. Meta multi-task learning for sequence modeling. In *Proceedings of AAAI 2018*.

[4] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of EMNLP 2014*.

[5] Tim Dettmers, Minervini Pasquale, Stenetorp Pontus, and Sebastian Riedel. 2018. Convolutional 2D knowledge graph embeddings. In *Proceedings of AAAI 2018*.

[6] Zhengxiao Du, Xiaowei Wang, Hongxia Yang, Jingren Zhou, and Jie Tang. 2019. Sequential scenario-specific meta learner for online recommendation. In *Proceedings of KDD 2019*.
[7] 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 *Proceedings of WWW 2018*.
[8] Shanshan Feng, Lucas Vinh Tran, Gao Cong, Lisi Chen, Jing Li, and Fan Li. 2020. HME: A hyperbolic metric embedding approach for next-POI recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1429–1438.
[9] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of ICML 2017*.
[10] Huiji Gao, Jiliang Tang, Xia Hu, and Huan Liu. 2013. Exploring temporal effects for location recommendation on location-based social networks. In *Proceedings of RecSys 2013*. 93–100.
[11] Xiangnan He, Zhankui He, Jingkuan Song, Zhenguang Liu, Yu-Gang Jiang, and Tat-Seng Chua. 2018. NAIS: Neural attentive item similarity model for recommendation. *IEEE Transactions on Knowledge and Data Engineering* 30, 12 (2018), 2354–2366.
[12] Irina Higgins, Loïc Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew M Botvinick, Shakir Mohamed, and Alexander Lerchner. 2017. beta-VAE: Learning basic visual concepts with a constrained variational framework. In *Proceedings of ICLR 2017*.
[13] Wangcheng Kang and Julian J. McAuley. 2018. Self-attentive sequential recommendation. arXiv:1808.09781.
[14] Jongmin Kim, Taesup Kim, Sungwoong Kim, and Chang D. Yoo. 2019. Edge-labeling graph neural network for few-shot learning. In *Proceedings of CVPR 2019*.
[15] Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *Proceedings of ICLR 2017*.
[16] Hoyeop Lee, Jinbae Im, Seongwon Jang, Hyunsouk Cho, and Sehee Chung. 2019. MeLU: Meta-learned user preference estimator for cold-start recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1073–1082.
[17] 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 *Proceedings of the 29th ACM International Conference on Information and Knowledge Management*. 845–854.
[18] Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. 2016. Predicting the next location: A recurrent model with spatial and temporal contexts. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*. 194–200.
[19] Tongcun Liu, Jianxin Liao, Zhigen Wu, Yulong Wang, and Jingyu Wang. 2020. Exploiting geographical-temporal awareness attention for next point-of-interest recommendation. *Neurocomputing* 400 (2020), 227–237.
[20] Yanchi Liu, Chuanren Liu, Bin Liu, Meng Qu, and Hui Xiong. 2016. Unified point-of-interest recommendation with temporal interval assessment. In *Proceedings of KDD 2016*.
[21] Yiding Liu, Tuan-Anh Nguyen Pham, Gao Cong, and Quan Yuan. 2017. An experimental evaluation of point-of-interest recommendation in location-based social networks. *Proceedings of the VLDB Endowment* 10, 10 (2017), 1010–1021.
[22] Yong Liu, Wei Wei, Aixin Sun, and Chunyan Miao. 2014. Exploiting geographical neighborhood characteristics for location recommendation. In *Proceedings of CIKM 2014*.
[23] Yadan Luo, Zi Huang, Zheng Zhang, Ziwei Wang, Mahsa Baktashmotlagh, and Yang Yang. 2020. Learning from the past: Continual meta-learning via Bayesian graph modeling. In *Proceedings of AAAI 2020*.
[24] Weizhi Ma, Min Zhang, Yue Cao, Woojeong Jin, Chenyang Wang, Yiqun Liu, Shaoping Ma, and Xiang Ren. 2019. Jointly learning explainable rules for recommendation with knowledge graph. In *Proceedings of WWW 2019*.
[25] Deepak Nathani, Jatin Chauhan, Charu Sharma, and Manohar Kaul. 2019. Learning attention-based embeddings for relation prediction in knowledge graphs. In *Proceedings of ACL 2019*.
[26] Dai Quoc Nguyen, Tu Dinh Nguyen, Dat Quoc Nguyen, and Dinh Phung. 2018. A novel embedding model for knowledge base completion based on convolutional neural network. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. 327–333.
[27] Enrico Palumbo, Giuseppe Rizzo, and Raphaël Troncy. 2017. Entity2Rec: Learning user-item relatedness from knowledge graphs for top-N item recommendation. In *Proceedings of RecSys 2017*.
[28] Zheyi Pan, Yuxuan Liang, Weifeng Wang, Yong Yu, Yu Zheng, and Junbo Zhang. 2019. Urban traffic prediction from spatio-temporal data using deep meta learning. In *Proceedings of KDD 2019*.
[29] Tieyun Qian, Bei Liu, Quoc Viet Hung Nguyen, and Hongzhi Yin. 2019. Spatiotemporal representation learning for translation-based POI recommendation. *ACM Transactions on Information Systems* 37, 2 (Jan. 2019), Article 18, 24 pages.
[30] Andrei A. Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. 2019. Meta-learning with latent embedding optimization. In *Proceedings of ICLR 2019*.

[31] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy P. Lillicrap. 2016. One-shot learning with memory-augmented neural networks. arXiv:1605.06065.
[32] C. Shi, B. Hu, W. X. Zhao, and P. S. Yu. 2019. Heterogeneous information network embedding for recommendation. *IEEE Transactions on Knowledge and Data Engineering* 31, 2 (2019), 357–370.
[33] Jake Snell, Kevin Swersky, and Richard S. Zemel. 2017. Prototypical networks for few-shot learning. In *Proceedings of NIPS 2017*.
[34] 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 *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 214–221.
[35] Rui Sun, Xuezhi Cao, Yan Zhao, Junchen Wan, Kun Zhou, Fuzheng Zhang, Zhongyuan Wang, and Kai Zheng. 2020. Multi-modal knowledge graphs for recommender systems. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management*. 1405–1414.
[36] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip H. S. Torr, and Timothy M. Hospedales. 2018. Learning to compare: Relation network for few-shot learning. In *Proceedings of CVPR 2018*.
[37] Jiaxi Tang and Ke Wang. 2018. Personalized top-N sequential recommendation via convolutional sequence embedding. In *Proceedings of WSDM 2018*.
[38] Jiaxi Tang and Ke Wang. 2018. Personalized top-N sequential recommendation via convolutional sequence embedding. In *Proceedings of WSDM 2018*.
[39] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *Proceedings of ICLR 2018*.
[40] Oriol Vinyals, Charles Blundell, Timothy P. Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. 2017. Matching networks for one shot learning. In *Proceedings of CVPR 2017*.
[41] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. KGAT: Knowledge graph attention network for recommendation. In *Proceedings of KDD 2019*.
[42] Tianxin Wei, Ziwei Wu, Ruirui Li, Ziniu Hu, Fuli Feng, Xiangnan He, Yizhou Sun, and Wei Wang. 2020. Fast adaptation for cold-start collaborative filtering with meta-learning. In *Proceedings of ICDM 2020*.
[43] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2018. Session-based recommendation with graph neural networks. In *Proceedings of AAAI 2019*.
[44] Yuxia Wu, Ke Li, Guoshuai Zhao, and Xueming Qian. 2019. Long- and short-term preference learning for next POI recommendation. In *Proceedings of CIKM 2019*.
[45] Min Xie, Hongzhi Yin, Hao Wang, Fanjiang Xu, Weitong Chen, and Sen Wang. 2016. Learning graph-based POI embedding for location-based recommendation. In *Proceedings of the 25th ACM International Conference on Information and Knowledge Management*. 15–24.
[46] Feng Xue, Xiangnan He, Xiang Wang, Jiandong Xu, Kai Liu, and Richang Hong. 2019. Deep item-based collaborative filtering for top-n recommendation. *ACM Transactions on Information Systems* 37, 3 (2019), 1–25.
[47] Huaxiu Yao, Yiding Liu, Ying Wei, Xianfeng Tang, and Zhenhui Li. 2019. Learning from multiple cities: A meta-learning approach for spatial-temporal prediction. In *Proceedings of WWW 2019*.
[48] Zijun Yao. 2018. Exploiting human mobility patterns for point-of-interest recommendation. In *Proceedings of WSDM 2018*.
[49] Mao Ye, Peifeng Yin, Wang-Chien Lee, and Dik Lee. 2011. Exploiting geographical influence for collaborative point-of-interest recommendation. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 325–334.
[50] 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 networks In *Proceedings of IJCAI 2018*.
[51] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative knowledge base embedding for recommender systems. In *Proceedings of KDD 2016*.
[52] Jia-Dong Zhang, Chi-Yin Chow, and Yanhua Li. 2014. LORE: Exploiting sequential influence for location recommendations. In *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. 103–112.
[53] Zhiqian Zhang, Chenliang Li, Zhiyong Wu, Aixin Sun, Dengpan Ye, and Xiangyang Luo. 2020. NEXT: A neural network framework for next POI recommendation. *Frontiers of Computer Science* 14, 2 (2020), 314–333.
[54] Kangzhi Zhao, Yong Zhang, Hongzhi Yin, Jin Wang, Kai Zheng, Xiaofang Zhou, and Chunxiao Xing. 2020. Discovering subsequence patterns for next POI recommendation. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence*. 3216–3222.

Received October 2020; revised March 2021; accepted April 2021