DAN-SNR: A Deep Attentive Network for Social-aware Next Point-of-interest Recommendation

LIWEI HUANG, School of Computer Science and Technology, Beijing Institute of Technology, China and Beijing Institute of Remote Sensing, China
YUTAO MA, School of Computer Science, Wuhan University, Wuhan, China
YANBO LIU, Beijing Institute of Remote Sensing, China
KEQING HE, School of Computer Science, Wuhan University, Wuhan, China

Next (or successive) point-of-interest (POI) recommendation, which aims to predict where users are likely to go next, has recently emerged as a new research focus of POI recommendation. Most of the previous studies on next POI recommendation attempted to incorporate the spatiotemporal information and sequential patterns of user check-ins into recommendation models to predict the target user's next move. However, few of the next POI recommendation approaches utilized the social influence of each user's friends. In this study, we discuss a new topic of next POI recommendation and present a deep attentive network for social-aware next POI recommendation called DAN-SNR. In particular, the DAN-SNR makes use of the self-attention mechanism instead of the architecture of recurrent neural networks to model sequential influence and social influence in a unified manner. Moreover, we design and implement two parallel channels to capture short-term user preference and long-term user preference as well as social influence, respectively. By leveraging multi-head self-attention, the DAN-SNR can model long-range dependencies between any two historical check-ins efficiently and weigh their contributions to the next destination adaptively. We also carried out a comprehensive evaluation using large-scale real-world datasets collected from two popular location-based social networks, namely, Gowalla and Brightkite. Experimental results indicate that the DAN-SNR outperforms seven competitive baseline approaches regarding recommendation performance and is highly efficient among six neural-network-based methods, four of which utilize the attention mechanism.

CCS Concepts: • **Information systems → Social recommendation**;

Additional Key Words and Phrases: Next point-of-interest recommendation, location-based service, social influence, self-attention, embedding

ACM Reference format:
Liwei Huang, Yutao Ma, Yanbo Liu, and Keqing He. 2020. DAN-SNR: A Deep Attentive Network for Social-aware Next Point-of-interest Recommendation. *ACM Trans. Internet Technol.* 21, 1, Article 2 (December 2020), 27 pages.
https://doi.org/10.1145/3430504

This work was partially supported by the National Key Research and Development Program of China (No. 2018YFB1003801) and the National Science Foundation of China (Nos. 61972292, 62006023, and 61672387).
Authors' addresses: L. Huang, School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, and Beijing Institute of Remote Sensing, Beijing 100854, China; email: dr_huanglw@163.com; Y. Ma (corresponding author), K. He, School of Computer Science, Wuhan University, Wuhan 430072, China; emails: {ytma, hekeqing}@whu.edu.cn; Y. Liu, Beijing Institute of Remote Sensing, Beijing 100854, China; email: liuyanbonudt@163.com.

1533-5399/2020/12-ART2 $15.00
https://doi.org/10.1145/3430504

1 INTRODUCTION

1.1 Background

Location-based social networks (LBSNs), such as Foursquare,[1] Loopt,[2] and Yelp,[3] have become very popular among young people in the past decade. Users utilize mobile devices and location-based services (LBSs) to search out points of interest (POIs) in LBSNs, post their check-ins and reviews for POIs, and share their life experiences in the real world. Millions of users in LBSNs have generated a massive amount of check-in data, which provides an excellent opportunity to recommend possible POIs for users accurately. Owing to the significance and business value of POI recommendation, the research on POI recommendation has attracted attention from academia and industry. In general, the existing work of POI recommendation attempts to predict target users' preferences based on their historical check-ins and recommend a set of unvisited urban POIs to them [1–6].

The past few years have witnessed a fast-growing demand for human mobility prediction in urban tourism, product advertising, and other application fields [7]. *Next POI recommendation* has recently emerged as a new research focus of POI recommendation, and its research objective is to predict where target users are likely to go next [8]. To deal with this challenging task, researchers have proposed a few approaches to learn users' movement sequences based on their historical check-ins and train personalized POI recommendation models according to the most recent checked-in locations of users [8–11]. Due to the complexity and diversity of human mobility, it is difficult for most of the previous studies on next POI recommendation to achieve satisfying recommendation results. Therefore, some recent studies [12–15] attempted to leverage more available information of user check-ins, such as spatial information and temporal information, to train better recommendation models with deep learning and other new techniques.

1.2 Motivation

As we know, LBSNs are a specific type of online social network that allows users to interact with whomever they like in a virtual world. Intuitively, a user's decision on where to go next may be affected by the user's friends (or called social influence) in LBSNs. Figure 1 illustrates the concept of social-aware next POI recommendation. Given user1's trajectory of check-ins (sorted in chronological order) at a hotel ($T_{t\text{-}5}$), a gym ($T_{t\text{-}3}$), and a hotel ($T_{t\text{-}1}$), which POIs can be recommended to the user at a given time point (T_t)? Suppose a next POI recommendation system does not consider the social influence of the user's friends. In that case, it may recommend a restaurant or a museum with equal probability according to the sequential patterns mined from the other four users (or called sequential influence). As shown in Figure 1, "restaurant" and "museum" appear equally likely to go after "hotel." After leaving a hotel, user2 and user4 like to visit a museum, while user3 and user5 prefer to go to a restaurant. Instead, a social-aware next POI recommendation system, which takes into account user1's friendships with user2 and user4 (see the two red lines between them in Figure 1), is more likely to recommend a museum to the user at the time point T_t.

A few previous works of POI recommendation have leveraged social influence to improve the quality of recommendations. They calculated the similarities between users regarding friendship and then designed recommendation models using collaborative filtering (CF) techniques [1, 16, 17]. Moreover, some previous studies [18, 19] employed network representation techniques to model users' friendships. However, the above studies showed small improvements in POI recommendation performance, because it is tough for them to accurately identify the behavioral correlations

[1] https://foursquare.com/.
[2] http://www.looptmix.com/.
[3] http://www.yelp.com/.

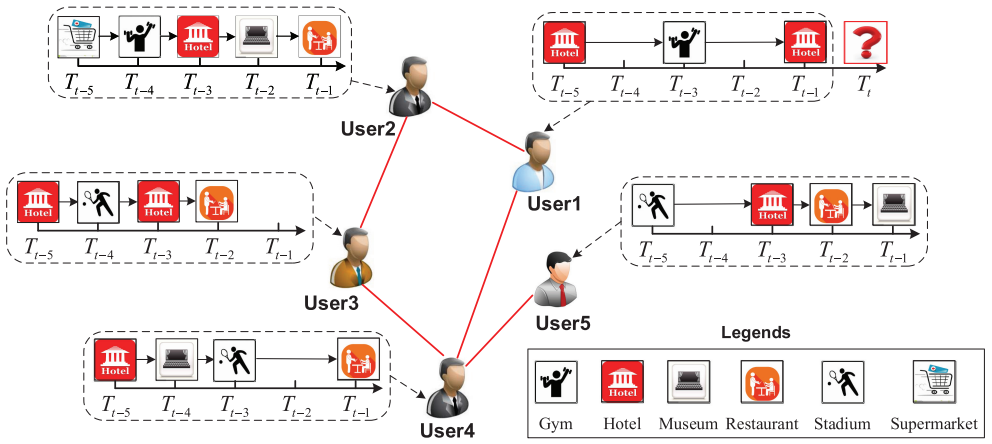

Fig. 1. An example of social-aware next POI recommendation.

between similar users, not just social relationships. In the next POI recommendation scenarios, each user's social influence is dynamic and context-dependent, which causes great difficulty for social-aware next POI recommendation. Therefore, the first problem to solve is the representation of dynamic social influence.

Because the next POI recommendation problem is, in essence, a sequence prediction problem, recurrent neural networks (RNNs) have recently been applied to modeling sequential influence for next POI recommendation [11, 15, 19]. RNNs treat each trajectory as a sequence of user check-ins and recursively compose each check-in behavior with its previous hidden state. Recurrent connections make RNNs applicable to sequence prediction tasks with arbitrary length. However, RNNs also have two disadvantages. First, due to the recursive nature of RNNs, it is hard for them to parallelize [20], which makes both offline training and online prediction very time-consuming. Second, fixed-size encoding vectors generated by RNN encoders sometimes do not represent both short and long sequences well [21]. Therefore, the second problem to solve is the efficient and effective joint learning of social influence and sequential influence in the spatiotemporal contexts.

1.3 Contribution

For the first problem mentioned above, we attempt to model (dynamic) social influence by capturing the behavioral correlations between each user and the user's friends in LBSNs. Considering the success of attention in natural language processing (NLP), we mine the global dependencies between a user's check-ins and his/her friends' check-ins using the self-attention mechanism [20]. It thus enables our approach to model the context-dependent social influence for next POI recommendation. Like previous studies [18, 19], we also use a graph embedding method to learn network-based user embeddings for each user in a shared latent space.

For the second problem mentioned above, we attempt to find a solution from two aspects. On the one hand, we model sequential influence by capturing the spatiotemporal correlations between each user's check-ins. Due to the disadvantages of RNNs, we also leverage the self-attention mechanism to mine the long-range dependencies between a user's check-ins, which can facilitate the joint learning process of social influence and sequential influence. On the other hand, we make use of self-attention to model sequential influence and social influence simultaneously. Moreover, we represent each check-in behavior by embedding user check-ins' spatiotemporal contexts into a compact vector, thus enabling our approach to better model user preference by considering geographical and temporal influences.

In brief, the technical contributions of this work are three-fold.

(1) We first introduce the self-attention mechanism to model dynamic and context-dependent social influence for next POI recommendation. More specifically, we utilize the self-attention mechanism to capture the behavioral correlations between each user's check-ins and his/her friends' check-ins. Also, we learn social-network-based user embeddings by using a graph embedding method.
(2) We present a **d**eep **a**ttentive **n**etwork for **s**ocial-aware **n**ext POI **r**ecommendation (DAN-SNR), which utilizes the self-attention mechanism instead of the architecture of RNNs as a basic framework to model sequential influence and social influence simultaneously. The advantages of the self-attention mechanism can facilitate the parallelization of modeling to speed up the whole joint learning process.
(3) To model user preference better, we take geographical and temporal influences into account and embed user check-ins' spatiotemporal contexts into a compact vector. More specifically, we construct a location-to-location (L2L) graph based on the distance between POIs to model two-dimensional geographical influence in LBSNs.

To demonstrate the DAN-SNR's effectiveness, we furthermore evaluated it with two real-world LBSN datasets, i.e., Gowalla[4] and Brightkite.[5] Experiment results indicate that the DAN-SNR performs better than seven competing baseline approaches of next POI recommendation regarding commonly used evaluation metrics.

1.4 Organization

The remainder of this article is organized as follows: Section 2 reviews the work related to next POI recommendation and social-aware POI recommendation. Section 3 formulates the problem to solve in this study. Section 4 details the proposed deep attentive network for next POI recommendation. Section 5 presents experiment setups and results. Section 6 discusses some issues related to experimental results. Section 7 concludes this article and provides an overview of our future work.

2 RELATED WORK

2.1 Next POI Recommendation

2.1.1 Machine Learning–based Approach. As mentioned in Section 1.2, the next POI recommendation problem is a sequence prediction problem, which needs to mine and utilize users' sequential influence to predict where target users are likely to go next. Many early studies on next POI recommendation employed the Markov chain model to learn sequential influence. For example, Cheng et al. [8] designed a matrix factorization model based on a factoring personalized Markov chain and localized regions, namely, FPMC-LR, to recommend a successive POI for target users. Zhang et al. [22] proposed an approach called LORE, which uses an additive Markov chain to predict the sequential probabilities on a location-location transition graph. Also, LORE incorporates sequential influence, geographical influence, and social influence into a unified recommendation framework. Ye et al. [23] proposed a framework using a mixed hidden Markov model to mine the dependencies between POI categories of user check-ins. The framework can significantly reduce prediction space and precisely express the semantic meaning of user activities. Since the Markov chain model can model latent check-in behavior patterns, it is still used by a few subsequent studies. For example, Li et al. [24] recently proposed a personalized pattern distribution model (PPDM) for both

[4] https://en.wikipedia.org/wiki/Gowalla.
[5] https://brightkite.com/.

the next and next new POI recommendation tasks. This model integrates a personalized Markov chain with contextual features, e.g., time of day, day of the week, and POI category. However, some recent works on human mobility have revealed that individuals' movement behavior is not precisely a stochastic process [25], making it hard to meet the Markov chain model's underlying assumption.

Matrix factorization has been widely used in recommender systems. In addition to the Markov chain model, it is another commonly used technique to mine sequential patterns of users. For example, Feng et al. [9] proposed a personalized ranking metric embedding (PRME) method to model the user-POI distance and the POI-POI distance in two different hidden spaces, respectively. To further improve the recommendation performance, they developed a PRME-G model that fuses sequential information, individual preference, and geographical influence. Zhao et al. [10] proposed a spatial-temporal latent ranking (STELLAR) method for next POI recommendation, which models the interactions between users and POIs in the fine-grained temporal contexts using a ranking-based pair-wise tensor factorization framework. Liu et al. [26] developed a "Where and When to gO" (WWO) recommender system that recommends possible locations to target users at a specific time point. In particular, the system uses a unified tensor factorization framework to model both static user preference and dynamic sequential influence. He et al. [27] also investigated the personalized next POI recommendation problem. They designed a unified tensor-based prediction model that combines the observed sequential patterns and latent behavior preference for each user. However, these approaches built based on matrix factorization often suffer from the cold-start problem.

2.1.2 Deep Learning–based Approach. Deep learning has recently achieved great success in NLP and computer vision. Some recent works of next POI recommendation began to use RNNs and their variants, such as long short-term memory (LSTM) and gated recurrent unit (GRU), to model sequential influence and temporal dynamics. For example, Liu et al. [11] extended the architecture of RNNs and proposed a spatial-temporal recurrent neural network called ST-RNN to predict the next location. Due to different sequential patterns in mobile paths, Yang et al. [19] employed the RNN and GRU models to represent the short-term and long-term check-in contexts, respectively. They proposed a neural network model to characterize social network structure and user trajectory jointly. Li et al. [28] proposed a temporal and multi-level context attention model named TMCA. This model was built based on an LSTM-based encoder-decoder framework. The proposed model can learn the spatial-temporal representations of historical check-ins and integrate embedding-based contextual factors in a unified manner. Wu et al. [29] designed a long-term and short-term preference learning (LSPL) model. To capture sequential patterns and user preference better, they trained two LSTM networks for location-based and category-based sequences. Similarly, Zhao et al. [13] proposed a spatiotemporal gated network (STGN) by improving an LSTM network. They designed spatiotemporal gates that can capture the spatiotemporal relationships between successive check-ins.

Besides, the attention mechanism [30] has recently been introduced to the architecture of RNNs for next POI recommendation. For example, Huang et al. [15] designed an attention-based spatiotemporal LSTM network named ATST-LSTM, which can capture the most critical piece of a user's check-in sequence to predict the next POI. Feng et al. [31] developed an attentional recurrent network called DeepMove for mobility prediction from lengthy and sparse user trajectories. In particular, they utilized the periodicity nature of human mobility to augment the recurrent neural network. Gao et al. [32] proposed a variation-attention-based next (VANext) POI prediction model to overcome the sparsity problem in user check-ins, using historical mobility attention. Generally speaking, the combination of the attention mechanism with the architecture of RNNs did improve

the performance of next POI recommendation approaches based on RNNs. However, the above approaches based on attention and RNNs have the principal disadvantage of high complexity in time.

2.2 Social-aware POI Recommendation

There is a saying that goes: "Birds of a feather flock together." Inspired by the intuition that friends in LBSNs are more likely to have general preferences and similar behavior patterns, the information of social ties (or relationships) has been leveraged to improve the prediction quality of location-based recommender systems [4, 6, 33]. Previous studies on social-aware POI recommendation usually calculated the similarities between users regarding social relationships (more specifically, friendship) and fused them into the user-based CF approach [3, 5, 16, 17, 34, 35]. For example, Cheng et al. [1] incorporated user similarity and geographical influence into a matrix factorization model. Ying et al. [36] also incorporated user similarity into a random walk approach referred to as Urban POI-Walk. Similarly, Kefalas et al. [37] proposed an algorithm called Random Walk with Restart on Heterogeneous Spatio-Temporal (RWR-HST), which can adequately capture the user-user similarity and the user-location relevance. Besides, Kefalas et al. [38] extended the user-based CF approach by leveraging the spatial influence of check-in history and the social influence of user reviews, which were often used as side information in many previous studies.

Inspired by the word2vec technique, network representation has become very popular in social networks in recent years. Therefore, some recent studies employed the network embedding method to capture the social influence of friends [18, 19]. However, there is little research on next POI recommendation that has considered social influence. Compared with social-aware POI recommendation, the social influence in the next POI recommendation scenarios is dynamic and context-dependent in LSBNs. Hence, in this study, we need to model better the behavioral correlations between each user and his/her friends.

3 PROBLEM FORMULATION

Table 1 presents the notations used in this article.

Definition 1 (LBSN). An LBSN is a bipartite graph with two independent sets of vertices, i.e., users and POIs. We can conceptually build three basic graphs within an LBSN, including a user-user graph, a user-POI graph, and a POI-POI graph [6].

Definition 2 (POI). In an LSBN, a POI is a spatial item associated with a geographical location, e.g., a restaurant or a cinema.

Definition 3 (Check-in). A check-in is a behavior represented by a quintuple $c_t^u = (u, v_t^u, l_t^u, t, p)$, indicating that user u visited POI v_t^u on location l_t^u at time point t. Here, p is the position index of c_t^u in the user's trajectory (see Definition 5), and l_t^u represents the latitude and longitude of a POI visited by user u at time point t.

Definition 4 (Check-in Sequence). A user's check-in sequence is a collection of all the check-ins of the user, denoted by $C_u = \{c_{t_i}^u | 1 \le i \le T\}$ where $[t_1,t_T]$ indicates the duration of the user's historical check-ins. For simplicity, historical check-ins of all users are denoted by $C^U = \{C_{u_j} | 1 \le j \le |U|\}$.

Definition 5 (Trajectory). A user's trajectory is a set of consecutive check-ins in a session, denoted by $S_{t_k}^u = \{c_{t_{k-M+1}}^u, c_{t_{k-M+2}}^u, \ldots, c_{t_k}^u\}$, where M is the length of the trajectory. $S_{t_k}^u$ is a partially ordered subset of C_u, i.e., $C_u = \cup_k S_{t_k}^u$.

Table 1. Notations Used in This Article

Symbol	Description
u, U	a user and a set of users
$N(u)$	a set of direct (or one-hop) friends of user u in an LBSN
$u' \in N(u) \cup \{u\}$	an element of the set composed of user u and his/her direct friends
$c_t^u = (u, v_t^u, l_t^u, t, p)$	a check-in: user u visits POI v_t^u on location l_t^u at time point t (position p)
C_u	a set of check-ins performed by user u
$S_{t_k}^u$	a check-in trajectory of user u
C^U	a set of historical check-ins of all users
$\mathbf{u}, \mathbf{v}_t^u, \mathbf{l}_t^u, \mathbf{t}, \mathbf{p}$	embeddings of user u, POI v_t^u, location l_t^u, time t, and position p
$\mathbf{c}_{t_i}^{u,s}$	the latent representation of check-in $c_{t_i}^u$ generated by the feature embedding layer in the short-term channel
$\mathbf{c}_{t_i}^{u,l}$	the latent representation of $c_{t_i}^u$ generated by the feature embedding layer in the long-term and social channel
$\{\mathbf{W}\}$	a set of parameter matrices in the DAN-SNR
$\mathbf{c}_{t_i}^{u,s,k}$, $\mathbf{c}_{t_i}^{u',l,k}$	the check-in representations generated by the kth nonlinear layers in the two channels
$\alpha_{ij}^{u,s,k,h}$, $\alpha_{ij}^{u',l,k,h}$	the attention weights in the hth head of the kth self-attention layers in the two channels
$f(\cdot)$	the attention function
$\mathbf{g}_{t_i}^{u,s,k}$, $\mathbf{g}_{t_i}^{u',l,k}$	the check-in representations generated by the kth self-attention layers in the two channels
$\mathbf{c}_v^{u,s}$, $\mathbf{c}_v^{u,l}$	the latent representations of check-in that user u visits POI v at the next moment in the two channels
$\alpha_{iv}^{u,s}$, $\alpha_{iv}^{u',l}$	the attention weights in the vanilla attention layers in the two channels
$\mathbf{h}_{t_N}^{u,s}$, $\mathbf{h}_{t_N}^{u,l}$	the latent representations of user u generated by the vanilla attention layers in the two channels
$o_{t_{N+1},v}^u$	the probability that user u visits POI v at the next moment

Then, we present the problem definition of social-aware next POI recommendation as follows:

Given check-in sequences of all users in an LBSN G at a time point t_N, the goal of social-aware next POI recommendation is to predict the most likely location v that user u will visit at the next moment t_{N+1}, i.e., $\max o_{t_{N+1},v}^u$.

4 DEEP ATTENTIVE NETWORK FOR SOCIAL-AWARE NEXT POI RECOMMENDATION

4.1 Overall Framework

We assume that each user's check-in behavior is affected by both personal preference and friendship. Every user has their long-term and short-term preferences. As for social influence, we consider the behavioral correlations between each user and the user's friends. In this study, we design two parallel channels, namely, a short-term channel (STC) and a long-term and social channel (LTSC), to simultaneously model personal preference and social influence. More specifically, the goal of the STC is to learn short-term preference. It takes each user's current trajectory as an input, which represents the user's ongoing sequential influence. Besides, the goal of the LTSC is to learn social influence and long-term preference simultaneously. The LTSC takes all historical check-ins

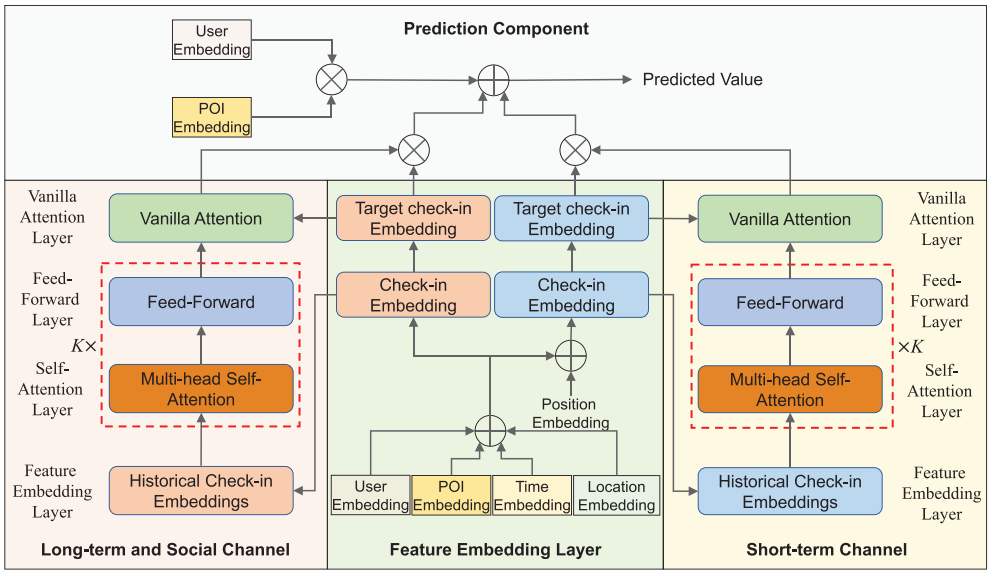

Fig. 2. The architecture of the DAN-SNR.

of each user and his/her friends in an LBSN as an input. In particular, we use all historical check-ins previous to the current time point to capture a user's long-term preference.

Because self-attention with time encoding is efficient in both the training and prediction phases, it can be an appropriate replacement for complex RNN structures in sequential behavior encoding [20]. Thus, the two channels of the DAN-SNR also utilize the self-attention mechanism (more specifically, multi-head self-attention) to obtain a representation of each input check-in. In each of the two channels, we perform vanilla attention between the check-in representations and the candidate POI vectors to select valuable check-ins that have more influence on a user's next-step behavior. The final description of the user is then obtained by combining the outputs of the two channels. At last, the DAN-SNR generates the user's preference score for the target POI. As shown in Figure 2, the DAN-SNR's overall framework has several building blocks, including a feature embedding layer, K identical nonlinear layers, each of which contains a self-attention layer and a feed-forward layer, a vanilla attention layer, and a prediction component. In the following subsections, we will present a bottom-up introduction to the DAN-SNR layer by layer.

4.2 Feature Embedding Layer

According to the definition of check-ins, in this study, such behavior has five types of features: user information (more specifically, friendship), POI information, spatial information, temporal information, and position information (in the whole trajectory). We leverage the five types of information in the form of embedding to obtain each user's latent representation. Next, we will introduce different embedding techniques used in this study.

4.2.1 User Embedding. Graph embedding (also known as network embedding) has recently been used in many essential tasks on graphs, such as classification and link prediction. Because an LBSN is a bipartite graph in this study, we adopt a graph embedding method, i.e., node2vec [39], to model user information from friendship. The graph embedding method encodes each node in an LBSN into a low dimensional vector and maintains the structure information of the LBSN. For user u, the graph embedding method outputs a feature vector $\mathbf{u}$. By learning the network-based user

embedding for each user in a shared latent space, the DAN-SNR can model social relationships for next POI recommendation.

4.2.2 POI Embedding. To generate POI embeddings, we first create a matrix $\mathbf{V} \in \mathbb{R}^{|V| \times \boldsymbol{d}}$ for POIs. Here, V is the set of POIs, and d is the dimension of latent variables. For POI v_t^u in check-in c_t^u, we perform a direct lookup on the matrix and obtain the corresponding POI embedding $\mathbf{v}_t^u$. More specifically, the POI ID is set to the value we want to look up, and the matrix elements are sorted in ascending order.

4.2.3 Time Embedding. Human mobility is affected by circadian rhythms, habits and customs, and other factors [40]. The next POI recommendation task is thus time-dependent. In other words, temporal information is critical to analyzing individual check-in behavior. However, it is difficult to learn a proper embedding directly from the continuous-time nature of check-ins using embedding concatenation or addition [21]. In this study, we use a temporal encoding method proposed by Zhou et al. [21]. First, we slice the elapsed time, w.r.t. the ranking time, into intervals whose length grows exponentially. For example, we can map the time in the range $[0, 1), [1, 2), [2, 4), \ldots, [2^k, 2^{k+1})$ to a categorial feature of $0, 1, 2, \ldots, k + 1$. Different groups of check-in behavior may have different granularities of time slicing. Second, we perform a direct lookup on the categorial time feature and obtain the time embedding $\mathbf{t}$ of time point t.

4.2.4 Location Embedding. Previous studies indicate that modeling the geographical influence of user check-ins is essential to POI recommendation in LBSNs. Assuming that strong spatial correlations exist between successive check-ins in short intervals [10], a few researchers attempted to model geographical influence in terms of the geographic distance from users' current locations [8–11]. However, these studies have some limitations, e.g., they were confined to the one-dimensional geographical influence of locations in a trajectory. Hence, in this study, we attempt to characterize two-dimensional geographical influence via an L2L graph that represents the proximity between POIs regarding geographic distance. An L2L graph is, in essence, a weighted undirected graph, where a vertex represents a POI, a link denotes the spatial correlation between POIs, and the weight of a link indicates geographic distance.

After constructing an L2L graph, we apply the graph embedding method, node2vec [39], on the L2L graph to encode each location into a low dimensional vector. For location l_t^u, node2vec outputs an embedded feature vector $\mathbf{l}_t^u$.

4.2.5 Position Embedding. As shown in Figure 2, the DAN-SNR does not contain any recurrence or convolution. To better model sequential influence, we use the timing signal approach proposed by Vaswani et al. [20]. Compared with traditional methods for learned and fixed positional encodings, the timing signal approach does not introduce additional parameters. Considering the relative positions of POIs in a trajectory, the position embedding $\mathbf{p}$ of position p can be formulated using the Sine and Cosine functions of different frequencies as follows:

$$\mathbf{p}_{(2i)} = \sin(p/10000^{2i/d}) \tag{1}$$

$$\mathbf{p}_{(2i+1)} = \cos(p/10000^{2i/d}) \tag{2}$$

where i is the dimension, and d is the dimension of latent variables. Each dimension of the position embedding corresponds to a sinusoid. For each embedding of relative positions, Equation (1) and Equation (2) are used for even and odd dimensions, respectively. The wavelengths form a geometric progression that ranges from 2π to $10000 \times 2\pi$, making it easy for our approach to extrapolate the results when encountering trajectory lengths longer than a fixed value.

4.2.6 Concatenation of Different Embeddings. After calculating the embeddings of users, POIs, locations, time, and positions, we then carry out a concatenation operation on them to obtain the hidden representation of each check-in. When modeling short-term user preference, sequential check-in patterns play an essential role in predicting the target user's next POI. Thus, we consider the position embedding in the STC. Instead, the position embedding is not used in the LTSC, because we take into account all the historical check-ins of each user and his/her friends. In other words, we believe that short-term sequential patterns have a limited impact on a user's long-term preference and social influence.

For user u, we feed feature vectors $\mathbf{u}$, $\mathbf{v}_t^u$, $\mathbf{l}_t^u$, $\mathbf{t}$, and $\mathbf{p}$ into a d-dimensional fully connected layer, which outputs the latent representation of each check-in $\mathbf{c}_t^{u,s}$ (see Equation (3)) in the STC and $\mathbf{c}_t^{u,l}$ (see Equation (4)) in the LTSC.

$$\mathbf{c}_t^{u,s} = \text{sigmoid}(\mathbf{W}_u\mathbf{u} + \mathbf{W}_v\mathbf{v}_t^u + \mathbf{W}_l\mathbf{l}_t^u + \mathbf{W}_t\mathbf{t} + \mathbf{W}_p\mathbf{p}) \tag{3}$$

$$\mathbf{c}_t^{u,l} = \text{sigmoid}(\mathbf{W}_u\mathbf{u} + \mathbf{W}_v\mathbf{v}_t^u + \mathbf{W}_l\mathbf{l}_t^u + \mathbf{W}_t\mathbf{t}) \tag{4}$$

where $\mathbf{W}_u \in \mathbb{R}^{d\times d}$, $\mathbf{W}_v \in \mathbb{R}^{d\times d}$, $\mathbf{W}_l \in \mathbb{R}^{d\times d}$, $\mathbf{W}_t \in \mathbb{R}^{d\times d}$, and $\mathbf{W}_p \in \mathbb{R}^{d\times d}$ are transition matrices. Note that the two channels of the DAN-SNR share the same parameters of the concatenation operation.

4.3 Self-attention Layer in Channels

As mentioned in the Introduction section, modeling various types of check-in information in a unified manner is quite challenging for next POI recommendation. For example, check-in behavior has three main temporal properties: periodicity, non-uniformness, and consecutiveness [4]. Location information of check-ins also has three unique features: hierarchical data, measurable distance, and sequential ordering [6]. Besides, modeling spatiotemporal characteristics of check-ins often relies on different prior assumptions. For example, the spatial distribution of a user's visited locations follows a specific distribution, e.g., the power law. A user's check-ins are periodic in one day or one week. To address the above problems, in this study, the DAN-SNR uses the self-attention mechanism to model the interactions between user check-ins regardless of what type of check-in information is involved, which can capture social, sequential, temporal, and spatial influence in a unified way. Moreover, the self-attention mechanism can automatically measure the degree of behavioral correlations between check-ins and adjust the attention weights accordingly to predict the next POI. Thus, the DAN-SNR can work without any prior assumptions.

As shown in Figure 2, the self-attention layer's primary goal is to capture two types of behavioral correlations between check-ins. In the STC, the self-attention layer's outputs stand for the representative sequence of check-ins that considers the impact of each user's short-term preference. In the LTSC, the self-attention layer's outputs indicate the representative series of check-in behavior that takes into account each user's long-term preference and his/her friends' actions. In this work, we leverage a similar multi-head self-attention structure proposed by Vaswani et al. [20] for the machine translation task [41], with some customized settings. We can utilize the multi-head self-attention to calculate the behavioral correlations between all users' check-ins in theory. However, this will cause a tremendous amount of computation. Therefore, we model social influence by capturing only the behavioral correlations between check-ins of direct (or one-hop) friends in an LBSN.

4.3.1 Self-attention in the STC. Given the current trajectory of user u $S_{t_N}^u = \{c_{t_{N-M+1}}^{u,s}, c_{t_{N-M+2}}^{u,s}, \ldots, c_{t_N}^{u,s}\}$ in the STC, let $\mathbf{C}_{t_N}^{u,s,k} = [\mathbf{c}_{t_{N-M+1}}^{u,s,k}, \mathbf{c}_{t_{N-M+2}}^{u,s,k}, \ldots, \mathbf{c}_{t_N}^{u,s,k}] \in \mathbb{R}^{d\times M}$ be a matrix that consists of latent feature vectors generated by the kth nonlinear layer in the STC. Here, $\mathbf{C}_{t_N}^{u,s}$ represents a matrix that consists of hidden feature vectors generated by the feature

embedding layer, d is the dimension of hidden variables, and M is the length of the user trajectory. There may be one or more historical trajectories containing previous check-ins before the first one in the current trajectory. We can utilize the multi-head self-attention to generate a new representation of check-ins in the current trajectory, described below.

$$\mathbf{r}_{t_i}^{u,s,k} = \text{concat}(\text{head}_1^{u,s,k,t_i}, \text{head}_2^{u,s,k,t_i}, \ldots, \text{head}_H^{u,s,k,t_i})\mathbf{W}^{s,k} \tag{5}$$

$$\text{head}_h^{u,s,k,t_i} = \sum_{j=N-M+1}^{N} \alpha_{ij}^{u,s,k,h} \mathbf{c}_{t_j}^{u,s,k-1,h} \tag{6}$$

where concat$(\cdot)$ denotes a concatenation operation, H is the number of heads in the multi-head self-attention, $\mathbf{W}^{s,k} \in \mathbb{R}^{d \times d}$ is a parameter matrix, $\mathbf{c}_{t_j}^{u,s,k-1,h}$ is the hidden feature vector of the hth head divided from $\mathbf{c}_{t_j}^{u,s,k-1}$, and $\alpha_{ij}^{u,s,k,h}$ is the weight of attention.

Then, we will introduce the calculation process of the attention weight matrix $\mathbf{A}^{u,s,k,h} = (\alpha_{ij}^{u,s,k,h})$ in detail. For each pair of latent feature vectors of the input to the kth nonlinear layer in the STC, i.e., $(\mathbf{c}_{t_i}^{u,s,k-1,h}, \mathbf{c}_{t_j}^{u,s,k-1,h})$, the attention weight $\alpha_{ij}^{u,s,k,h}$ measures the degree of the former's impact on the latter. More specifically, we calculate this parameter using the following equation:

$$\alpha_{ij}^{u,s,k,h} = \frac{\exp(f(\mathbf{c}_{t_i}^{u,s,k-1,h}, \mathbf{c}_{t_j}^{u,s,k-1,h}))}{\sum_{j=N-M+1}^{N} \exp(f(\mathbf{c}_{t_i}^{u,s,k-1,h}, \mathbf{c}_{t_j}^{u,s,k-1,h}))} \tag{7}$$

where $f(\mathbf{c}_{t_i}^{u,s,k-1,h}, \mathbf{c}_{t_j}^{u,s,k-1,h})$ is an attention function. As mentioned above, we use the dot-product attention as the attention function in this study. It is because Vaswani et al. [20] found that the additive attention outperforms the dot-product attention when d is large. As with Reference [20], we also define the attention function with a scale, described as follows:

$$f\left(\mathbf{c}_{t_i}^{u,s,k-1,h}, \mathbf{c}_{t_j}^{u,s,k-1,h}\right) = \frac{\mathbf{c}_{t_i}^{u,s,k-1,h}(\mathbf{c}_{t_j}^{u,s,k-1,h})^T}{\sqrt{d}} \tag{8}$$

He et al. [42] indicated that the depth of representations is essential to achieve excellent performance in visual recognition tasks. Inspired by their idea, we construct our model with residual learning [42], which has been proven to be very useful for training deep neural networks. In both the two channels, we add a residual connection to each self-attention layer. Following the work of Vaswani et al. [20], we then apply layer normalization [43] after the residual connection to stabilize the activations of deep neural networks. Given an input $\mathbf{c}_{t_i}^{u,s,k-1}$ in the STC, the output $\mathbf{g}_{t_i}^{u,s,k}$ of the kth self-attention layer in this channel is computed by the following equation:

$$\mathbf{g}_{t_i}^{u,s,k} = \text{layer\_norm}(\mathbf{c}_{t_i}^{u,s,k-1} + \mathbf{r}_{t_i}^{u,s,k}) \tag{9}$$

where layer_norm$(\cdot)$ denotes the layer normalization function.

4.3.2 Self-attention in the LTSC. Given user u and the user's direct friends $N(u)$ in the LTSC, we consider all historical check-ins of u and his/her friends at time point t_N, i.e., $L_{t_N}^u = \cup_{t_1}^{t_N} \cup_{u' \in N(u) \cup \{u\}} \{C_{t_i}^{u'}\}$. By using the multi-head self-attention, we can calculate the attention weight $\alpha_{ij}^{u',l,k,h}$ for each pair of check-ins $c_{t_i}^{u'}, c_{t_j}^{u'} \in L_{t_N}^u$ in the LTSC, according to Equation (10) and Equation (11):

$$\alpha_{ij}^{u',l,k,h} = \frac{\exp(f(\mathbf{c}_{t_i}^{u',l,k-1,h}, \mathbf{c}_{t_j}^{u',l,k-1,h}))}{\sum_{j=1}^{N} \exp(f(\mathbf{c}_{t_i}^{u',l,k-1,h}, \mathbf{c}_{t_j}^{u',l,k-1,h}))} \tag{10}$$

$$f\left(\mathbf{c}_{t_i}^{u',l,k-1,h}, \mathbf{c}_{t_j}^{u',l,k-1,h}\right) = \frac{\mathbf{c}_{t_i}^{u',l,k-1,h}(\mathbf{c}_{t_j}^{u',l,k-1,h})^T}{\sqrt{d}} \tag{11}$$

Then, we can generate a new vector representation $\mathbf{r}_{t_i}^{u',l,k}$ of check-in $c_{t_i}^{u'} \in L_{t_N}^{u}$ using the kth self-attention layer in the LTSC, described below:

$$\mathbf{r}_{t_i}^{u',l,k} = \text{concat}(\text{head}_1^{u',l,k,t_i}, \text{head}_2^{u',l,k,t_i}, \ldots, \text{head}_H^{u',l,k,t_i})\mathbf{W}^{l,k} \tag{12}$$

$$\text{head}_h^{u',l,k,t_i} = \sum_{j=1}^{N} \alpha_{ij}^{u',l,k,h} \mathbf{c}_{t_j}^{u',l,k-1,h} \tag{13}$$

where $\mathbf{c}_{t_j}^{u',l,k-1}$ is the output of the $(k-1)$th nonlinear layer, $\mathbf{c}_{t_j}^{u',l,k-1,h}$ is the hidden feature vector of the hth head divided from $\mathbf{c}_{t_i}^{u',l,k-1}$, and $\mathbf{W}^{l,k} \in \mathbb{R}^{d\times d}$ is a parameter matrix.

After applying layer normalization [43] to the residual connection to the self-attention layer in the LTSC, we can obtain the output of this layer, $\mathbf{g}_{t_i}^{u',l,k}$, in a similar form of Equation (9).

4.4 Feed-forward Layer in Channels

The learning capability of neural networks depends on highly flexible nonlinear transformations. Unlike neural networks, the self-attention mechanism encodes an input sequence into an output sequence using weighted sum operations. Thus, it has limited capability to represent latent features. To improve the DAN-SNR's representation capability, we employ a fully connected feed-forward network to deal with the output from each of the self-attention layers in two channels. Each feed-forward layer of the DAN-SNR consists of two transformations and a rectified linear unit (ReLU) activation.

Given an input $\mathbf{g}_{t_i}^{u,s,k}$ generated by the kth self-attention layer in the STC, we can calculate the output $\mathbf{c}_{t_i}^{u,s,k}$ of the feed-forward layer using the following equation:

$$\mathbf{c}_{t_i}^{u,s,k} = \text{relu}(\mathbf{g}_{t_i}^{u,s,k}\mathbf{W}_1^s)\mathbf{W}_2^s \tag{14}$$

where $\text{relu}(\cdot)$ denotes the ReLU activation function, and $\mathbf{W}_1^s, \mathbf{W}_2^s \in \mathbb{R}^{d\times d}$ are two trainable parameter matrices. The linear transformations have the same parameters across different check-in representations in one feed-forward layer, but they vary from layer to layer.

Similarly, we can obtain the output of the kth feed-forward layer in the LTSC, i.e., $\mathbf{c}_{t_i}^{u',l,k}$.

4.5 Vanilla Attention Layer in Channels

By using the K nonlinear layers, we obtain an abstract representation of each check-in behavior, which encodes social influence, sequential influence, spatial influence, and temporal influence simultaneously. Next, we will leverage these check-in representations to predict the target user's next POI. Because not all the historical check-ins of a user and his/her friends have the same effect on the user's next-step behavior, we need to pay attention to those more useful ones. Therefore, we design a vanilla attention layer to capture the correlations between the representations of historical check-ins regarding the next-step movement. By leveraging the attention mechanism, the vanilla attention layer can help select the representative check-ins that characterize user preference and social influence and assign different weights to them in a flexible, efficient way.

4.5.1 Vanilla Attention in the STC. For the vanilla attention layer in the STC, we can obtain the hidden representation of u concerning v, i.e., $\mathbf{h}_{t_N}^{u,s}$, which is calculated by the following operation:

$$\mathbf{h}_{t_N}^{u,s} = \frac{1}{M} \sum_{i=N-M+1}^{N} \alpha_{iv}^{u,s} \, \mathbf{c}_{t_i}^{u,s,K} \tag{15}$$

$$\alpha_{iv}^{u,s} = \text{softmax}(f(\mathbf{c}_{t_i}^{u,s,K}, \mathbf{c}_v^{u,s})) \tag{16}$$

where $\mathbf{c}_v^{u,s}$ is the latent representation of the check-in behavior that u visits v at a specific time point t_{N+1} in the STC. Compared with $\mathbf{c}_v^{u,l}$, $\mathbf{c}_v^{u,s}$ also considers the position embedding in addition to the other four embeddings:

$$\mathbf{c}_v^{u,s} = \text{sigmoid}(\mathbf{W}_u\mathbf{u} + \mathbf{W}_v\mathbf{v}_{t_{N+1}}^u + \mathbf{W}_l\mathbf{l}_{t_{N+1}}^u + \mathbf{W}_t\mathbf{t}_{N+1} + \mathbf{W}_p\mathbf{p}_{N+1}) \tag{17}$$

4.5.2 Vanilla Attention in the LTSC. Given user u, his/her direct friends $N(u)$, and their check-in history before the time point t_N, the output of each check-in representation generated by the last feed-forward layer is $\mathbf{c}_{t_i}^{u',l,K}$, where $u' \in N(u) \cup \{u\}$ and $t_i \in [t_1, t_N]$. For each candidate POI v, we concatenate different embeddings and then employ an attention network and the softmax function to calculate the normalized attention weight, which follows a similar procedure in the self-attention layers:

$$\mathbf{c}_v^{u,l} = \text{sigmoid}(\mathbf{W}_u\mathbf{u} + \mathbf{W}_v\mathbf{v}_{t_{N+1}}^u + \mathbf{W}_l\mathbf{l}_{t_{N+1}}^u + \mathbf{W}_t\mathbf{t}_{N+1}) \tag{18}$$

$$\alpha_{iv}^{u',l} = \text{softmax}\Big(f\Big(\mathbf{c}_{t_i}^{u',l,K}, \mathbf{c}_v^{u,l}\Big)\Big) \tag{19}$$

$$f\Big(\mathbf{c}_{t_i}^{u',l,K}, \mathbf{c}_v^{u,l}\Big) = \frac{\mathbf{c}_{t_i}^{u',l,K}(\mathbf{c}_v^{u,l})^T}{\sqrt{d}} \tag{20}$$

where $\mathbf{c}_v^{u,l}$ is the latent representation of the check-in behavior that u visits v at the next moment t_{N+1} in the LTSC, and $\alpha_{ij}^{u,l}$ is the attention weight for each pair of $\mathbf{c}_{t_i}^{u',l,K}$ and $\mathbf{c}_v^{u,l}$. Once the vanilla attention layer outputs the attention weights, the hidden representation of u concerning v is calculated using the following equation:

$$\mathbf{h}_{t_N}^{u,l} = \frac{1}{N} \sum_{i=1}^{N} \alpha_{iv}^{u',l} \, \mathbf{c}_{t_i}^{u',l,K} \tag{21}$$

4.6 Prediction Component

In this study, a user's preference score is defined as a function of three embeddings, namely, $\mathbf{h}_{t_N}^{u,s}$, $\mathbf{h}_{t_N}^{u,l}$, and $\mathbf{u}$. We recommend possible POIs for the target user by calculating the dot-product of user and POI representations, which is similar to those previous studies using matrix factorization. Finally, the predicted probability that user u visits candidate POI v at time point t_{N+1} (i.e., the preference score) can be obtained by the following equation:

$$o_{t_{N+1},v}^u = \Big(\mathbf{h}_{t_N}^{u,s}\Big)^T \mathbf{c}_v^{u,s} + \Big(\mathbf{h}_{t_N}^{u,l}\Big)^T \mathbf{c}_v^{u,l} + (\mathbf{u})^T\mathbf{v} \tag{22}$$

where the last item of the equation denotes the inherent interest of user u in POI v.

4.7 Model Training

For user u, we build a training instance $I_N = <S_{t_N}^u, L_{t_N}^u, \{(v, v')\}>$ at time point t_N, including the current trajectory $S_{t_N}^u$, a set of historical check-ins of the user and his/her friends $L_{t_N}^u$, and all pairs of positive and negative POIs $\{(v, v')\}$ at the next moment t_{N+1}. Here, v and v' represent a positive (or called observed) POI and a negative (or called unobserved) POI, respectively. For the construction process of training instances, please refer to *Algorithm 1*.

ALGORITHM 1: Constructing training instances

Input: an LBSN G and a set of historical check-in sequences of all users C^U
Output: a set of training instances D

Initialize $D = \cup_u D^u = \emptyset$
For each user u in G **do**
  For each user trajectory $S^u_{t_{N+1}}$ in C_u **do**
    Get the set of historical check-ins of u and his/her friends at t_N, $L^u_{t_N} = \cup_{t_1}^{t_N} \cup_{u' \in N(u) \cup \{u\}} \{C^{u'}_{t_i}\}$;
    Get a positive sample $v^u_{t_{N+1}}$ that u visited at t_{N+1} from $c^u_{t_{N+1}}$;
    Get the set of negative samples $\{v'^u_{t_{N+1}}\}$ by the sampling method;
    Add a training instance $<S^u_{t_N}, L^u_{t_N}, \{(v^u_{t_{N+1}}, v'^u_{t_{N+1}})\}>$ to D^u;
  End for
End for
$D = \cup_u D^u$;
Return the training set D;

We use the Bayesian personalized ranking (BPR) [44] rather than the point-wise loss to define the loss function for model parameter learning. BPR can make use of the unobserved user-POI data by learning a pair-wise ranking loss in the DAN-SNR's training process. Moreover, BPR considers the relative order of POIs to predict users' preference scores, according to an underlying assumption that each user prefers the observed POI (or called positive example) over the unobserved POIs (also known as negative examples).

The BPR loss function requires pairs of two scores: one for the target POI (i.e., the actual next POI) and the other for a negative sample (i.e., any POI except the target POI). However, calculating scores for all pairs of (v, v') is not practical in real-world application scenarios with millions of items [45]. Thus, we use a sampling method similar to the negative sampling mechanism used in word2vec [46] to select a fraction of POIs as negative samples during the training process. Because POI's geographical information has an important impact on predicting a user's next movement, we randomly choose negative samples from the observed POIs located in the same city. Suppose the number of all the observed POIs situated in the same town are smaller than the size of negative samples. In that case, we employ the popularity-based sampling method [45] to generate the remaining negative samples.

Then, we use the maximum a posterior (MAP) estimation to learn the DAN-SNR's parameters, described below:

$$p(u, t_{N+1}, v \succ v') = g(o^u_{t_{N+1}, v} - o^u_{t_{N+1}, v'}) \tag{23}$$

where $g(\cdot)$ denotes a nonlinear function defined as:

$$g(x) = \frac{1}{1 + e^{-x}} \tag{24}$$

By integrating the pair-wise loss function and a regularization term, we can solve the DAN-SNR's objective function for the next POI recommendation task as follows:

$$\begin{aligned} J &= -\sum_{I_N} \sum_{u} \sum_{(v, v')} \ln p(u, t_{N+1}, v \succ v') + \frac{\lambda}{2} ||\Theta||^2 \\ &= \sum_{I_N} \sum_{u} \sum_{(v, v')} \ln \left(1 + e^{-(o^u_{t_{N+1}, v} - o^u_{t_{N+1}, v'})}\right) + \frac{\lambda}{2} ||\Theta||^2 \end{aligned} \tag{25}$$

Table 2. Statistics of the Experimental Datasets

Dataset	#Users	#Check-ins	#POIs	#Friendships	#Trajectories
Gowalla	1,947	569,651	25,322	10,274	231,192
Brightkite	2,987	1,939,499	14,259	17,808	876,755

where λ determines the power of regularization, and Θ indicates the parameter set. Note that the objective function is optimized by the stochastic optimization method, Adam [47]. For the whole training process of the DAN-SNR, please refer to *Algorithm 2*.

ALGORITHM 2: Training the DAN-SNR

Input: a training set D
Output: the parameter set of the DAN-SNR Θ

```
Initialize the parameter set Θ;
While (exceed(maximum number of iterations) == FALSE) do
    Randomly select a batch of training instances D_b from D;
    For each user u in D_b do
        For each pair of (v, v') of u in D_b do
            Calculate the probabilities o^u_{t_{N+1},v} and o^u_{t_{N+1},v'} according to Equation (22);
        End for
    End for
    Find Θ minimizing the objective function (Equation (25)) with D_b;
End while
Return the parameter set Θ;
```

5 EXPERIMENT SETUPS AND RESULTS

5.1 Datasets

Two publicly available LBSN datasets [5] (i.e., Gowalla and Brightkite) are used for our evaluation in this study. A check-in record consists of user ID, check-in timestamp, POI ID, and the corresponding location in the two datasets. First, we preprocessed the two datasets to filter out inactive users who have fewer than 20 check-in records and unpopular POIs that have been visited for less than 20 times [15]. Second, we constructed check-in trajectories for each user. According to the definition of user trajectory, we split each user's check-in sequence into trajectories of different lengths. As with Cheng et al.'s work [1], the interval threshold for any two successive check-ins was set to six hours. In other words, if the time interval between two consecutive check-ins is more than six hours, the two check-ins belong to two different trajectories. To alleviate the cold-start problem for new users, we removed those users with fewer than five trajectories from the two datasets. Next, we built two separate user-user graphs composed of the remaining users and their friendships in the two datasets. Finally, we obtained two experimental datasets derived from the original datasets. Table 2 shows the statistics of the two experimental datasets.

5.2 Baseline Approaches

To validate the DAN-SNR's effectiveness in the next POI recommendation task, we compare it with the following seven competitive approaches.

(1) *FPMC-LR* (short for factorized personalized Markov chain for localized regions) [8]. It is a matrix factorization method that uses a Markov chain model to model users' customized

Table 3. Summary of the Eight Approaches used in this Study

Feature	FPMC-LR	PRME-G	ST-RNN	GRU4Rec+ST	SASRec+ST	ATST-LSTM	DGRec+ST	DAN-SNR
SE	✓	✓	✓	✓	✓	✓	✓	✓
SP	✓	✓	✓	✓	✓	✓	✓	✓
TE	✗	✗	✓	✓	✓	✓	✓	✓
SO	✗	✗	✗	✗	✗	✗	✓	✓
AT	✗	✗	✗	✗	✓	✓	✓	✓

SE, SP, TE, SO, and AT denote whether the given approach considers the sequential information, spatial information, temporal information, social information, and attention mechanism, respectively.

sequential transitions. As an extension of FPMC [48], this method embeds the personalized Markov chain in check-in sequences and the localized regions in users' movement constraint.

(2) *PRME-G* (short for personalized ranking metric embedding with geographical influence) [9]. It is a metric embedding approach that models users' personalized check-in sequences by embedding users and POIs into a shared latent space. As an extension of PRME, this method utilizes a simple weighting scheme to fuse geographical influence.

(3) *ST-RNN* (short for spatial-temporal recurrent neural networks) [11]. It is an RNN-based model that incorporates spatial-temporal contexts in a recurrent architecture. This method extends an RNN and can model both temporal influence and geographical influence in each layer with specific transition matrices.

(4) *GRU4Rec+ST* (short for a gated recurrent unit for recommendations with spatial and temporal contexts) [49]. GRU4Rec is a session-based recommendation model that adopts an RNN-based framework. However, it is not designed for the next-POI recommendation task. In this study, we extend the GRU4Rec by embedding the spatial and temporal contexts of user check-ins into a compact vector representation.

(5) *SASRec+ST* (short for self-attention based sequential recommendation model with spatial and temporal contexts) [50]. SASRec is a sequential recommendation model that utilizes the self-attention mechanism. In the next POI recommendation scenarios, we also extend the SASRec by embedding the spatial and temporal contexts of user check-ins into a compact vector representation.

(6) *ATST-LSTM* (short for an attention-based spatiotemporal long and short-term memory) [15]. It is an RNN-based model that leverages the attention mechanism. This method can model both temporal influence and geographical influence in each step and pay more attention to those relevant historical check-in records in a check-in sequence.

(7) *DGRec+ST* (short for a dynamic-graph-attention neural network for recommendations) [51]. DGRec is a session-based social recommender system that can capture social influence with a dynamic graph attention neural network. To compare with other approaches in the next POI scenarios, we incorporate the spatial and temporal information of check-ins in the same way.

Table 3 summarizes the eight approaches used in this study. Generally speaking, they are classified into three categories of commonly used methods. First, the sequential POI recommendation approach using Markov chains (such as the FPMC-LR), embedding learning (such as the PRME-G), and neural networks (such as the ST-RNN and GRU4Rec+ST). Second, the attention-based POI recommendation approach, such as the SASRec+ST and ATST-LSTM. Third, the hybrid approach fuses sequential influence and social influence, such as the DGRec+ST and DAN-SNR.

5.3 Evaluation Metrics

We evaluate the recommendation performance of all the eight approaches regarding two commonly used metrics: Recall@k and normalized discounted cumulative gain@k (*NDCG@k*), where k equals to five or ten. Note that we do not choose Precision@k and F1-score@k as primary evaluation measures. The main reasons are two-fold. First, *P@k* (short for Precision@k) has a strong positive correlation with *R@k* (short for Recall@k). Second, *R@k* is more useful than *P@k* to show a recommendation approach's capability of searching out more candidate POIs in the next POI recommendation scenarios.

R@k measures how many of the actual POIs in the test set are hit by the top-k recommended items, formally defined as

$$R@k = \frac{1}{N}\sum_{u=1}^{N} R_u@k = \frac{1}{N}\sum_{u=1}^{N}\frac{|S_u(k) \cap V_u|}{|V_u|} \tag{26}$$

where $S_u(k)$ denotes a set of the top-k POIs recommended to user u, and V_u means a collection of POIs that the user visits at the next moment in the test set. Note that $|V_u| = 1$.

NDCG@k evaluates the ranking performance of a recommendation approach by considering the positions of actually visited POIs, formally defined as

$$NDCG@k = \frac{1}{N}\sum_{u=1}^{N}\frac{1}{Z_u}\sum_{j=1}^{k}\frac{2^{I(|\{s_u^j\}\cap V_u|)}-1}{log_2(j+1)} \tag{27}$$

where $I(\cdot)$ is an indicator function, s_u^j is the jth recommended item in $S_u(k)$, and Z is a normalization constant that is the maximum value of *DCG@k*.

In the training process, we evaluate the efficiency of neural network–based approaches in terms of running time per batch.

5.4 Settings

We carried out our experiment on a Lenovo ThinkStation P910 Workstation with dual processors (2 x Intel Xeon E5-2660 v4, 2.0 GHz) and one graphics processing unit (GPU, NVIDIA TITAN X Pascal, 12 GB). The operating system of the workstation was Microsoft Windows 10 (64-bit). All the program code used in our experiment was written in Python 3.7, and the deep learning framework we employed was TensorFlow[6] 1.2.0.

We built an L2L graph for each of the two datasets. However, the number of edges in the L2L graph grew exponentially as the number of nodes increased, which degraded graph embedding efficiency. To address this problem, we used an approximate solution to construct new L2L graphs according to the distance effect in human mobility [52]. We picked out a certain number of POIs with the shortest distance from the target POI and rebuilt a new L2L graph for each POI. The number of selected POIs was set to 20, which was far smaller than the number of POIs in the original L2L graphs. This solution facilitated the graph embedding process on L2L graphs.

The dimension size of five types of feature embeddings was set to 256. Feature embeddings were concatenated as the initial representation of a check-in behavior in a fully connected layer. The number of neurons in the fully connected layer was therefore set to 256. The lengths of the LTSC and STC were set to 200 and 50, respectively. Here, the maximum length of user trajectories in the two LBSN datasets was 50. Because long sequences of user check-ins caused high computation complexity, we assigned the length of the LSTC an appropriate value, i.e., 200. If the number of user check-ins in the input data is smaller than the length of a channel, we conducted the padding

[6] https://www.tensorflow.org/.

Table 4. Comparison of Different Methods in Recommendation Performance

Methods \ Metrics	Gowalla				Brightkite			
	R@5	*NDGG*@5	*R*@10	*NDGG* @10	*R*@5	*NDGG* @5	*R*@10	*NDGG* @10
FPMC-LR	0.0543	0.1133	0.1297	0.1195	0.1197	0.1267	0.1407	0.1506
PRME-G	0.0769	0.1276	0.1481	0.1317	0.1245	0.1331	0.1612	0.1708
ST-RNN	0.0904	0.1282	0.1645	0.1648	0.1736	0.1632	0.1913	0.1854
GRU4Rec+ST	0.1004	0.1431	0.1885	0.1652	0.1853	0.1845	0.2214	0.2145
SASRec+ST	0.1227	0.1502	0.1964	0.1858	0.1934	0.1851	0.2415	0.2234
ATST-LSTM	0.1336	0.1537	0.1961	0.1898	0.1965	0.1863	0.2598	0.2328
DGRec+ST	0.1576	0.1599	0.2214	0.2057	0.2045	0.1933	0.2811	0.2608
DAN-SNR	**0.1832**	**0.1783**	**0.2554**	**0.2219**	**0.2332**	**0.2214**	**0.3052**	**0.2823**

operation; otherwise, we chose only the latest 200 or 50 check-ins as input to the corresponding channel. In the two channels of the DAN-SNR, the number of the identical nonlinear layers was set to six; moreover, the number of attention heads in the self-attention layers was set to eight.

We apportioned user trajectories sorted in chronological order into training and test sets for the two datasets, with an 80-20 split. More specifically, the top 80% of trajectories were used as the training set, and the remainder of trajectories was used as the test set. For each target POI, the number of negative samples was set to 500. The batch size was set to 50. We employed Adam [47] as the optimizer and applied exponential decay in which the learning rate started at 0.001, and the decay rate was set to 0.96.

For more details of the proposed approach's settings, please refer to the source code publicly available for download at https://github.com/drhuangliwei/DAN-SNR.

5.5 Results

5.5.1 Recommendation Performance. Table 4 presents a comparison of the eight approaches in recommendation performance on the two datasets. The numbers shown in bold represent the best result of each column in Table 4. In general, the more the information, the better the approach. Because the PFMC-LR and PRME-G only use sequential and spatial data, they performed the worst among the eight methods regarding the two evaluation metrics. Besides, the FPMC-LR employs the Markov chain method to model sequential patterns, implying that it cannot capture the long-term sequential influence. Compared with the PFMC-LR and PRME-G, the ST-RNN and GRU4Rec+ST made improvements in recommendation performance, because they consider temporal and geographical influence further. Although the SASRec+ST and ATST-LSTM incorporate the same information used by the ST-RNN and GRU4Rec+ST, they achieved better results on the two datasets. The results indicate that leveraging the attention mechanism to model sequential influence can indeed improve recommendation performance. The DGRec+ST and DAN-SNR take into account four types of information and leverage the attention mechanism. As a result, they achieved the best results on both the two datasets.

Compared with the state-of-the-art DGRec+ST, the *R*@5, *NDGG*@5, *R*@10, and *NDGG*@10 values of the DAN-SNR were increased by 16.24%, 11.51%, 15.36%, and 7.88%, respectively, on the Gowalla dataset. For the Brightkite dataset, the performance improvements regarding the above four evaluation metrics were 14.03%, 14.54%, 8.57%, and 8.24%, respectively. The primary reasons that contribute to the state-of-the-art results are three-fold: First, the proposed approach makes full use of all four types of information available. Second, the DAN-SNR uses the self-attention mechanism to model user preference, and the results indicate that it is more effective than the

Table 5. Comparison of Different Methods in Training Efficiency

Dataset	Metric	ST-RNN	GRU4Rec+ST	SASRec+ST	ATST-LSTM	DGRec+ST	DAN-SNR
	#Params	5,447,680	14,083,306	**2,242,560**	10,383,800	6,814,976	18,650,112
Gowalla	Time(s)/batch	0.52	3.62	**0.15**	2.24	2.03	0.25
	NDGG@5	0.1282	0.1431	0.1502	0.1537	0.1599	**0.1783**
Brightkite	Time(s)/batch	0.54	3.75	**0.17**	2.38	2.36	0.28
	NDGG@5	0.1632	0.1845	0.1851	0.1863	0.1933	**0.2214**

Markov chain model (such as the PFMC-LR) and RNN-based neural networks (such as the ST-RNN, GRU4Rec+ST, ATST-LSTM, and DGRec+ST). Third, although the DGRec+ST also incorporates social information, it only models users' short-term preferences within a session using an RNN architecture. Instead, our approach considers more historical check-ins of each user and the user's friends in a unified manner and measure the behavioral correlations between different check-ins adaptively using the self-attention mechanism.

5.5.2 Model Training Efficiency. We then conducted an efficiency analysis on six neural-network-based methods, i.e., the ST-RNN, GRU4Rec+ST, SASRec+ST, ATST-LSTM, DGRec+ST, and DAN-SNR. Note that the last four approaches utilize the attention mechanism. To compare them in the same settings, the batch size, the embedding dimension, and the trajectory length of the other five approaches were set to 50, 256, and 50, respectively. For each of the two datasets, we calculated the time to training a batch under the same experimental environment on the whole dataset.

Table 5 shows a comparison of the six approaches in training efficiency on the two datasets. The numbers shown in bold represent each row's best result in Table 5, and #Params denotes the number of parameters involved in an approach. It is evident from Table 5 that the SASRec+ST and DAN-SNR, which leverage only the self-attention mechanism, work faster than the other four RNN-based methods, namely, ST-RNN, GRU4Rec+ST, ATST-LSTM, and DGRec+ST. The SASRec+ST runs a batch with the minimum amount of time, followed by the DAN-SNR with the maximum parameters. Compared with the SASRec+ST, our approach achieved 18.7% and 19.6% improvements in *NDCG*@5 on the two datasets, respectively. This result indicates that the DAN-SNR can make a better trade-off between performance and efficiency than the other five approaches, although it needs much more GPU memory than the SASRec+ST in the training process. Note that one standard GPU card with 4 GB memory is sufficient to train the DAN-SNR.

5.6 Attention Visualization

5.6.1 Self-attention Visualization. Since we utilize the self-attention mechanism to model the dependencies between any two historical check-ins, we visualize such dependencies in the STC's self-attention layer in this subsection. Figure 3 illustrates a qualitative analysis of a Gowalla user's 12 check-in records in three days. In Figure 3, the left part plots the user's trajectory with Bing Maps.[7] The right part depicts the correlations between check-ins in the self-attention layer of the fifth and sixth nonlinear layers, which correspond to the lower and upper rows, respectively. For each check-in of the user's trajectory, we display the check-in time and POI category. The self-attention weights of all the check-ins in this trajectory are visualized with correlation lines and color intensity. The darker the line, the higher the attention weight.

[7] https://www.microsoft.com/en-us/maps.

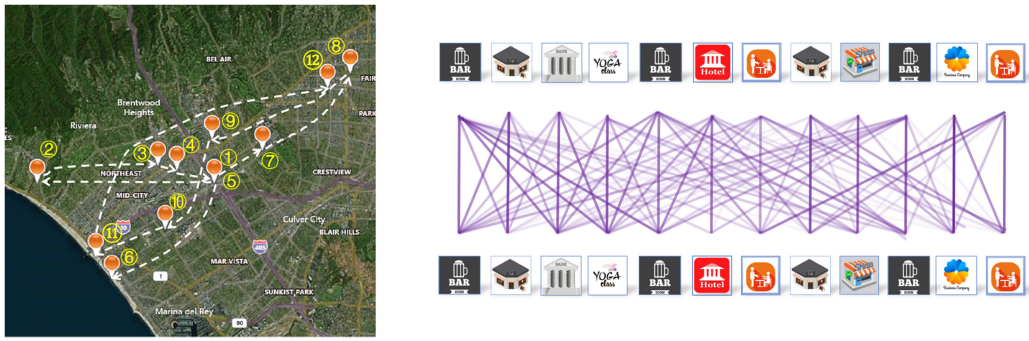

Fig. 3. An example of visualizing the dependencies between historical check-ins via the self-attention mechanism.

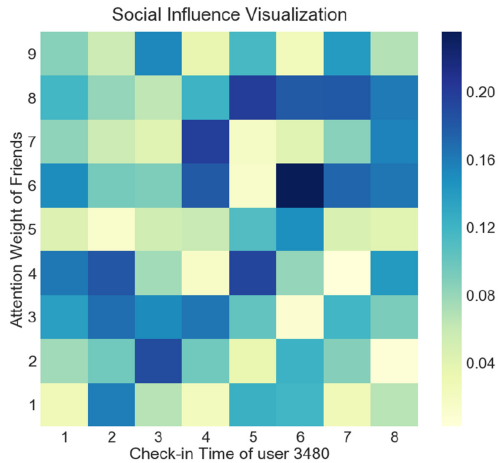

Fig. 4. An example of visualizing the social influence of friends on a user's movement over time.

We find some interesting behavioral patterns of the user from Figure 3. First, a check-in behavior at one moment correlates positively with a small number of historical check-ins. For example, the sixth check-in has no significant correlations with previous ones except the fourth and fifth check-ins. This finding reflects the randomness and uncertainty of individual behavior. Second, long-range dependencies exist among the user's check-ins. For example, although there is a long distance between the tenth and first check-ins, there is a relatively high correlation between them, mainly due to the periodicity of the user's daily habit. In brief, the results mentioned above suggest that the DAN-SNR can indeed model users' sequential patterns better via the self-attention mechanism.

5.6.2 Social Influence Visualization. Because the DAN-SNR weighs the contribution of check-ins of a user's friends by the self-attention mechanism, we picture the effect of social influence on individual behavior over time in this subsection. Figure 4 displays the impact of a randomly selected Gowalla user's friends on the user's current check-in behavior across eight consecutive timestamps. In this heat map, the X-axis represents the eight successive check-ins of the user, the Y-axis represents the user's nine friends, and each cell (i, j) denotes the impact of the jth friend on the ith check-in behavior. Note that we use the attention weights from the vanilla attention layer

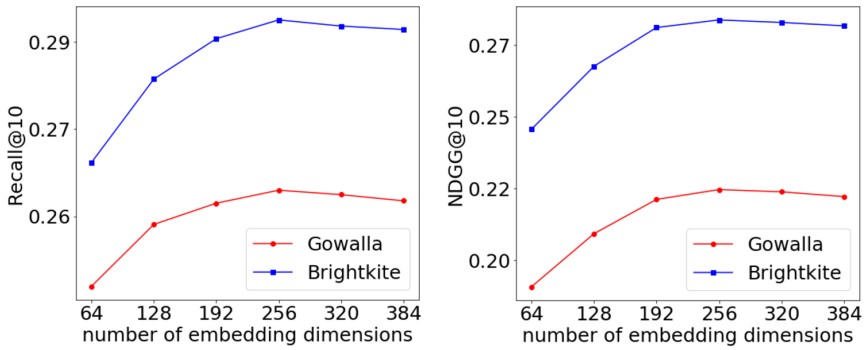

Fig. 5. Performance tuning with different embedding dimensions.

of the LTSC to express the effect of a friend's historical check-in on the user's current movement. Furthermore, we represent a friend's total impact by averaging the friend's attention weights of all historical check-ins.

Figure 4 shows that the social influence of the user's friends varies from person to person. For example, in the sixth column, the user's sixth friend has a significant impact on this check-in, which does not appear to be affected by the third friend. Besides, the influence of the same friend on the user's movement changes over time. For example, cells (5, 6) and (6, 6) look nearly opposite: one very white and one very dark, suggesting that the sixth friend has the opposite effect on the two successive check-ins. Moreover, this result indicates that the social influence of users is dynamic and context-dependent.

6 DISCUSSION

6.1 Sensitive Analysis of Parameters

6.1.1 Number of Embedding Dimensions. The number of embedding dimensions is essential to the DAN-SNR. The higher this parameter, the stronger the representation ability of our approach. However, high values of this parameter may lead to the overfitting problem. Figure 5 presents this parameter's effect on the DAN-SNR's recommendation performance on the Gowalla and Brightkite datasets. When the number of embedding dimensions is below 200, the *R*@10 and *NDCG*@10 values significantly increase with this parameter. This result implies that the representation ability of our approach has grown remarkably. As the number of embedding dimensions exceeds 256, there is a slight decline in recommendation performance regarding *R*@10 and *NDCG*@10, suggesting that the representation ability of the DAN-SNR has reached its limit. Therefore, the number of embedding dimensions was set to 256 in this study.

6.1.2 Number of Negative Samples. The number of negative samples is another critical parameter of the DAN-SNR. The performance of recommendation models will get improved with the increase of this parameter. This finding has been proved by some previous studies on representation learning [15, 53]. However, if all unobserved data is used as negative examples, a model's computational complexity will increase substantially. Figure 6 shows this parameter's effect on the DAN-SNR's recommendation performance on the two datasets. Note that "All" in Figure 6 denotes a specific case that all unobserved POIs were selected to be negative samples without data sampling. As the number of negative samples increases, the performance of our approach tends to improve considerably. However, the *R*@10 and *NDCG*@10 values remain unchanged after this parameter exceeds 500. Moreover, there is a visible decrease in the DAN-SNR's performance when

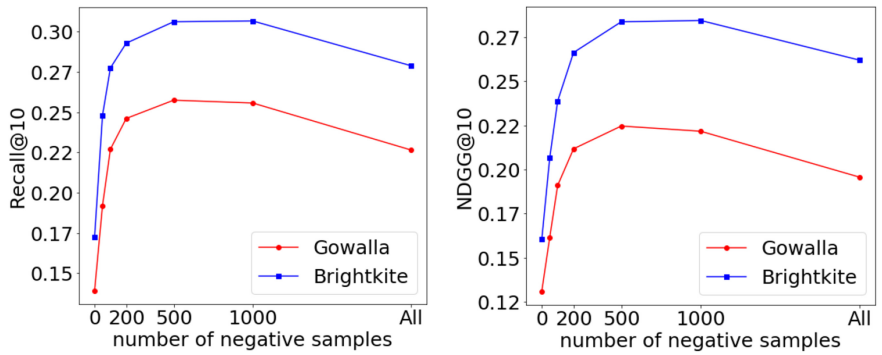

Fig. 6. Performance tuning with different negative samples.

Table 6. Comparison between Our Approach and Its Variants in Recommendation Performance

Methods \ Metrics	Gowalla				Brightkite			
	R@5	*NDGG*@5	*R*@10	*NDGG*@10	*R*@5	*NDGG*@5	*R*@10	*NDGG*@10
DAN-SNR-self	0.1540	0.1413	0.2123	0.1931	0.1897	0.1784	0.2674	0.2364
DAN-SNR-social	0.1090	0.0960	0.1496	0.1573	0.1442	0.1204	0.1612	0.1772
DAN-SNR	0.1832	0.1783	0.2554	0.2219	0.2332	0.2214	0.3052	0.2823

this parameter is higher than 1,000. Therefore, the number of negative samples was set to 500 in our experiment.

6.2 Ablation Study

6.2.1 Personal Preference Versus Social Influence. For each target user, the DAN-SNR generates his/her final representation that combines the user's past behavior and context-dependent social influence. To see how the above two "features" affect our method's performance, we designed two DAN-SNR variants: DAN-SNR-self and DAN-SNR-social. The former removed all check-ins of each user's friends and leveraged only the user's historical check-ins. The latter took into account only social influence while disregarding users' personal preferences mined from their past behavior. The difference between the two variants is on what channel they used. DAN-SNR-self employs two channels to model long-term and short-term user preferences, respectively, while DAN-SNR-social uses one channel to model social influence.

Table 6 presents the recommendation performance of our approach and its two variants on the two datasets. As shown in Table 6, DAN-SNR-self outperforms DAN-SNR-social across the two datasets, suggesting that a user's personal preference contributes more to the user's next move than his/her friends' influence. Therefore, modeling user preference based on historical check-ins is a necessary part of our method. Once the DAN-SNR makes use of both the two "features," there is a marked increase in the values of the two evaluation metrics, which is, of course, the primary motivation of this study. As a result, it is crucial to model user preference and social influence in the next POI recommendation task to improve recommendation performance further.

6.2.2 Short-term Preference Versus Long-term Preference. Since user preference plays an essential role in personalized POI recommendation, the DAN-SNR provides a mechanism to encode users' short-term and long-term preferences. We then studied the impact of two different user preferences on recommendation performance and designed two variants: DAN-SNR-long

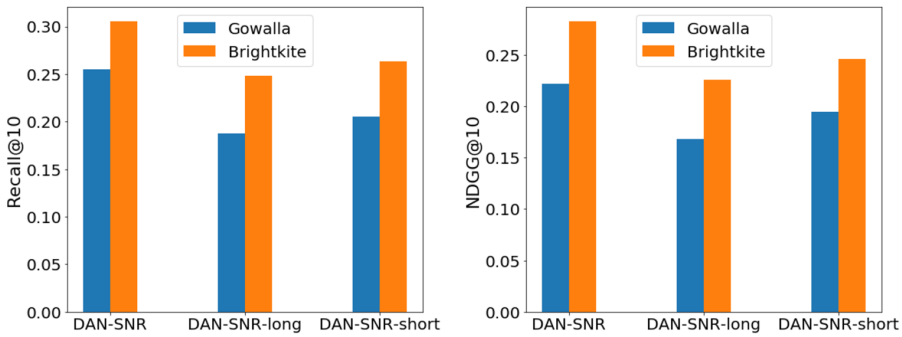

Fig. 7. Effect of two different types of user preferences on recommendation performance.

and DAN-SNR-short. The former took into consideration long-term user preference, while the latter leveraged short-term user preference. Note that we deleted the "feature" of social influence from both the two variants.

Figure 7 displays a comparison between our approach and its variants in performance on the Gowalla and Brightkite datasets. DAN-SNR-short achieved, unsurprisingly, higher values of *R*@10 and *NDCG*@10 than DAN-SNR-long across the two datasets. The main reason is that short-term preference can capture users' changing needs in context-dependent scenarios better. Therefore, short-term user preference has a more significant impact on recommendation performance. However, it is worth noting that DAN-SNR-short offered a slight performance advantage over DAN-SNR-long. Compared with the former, DAN-SNR-long analyzed each user's current check-in trajectory and did not extract sequential patterns from these trajectories. In other words, long-term user preference captures long-range dependencies between historical check-ins. When our method took into account long-term and short-term preferences together, its performance was improved significantly.

We further analyzed the impact of the number of check-ins in user trajectories on the quality of recommendations after fixing the "feature" of long-term preference. There was a significantly positive correlation between the actual length of user trajectories without padding and the recommendation quality for the users with check-in sequences of similar size in the test set. This result indicated that our approach could recommend more accurate POIs for those users with a larger number of check-ins in their trajectories. In other words, DAN-SNR's recommendation performance will become better if the target users have a more extended history of check-ins.

6.3 Limitations

The cold-start problem is a common problem in recommender systems. Considering the challenge of the next POI recommendation task, we do not attempt to address the cold-start problem of new users in this study. As mentioned in Section 5.1, we filtered out inactive users with fewer than five trajectories from the two datasets to alleviate the user cold-start problem. Besides, we do not care whether the recommended next POI is new (or unvisited) or not in this study. Here, a new POI is a place the target user had never visited. Although some recent studies [9, 24] report their promising results in the next new POI recommendation task, our approach's performance remains mostly unexplored in such a scenario. Because the next new POI recommendation problem is a far more challenging task, we systematically investigate it in the future. In brief, the proposed approach has limitations in coping with the cold-start problem in next POI recommendation, and our future work is to address it.

An online social network structure evolves with the change of social relationships, despite the structural self-similarity [54]; the same is true for this study. Since the two experimental datasets did not provide information about when two users became good friends (or the users were not friends anymore), we have to assume that the friendship between LBSN users remains unchanged. In other words, the structure of the user-user graph in an LBSN keeps constant. Therefore, our approach cannot characterize the time-varying social influence in LBSNs. Even so, we believe that our future work that considers the temporality in friendships is non-trivial to the research of social-aware next POI recommendation.

7 CONCLUSION

Next (or successive) POI recommendation is a challenging task of POI recommendation and has been studied in recent years. In this study, we discuss a new research topic, i.e., social-aware next POI recommendation. By incorporating sequential influence, temporal influence, spatial influence, and social influence, we design and implement a deep attentive neural network called DAN-SNR. More specifically, the DAN-SNR can model the context-dependent social influence by capturing the behavioral correlation between the target user and his/her friends. By leveraging the self-attention mechanism rather than using the RNN architecture, it can better model long-range dependencies between each user's historical check-ins regardless of the distance between them. Besides, experimental results on two public LBSN datasets indicate that the proposed approach outperforms seven competitive baselines regarding two commonly used metrics, i.e., *Recall* and *NDCG*.

It is worth noting that our work may play a promising role in many application scenarios, such as location-based item recommendation, mobile advertising, and travel assistant. In particular, a smart travel assistant can create personalized user profiles by automatically learning historical check-in records of a user and the mobility behaviors of the user's friends and then recommend possible POIs to go next in real-time.

In the future, on the one hand, we will incorporate more context information, such as visual and text information associated with users and POIs, into the DAN-SNR to improve performance further. On the other hand, social-aware next POI recommendation is, in essence, a kind of graph-structured recommendations. We plan to effectively embed users and POIs in ever-changing LBSNs using new graph embedding techniques, such as graph convolutional networks that have recently been proved to be the state-of-the-art representation learning method in many applications.

REFERENCES

[1] Chen Cheng, Haiqin Yang, Irwin King, and Michael R. Lyu. 2012. Fused matrix factorization with geographical and social influence in location-based social networks. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence*. AAAI Press, 17–23.

[2] Huiji Gao, Jiliang Tang, Xia Hu, and Huan Liu. 2013. Exploring temporal effects for location recommendation on location-based social networks. In *Proceedings of the 7th ACM Conference on Recommender Systems*. ACM, 93–100. DOI: https://doi.org/10.1145/2507157.2507182

[3] Liwei Huang, Yutao Ma, and Yanbo Liu. 2015. Point-of-interest recommendation in location-based social networks with personalized geo-social influence. *China Commun.* 12, 12 (2015), 21–31. DOI: https://doi.org/10.1109/CC.2015.7385525

[4] 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. *PVLDB* 10, 10 (2017), 1010–1021. DOI: https://doi.org/10.14778/3115404.3115407

[5] Eunjoon Cho, Seth A. Myers, and Jure Leskovec. 2011. Friendship and mobility: user movement in location-based social networks. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1082–1090. DOI: https://doi.org/10.1145/2020408.2020579

[6] Jie Bao, Yu Zheng, David Wilkie, and Mohamed Mokbel. 2015. Recommendations in location-based social networks: A survey. *GeoInformatica* 19, 3 (2015), 525–565. DOI: https://doi.org/10.1007/s10707-014-0220-8

[7] V. S. Subrahmanian and Srijan Kumar. 2017. Predicting human behavior: The next frontiers. *Science* 355, 6324 (2017), 489. DOI: https://doi.org/10.1126/science.aam7032

[8] Chen Cheng, Haiqin Yang, Michael R. Lyu, and Irwin King. 2013. Where you like to go next: Successive point-of-interest recommendation. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*. AAAI Press, 2605–2611.

[9] Shanshan Feng, Xutao Li, Yifeng Zeng, Gao Cong, Meng Yeow Chee, and Quan Yuan. 2015. Personalized ranking metric embedding for next new POI recommendation. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*. AAAI Press, 2069–2075.

[10] Shenglin Zhao, Michael R. Lyu, and Irwin King. 2016. STELLAR: Spatial-temporal latent ranking for successive point-of-interest recommendation. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*. AAAI Press, 315–322.

[11] 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*. AAAI Press, 194–200.

[12] Lu Zhang, Zhu Sun, Jie Zhang, Horst Kloeden, and Felix Klanner. 2020. Modeling hierarchical category transition for next POI recommendation with uncertain check-ins. *Inf. Sci.* 515 (2020), 169–190. DOI: https://doi.org/10.1016/j.ins.2019.12.006

[13] 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 *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*. AAAI Press, 5877–5884. DOI: https://doi.org/10.1609/aaai.v33i01.33015877

[14] Haochao Ying, Jian Wu, Guandong Xu, Yanchi Liu, Tingting Liang, Xiao Zhang, and Hui Xiong. 2019. Time-aware metric embedding with asymmetric projection for successive POI recommendation. *World Wide Web* 22, 5 (2019), 2209–2224. DOI: https://doi.org/10.1007/s11280-018-0596-8

[15] Liwei Huang, Yutao Ma, Shibo Wang, and Yanbo Liu. 2019. An attention-based spatiotemporal LSTM network for next POI recommendation. *IEEE Trans. Serv. Comput.* (2019), 1–1. DOI: https://doi.org/10.1109/tsc.2019.2918310

[16] Huiji Gao, Jiliang Tang, and Huan Liu. 2012. gSCorr: Modeling geo-social correlations for new check-ins on location-based social networks. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*. ACM, 1582–1586. DOI: https://doi.org/10.1145/2396761.2398477

[17] Jia-Dong Zhang and Chi-Yin Chow. 2013. iGSLR: Personalized geo-social location recommendation: a kernel density estimation approach. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 334–343. DOI: https://doi.org/10.1145/2525314.2525339

[18] Liwei Huang, Yutao Ma, Yanbo Liu, and Arun Kumar Sangaiah. 2020. Multi-modal Bayesian embedding for point-of-interest recommendation on location-based cyber-physical-social networks. *Fut. Gen. Comput. Syst.* 108 (2020), 1119–1128. DOI: https://doi.org/10.1016/j.future.2017.12.020

[19] Cheng Yang, Maosong Sun, Wayne Xin Zhao, Zhiyuan Liu, and Edward Y. Chang. 2017. A neural network approach to jointly modeling social networks and mobile trajectories. *ACM Trans. Inf. Syst.* 35, 4 (2017), 36:1–36:28. DOI: https://doi.org/10.1145/3041658

[20] 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 *Proceedings of the International Conference on Advances in Neural Information Processing Systems*. 5998–6008.

[21] Chang Zhou, Jinze Bai, Junshuai Song, Xiaofei Liu, Zhengchao Zhao, Xiusi Chen, and Jun Gao. 2018. ATRank: An attention-based user behavior modeling framework for recommendation. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*. AAAI Press, 4564–4571.

[22] 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*. ACM, 103–112. DOI: https://doi.org/10.1145/2666310.2666400

[23] Jihang Ye, Zhe Zhu, and Hong Cheng. 2013. What's your next move: User activity prediction in location-based social networks. In *Proceedings of the SIAM International Conference on Data Mining*. SIAM, 171–179. DOI: https://doi.org/10.1137/1.9781611972832.19

[24] Xin Li, Dongcheng Han, Jing He, Lejian Liao, and Mingzhong Wang. 2019. Next and next new POI recommendation via latent behavior pattern inference. *ACM Trans. Inf. Syst.* 37, 4 (2019), 46:1–46:28. DOI: https://doi.org/10.1145/3354187

[25] Laura Alessandretti, Piotr Sapiezynski, Vedran Sekara, Sune Lehmann, and Andrea Baronchelli. 2018. Evidence for a conserved quantity in human mobility. *Nat. Hum. Behav.* 2 (2018), 485–491. DOI: https://doi.org/10.1038/s41562-018-0364-x

[26] Yanchi Liu, Chuanren Liu, Bin Liu, Meng Qu, and Hui Xiong. 2016. Unified point-of-interest recommendation with temporal interval assessment. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1015–1024. DOI: https://doi.org/10.1145/2939672.2939773

[27] Jing He, Xin Li, Lejian Liao, Dandan Song, and William K. Cheung. 2016. Inferring a personalized next point-of-interest recommendation model with latent behavior patterns. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*. AAAI Press, 137–143.

[28] Ranzhen Li, Yanyan Shen, and Yanmin Zhu. 2018. Next point-of-interest recommendation with temporal and multi-level context attention. In *Proceedings of the IEEE International Conference on Data Mining*. IEEE Computer Society, 1110–1115. DOI: https://doi.org/10.1109/icdm.2018.00144

[29] Yuxia Wu, Ke Li, Guoshuai Zhao, and Xueming Qian. 2019. Long-and short-term preference learning for next POI recommendation. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. ACM, 2301–2304. DOI: https://doi.org/10.1145/3357384.3358171

[30] Volodymyr Mnih, Nicolas Heess, Alex Graves, and Koray Kavukcuoglu. 2014. Recurrent models of visual attention. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems*. 2204–2212.

[31] 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 the 27th International Conference on World Wide Web*. ACM, 1459–1468. DOI: https://doi.org/10.1145/3178876.3186058.

[32] Qiang Gao, Fan Zhou, Goce Trajcevski, Kunpeng Zhang, Ting Zhong, and Fengli Zhang. 2019. Predicting human mobility via variational attention. In *Proceedings of the 28th International Conference on World Wide Web*. ACM, 2750–2756. DOI: https://doi.org/10.1145/3308558.3313610

[33] Pavlos Kefalas, Panagiotis Symeonidis, and Yannis Manolopoulos. 2016. A graph-based taxonomy of recommendation algorithms and systems in LBSNs. *IEEE Trans. Knowl. Data Eng.* 28, 3 (2016), 604–622. DOI: https://doi.org/10.1109/TKDE.2015.2496344

[34] Huiji Gao, Jiliang Tang, and Huan Liu. 2012. Exploring social-historical ties on location-based social networks. In *Proceedings of the 6th International AAAI Conference on Weblogs and Social Media*. AAAI Press, 114–121.

[35] Huayu Li, Yong Ge, Richang Hong, and Hengshu Zhu. 2016. Point-of-interest recommendations: Learning potential check-ins from friends. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 975–984. DOI: https://doi.org/10.1145/2939672.2939767

[36] Josh Jia-Ching Ying, Wen-Ning Kuo, Vincent S. Tseng, and Eric Hsueh-Chan Lu. 2014. Mining user check-in behavior with a random walk for urban point-of-interest recommendations. *ACM Trans. Intell. Syst. Technol.* 5, 3 (2014), 40:1–40:26. DOI: https://doi.org/10.1145/2523068

[37] Pavlos Kefalas, Panagiotis Symeonidis, and Yannis Manolopoulos. 2018. Recommendations based on a heterogeneous spatio-temporal social network. *World Wide Web* 21, 2 (2018), 345–371. DOI: https://doi.org/10.1007/s11280-017-0454-0

[38] Pavlos Kefalas and Yannis Manolopoulos. 2017. A time-aware spatio-textual recommender system. *Expert Syst. Appl.* 78, (2017), 396–406. DOI: https://doi.org/10.1016/j.eswa.2017.01.060

[39] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 855–864. DOI: https://doi.org/10.1145/2939672.2939754

[40] Tao Xu, Yutao Ma, and Qian Wang. 2018. Cross-urban point-of-interest recommendation for non-natives. *Int. J. Web Serv. Res.* 15, 3 (2018), 82–102. DOI: https://doi.org/10.4018/ijwsr.2018070105

[41] Bahdanau Dzmitry, Cho Kyunghyun, and Bengio Yoshua. 2014. Neural machine translation by jointly learning to align and translate. *ArXiv.org*, arXiv:1409.0473 (2014).

[42] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, 770–778. DOI: https://doi.org/10.1109/cvpr.2016.90

[43] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *arXiv.org*, arXiv:1607.06450 (2016).

[44] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2012. BPR: Bayesian personalized ranking from implicit feedback. *Arxiv.Org*, arXiv:1205.2618 (2012).

[45] Balázs Hidasi and Alexandros Karatzoglou. 2018. Recurrent neural networks with top-k gains for session-based recommendations. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. ACM, 843–852. DOI: https://doi.org/10.1145/3269206.3271761

[46] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems*. 3111–3119.

[47] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *ArXiv.org*, arXiv:1412.6980 (2014).

[48] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized Markov chains for next-basket recommendation. In *Proceedings of the 19th International Conference on World Wide Web*. ACM, 811–820. DOI: https://doi.org/10.1145/1772690.1772773

[49] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. *ArXiv.org*, arXiv:1511.06939 (2015).

[50] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *Proceedings of the IEEE International Conference on Data Mining*. IEEE Computer Society, 197–206. DOI: https://doi.org/10.1109/icdm.2018.00035

[51] Weiping Song, Zhiping Xiao, Yifan Wang, Laurent Charlin, Ming Zhang, and Jian Tang. 2019. Session-based social recommendation via dynamic graph attention networks. In *Proceedings of the 12th ACM International Conference on Web Search and Data Mining*. ACM, 555–563. DOI: https://doi.org/10.1145/3289600.3290989

[52] Chaoming Song, Zehui Qu, Nicholas Blumm, and Albert-László Barabási. 2010. Limits of predictability in human mobility. *Science* 327, 5968 (2010), 1018–1021. DOI: https://doi.org/10.1126/science.1177170

[53] Chen Ting, Kornblith Simon, Norouzi Mohammad, and Hinton Geoffrey. 2020. A simple framework for contrastive learning of visual representations. *arXiv.org*, arXiv:2002.05709 (2020).

[54] Hanhua Chen, Hai Jin, and Shaoliang Wu. 2016. Minimizing inter-server communications by exploiting self-similarity in online social networks. *IEEE Trans. Parallel Distrib. Syst.* 27, 4 (2016), 1116–1130. DOI: https://doi.org/10.1109/TPDS.2015.2427155

Received April 2020; revised September 2020; accepted October 2020