TimeSAN: A Time-Modulated Self-Attentive Network for Next Point-of-Interest Recommendation

Jiayuan He, Jianzhong Qi*, Kotagiri Ramamohanarao

School of Computing and Information Systems, The University of Melbourne, Australia

hjhe@student.unimelb.edu.au, {jianzhong.qi,kotagiri}@unimelb.edu.au

***Abstract*—Next Point-of-Interest (POI) recommendation aims to rank a list of POIs by their attractiveness to users based on the users' historical records of POI visits. This task is challenging, because user preferences may be influenced by various contextual factors. In this paper, we consider the temporal contextual factor, i.e., the time of users' POI visits. Previous attempts for modelling the impact of temporal contexts can be categorized into two groups: factorization based methods and recurrent neural network based methods. The first group adds a time dimension to their latent recommendation spaces, which may suffer from the data sparsity problem due to the additional dimension. The second group uses time-aware contextual gates to update the hidden and cell states in RNNs, which may have limited capability in capturing long-range temporal dynamics. In this paper, we propose a time-modulated self-attentive network (TimeSAN) for next POI recommendation. This model learns the relevance between a user's next POI visit and her historical visits via the self-attention mechanism, where the relevance is modulated by the temporal contextual influence. The learned time-aware relevance is further fused with users' long-term interests to provide final recommendations. We conduct extensive experiments on real-world datasets. The results confirm that TimeSAN outperforms previous methods consistently and significantly in recommendation accuracy, while attaining a high model training efficiency.**

I. Introduction

The rapid growth of location-based social networks (LBSNs), such as Foursquare[1] and Yelp[2], has enabled researchers to incorporate better personalization in location-based recommendation tasks. A widely studied location-based task is the *next Point-of-Interest (POI) recommendation*, which aims to rank a list of POIs by their attractiveness to users, based on the users' historical records of POI visits (e.g., check-ins) [11].

Temporal contexts of users' POI visits are crucial to POI recommendations. We use Fig. 1 to illustrate how users' check-ins are influenced by the temporal contexts. We present an example of three users' check-ins within a day. We observe that the preferences of the three users change with the time: they prefer to visit "refreshing" POIs (e.g., Cafe) in the morning and "relaxation" POIs (e.g., Bar) at night. Suppose that we need to make a recommendation for User 3 at 20:00. Models that are solely based similarity but agnostic to temporal contexts may suggest "Pool" for User 3, since User 2 and 3 are more similar according to their previous check-ins. However, if we take the recommendation time "20:00" into consideration, "Bar" is a more preferable choice for User 3.

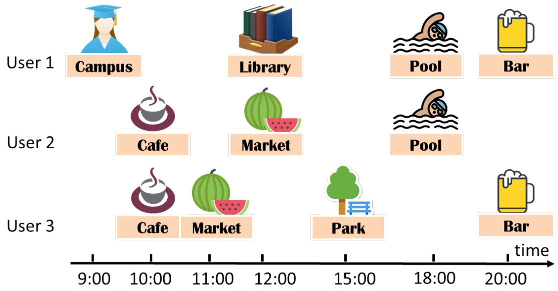


Fig. 1: An example of three users' check-ins within a day.

Capturing the temporal influence is nontrivial, mainly for two reasons. First, real-world check-in datasets are sparse, making it difficult to infer a complete time-varying profile for each user. Secondly, the number of possible timestamps that are need to be considered is large and even infinite, which calls for a recommendation model that utilizes the impact of timestamps collaboratively and effectively.

Existing studies that consider the temporal contexts can be categorized into two groups. The first group is developed based on factorization. The entire set of possible timestamps is mapped into a finite set of time slots, assuming that the timestamps within the same time slot have the same influence on user check-ins [26]. Then a low-dimensional latent space is used to capture the features of the time slots, which are further combined with users' other types of preferences. However, it is difficult to determine which time granularity to be used for time slot mapping. If certain time granularity has significant impact on users' preferences but is not incorporated, the model may not be able to make accurate recommendations. On the other hand, if a model considers all possible time granularity, the user-POI check-in matrix within each time slot will be too sparse, which will also deteriorate the model performance.

The second group is built based on recurrent neural networks (RNNs). RNN cells are used to process users' historical check-ins sequentially and predict users' next POI visits. To incorporate the temporal influences, various time-aware cell structures are proposed [6, 28]. However, RNNs may not effectively capture long-range temporal dynamics, since the historical records are represented only by the cell and hidden states with limited dimensions. Moreover, RNN models are inefficient in training, since the intrinsic recurrent structure cannot fully utilize parallel computation.

*Primary contact

[1] https://foursquare.com/

[2] https://www.yelp.com/

To address the above limitations, we introduce a time-modulated self-attentive network, TimeSAN, for time-aware next POI recommendation. TimeSAN treats the POI recommendation as a sequence-to-sequence translation task, which aims to translate the sequence consisting of a user's historical check-ins to the sequence consisting of the user's future check-ins. TimeSAN adapts the the multi-head self-attention mechanism [20] for the translation between check-in sequences. To consider the influence of temporal contexts, TimeSAN enhances the vanilla self-attention mechanism with a *time-modulated self-attention module*, which allows the attention among the temporal contexts to modulate the self-attention of the input check-in sequence. Further, TimeSAN introduces a temporal encoding method which maps timestamps to their multi-scale periodic representations. The proposed encoding method is shown to be more effective in capturing the features of temporal information compared with existing encoding methods. Finally, thanks to the inherited self-attention architecture, TimeSAN retains the state-of-the-art training speed, and is more than one order of magnitude faster compared with the RNN-based methods.

To summarize, the contributions of this paper are as follows.

- We propose a time-modulated self-attentive network for time-aware next POI recommendation. The model utilizes a proposed *time-modulated self-attention module* to capture the influence of temporal contexts on users' preferences. Moreover, we propose a multi-scale periodic encoding for temporal contexts which facilitates capturing temporal dynamics in user preferences.
- We conduct an extensive evaluation on the learning capability of the proposed model using synthetic data. The results confirm that TimeSAN can capture the impact of temporal contexts effectively. The results also confirm that, compared with existing time encoding methods, the proposed method provides a more effective representation of the temporal contexts.
- We conduct extensive experiments on real-world datasets. The experimental results confirm that the proposed model outperforms the state-of-the-art recommendation models significantly and consistently. In particular, TimeSAN improves the recommendation accuracy by up-to 17.1%. Meanwhile, TimeSAN retains high training efficiency, and is faster than RNN-based models by more than one order of magnitude.

II. Related work

We review three groups of related studies: (1) time-agnostic POI recommendations; (2) time-aware POI recommendations; and (3) attention mechanisms.

A. Time-Agnostic POI Recommendations

Many models have been proposed for POI recommendations, such as user-based collaborative filtering (CF) [2] and matrix factorization (MF) [10]. To enhance the prediction accuracy, several contextual factors have also been studied and incorporated, e.g., geographical locations of POIs [10], and social connections of users [3]. Recently, sequential patterns in users' check-ins have attracted lots of interests. Many models for capturing sequential patterns have been proposed, e.g, Markov-chain models which capture the patterns using the transition probabilities between POIs [4, 14] and recurrent-neural network models [15] which utilize RNNs to model the long- and short-term dynamics in users' check-ins.

B. Time-Aware POI Recommendations

There exist two categories of studies on time-aware POI recommendations. The first category is based on factorization. Gao et al. [5] propose a recommendation model based on matrix factorization. They split the time axis into hourly slots, which result in 24 sub-matrices from the original user-POI check-in matrix. To overcome the data sparsity, they propose a smoothing method that jointly factorizes the sub-matrices and estimates users' hourly preferences. Liu et al. [13] embed users and POIs into a low-dimensional latent space. To incorporate temporal contexts, they consider 168 time slots (24 hours per day $\times$ 7 days per week), which are embedded into the same latent space, where the proximity between a time slot and a POI captures the characteristics of their interaction.

The second category of studies seek to use RNNs to model the temporal impact. Zhu et al. [28] propose the Time-LSTM, a variant of long- and short-term memory (LSTM), for time-aware recommendation. Specifically, they introduce a *time gate* into a LSTM cell. At each step, a Time-LSTM cell takes two inputs: an input POI (check-in) and an input timestamp. It uses the input POI to update the current hidden and cell states, where the update mechanism is controlled by time gate using the input timestamp. Huang et al. [7] also use RNN for time-aware POI recommendation. They propose the ATST-LSTM network that aggregates the latent features of each check-in and the associated temporal context at each step. To enhance the learning of long-range temporal dependencies, ATST-LSTM augments the RNN network with an attention layer on top. However, due to the recurrent structure of RNNs, these models cannot fully utilize parallel acceleration.

C. Attention Mechanisms

Attention mechanisms have attracted extensive interests due to its effectiveness in a wide range of tasks including machine translation, acoustic modelling, and image captioning. There exist some studies on recommendations using attention mechanisms. Chen et al. [1] augment Bayesian pairwise ranking (BPR) with an attention layer that simultaneously capture two types of attentions: the user-item level attention and the user-feature level attention. Self-Attentive sequential recommender (SASRec) [8] is a recent recommendation system that is based on self-attention mechanism. This model is the most related work to ours since it's also a model solely based on self-attention, without relying on other deep neural networks such as RNNs and CNNs. However, it differs from our work because it does not consider the temporal contextual influences on user preferences.

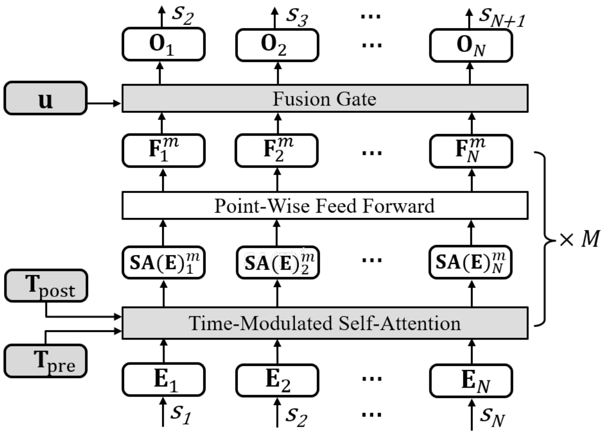


Fig. 2: The architecture of TimeSAN.

III. Problem Definition

Let $\mathcal{U}$ be a set of users, $\mathcal{L}$ be a set of POIs, and $\mathcal{S}$ be a set of check-ins post by users in $\mathcal{U}$. The check-ins of a user u form a chronologically ordered check-in sequence: $S^u = (\langle s_1^u, t_1^u \rangle, \langle s_2^u, t_2^u \rangle, \dots, \langle s_{|S^u|}^u, t_{|S^u|}^u \rangle)$. The tuple $\langle s_i^u, t_i^u \rangle$, $i \in [1, |S^u|]$ represents the i-th check-in of user u, where s_i^u represents the POI ID, and t_i^u represents the timestamp of the check-in. Given a recommendation time t_{rec}, the task of *time-aware POI recommendation* is to recommend a POI $l_i \in \mathcal{L}$, such that $r(l_i|u, S^u, t_{rec}) > r(l_j|u, S^u, t_{rec})$, $\forall l_j \in \mathcal{L}, j \neq i$. Here, $r(l_i|u, S^u, t_{rec})$ represents the ranking score of POI l_i for user u at time t_{rec}.

IV. Model architecture

Our proposed model, TimeSAN, treats the task of time-aware next POI recommendation as a sequence-to-sequence translation task which aims to convert an input sequence $(s_1, \dots, s_N)$ to a target sequence $(s_2, \dots, s_{N+1})$. When making a recommendation for user u, the input sequence $(s_1, \dots, s_N)$ will be constructed from u's N most recent check-ins, and s_{N+1} in the output sequence will be the recommendation for u. We illustrate the architecture of our proposed TimeSAN model in Figs. 2. Next, we detail the structure of the proposed model.

A. Embedding Layer

1) Embedding of a POI sequence: TimeSAN maintains two embedding matrices: a POI embedding matrix $\mathbf{L} \in \mathbb{R}^{|\mathcal{L}| \times d}$ and a positional encoding matrix $\mathbf{P} \in \mathbb{R}^{|N| \times d}$, where d represents the latent dimensions of the model. Given an input sequence $S = (s_1, \dots, s_N)$, the latent embedding of the i-th check-in is computed as $\mathbf{E}_i = \mathbf{L}(s_i) + \mathbf{P}(i)$, where $\mathbf{L}(s_i)$ represents the embedding vector of POI s_i and $\mathbf{P}(i)$ represents the positional encoding of the index i. Then the embedding of S is computed as the concatenation of the latent embeddings of the N check-ins: $\mathbf{E} = [\mathbf{E}_1; \mathbf{E}_2; \dots; \mathbf{E}_N]$.

2) Embedding of temporal context: In addition to the input and target check-in POI sequences, we also extract two sequences of temporal contexts: $T_{pre} = \{t_1, \dots, t_N\}$ and $T_{post} = \{t_2, \dots, t_{N+1}\}$. Note that we follow previous studies and assume that the recommendation time t_{N+1} is known for the time-aware recommendation purpose [24, 27]. In real applications, the actual recommendation time can be used as t_{N+1}.

We embed T_{pre} and T_{post} into a latent space as follows. Given a timestamp t, we first map t to a multi-dimensional feature vector, where the features can be chosen based on the temporal characteristics of users' check-ins. In this work, we map a timestamp t to a four-dimensional feature vector:

$$f(t) = [f_m, f_d, f_w, f_h] \tag{1}$$

Here, f_m, f_d, f_w, f_h represent the "month of a year", "day of a month", "day of a week", and "hour of a day" of t. For instance, the timestamp "2019-Mar-12 Friday 10:10:56" is mapped to vector $[3, 12, 5, 10]$. This vector forms an original feature vector of the timestamp.

The mapping in Eq. 1 is a straightforward way to represent timestamps. However, we find that it is not easy for neural networks to learn the *relative* distance between timestamps using such representations. A main reason is that the similarity of timestamps is computed as their dot-products, which do not directly reflect the relative distances between vectors. Inspired by previous studies [20], we propose a multi-scale periodic time encoding that maps the feature vector of a timestamp with a set of sin and cos functions at different frequencies. In particular, we expand each feature (e.g., f_m) in $f(t)$ to a multi-dimensional vector (e.g., $\mathbf{f_m}$), where the i-th dimension (e.g., $\mathbf{f_m}(f_m, i)$) is computed through a sin/cos function:

$$\mathbf{f_x}(f_x, 2i) = \texttt{sin}(f_x / w_x^{2i/d_f}) \tag{2}$$

$$\mathbf{f_x}(f_x, 2i+1) = \texttt{cos}(f_x / w_x^{2i/d_f}) \tag{3}$$

Here, $x \in \{m, d, w, h\}$ represents a feature from the feature set, d_f represents the latent dimensionality of each feature, and w_x represents the width of value range of the corresponding feature (e.g., $w_m = 12$). The final time encoding of timestamp t is computed as a $4d_f$-dimensional vector $\mathbf{f}(t)$ by concatenating the expanded vectors of all features:

$$\mathbf{f}(t) = [\mathbf{f_m}, \mathbf{f_d}, \mathbf{f_w}, \mathbf{f_h}] \tag{4}$$

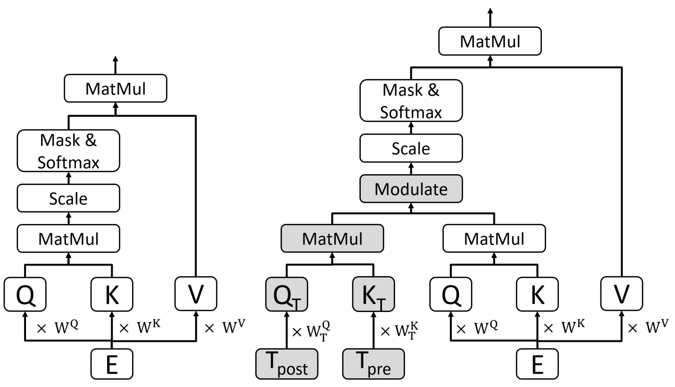
As shown by the evaluation conducted later in Section V, the proposed multi-scale time encoding in Eq. 4 is more effective in capturing the features of temporal contexts compared with the encoding in Eq. 1. This is because the proposed time encoding is much more sensitive to the relative distance, since it allows the neural networks to examine the difference of these timestamps at different periodic scales. With Eq. 4, we compute the embeddings for the temporal contexts T_{pre} and T_{post} as follows:

$$\mathbf{T}_{\text{pre}} = [\mathbf{f}(t_1); \mathbf{f}(t_2); \dots; \mathbf{f}(t_N)] \tag{5}$$

$$\mathbf{T}_{\text{post}} = [\mathbf{f}(t_2); \mathbf{f}(t_3); \dots; \mathbf{f}(t_{N+1})] \tag{6}$$

B. Time-Modulated Self-Attention

We propose a time-aware self-attention module to capture the temporal dynamics in the inputs. It computes the attention among temporal contexts first, and then let the temporal



(a) Self-attention in [20] (b) Time-modulated self-attention

Fig. 3: Comparison of self-attention in [20] and time-modulated attention in TimeSAN.

attention modulate the self-attention of the input sequence. We illustrate the comparison of the self-attention in [20] and the time-modulated attention in TimeSAN in Fig. 3.

Specifically, we compute three linear projections of $\mathbf{E}$, yielding a query $\mathbf{Q}$, a key $\mathbf{K}$, and a value $\mathbf{V}$ representation of the check-in sequence, respectively. The three representations are computed as: $\mathbf{Q} = \mathbf{E}\mathbf{W}^Q$, $\mathbf{K} = \mathbf{E}\mathbf{W}^K$, and $\mathbf{V} = \mathbf{E}\mathbf{W}^V$, where $\mathbf{W}^Q \in \mathbb{R}^{d \times d}$, $\mathbf{W}^K \in \mathbb{R}^{d \times d}$, and $\mathbf{W}^V \in \mathbb{R}^{d \times d}$ represent three learnable projection matrices. We also compute a query representation $\mathbf{Q}_{\mathrm{T}}$ and a key representation $\mathbf{K}_{\mathrm{T}}$, from $\mathbf{T}_{\mathrm{post}}$ and $\mathbf{T}_{\mathrm{pre}}$, respectively:

$$\mathbf{Q}_T = \mathbf{T}_{\mathrm{post}} \cdot \mathbf{W}_T^Q \tag{7}$$

$$\mathbf{K}_T = \mathbf{T}_{\mathrm{pre}} \cdot \mathbf{W}_T^K \tag{8}$$

where $\mathbf{W}_T^Q \in \mathbb{R}^{4d_f \times d_t}$ and $\mathbf{W}_T^K \in \mathbb{R}^{4d_f \times d_t}$ are two linear projection matrices. Then the time-modulated self-attention $\mathbf{SA}(\mathbf{E})$ of the input sequence is computed as:

$$\mathbf{SA}(\mathbf{E}) = \mathtt{softmax}\big(\frac{\mathtt{mod}(\mathbf{Q} \cdot \mathbf{K}^\top, \mathbf{Q}_T \cdot \mathbf{K}_T^\top)}{\sqrt{d}}\big) \cdot \mathbf{V} \tag{9}$$

The choice of the modulate function $\mathtt{mod}(\cdot, \cdot)$ depends on the characteristics of datasets. We use element-wise multiplication wrapped with a softsign activation function, since it yields best results in our empirical experiments:

$$\mathtt{mod}(\mathbf{X}_1, \mathbf{X}_2) = \mathtt{softsign}(\mathbf{X}_1 \odot \mathbf{X}_2) \tag{10}$$

Here, $\odot$ represents element-wise multiplication of matrices, $\mathbf{X}_1$ and $\mathbf{X}_2$ represent the input matrices of the function. We compare the proposed time-modulated self-attention with the self-attention in [20] in Fig. 3. The intuition behind the time-modulation mechanism is that we not only consider the positional attention ($\mathbf{Q} \cdot \mathbf{K}^\top$), but also the attention between the target and history temporal context ($\mathbf{Q}_T$ and $\mathbf{K}_T$). For example, if the target temporal context is Saturday, the modulation mechanism allows increasing (or decreasing) the contribution of a Friday (or Wednesday) check-in to the output, since the temporal context Friday (or Wednesday) is close (or far) from Saturday.

In real applications, the check-ins $(s_{i+1}, \ldots, s_N)$ in the input should not be known when predicting s_{i+1}. Note that s_{i+1} is the i-th element in the target sequence since the target sequence is the input sequence left-shifted by one. Thus, we disable the attention links between the i-th output and the $(i+1)$- to N-th input check-ins by applying a positional mask $\mathbf{\Lambda}$. The mask value $\mathbf{\Lambda}_{ij}$ for the attention between the i-th output and the j-th input is set as 0 if $j > i$ and 1 otherwise. Note that since the query temporal sequence T_{post} is the key temporal sequence T_{pre} left-shifted by one, the temporal context at the recommendation time is still incorporated after applying $\mathbf{\Lambda}$ to achieve time-aware recommendation.

C. Point-Wise feed forward layer

Let $\mathbf{SA}_i$ be the output on i-th position of the time-aware attention layer, the output on i-th position after the point-wise feed forward (PFF) layer is computed as:

$$\mathbf{F}_i = \mathtt{ReLU}(\mathbf{SA}_i \cdot \mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2 \tag{11}$$

where $\mathbf{W}_1 \in \mathbb{R}^{d \times d}$, $\mathbf{W}_2 \in \mathbb{R}^{d \times d}$ represent two learnable projection matrices, and $\mathbf{b}_1 \in \mathbb{R}^{d \times 1}$, and $\mathbf{b}_2 \in \mathbb{R}^{d \times 1}$ represent the biases of the first and second feed forward layer. The self-attention layer and PFF layer together form a *self-attentive block*. We allow stacking multiple blocks to capture deeper semantics in the input sequence. We use $\mathbf{SA}_i^m$ and $\mathbf{F}_i^m$ to represent the output of the attention layer, and PFF layer of the m-th block in Fig. 2.

D. Incorporating Long-Term Interest

Although self-attention mechanisms can effectively capture the semantic dependencies among input check-ins, the calculation of similarity between query and key is merely controlled by the trained parameter matrices $\mathbf{W}^Q$ and $\mathbf{W}^K$. Thus, it is difficult for self-attentive networks to capture the global features [21]. Thus, we propose a soft fusion mechanism to incorporate users' global (long-term) interest into the proposed model. Specifically, given a user check-in sequence S^u, we utilize a user embedding $\mathbf{u}$. Intuitively, this user embedding can be seen as a global bias of the sequence. We fuse this global features with the sequence representation after the M-th PFF layer to generate the output $\mathbf{O}$ of the model. Following Shaw et al. [18], we capture the global context via addition:

$$\mathbf{O} = \boldsymbol{\lambda} \odot \mathbf{F}^M + (1 - \boldsymbol{\lambda}) \odot \mathbf{u} \tag{12}$$

where $\mathbf{F}^M$ is the output of the PFF layer of the M-th block. The weight matrix $\boldsymbol{\lambda}$ controls the importance of each element in the final output. To ensure a soft and smooth fusion, we learn the values in $\boldsymbol{\lambda}$ as follows:

$$\boldsymbol{\lambda} = \mathtt{sigmoid}(\mathbf{W}^{(\lambda 1)}\mathbf{F}^M + \mathbf{W}^{(\lambda 2)}\mathbf{u} + \mathbf{b}^\lambda) \tag{13}$$

Here, $\mathbf{W}^{(\lambda 1)} \in \mathbb{R}^{d \times d}$ and $\mathbf{W}^{(\lambda 2)} \in \mathbb{R}^{d \times d}$ are two learnable projection matrices, and $\mathbf{b}^\lambda \in \mathbb{R}^d$ represents the bias vector.

E. Model training

The final output $\mathbf{O} \in \mathbb{R}^{N \times d}$ is a latent representation of the predicted sequence. Let $\mathbf{O}_i$ $(i \in [1, N])$ be the i-th vector in $\mathbf{O}$ and l_j be a POI in $\mathcal{L}$, we compute the ranking score $r(l_j | u, S^u_{1:i}, t_i)$ of l_j being recommended given user u, the user's previous check-ins $S^u_{1:i}$, and the recommendation time t_i, as follows:

$$r(l_j | u, S^u_{1:i}, t_i) = \mathbf{O}_i \cdot \mathbf{L}(l_j) \tag{14}$$

To train the model, we first take the most recent $N+1$ check-ins of each user to form a training sequence $\widetilde{S}$. If a user has less than $N+1$ check-ins, we pad a dummy check-in to the sequence until it reaches $N+1$. For each constructed sequence, we generate a training instance, where the input sequence $\widetilde{S}_{in} = \widetilde{S}_{1:N}$, and the target sequence $\widetilde{S}_{pos} = \widetilde{S}_{2:N+1}$. We adopt negative sampling to accelerate the training process. Specifically, for each POI in the target sequence, we generate a negative POI. These sampled POIs form a negative sequence $\widetilde{S}_{neg}$. We adopt the binary cross-entropy loss as the objective function of the training process:

$$-\sum_{\widetilde{S} \in \mathcal{S}} \sum_{i \in [1,N]} \delta(\widetilde{S}, i) \Big(\log\big(\sigma(r_{pos,i})\big) + \log(1 - \sigma\big(r_{neg,i})\big) \Big) \tag{15}$$

Here, $r_{pos,i}$ represents the ranking score of the i-th POI in $\widetilde{S}_{pos}$, and $r_{neg,i}$ represents the ranking score of the i-th POI in $\widetilde{S}_{neg}$, and $\delta(\widetilde{S}, i)$ is an indicator which equals to 0 if $\tilde{s}_i$ is a dummy check-in. The model is trained using Stochastic Gradient Descend (SGD).

V. Evaluation Using Synthetic Data

In this section, we use synthetic data to study the capability of capturing temporal dynamics of TimeSAN. Consider a simple sequential prediction task. Given a sequence $S = \{\langle y_1, x_1 \rangle, \ldots, \langle y_N, x_N \rangle\}$ where y_i represents the observation value and x_i represents the context value of the i-th element, we aim to predict the target y_{N+1} given its context x_{N+1}. We require that $x_1 < x_2 < \ldots < x_{N+1}$ to keep the context sequence ordered. We use a simple periodic function to generate the observation data:

$$y_i = sin(2\pi \cdot x_i / 100) + \epsilon \tag{16}$$

Here, ϵ represents random noise which conforms Gaussian distribution: $\epsilon \sim \mathcal{N}(0, 0.01)$. We generate 1,280 random sequences, where we first generate context values by drawing random samples from $[0, 10^3]$ and then compute the corresponding observation values using Eq. 16. We set the number of elements in each sequence as 250. We treat the last element in each sequence as testing data, the second last element as validation data, and use the rest of the elements as training data. We use the mean square error (MSE) as the loss function, and record the MSE of all tested models. In this experiment, we consider three methods: (1) the proposed TimeSAN model; (2) the proposed TimeSAN model without using the proposed multi-scale time encoding; and (3) vanilla self-attention in [20]. We set the attention dimensionality d as 10 for all methods, the feature dimensionality d_f as 10, the time dimensionality d_t as 1 for method (1).

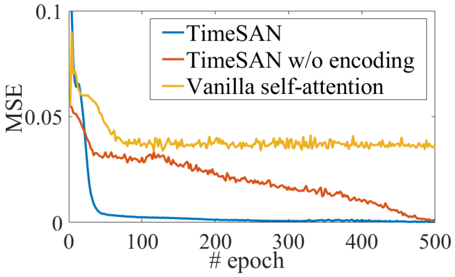


Fig. 4: Predicting accuracy vs. number of training epochs.

For all methods, we present their MSEs vs. number of training epochs in Fig. 4. The results confirm that TimeSAN is able to capture the temporal contexts effectively. Specifically, its MSE is much lower than that of vanilla self-attention. It also converges most quickly. Method (2) can reach almost the same accuracy as TimeSAN, but it converges much slower than TimeSAN: it reaches similar accuracy with that of TimeSAN after 500 epochs of training. This confirms that the proposed time encoding method can help the networks to capture the features of temporal contexts more effectively.

VI. Experiments

We evaluate the effectiveness and the efficiency of the proposed algorithms empirically in this section.

A. Datasets and Evaluation Protocols

1) Datasets: We conduct experiments using two publicly available large-scale LBSN checkin datasets, Foursquare and Gowalla, to show the general applicability of the proposed algorithms across different LBSN platforms. The Foursquare dataset [22] contains checkins in Tokyo from April 2012 to February 2013. The Gowalla dataset [3] contains checkins from March 2009 to October 2010. We follow previous studies [7, 27] and remove users who have less than 20 check-ins. Table I summarizes the statistics of the two datasets. We split the check-in sequence S^u for each user u into three parts: (1) the most recent check-in $S^u_{|S_u|}$ for testing, (2) the second most recent check-in $S^u_{|S_u|-1}$ for validation, and (3) all remaining check-ins for training.

TABLE I: Summarization of dataset statistics.

Dataset	# user	# POI	# check-in	avg. # check-in/user
Foursquare	2,293	61,858	573,703	250.2
Gowalla	21,809	34,140	1,757,997	80.6

2) Evaluation protocols: We report the performances of all models in two popular top-N metrics, namely **HR** (Hit Ratio) [23] and **NDCG** [12] (Normalized Discounted Cumulative Gain). Specifically, HR@K equals the percentage of the "hitted" cases in which the ground-truth POI is included in the top K recommendations. NDCG@K is a position-aware metric that penalizes if the ground-truth POI is hitted but ranks low in the top K recommendations.

B. Baselines

We compare our proposed TimeSAN model with three groups of baseline models: (1) factorization based models; (2) RNN based models; and (3) self-attention based models. We summarize the baseline models as follows:

1) Factorization based models: The first group contains models based on factorization:

Bayesian Pairwise Ranking (BPR). BPR is a classical model for learning personalized rankings from implicit feedback [16].

Factorized Personalized Markov Chain (FPMC). FPMC is a ranking-based pairwise tensor factorization framework, which jointly considers the first-order Markov Chain transition probabilities between POIs, and users' long-term interest [17].

Spatial-Temporal Latent Ranking (STELLAR). STELLAR is also based on tensor factorization. In addition to users' long-term interests and POI-to-POI transition probabilities, it also considers the influence of temporal contexts [27]. The model proposes a binary encoding for timestamps, and then learns the latent representations of the encoded timestamps to make recommendations.

2) RNN based models: The second group contains models based on RNN.

Vanilla RNN (RNN). RNN is a traditional recurrent model, which only considers the chronological order of user check-ins, but ignores temporal information associated with the check-ins [25].

Time-LSTM (TLSTM).[3] Time-LSTM is based on Long-Short Term Memory (LSTM). The model uses time gates to control the influences of the historical check-ins on users' future check-ins [28].

Attention-Based Spatiotemporal LSTM (ASLSTM).[4] ASLSTM extends LSTM models with the attention mechanism, making LSTM to focus on relevant historical check-in records selectively. The model also incorporates spatial and temporal contextual information [7].

3) Self-Attention based models: The final group contains models based on self-attention.

Self-Attentive Sequential Recommender (SASRec). SASRec[5] is a sequential recommender that captures the sequential patterns in user preferences using multi-head self-attention [20].

C. Implementation Details

We set the maximum sequence length N as 250 and 150 in Foursquare and Gowalla, respectively. We use Adam optimizer [9] for training. We optimize the performance of TimeSAN and SASRec by varying the number of blocks from $\{1, 2, 3\}$, the number of attention heads from $\{1, 2, 3\}$, dropout rate from $\{0, 0.1, 0.2, 0.5\}$. For all models, we optimize their performances, varying the latent dimensionality d from $\{10, 20, 50, 80, 100\}$, learning rate from $\{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$. We fix the batch size as 128 for all models to ensure fair comparison. We run each set of experiments for 10 times, and record the average performace of each model.

D. Experimental Results

1) Overall recommendation accuracy: We summarize the overall recommendation accuracy of all models in Table II. For each metric, we highlight the best result in **bold**, and underline the second best result. The experimental results show that the proposed TimeSAN model outperforms all baselines consistently and significantly. Specifically, TimeSAN achieves up to 17.9% and 12.2% improvement over SASRec, on Foursquare and Gowalla respectively. In addition, we find that models which incorporate temporal dynamics (e.g., Stella, TLSTM, ASLSTM) have higher recommendation accuracy, compared with the other models within the same group. This confirms the importance of learning the influence of temporal contexts. Finally, we observe that factorization based methods tend to perform better compared with RNN based methods. This is consistent with the findings in previous studies [8, 19]. A main reason is that real-world check-in datasets are usually very sparse, which make RNN based methods ineffective due to overfitting.

2) Ablation study: Table III summarizes the results of ablation study on TimeSAN. In the default setting of TimeSAN, we set the number of blocks as 2 ($M = 2$). We do not use positional encoding (PE) in the default setting of TimeSAN, because the time-modulated attention module already considers the chronological information of the check-in sequences. We see that adding positional encoding to TimeSAN degrades the overall performance slightly. This is because TimeSAN utilizes a time-modulation attention module and does not rely on the positional encoding to capture the chronological information about the inputs. This also confirms that the proposed time-modulation attention module is effective in capturing time dynamics in the input check-in sequences. The components of user embedding (cf. Section IV-D) and time-modulation (cf. Section IV-B) both contribute positively to the overall performance. The contribution of user embedding on Foursquare is more significant compared with that on Gowalla. This is because Foursquare is denser than Gowalla, where each user has a longer check-in sequence. Therefore, it is more important to use a user embedding to capture the long-term context in the input sequence. When varying the number of time-modulated self-attention blocks of TimeSAN, we observe a significant degradation in performance if no self-attention block is used ($M = 0$). This is because when $M = 0$, TimeSAN will make prediction solely based on users' most recent check-in and the user's embedding which represents user's long-term interest, ignoring the contextual information carried by non-neighboring check-ins. When we set $M > 0$, we find that the performance on both datasets degrades when $M = 3$ due to overfitting. On Foursquare dataset, the optimum value of M is 2 (as in default $M = 2$), while the optimum value of M is 1 in Gowalla. This is because if the dataset is dense, deep neural structure is needed to capture the complex interaction.

[3] https://github.com/ZJULearning/time_lstm

[4] https://github.com/drhuangliwei/An-Attention-based-Spatiotemporal-LSTM-Network-for-Next-POI-Recommendation

[5] https://github.com/kang205/SASRec

TABLE II: Overall recommendation accuracy of all baseline models and TimeSAN (proposed).

Dataset	Metric	(1) Factorization-based			(2) RNN-based			(3) Self-attention-based		Improv.	Improv.	Improv.
		BPR	FPMC	Stellar	RNN	TLSTM	ASLSTM	SASRec	TimeSAN	over (1)	over (2)	over (3)
Foursquare	HR@5	0.1801	0.2268	0.2700	0.2157	0.2396	0.2166	0.2686	**0.3144**	16.4%	31.2%	17.1%
	NDCG@5	0.1245	0.1788	0.2006	0.1543	0.1631	0.1471	0.1837	**0.2165**	7.9%	32.7%	17.9%
	HR@10	0.2673	0.2787	0.3341	0.2934	0.3281	0.3238	0.3689	**0.4091**	22.4%	26.3%	10.9%
	NDCG@10	0.1526	0.1804	0.2214	0.1794	0.1918	0.1819	0.2160	**0.2465**	11.3%	28.5%	14.1%
	HR@20	0.3567	0.3450	0.4091	0.3602	0.4145	0.4288	0.4387	**0.4784**	16.9%	11.6%	9.0%
	NDCG@20	0.1751	0.1971	0.2404	0.1963	0.2129	0.2085	0.2337	**0.2641**	9.9%	24.0%	13.0%
Gowalla	HR@5	0.4121	0.4323	0.4345	0.3607	0.3737	0.3997	0.4242	**0.4629**	6.5%	15.8%	9.1%
	NDCG@5	0.3136	0.3324	0.3362	0.2784	0.2913	0.3003	0.3267	**0.3664**	9.0%	22.0%	12.2%
	HR@10	0.4837	0.5070	0.5101	0.4373	0.4630	0.5115	0.5040	**0.5382**	5.5%	5.2%	6.8%
	NDCG@10	0.3367	0.3566	0.3607	0.3032	0.3347	0.3499	0.3522	**0.3908**	8.3%	11.7%	11.0%
	HR@20	0.5497	0.5751	0.5832	0.5159	0.5395	0.5684	0.5833	**0.6061**	3.9%	6.6%	3.9%
	NDCG@20	0.3534	0.3739	0.3792	0.3230	0.3477	0.3601	0.3722	**0.4079**	7.6%	13.3%	9.6%

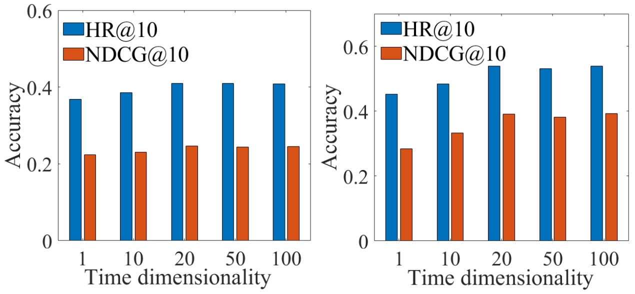
TABLE III: Ablation study of TimeSAN. With PE: add positional encoding; w/o user: remove user embedding; w/o time: remove time-modulation. Performance improvement is highlighted in bold. Significant performance degradation (more than 10%) is marked with ↓.

setting	Foursquare		Gowalla	
	HR@10	NDCG@10	HR@10	NDCG@10
default ($M = 2$)	**0.4091**	**0.2465**	0.5382	0.3908
with PE	0.3977	0.2414	0.5314	0.3854
w/o user	0.3947	0.2393	0.5322	0.3816
w/o time	0.3803	0.2317	0.5183	0.3726
M = 0	0.3650↓	0.2187↓	0.4778↓	0.3373↓
M = 1	0.3951	0.2374	**0.5427**	**0.3913**
M = 3	0.3929	0.2408	0.5216	0.3673

3) Impact of time dimensionality d_t: We vary the value of d_t for TimeSAN from $[1, 10, 20, 50, 100]$ and show the corresponding HR@10 and NDCG@10 in Fig. 5. In this set of experiments, we fix the latent dimensionality d at 100, and the number of self-attentive blocks M at 2. The other settings are as described in Section VI-C. As shown in the figure, TimeSAN reaches the highest recommendation accuracy when $d_t = 20$. The recommendation accuracy of the model becomes lower when the value of d_t decreases from 20 to 1, because the time-modulation module is not able to capture the complex temporal dynamics with limited dimensions. When d_t increases beyond 20, there is also a slight fluctuation in recommendation accuracy, but overall, the recommenation accuracy remains stable and does not increase with d_t anymore.

TABLE IV: Training speed of all models (sec/epoch).

TimeSAN (M=1)	TimeSAN (M=2)	SASRec	TLSTM	ASLSTM	RNN
0.57	0.91	0.44	5.9	5.8	3.6



(a) Foursquare

(b) Gowalla

Fig. 5: Varying the latent dimensionality d_t.

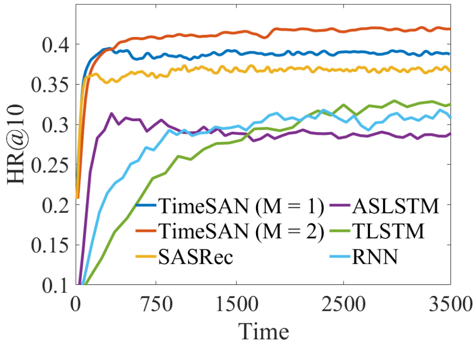


Fig. 6: Performances (HR@10) of all deep neural network based models on Foursquare dataset.

4) Training efficiency: We evaluate the training efficiency of the proposed model and all baseline models that are based on deep neural networks, i.e., Groups (2) and (3). We illustrate the HR@10 on Foursquare dataset of these models as a function of training time in Fig. 6. For TimeSAN, we present the results where the number of blocks are set as 1 and 2, namely TimeSAN ($M = 1$) and TimeSAN ($M = 2$), respectively. For SASRec, we fix the number of blocks as 1 ($M = 1$), since the performance of SASRec decreases on Foursquare dataset if we increase M beyond

1. The results confirm that the proposed model retains high training efficiency compared with the baseline models that are also based on deep neural networks. Specifically, the training speeds (sec/epoch) of TimeSAN ($M = 1$), TimeSAN ($M = 2$), SASRec, TLSTM, ASLSTM, and RNN are 0.57, 0.91, 0.44, 5.9, 5.8, and 3.6, respectively. TimeSAN ($M = 1$) has a very close training speed to SASRec, and is only slightly slower due to additional parameters introduced by the time-modulation module. TimeSAN ($M = 2$) is slightly more slower since it has two blocks, and is doubled in the model size. Compared with RNN based methods, TimeSAN has much higher training speed. In particular, compared with the two models TLSTM and ASLSTM, which also capture temporal contexts, TimeSAN ($M = 1$) is faster by an order of magnitude. In addition to training speed, TimeSAN also converges much faster compared with RNN based models.

VII. Conclusions

We proposed a time-modulated self-attention network, TimeSAN, for time-aware next POI recommendation. TimeSAN uses a time-modulated self-attention module to effectively learn the temporal dynamics in users' interests. Specifically, TimeSAN learns the relevance between users' history records and next POI visits via the self-attention mechanism, and modulates the relevance with the temporal contexts. Further, we propose a fusion mechanism to enable more effective learning of users' long-term interests. We demonstrate that the proposed time-modulated self-attention module can effectively capture the influence of temporal contexts using synthetic data. We also conduct extensive experiments on real-world check-in datasets. The experimental results show that the proposed TimeSAN model outperforms the state-of-the-art recommendation models consistently and significantly in recommendation accuracy, while retains high training efficiency.

References

[1] J. Chen, H. Zhang, X. He, L. Nie, W. Liu, and T.-S. Chua. Attentive collaborative filtering: Multimedia recommendation with item-and component-level attention. In *SIGIR*, 2017.

[2] X. Chen, Y. Zeng, G. Cong, S. Qin, Y. Xiang, and Y. Dai. On information coverage for location category based point-of-interest recommendation. In *AAAI*, 2015.

[3] E. Cho, S. A. Myers, and J. Leskovec. Friendship and mobility: user movement in location-based social networks. In *SIGKDD*, 2011.

[4] S. Feng, X. Li, Y. Zeng, G. Cong, Y. M. Chee, and Q. Yuan. Personalized ranking metric embedding for next new poi recommendation. In *IJCAI*, 2015.

[5] H. Gao, J. Tang, X. Hu, and H. Liu. Exploring temporal effects for location recommendation on location-based social networks. In *RecSys*, 2013.

[6] B. Hidasi and A. Karatzoglou. Recurrent neural networks with top-k gains for session-based recommendations. In *CIKM*, 2018.

[7] L. Huang, Y. Ma, S. Wang, and Y. Liu. An attention-based spatiotemporal lstm network for next poi recommendation. In *IEEE Trans. on Services Computing*, 2019.

[8] W.-C. Kang and J. McAuley. Self-attentive sequential recommendation. In *ICDM*, 2018.

[9] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *arXiv preprint arXiv:1412.6980*, 2014.

[10] D. Lian, C. Zhao, X. Xie, G. Sun, E. Chen, and Y. Rui. Geomf: joint geographical modeling and matrix factorization for point-of-interest recommendation. In *SIGKDD*, 2014.

[11] B. Liu, Y. Fu, Z. Yao, and H. Xiong. Learning geographical preferences for point-of-interest recommendation. In *SIGKDD*, 2013.

[12] N. N. Liu and Q. Yang. Eigenrank: a ranking-oriented approach to collaborative filtering. In *SIGIR*, 2008.

[13] Q. Liu, S. Wu, L. Wang, and T. Tan. Predicting the next location: A recurrent model with spatial and temporal contexts. In *AAAI*, 2016.

[14] X. Liu, Y. Liu, K. Aberer, and C. Miao. Personalized point-of-interest recommendation by mining users' preference transition. In *CIKM*, 2013.

[15] E. Palumbo, G. Rizzo, R. Troncy, and E. Baralis. Predicting your next stop-over from location-based social network data with recurrent neural networks. In *RecSys*, 2017.

[16] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *UAI*, 2009.

[17] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme. Factorizing personalized markov chains for next-basket recommendation. In *WWW*, 2010.

[18] P. Shaw, J. Uszkoreit, and A. Vaswani. Self-attention with relative position representations. In *NAACL*, 2018.

[19] J. Tang and K. Wang. Personalized top-n sequential recommendation via convolutional sequence embedding. In *WSDM*, 2018.

[20] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

[21] B. Yang, J. Li, D. F. Wong, L. S. Chao, X. Wang, and Z. Tu. Context-aware self-attention networks. In *AAAI*, 2019.

[22] D. Yang, D. Zhang, V. W. Zheng, and Z. Yu. Modeling user activity preference by leveraging user spatial temporal characteristics in lbsns. In *IEEE Trans. on Systems, Man, and Cybernetics: Systems*, 2014.

[23] F. Yuan, A. Karatzoglou, I. Arapakis, J. M. Jose, and X. He. A simple convolutional generative network for next item recommendation. In *WSDM*, 2019.

[24] Q. Yuan, G. Cong, Z. Ma, A. Sun, and N. M. Thalmann. Time-aware point-of-interest recommendation. In *SIGIR*, 2013.

[25] Y. Zhang, H. Dai, C. Xu, J. Feng, T. Wang, J. Bian, B. Wang, and T.-Y. Liu. Sequential click prediction for sponsored search with recurrent neural networks. In

AAAI, 2014.

[26] S. Zhao, T. Zhao, I. King, and M. R. Lyu. Geo-teaser: Geo-temporal sequential embedding rank for point-of-interest recommendation. In *WWW*, 2017.

[27] S. Zhao, T. Zhao, H. Yang, M. R. Lyu, and I. King. Stellar: spatial-temporal latent ranking for successive point-of-interest recommendation. In *AAAI*, 2016.

[28] Y. Zhu, H. Li, Y. Liao, B. Wang, Z. Guan, H. Liu, and D. Cai. What to do next: Modeling user behaviors by time-lstm. In *IJCAI*, 2017.