A Category-Aware Deep Model for Successive POI Recommendation on Sparse Check-in Data

Fuqiang Yu
School of Software & Joint SDU-NTU Centre for Artificial Intelligence Research (C-FAIR)
Shandong University, Jinan, China
yfq_sdu@163.com

Lizhen Cui*
School of Software & Joint SDU-NTU Centre for Artificial Intelligence Research (C-FAIR)
Shandong University, Jinan, China
clz@sdu.edu.cn

Wei Guo
School of Software & Joint SDU-NTU Centre for Artificial Intelligence Research (C-FAIR)
Shandong University, Jinan, China
guowei@sdu.edu.cn

Xudong Lu
School of Software & Joint SDU-NTU Centre for Artificial Intelligence Research (C-FAIR)
Shandong University, Jinan, China
dongxul@sdu.edu.cn

Qingzhong Li
School of Software & Joint SDU-NTU Centre for Artificial Intelligence Research (C-FAIR)
Shandong University, Jinan, China
lqz@sdu.edu.cn

Hua Lu*
Department of Computer Science
Aalborg University, Aalborg, Denmark
luhua@cs.aau.dk

ABSTRACT

As considerable amounts of POI check-in data have been accumulated, successive point-of-interest (POI) recommendation is increasingly popular. Existing successive POI recommendation methods only predict where user will go next, ignoring when this behavior will occur. In this work, we focus on predicting POIs that will be visited by users in the next 24 hours. As check-in data is very sparse, it is challenging to accurately capture user preferences in temporal patterns. To this end, we propose a category-aware deep model *CatDM* that incorporates POI category and geographical influence to reduce search space to overcome data sparsity. We design two deep encoders based on LSTM to model the time series data. The first encoder captures user preferences in POI categories, whereas the second exploits user preferences in POIs. Considering clock influence in the second encoder, we divide each user's check-in history into several different time windows and develop a personalized attention mechanism for each window to facilitate CatDM to exploit temporal patterns. Moreover, to sort the candidate set, we consider four specific dependencies: user-POI, user-category, POI-time and POI-user current preferences. Extensive experiments are conducted on two large real datasets. The experimental results demonstrate that our CatDM outperforms the state-of-the-art models for successive POI recommendation on sparse check-in data.

CCS CONCEPTS

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

*Corresponding Authors

WWW '20, April 20–24, 2020, Taipei, Taiwan

ACM ISBN 978-1-4503-7023-3/20/04.
https://doi.org/10.1145/3366423.3380202

KEYWORDS

POI recommendation, sparse data, category-aware, deep model

ACM Reference Format:
Fuqiang Yu, Lizhen Cui, Wei Guo, Xudong Lu, Qingzhong Li, and Hua Lu. 2020. A Category-Aware Deep Model for Successive POI Recommendation on Sparse Check-in Data. In *Proceedings of The Web Conference 2020 (WWW '20), April 20–24, 2020, Taipei, Taiwan.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3366423.3380202

1 INTRODUCTION

Location-based social networks (LBSNs) such as GoWalla[1], JiePang[2] and Foursquare[3] have grown rapidly and become increasingly popular in recent years. On such online platforms, users are allowed to check-in at point-of-interests (POIs) using their mobile devices. As a result, LBSNs have accumulated considerable amounts of user check-in data about POI visits. Such user-generated check-in data offers valuable opportunities for understanding user behaviors and making POI recommendations to users accordingly.

Conventional POI recommendation [3] suggests POIs from a general, static point of view, ignoring a user's recent visits. In contrast, successive POI recommendation [23] is based on the intuition that a user's recent visits influence her/his next visits. However, most existing works for successive POI recommendation only predict where a user will go, ignoring when it will occur. This flaw makes such successive POI recommendation barely meaningful in many practical scenarios. For example, if the next predicted behavior of a user only occurs after a few weeks or even months, the successive POI recommendation makes little sense.

In this work, we focus on making successive POI recommendation to users in the next 24 hours, a more meaningful and rational task. However, it faces two big challenges: (1) Temporal patterns. In addition to preference variance [16, 25, 32], this task attempts to understand the temporal patterns of user behaviors throughout a day and how user behaviors change in different days in order to

[1]https://blog.gowalla.com/
[2]https://jiepang.app/
[3]https://foursquare.com/

make recommendation for the next 24 hours. Therefore, finding out how long-short-term interests affect user behaviors in the next 24 hours is the key to making meaningful recommendations. (2) High sparsity [4, 11]. Check-in data is very sparse as each user checks-in at a limited number of POIs and vice versa. For example, we randomly pick 6 users and 60 POIs from the dataset NYC (detailed in Section 5.1.1), and plot the user-POI matrix in Figure 1. The intensity of the color indicates the frequency of a user visiting a venue. Clearly, the density of check-in matrix is fairly low. Specifically, the data sparsity[4] is up to 99.51% and 99.58% on our two datasets (detailed in Section 5.1.1), respectively. With high sparse check-in data, it is highly challenging to accurately capture user preferences and recommend POI for a user to visit in subsequent hours.

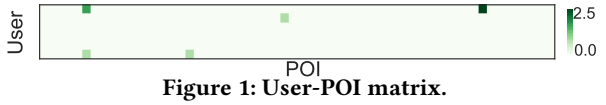


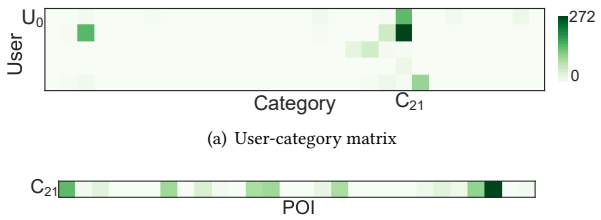
Figure 1: User-POI matrix.

Some successive POI recommendation methods [25, 29, 32] consider temporal influence of users' check-ins to better capture user preferences. However, those methods fail to capture sequential information from a user's check-in behaviors since they only model the relationship between two consecutive POIs. Recently, Recurrent Neural Networks (RNN) has shown its promising performance in modeling sequential information and correlations between POIs from a user's recent check-ins [10, 13, 31]. Employing RNN to model user's sequential check-in data is a good choice. However, it still suffers from data sparsity and lacks the ability of accurately exploiting temporal patterns mentioned above.

Our task in this study demands that we appropriately capture user's daily periodic patterns. To this end, we divide a user's check-in history into several time windows and develop a personalized attention mechanism for each time window, according to clock influence of the day. Long-short-term interests are captured by weighted averaging hidden layers of LSTM [8] in each time window.

To overcome the data sparsity problem, we incorporate context information like POI categories and geographical influence. Taking 5 users and 30 categories as examples, Figure 2(a) illustrates the user-category matrix on the dataset NYC (detailed in Section 5.1.1). We observe that POI category c_{21} is visited by user u_0 more than 150 times, whereas some other categories are visited only a few times. For further observation and analysis, we show all the POIs that belong to the category c_{21} and the frequencies of u_0 visiting these POIs in Figure 2(b). It is clear from these statistics that a user visits different POIs of the same category that he/she prefers. Such a visiting pattern, i.e., a user often visits a specific POI category frequently but each individual POI of that category infrequently, should be considered in successive POI recommendation. Meanwhile, according to the observation from Figure 1 and Figure 2, the density of the user-category matrix is higher than that of the user-POI matrix. Hence, effectively utilizing the POI categories can alleviate the impact of data sparsity. In other words, although it is hard to dig out user preferences in POIs, we can make up for this deficiency by mining user preferences in POI categories. This also conforms with intuitions. For example, it is easy to imagine how much a user likes a venue can depend on the category of this venue.

[4]Data sparsity is the percentage of missing values in a data matrix.



(a) User-category matrix

(b) Frequencies of U_0 visiting the POIs that belong to category C_{21}

Figure 2: User check-in frequency matrix.

As POI categories play an important role in capturing latent user preferences, we need to establish appropriate models for POI categories as well. Furthermore, we found that geographical influence can also be used to filter POIs and reduce search space according to the analysis in Section 4.1.

Considering all these factors, we propose a POI category-aware deep model, named CatDM, for successive POI recommendation on sparse data. CatDM contains two encoders based on LSTM. The first encoder models the time series of POI categories from sparse check-in data, and its main purpose is to capture user preferences in POI categories. The first encoder and the next two filtering layers form a reasonable filter capable of reducing search space, i.e., reducing the number of candidates from which recommended POIs are selected finally. User preferences in POI categories and geographical influence play important roles in the process of generating candidate set of POIs. Note that other approaches could also utilize our designed filter in reducing search space to improve the performance. The second LSTM encoder is able to exploit temporal patterns with a design of different time windows, and thus is utilized in modeling the time series of POIs to mine user preferences in POIs. Finally, to rank the candidates, four specific correlations are considered in calculating the probability of each POI in the candidate set: (1) correlation between user and POI; (2) correlation between user and POI category; (3) correlation between POI and temporal influence; (4) correlation between POI and user's current preferences in POIs. Furthermore, in order to overcome the lack of training data, we design a sequential loss function for the second LSTM encoder.

This work makes the following main contributions: (1) To overcome the data sparsity problem of the check-in data, we incorporate POI categories and geographical influence to reduce search space through a two-layer filter architecture, which can be used in many other approaches to improve performance. (2) We design two different LSTM encoders to mine user preferences in POI categories and POIs, respectively. To make LSTM exploit temporal patterns of user behaviors more accurately and finely, we divide a user's check-in history into several different time windows and develop a personalized attention mechanism for each time window using clock influence of the day. (3) We consider four specific correlations to rank POIs in the candidate for recommendation. (4) We conduct extensive experiments on two real-world datasets. The results demonstrate that our model outperforms the state-of-the-art methods for successive POI recommendation on sparse data.

The rest of this paper is organized as follows. Section 2 reviews the related work. Section 3 defines the problem and introduces LSTM. Section 4 details our category-aware deep model for successive POI recommendation. Section 5 reports on experiments. Section 6 concludes the paper and discusses future work directions.

2 RELATED WORK

Conventional POI Recommendation Conventional POI recommendation mainly considers geographical information [2, 9, 17, 24] and temporal influence [5, 18]. Ye et al. [24] develop a collaborative filtering (CF) based approach that recommends POIs according to their geographical influence and users' social influence. Liu et al. [14] propose a matrix factorization (MF) based approach that considers contextual and social information. Gao et al. [5] utilize nonnegative MF techniques to capture temporal effects. Yuan et al. [26] combine temporal effects with user-oriented CF and use the Bayes rule to capture the spatial-temporal influence collectively. Furthermore, Gao et al. [6] leverage user-generated contents to improve POI recommendation, and Liu et al. [15] employ MF to predict a user's preference transitions over categories. Moreover, Liu and Xiong [12] make POI recommendation by exploiting an aggregated LDA model to learn the topics that interest users.

Successive POI Recommendation Unlike conventional POI recommendation, successive POI recommendation attaches importance to a user's most recent check-in(s). Markov chain (MC) models are often used to model user sequential behaviors and to predict a user's next POI given her/his short sequence of POIs visited most recently. The factorized personalized Markov chain (FPMC) model [20] includes a common Markov chain and a normal matrix factorization model that factorizes the tensor of transition cube. Zhao et al. [32] exploit the personalized Markov chain in the check-in sequence and take user's movement constraint into account when recommending POIs. Zhang et al. [28] employ Personalized Ranking Metric Embedding (PRME) to extend FPMC by modeling user-location distance and location-location distance in two different vector spaces. Also, Zhang et al. [32] develop a ranking-based pairwise tensor factorization framework that contains a fine-grained modeling of user-POI, POI-time, and POI-POI interactions.

Nowadays, Recurrent Neural Network (RNN) is widely employed in successive POI recommendation as well, as it shows promising performance for mining sequential information [30]. Since a user's check-in records form a sequence, RNN or its variant (LSTM) is applied in modeling user behaviors. For example, Liu et al. [13] propose Spatial Temporal Recurrent Neural Networks (ST-RNN) to model local temporal and geographical influence. It utilizes two matrices, time-specific transition matrix and distance-specific transition matrix, to incorporate the influence of different time intervals and different geographical distances, respectively. Kong et al. [10] propose HST-LSTM, which can combine spatial-temporal influence, to predict the next location. Through an encoder and decoder manner, it models the visit session to improve the prediction performance. Zhao et al. [31] propose ST-CLSTM to mine the spatial-temporal relation between successive check-ins, by using time gate and distance gate to control the long term and short term interests update. However, data sparsity is still an important factor limiting model performance. Although it is possible to mine sequential information from a user's recent check-ins, it is still unknown how to exploit temporal patterns accurately. Unlike previous works, our study addresses data sparsity and designs a two-layer filter to overcome this problem. Moreover, we introduce time windows in recommendation and a personalized weighting mechanism, which is helpful for LSTM to capture the temporal patterns.

Category-aware Approach Since POI category plays an important role in user preferences, recent methods for successive POI recommendation mine user preferences for several specific POIs from POI categorical influence, temporal influence, social influence, geographical influence or a combination of part of them. Yang et al. [21] propose a location based social matrix factorization algorithm to take information of POI categories into account. Bao et al. [1] model the POI categories with a weighted category hierarchy (WCH). Zhang et al. [27] propose to incorporate geographical correlations, social correlations and categorical correlations to calculate the preference score in POI. He et al. [7] develop a category-based method to make successive POI recommendation using the geographical influence and category ranking influence.

However, these works merely take POI categories as an specific influence to improve the performance. None of them consider using user preferences in POI categories to reduce search space. In our work, we aim to capture user preferences in POI categories as well as in POIs. In particular, we devise a deep encoder to capture user preferences in POI categories. Through mining user preferences in POI categories, our method is able to reduce search space to overcome the data sparsity problem.

3 PRELIMINARIES

3.1 Problem Formulation

Let $\mathcal{U} = \{u_1, u_2, \ldots, u_M\}$ be a set of LBSN users, $\mathcal{V} = \{v_1, v_2, \ldots, v_L\}$ be a set of POIs, and $C = \{c_1, c_2, \ldots, c_P\}$ be a set of POI categories. Each POI is associated with its longitude and latitude coordinates and falls into one of the POI categories in C.

Definition 1 (Check-in). A check-in is a 4-tuple (u, v_l, c_p, t_s) where $u \in \mathcal{U}$, $v_l \in \mathcal{V}$, and $c_p \in C$. It means that the user u visited POI v_l from category c_p at a past time t_s.

Definition 2 (User Check-in History). A user's check-in history $\mathcal{CH}_u$ is the user's full sequence of check-ins, i.e.,

$$\mathcal{CH}_u = \langle (u, v_1^u, c_1^u, t_1^u), (u, v_2^u, c_2^u, t_2^u), \ldots, (u, v_n^u, c_n^u, t_n^u) \rangle$$

where $u \in \mathcal{U}$, $c_i^u \in C$ and $v_i^u \in \mathcal{V}$ for $1 \le i \le n$.

Definition 3 (Check-in Database). Given a set $\mathcal{U}$ of LBSN users, the check-in database $\mathcal{CH}_\mathcal{U}$ of $\mathcal{U}$ is the set of all users' check-in history, i.e., $\mathcal{CH}_\mathcal{U} = \{\mathcal{CH}_u \mid u \in \mathcal{U}\}$.

Problem 1 (Top-N Successive POI Recommendation). Given a user u's check-in history $\mathcal{CH}_u$, the user's current location v_i^u, the current time t_i^u, top-N successive POI recommendation returns from $\mathcal{V}$ a sequence of N POIs $\langle v_{rec_1}, v_{rec_2}, \ldots, v_{rec_N} \rangle$ such that POI v_{rec_1} is most likely to be visited by user u during the next 24 hours, POI v_{rec_2} is the second likely to be visited, and so on.

3.2 Long Short-Term Memory Cell

Recurrent neural networks (RNN) models can maintain historical information, which enables the prediction of the current output combined with previous information. However, RNN cannot process long sequential data due to the vanishing gradient problem. Therefore, we use the Long Short-Term Memory [8] (LSTM), a special kind of RNN, instead. In particular, LSTM cells are used as the basic unit in our model for top-N successive recommendation. Compared with RNN, LSTM is able to solve the long range dependencies

and deal with the vanishing gradient problem. The input, forget and output gates control content to be memorized, content to be erased, and current content to be exposed, respectively. At a time step t, the LSTM cell is implemented as:

$$
\begin{aligned}
i_t &= \sigma(\mathbf{W}_{ix}x_t + \mathbf{W}_{ih}h_{t-1} + b_i) \\
f_t &= \sigma(\mathbf{W}_{fx}x_t + \mathbf{W}_{fh}h_{t-1} + b_f) \\
o_t &= \sigma(\mathbf{W}_{ox}x_t + \mathbf{W}_{oh}h_{t-1} + b_o) \\
\hat{c}_t &= \tanh(\mathbf{W}_{cx}x_t + \mathbf{W}_{ch}h_{t-1} + b_c) \\
c_t &= f_t \odot c_{t-1} + i_t \odot \hat{c}_t \\
h_t &= o_t \odot \tanh(c_t)
\end{aligned}
$$

Specifically, i, f and o are the input, forget and output gates, respectively; $\sigma(x)$ and $\odot$ represent the element-wise sigmoid function and the element-wise product, respectively. For a given sequence data $\langle v_1, v_2, ..., v_n \rangle$ containing n locations, each location is represented as a d-dimensional vector. An LSTM computes h_t which incorporates the current information and the information before time t. These representations effectively include a representation of a location in its particular context.

4 CATEGORY-AWARE DEEP MODEL

4.1 Overall Considerations

When modeling user behavior for top-N successive POI recommendation, we consider the following four important factors.

POI category: POI categories play an important role in capturing user preferences. For example, some people prefer to go shopping in leisure time, and some prefer to exercise in sports venues. Such latent preferences about POI categories can be learned from user check-in sequences.

Long-short-term user interest: A user's behavior in the long term exhibits some core interest that does not vary temporally. For example, a user who loves coffee may visit a coffeehouse regularly. Thus, a user may have a stable lifestyle and periodically visit certain POIs. On the other hand, a user's short-term interest has an immediate influence on the user's next visits. There can be a strong correlation between the POI that a user wants to visit next and the POI(s) that the user has visited most recently.

Clock influence: Figure 3 visualizes how people visit POIs differently in different times within a day. The data was collected from New York (NYC) and Tokyo (TKY) in Foursquare. The probability within each time period is computed by the ratio of the number of check-ins at that time to the total number of check-ins by all users for the entire time horizon of a dataset. Apparently, people have different POI visit behaviors or preferences in different hours. For example, users are more likely to be interested in restaurants around lunch or dinner hours, and thus it makes less sense to recommend restaurants to users in other hours. We use clock influence to capture this kind of user interest with respect to finer temporal granularity in top-N successive POI recommendation.

Geographical influence: Figure 4(a) shows the probability of the distance between two successive check-ins in the aforementioned two datasets. Figure 4(b) shows that about 86% successively visited pairs of POIs in NYC and TKY are apart less than 10 kilometers. This observation supports that geographical distances have impact on users' POI visiting and these distances can be used to filter POIs and reduce search space. Therefore, in our model, one part is designed to filter POIs according to geographical locations.

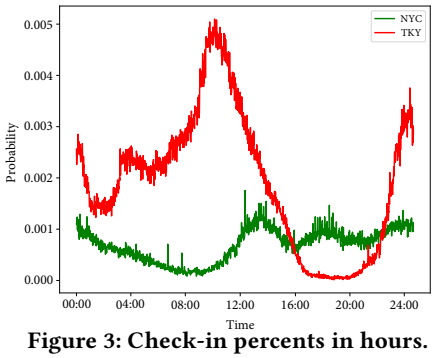
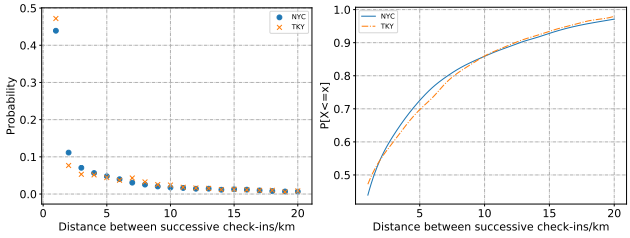


Figure 3: Check-in percents in hours.



(a) Probability (b) Cumulative distribution

Figure 4: Geographical influence in POI visiting.

4.2 Model Formulation

We propose a category-aware deep model, CatDM, to consider the aforementioned four factors. CatDM also features specific designs to overcome data sparsity. It contains two deep encoders illustrated in the left half of Figure 5, a two-layer filter depicted in the red box in the right half, and a candidate sorting component underneath.

4.2.1 Metric Embedding. As the carrier of information, the vector representation is critical to a model. However, using one-hot encoding to represent each user, POI, POI category and time will make it hard to capture user preferences due to its sparsity. To this end, a good choice is to learn low dimensional embedding for each of them individually. The latent vectors $U_u, V_v, C_c, T_{t_z^{id}} \in \mathbb{R}^d$ represent the latent features of user u, POI v, POI category c and time t_z respectively, and t_z^{id} is the encoded id of t_z. How to encode time to a specific index in time latent space is a problem to be discussed here: (1) Time stamp is discrete value. Therefore, it is unrealistic to map all time stamps to different vectors due to the huge size of index. (2) Temporal patterns need to be considered.

To overcome the above issues, similar to [32], we divide each day into 12 time periods. In addition, we differentiate the days of a week into weekday and weekend. As a result, the index size of time embedding $|T|$ is 24. However, according to clock influence, user preferences are very different during a day, even at different times of the daytime, such as morning, noon and afternoon. Thus, dividing a day into different time periods evenly is not a good choice. Figure 3 clearly demonstrates that user behaviors in the day have many ups and downs in TKY, while those in NYC are relatively uniform. Therefore, the barrier is how to adapt the dividing method to all datasets. We solve this by using the probability density of user behaviors of visits during the day to determine the span of

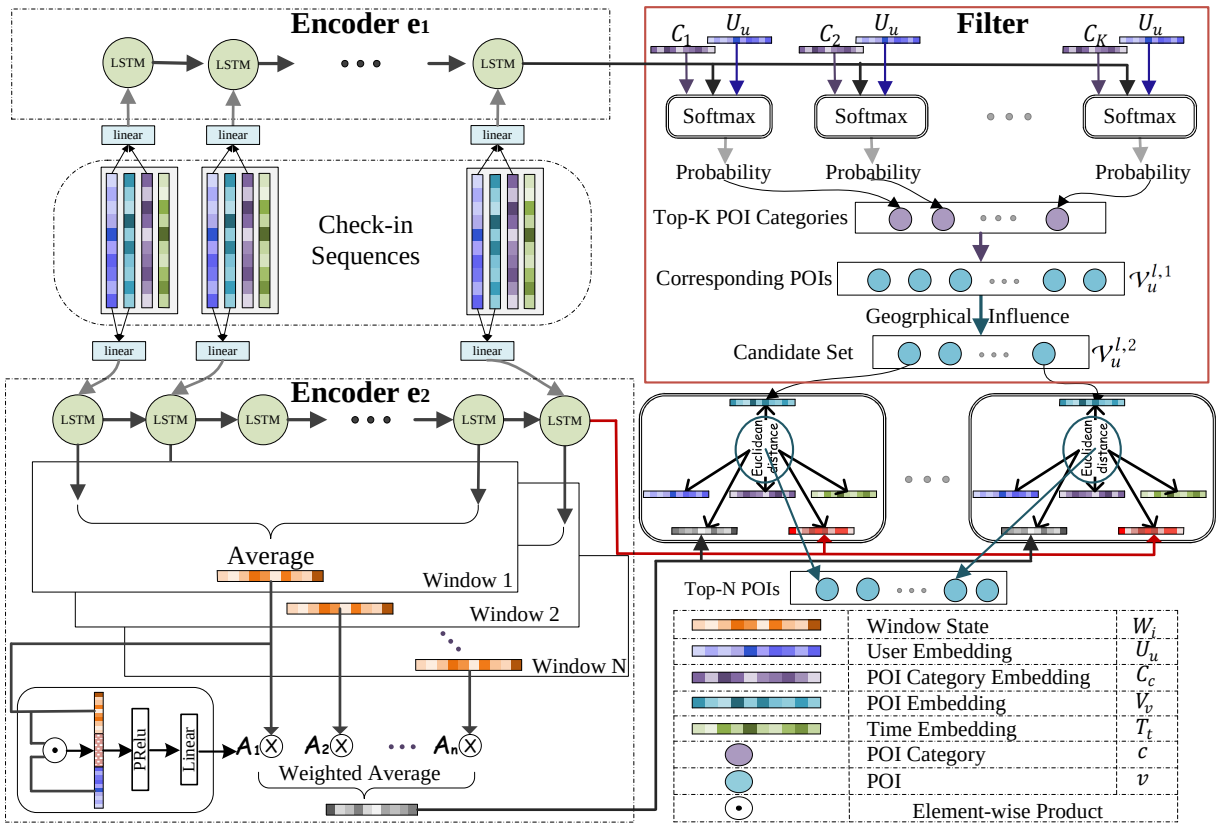


Figure 5: Category-aware deep model.

each time period. Given a time t_z, t_z^{day} represents if the time falls in weekday and t_z^m represents the specific moment of the day. The encoded index number t_z^{id} is defined as follows:

$$TP_i = [t_i^m, t_{i+1}^m), \int_{t_i^m}^{t_{i+1}^m} f(x_t)dx_t = \frac{1}{|TP|} \quad (1)$$

$$t_0^m = 00:00\ a.m;\ t_{|TP|}^m = 24:00\ p.m;\ i = 0, ..., |TP| - 1$$

$$t_z^{id} = \varphi\left(t_z^{day}\right) + \varphi\left(t_z^m\right) \quad (2)$$

$$\varphi\left(t_z^{day}\right) = \begin{cases} 0 & t_z^{day} \in weekday \\ 12 & t_z^{day} \in weekend \end{cases} \quad (3)$$

$$\varphi\left(t_z^m\right) = i,\ t_z^m \in \left[t_i^m, t_{i+1}^m\right) \quad (4)$$

where TP_i is the divided time period of a day and $|TP|$ is equal to 12. According to Figure 3, we derive probability distribution of user behaviors, i.e., $f(x_t)$, and probability density $\int_a^b f(x_t)dx_t$. The sum of all probabilities is 1, and the probability within each time period TP_i is $\frac{1}{|TP|}$. Thus, in the peak period of user visits the division could be finer-grained, while it could be relatively coarse-grained in the user's slack visit period.

4.2.2 Deep Encoder for User Preferences in POI Categories. Capturing user preferences in POI categories is the key to reduce search space in POI recommendation. To this end, we design an encoder based on LSTM, illustrated in the top-left part in Figure 5, to model the sequence of POI categories. Given a user u's check-in history $\langle (u, v_1^u, c_1^u, t_1^u), (u, v_2^u, c_2^u, t_2^u), \ldots, (u, v_n^u, c_n^u, t_n^u)\rangle$, the process of the encoder e_1 for category is as follows:

$$I_j^{e,1} = W_u^{e,1} \cdot U_u + W_c^{e,1} \cdot C_j^u \quad (5)$$

$$H_j^{e,1}, S_j^{e,1} = LSTM(I_j^{e,1}, S_{j-1}^{e,1});\ j = 1, 2, ..., n \quad (6)$$

Above, $U_u, C_j^u \in \mathbb{R}^d$ represent latent vectors of user u and POI category c_j^u, respectively; $W_u^{e,1}, W_c^{e,1} \in R^{d \times d}$ are two weight matrices to measure the latent influences of user and POI category, respectively. $I_j^{e,1}$ extracts information from the latent features of POI category and user, i.e., C_j^u and U_u. $H_j^{e,1}$ and $S_j^{e,1}$ are the output (hidden state) and state of the LSTM cell, respectively. They are computed by the current input layer and the previous state of the hidden layer. Since each time step incorporates the current information and all previous information as well, it implies some temporal information. Moreover, the final output of the encoder $H_n^{e,1}$ can be regarded as user u's current preferences in POI categories.

4.2.3 Reduce Search Space to Generate Candidates. Due to the low density of the user-POI matrix, it is difficult to search POIs in a large corpus for a user. Nevertheless, we believe that this sparsity can be overcomed if the search space of POIs for each user can be reduced in an appropriate way. Hence, the challenge is how to reduce the search space for each user to generate a valid candidate set. Figures 1 and 2 show that the user-category matrix is denser than the user-POI matrix, which means that learning the ranking of POI categories

is easier than learning the ranking of POI. Using the results of the high-level concept, namely POI categories, it is feasible to reduce the search space in the next layer. Additionally, user behavior in the next moment is affected by the current geographical location, as shown in Figure 4. Therefore, filtering POIs using geographical coordinates also helps to reduce the search space. All in all, to make recommendations on sparse check-in data accurately, we utilize user preferences in POI categories and geographical influence together to generate candidates.

Next, we elaborate on the process. In the first layer of filtering, we use user preferences in POI categories, i.e., $H_n^{e,1}$ in the previous step, to learn the ranking of all POI categories for every user. The main goal is to exclude the POI categories that a user is not interested in. As shown in the red rectangular box in Fig. 5, the binary classifier, i.e., softmax function σ, is employed to sort all POI categories in the first layer of filtering. Given a set of POI categories $C = \{c_1, c_2, \ldots, c_P\}$, the calculated probabilities are:

$$Input = W_n^{e,1} \cdot H_n^{e,1} + W_c^s \cdot C_i + W_u^s \cdot U_u \tag{7}$$

$$Y_i^s = \sigma(W^s \cdot Input + b^s);\ i = 1, 2, ..., P \tag{8}$$

$$\sigma(X)_j = \frac{exp(x_j)}{exp(x_1) + exp(x_2)}; j = 1, 2 \tag{9}$$

Here, three weight matrices $W_n^{e,1}, W_c^s, W_u^s \in R^{d \times d}$ are used to capture user preferences in that POI category c_i. Since W^s is of 2-by-d and $b^s \in R^2$ is a bias variable, the calculated probability Y_i^s is 2-dimensional, which corresponds to $P(c_i = 0)$ and $P(c_i = 1)$.

After sorting, top-K POI categories are obtained for each user. In the first layer, the model's search space is reduced from all POIs to the POIs that belong to the top-K POI categories, denoted as $\mathcal{V}_u^{l,1}$. Furthermore, when the geographic location of POI v is far from the user's current location, this POI can be excluded. According to Figure 4(a), if the distance between POI v and user's current location is large than 10 kilometers, the POI will be excluded in the second layer. The process is shown as follows:

$$\mathcal{V}_u^{l,1} = layer_1(\mathcal{V}_u) \tag{10}$$

$$\mathcal{V}_u^{l,2} = layer_2(\mathcal{V}_u^{l,1}) \tag{11}$$

Here, the final candidate set is $\mathcal{V}_u^{l,2} = \langle v_1, v_2, \ldots, v_q \rangle$ where q is the number of POIs in the candidate set.

We regard the POI categories with the groundtruth label of user u as the positive samples $\mathcal{P}_u$ and the rest categories as the negative samples $\mathcal{N}_u$. The objective function is as follows:

$$Obj_{cat} = \prod_u \Big(\prod_{c \in \mathcal{P}_u} P(\hat{y}_u(c) = 1|u, c) \prod_{c \in \mathcal{N}_u} P(\hat{y}_u(c) = 0|u, c) \Big) \tag{12}$$

where $\hat{y}_u(c)$ represents the predicted label of the category c. Given the user u and the category c, $P(\hat{y}_u(c)|u, c)$ represents the output of the softmax function.

4.2.4 Deep Encoder for User Preferences in POIs. The observation in Section 4.1 suggests that the clock influence is crucial to capture temporal patterns of user preference. User interests, especially for long-term interests, may be made up of several more granular interests. To dig out user preferences more finely, we designed multiple windows to distinguish behaviors and extract clock influence for different time periods of the day, illustrated in the bottom-left in Figure 5. As a side effect of this design, the weakness caused by data sparsity can be greatly mitigated. If the user's check-in sequences are directly embedded into one vector, the sparseness of the data will make it difficult for the model to capture global user preferences. Nevertheless, when we capture user preferences in different time windows separately, even if the user behavior information is missing in a few time windows, it will not affect other windows. Given a user u's check-in history $\langle (u, v_1^u, c_1^u, t_1^u), (u, v_2^u, c_2^u, t_2^u), \ldots, (u, v_n^u, c_n^u, t_n^u) \rangle$, the process of encoder e_2 for POI is as follows:

$$I_j^{e,2} = W_u^{e,2} \cdot U_u + W_v^{e,2} \cdot V_j^u \tag{13}$$

$$H_j^{e,2}, S_j^{e,2} = LSTM(I_j^{e,2}, S_{j-1}^{e,2}) \tag{14}$$

$$H^{e,2} = \langle H_1^{e,2}, H_2^{e,2}, \ldots, H_n^{e,2} \rangle \tag{15}$$

Above, $U_u, V_j^u \in \mathbb{R}^d$ represent latent vectors of user u and POI category v_j^u, respectively; $W_u^{e,2}, W_c^{e,2} \in R^{d \times d}$ are two weight matrices to measure the latent influences of user and POI, respectively. Since $H_n^{e,2}$ incorporates the current input and the previous sequential state, it can be regarded as one of the user u's current preferences in POIs. Meanwhile, $H_j^{e,2} \in R^d$ represents hidden states, and $H^{e,2}$ represents all the n hidden states which incorporate temporal information and some correlations from a check-in sequence. To mine the user preferences more finely, we partition $H^{e,2}$ into M separate windows, each corresponding to a specific time period of a day.

However, according to Section 4.2.1, dividing hours into different windows evenly is not a good choice. Therefore, we partition all the $H^{e,2}$ into M separate windows in the light of the division method in Section 4.2.1. Each window $\mathcal{W}_i^{e,2}$ is defined as follows:

$$\mathcal{W}_i^{e,2} = \left\{ H_j^{e,2} \mid t_j^u \in [t_i^{e,2}, t_{i+1}^{e,2}) \right\} \tag{16}$$

$$\int_{t_i^{e,2}}^{t_{i+1}^{e,2}} f(x_t)dx_t = \frac{1}{M},\ \ for\ \ [t_i^{e,2}, t_{i+1}^{e,2}); \tag{17}$$

$$j = 1, 2, ..., n;\ i = 0, 1, ..., M-1;$$

$$t_0^{e,2} = 00:00\ a.m;\ t_M^{e,2} = 24:00\ p.m$$

Here, similar to Section 4.2.1, we derive probability distribution of user behaviors, i.e., $f(x_t)$, and probability density $\int_a^b f(x_t)dx$. The sum of all probabilities is 1, and the probability within each window is $\frac{1}{M}$. M is equal to 12 here.

In fact, each window can be seen as a subsequence of user behaviors, which implies the user's long-short-term interests in that time period. To capture this, we use window state to combine all the user's check-in behaviors within each subsequence. Let $|\mathcal{W}_i^{e,2}|$ denote the number of check-ins within time window $\mathcal{W}_i^{e,2}$. Each window state W_i with size d is computed as follows:

$$W_i = \frac{\sum_{k=1}^{|\mathcal{W}_i^{e,2}|} H_k^{e,2}}{|\mathcal{W}_i^{e,2}|};\ \ H_1^{e,2}, ..., H_{|\mathcal{W}_i^{e,2}|}^{e,2} \in \mathcal{W}_i^{e,2} \tag{18}$$

The goal of this encoder is to learn a representation for user preferences. We achieve this by designing an effective and reasonable method to linearly combine M window states. However, there are two important issues worth particular treatment. On the one

hand, the preferences vary from person to person. Therefore, even for the same time period, i.e., window, if the users are different, the assigned weights should be different. On the other hand, the distribution of interests of a user during the day may be uneven. Thus, for the same user, the weight assigned to one time window should also differ from another.

Hence, the assigned weight of each window should be determined by the current user and the state of this window. Inspired by [33], we devise a weight calculation method for each window to achieve personalized window weighting for users. We calculate the weights $\alpha^u_{W_i}$ of each time window $\mathcal{W}^{e,2}_i$ using user embedding U_u and window state W_i, shown as follows:

$$A^u_{W_i} = PRelu(concat(U_u, \omega(U_u, W_i), W_i)) \tag{19}$$

$$\omega(x, y) = x \odot y = [x_1 y_1, x_2 y_2, ..., x_d y_d] \tag{20}$$

$$\alpha^u_{W_i} = \sum_{o_j \in A^u_{W_i}} o_j \tag{21}$$

Here, user embedding U_u and W_i are vectors form R^d. Operator $\odot$ represents the element-wise product. The i-th element in $\omega(U_u, W_i)$ is $x_i y_i$, where x_i and y_i are the i-th element in U_u and W_i, respectively. Then U_u, $\omega(U_u, W_i)$ and W_i are concatenated as vectors from R^{3d} using function $concat$. Note that U_u ensures a window's weight relates to user preference, W_i ensures the weight relates to window state, and $\omega(x, y)$ calculates the relationship between user preference and window state. Subsequently, function $PRelu$ is used to guarantee that all elements are non-negative. Since o_j is the element of $A^u_{W_i}$, the resulting weight $\alpha^u_{W_i}$ should have size 1. Finally, in addition to the $H^{e,2}_n$, another user u's current preferences in POIs $P^{poi}_u \in R^d$ is obtained by the following equation.

$$P^{poi}_u = \frac{\sum_{i=1}^{M} \alpha^u_{W_i} W_i}{\sum_{j=1}^{M} \alpha^u_{W_j}} \tag{22}$$

4.2.5 Ranking Candidate Set. To accurately sort the POIs in the candidate set, we take into account four specific correlations simultaneously: (1) correlation between user and POI; (2) correlation between user and POI category; (3) correlation between POI and temporal influence; (4) correlation between POI and user's current preferences in POIs. Since each user, POI, POI category and time is embedded in a latent space, Euclidean distance [25] is a reasonable and efficient indicator to measure the correlations between them. For example, given a user u and POI v, the lower the Euclidean distance $||U_u - V_v||^2_2$ between them, the greater the correlation between them. Likewise, $||T_{t^{id}_z} - V_v||^2_2$ is related to the probability that POI v will be considered under the current time t_z. According to the analysis in Section 4.2.2, user preferences is affected by category as well, i.e., $||U_u - C_c||^2_2$. Moreover, both $H^{e,2}_n$ and P^{poi}_u represent a user's current preferences in POIs. Hence, the correlation between POI and the user's current preferences is defined as $||H^{e,2}_n - V_v||^2_2 + ||P^{poi}_u - V_v||^2_2$. Given a user u, his/her current location v_n and current time t_n, we rank the candidate set $\mathcal{V}^{l,2}_u$ by calculating the Euclidean distances for each $v \in \mathcal{V}^{l,2}_u$:

$$\begin{aligned} E^n_{u,v} =& ||U_u - V_v||^2_2 + ||T_{t^{id}_n} - V_v||^2_2 + ||U_u - C_c||^2_2 \\ &+ ||H^{e,2}_n - V_v||^2_2 + ||P^{poi}_u - V_v||^2_2 \end{aligned} \tag{23}$$

4.3 Parameter Learning

We use $\mathcal{P}^{poi}_u$ to denote the set of positive sample POIs in the groundtruth label of user u, and $\mathcal{N}^{poi}_u$ the set of negative sample POIs. The goal of training is to minimize E^n_{u,v_i} for $v_i \in \mathcal{P}^{poi}_u$, and maximize E^n_{u,v_i} for $v_i \in \mathcal{N}^{poi}_u$. Due to the lack of training data, user groundtruth labels and negative samples are insufficient.

To this end, we design a sequential loss function for the second LSTM encoder. At each step i of the second encoder, we take the POI of next input v_{i+1} as a positive sample, i.e., the next location we need to predict, and randomly select a negative sample for it. This way, we can calculate the Euclidean distances and the corresponding loss for each time step. However, calculating window state W in each time step will increase the computational complexity. Hence, we choose to calculate the window state W only when sorting the candidate set, i.e, the final time step n. While for each time step i, we only take $H^{e,2}_i$ as the user's current preference in POIs. The loss function is defined as follows:

$$\begin{aligned} \hat{E}^i_{u,v_{i+1}} =& ||U_u - V_{v_{i+1}}||^2_2 + ||T_{t^{id}_i} - V_{v_{i+1}}||^2_2 \\ &+ ||U_u - C_{c_{i+1}}||^2_2 + ||H^{e,2}_i - V_{v_{i+1}}||^2_2 \end{aligned} \tag{24}$$

$$\begin{aligned} \mathcal{L}_{poi} &= \underset{\Theta}{argmin} - \sum_{u \in \mathcal{U}} \prod_{v_{i+1} \in C\mathcal{H}_u}^{1 \le i \le n-1} (\hat{E}^i_{u,\hat{v}_{i+1}} - \hat{E}^i_{u,v_{i+1}}) \\ &= \underset{\Theta}{argmin} - \sum_{u \in \mathcal{U}} \sum_{v_{i+1} \in C\mathcal{H}_u}^{1 \le i \le n-1} ln(\theta(\hat{E}^i_{u,\hat{v}_{i+1}} - \hat{E}^i_{u,v_{i+1}})) \end{aligned} \tag{25}$$

where $\hat{v}_{i+1}$ a randomly selected negative sample for v_{i+1}, and θ is the logistic function $\theta(x) = \frac{1}{1+e^{-x}}$. To alleviate overfitting, we add parameter regularization and thus $\mathcal{L}_{poi}$ is rewritten as follows:

$$\mathcal{L}_{poi} = \underset{\Theta}{argmin} - \sum_{u \in \mathcal{U}} \sum_{v_{i+1} \in C\mathcal{H}_u}^{1 \le i \le n-1} ln(\theta(\hat{E}^i_{u,\hat{v}_{i+1}} - \hat{E}^i_{u,v_{i+1}})) - \lambda||\Theta|| \tag{26}$$

where the parameter set Θ include latent matrices U, V, C and T of user, POI, POI category and time, respectively.

5 EXPERIMENTS

5.1 Experimental Setup

5.1.1 Datasets. We evaluate the models on two real datasets: the Foursquare check-ins in New York and Tokyo [22] from 12 April 2012 to 16 February 2013. They are denoted as NYC and TKY, respectively. We remove the POIs visited by fewer than 5 users, as they are outliers in the data.

After pre-processing, the NYC dataset covers 1,083 users, 9,989 POIs and 233 venue categories in 179,468 check-ins, and the TKY dataset covers 2,293 users, 15,177 POIs and 216 venue categories in 494,807 check-ins. Table 1 gives the statistics of the two datasets. We use $|\mathcal{U}|$ to denote the number of users, $|\mathcal{V}|$ the number of POIs, $|C|$ the number of POI categories, and $|C\mathcal{H}|$ the total number of

check-in records. The data *sparsity* is computed as $\frac{|ZC|}{|\mathcal{U}|\times|\mathcal{V}|}$, where $|ZC|$ is the number of zeros in the user-POI check-in matrix. As shown in Table 1, both datasets are highly sparse even after preprocessing. Therefore, it is hard to infer where a user will go in the next few hours using such sparse datasets.

We split each dataset into a training set, a validation set and a test set. We take the check-ins in the first 8 months as training set, the ninth month as valid set, and the last month as test set. In the test data, we pick each user's last 24-hour check-ins, use the first POI as current location, and the rest as groundtruth. The number of divided windows, i.e., M, is 12. Given a user u and u's current location, our model recommends top-N successive POIs for the next 24 hours.

Table 1: Basic statistics of Foursquare datasets.

	$\lvert\mathcal{U}\rvert$	$\lvert\mathcal{V}\rvert$	$\lvert C\rvert$	$\lvert\mathcal{CH}\rvert$	$\frac{\lvert ZC\rvert}{\lvert\mathcal{U}\rvert\times\lvert\mathcal{V}\rvert}$
NYC	1,083	9,989	233	179,468	99.51%
TKY	2,293	15,177	216	494,807	99.58%

5.1.2 Performance Metrics. To evaluate top-N successive POI recommendation, we adopt performance metrics *Pre@N* and *Rec@N*, where N is the number of recommended POIs in a result. These are general metrics for POI recommendation used in previous works [24, 26]. The *Pre@N* is the ratio of recovered POIs to the N recommended POIs in a day, and *Rec@N* is the ratio of recovered POIs to the groundtruth. Given a user u's history $\mathcal{CH}_u$, her/his current location v_u and the current time t_u, we use the next locations of check-in in 24 successive hours as the groundtruth V_u^T. Let V_u^R be the set of recommendation result. The definitions of *Pre@N* and *Rec@N* are shown as follows:

$$Pre@N = \frac{1}{|\mathcal{U}|}\sum_{u\in\mathcal{U}}\frac{|V_u^T \cap V_u^R|}{N} \tag{27}$$

$$Rec@N = \frac{1}{|\mathcal{U}|}\sum_{u\in\mathcal{U}}\frac{|V_u^T \cap V_u^R|}{|V_u^T|} \tag{28}$$

5.1.3 Baseline Methods. We choose representative baselines from typical research works. The baselines are works for conventional POI recommendation and successive POI recommendation, which consider POI category, temporal influence, geographical influence or the combination of them. The characteristics of all baselines and our CatDM are listed in Table 2.

5.2 Effect of Parameters

5.2.1 Effect of Hidden Size d. The parameter d is responsible for determining the dimensionality of embedded vectors of POI, POI category and user, so it impacts the model performance. In order to find out how the hidden size d affects model performance and to select the best setting, we conduct several experiments. We use precision *Pre@5* to evaluate the effect, since the performance with other indicators are similar. According to results illustrated in Figure 6(a), we set $d = 60$ for NYC and $d = 70$ for TKY.

5.2.2 Number of Unfiltered POI Categories K. It determines the number of POI categories after the first layer of filter works. Figure 6(b) shows *Rec@K* of predicted top-K POI categories. The recalls in NYC and TKY are close to 1 when K=130 and almost stop growing when K>130. Therefore, to reduce search space and to ensure high recall simultaneously, we set K=130.

5.3 Results on Successive POI Recommendation

For successive POI recommendation, the performance results of all implemented methods are presented in Figure 7. We observe that GeoSoCa and Rank-GeoFM perform worst. As a calculating-based method, GeoSoCa cannot improve its ability through training process. Also, to adapt to the data, GeoSoCa cannot consider social correlations. For Rank-GeoFM, it only considers geographical factorization to predicts POIs. Although LSTM-based model can capture the sequential information, neither HST-LSTM nor ST-CLSTM performs well. This can be due to the following reason. LSTM-based model requires a lot of data to support training, while both NYC and TKY datasets are fairly sparse. Compared with HST-LSTM and ST-CLSTM, ST-RNN shows a higher performance. The reason might be that ST-RNN learns parameters based on Bayesian Personalized Ranking [19], and the pairwise learning method alleviating the issue of lack of data. The good performance of LBPR shows that POI category plays an effective role in capturing user preferences. Furthermore, CatDM performs better than STELLAR on both datasets. For example, in terms of *Pre@5*, CatDM outperforms STELLAR by 83% on NYC, and 42% on TKY. MEAP-T achieves considerable performance because it captures three correlations, user-POI, POI-POI and POI-time, and focuses on asymmetric property of successive check-ins, which is ignored by others. CatDM outperforms it because CatDM captures temporal patterns by using clock influence to divide time windows more finely. Moreover, CatDM takes into account user preferences in POI categories to reduce search space.

Moreover, it is noteworthy that the performance advantage of our model on NYC is much greater than that on TKY. Take *Pre@5* as an example. CatDM outperforms MEAP-T by about 23% on NYC, while only by about 19% on TKY. Although the two datasets feature similar sparsity, the number of users and the average number of check-ins per user in NYC are much lower than the counterparts in TKY. As a result, it is harder to model user behaviors and exploit user preferences accurately on NYC that contains less data. Nevertheless, the results verify that our model is more suitable for sparse data.

5.4 Results on Successive POI Category Recommendation

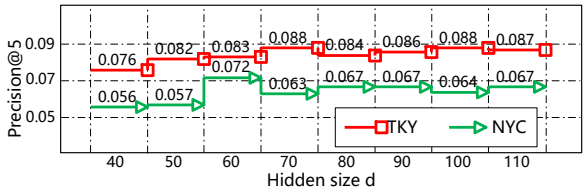
As each POI belongs to a category, the same way can be used to calculate the *Pre@N* and *Rec@N* of the recommended POI categories. Therefore, to show that our CatDM can accurately capture user preferences in POI categories, we conduct several experiments on POI category recommendation. The performance results are shown in Figure 8. We can observe that CatDM always performs best. Compared with MEAP-T, for *Pre@5*, the improvements of CatDM are about 16% on NYC and about 12% on TKY. We attribute the advantage to a particular component in CatDM, namely the deep encoder that captures user preferences in POI categories.

Table 2: Baselines and characteristics of them.

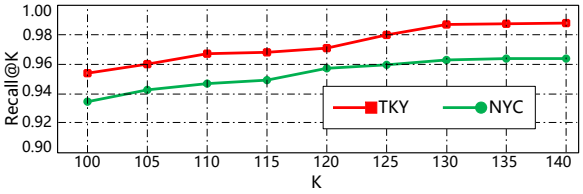
Approach	Category	Temporal influence	Geographical influence	POI category	Social influence	RNN-based
GeoSoCa [27]	Traditional	×	✓	✓	✓	×
LBPR [7]	Successive	✓	✓	✓	×	×
Rank-GeoFM [11]	Traditional	✓	✓	×	×	×
STELLAR [32]	Successive	✓	✓	×	×	×
MEAP-T [25]	Successive	✓	✓	×	×	×
ST-RNN [13]	Successive	✓	✓	×	×	✓
HST-LSTM [10]	Successive	✓	✓	×	×	✓
ST-CLSTM [31]	Successive	✓	✓	×	×	✓
CatDM	Successive	✓	✓	✓	×	✓

Table 3: Performance of different versions of CatDM.

	NYC						TKY					
	Pre@5	*Rec*@5	*Pre*@10	*Rec*@10	*Pre*@15	*Rec*@15	*Pre*@5	*Rec*@5	*Pre*@10	*Rec*@10	*Pre*@15	*Rec*@15
CatDM	**0.0720**	**0.2407**	**0.0458**	**0.3113**	**0.0347**	**0.3468**	**0.0882**	**0.2148**	**0.0590**	**0.2739**	**0.0456**	**0.3384**
CatDM$_{NW}$	0.0607	0.1876	0.0378	0.2305	0.0285	0.2785	0.0688	0.1460	0.0476	0.1993	0.0374	0.2311
CatDM$_{EW}$	0.0670	0.2372	0.0421	0.2808	0.0305	0.3022	0.0749	0.1798	0.0498	0.2333	0.0390	0.2793
CatDM$_{NF}$	0.0683	0.2404	0.0433	0.2847	0.0310	0.3339	0.0803	0.1968	0.0571	0.2684	0.0456	0.3094



(a) Hidden size



(b) Top-K POI categories

Figure 6: Impact of parameters.

5.5 Results on Different Versions of CatDM

To verify the effectiveness of several key parts designed in our model, we conduct more experiments to evaluate CatDM variants with and without such designs.

CatDM$_{NW}$. Different time windows are developed to exploit user preferences in different time periods. Also, it is a crucial part to overcome data sparsity. Hence, it is necessary to design experiments to demonstrate its effectiveness. We exclude the correlation between P_u^{poi} and V_v, when sorting the candidate set. Then we obtain the variant CatDM$_{NW}$ without dividing windows.

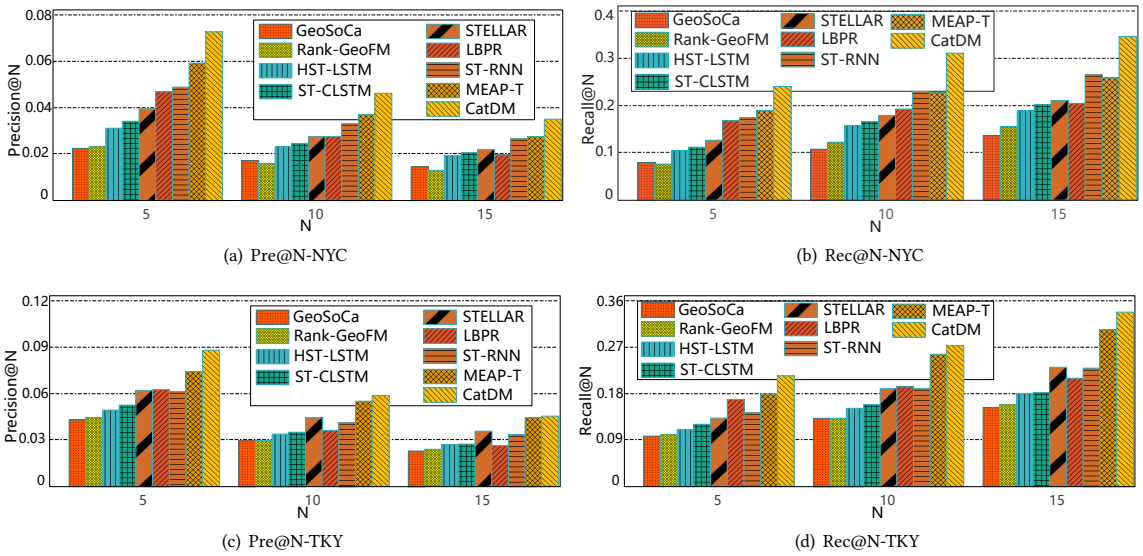
CatDM$_{EW}$. CatDM utilizes the probability density of user check-in behaviors during a day to allocate the time windows, which can adapt to all datasets in general (detailed in Section 4.2.4). To verify this design, we implement the variant CatDM$_{EW}$, which divides 24 hours a day into 12 windows evenly.

CatDM$_{NF}$. We briefly discuss the reasons of using a filter in Section 4.2.3. To further explore the specific functions of the filter in our model, we also implement a method CatDM$_{NF}$ without filter.

Table 3 presents the experimental results of these CatDM variants. Overall, CatDM outperforms its variants on both datasets. This indicates that each designed part takes important effect. The big difference between the results of CatDM and CatDM$_{NW}$ on NYC clearly demonstrates that having different windows can capture user preferences more accurately. It shows that uneven time windows can also appropriately strengthen the model's capabilities. However, we can observe that the difference between the performance results of CatDM$_{EW}$ and CatDM on TKY is higher than that on NYC. This is due to the non-uniformity of distribution of the check-in frequency in TKY. In such a case, evenly dividing the 24 hours to fixed-length intervals cannot distinguish peak periods from slack periods of user visits, and consequently cannot effectively capture user preferences for different time periods of the day. Moreover, the performance results of CatDM and CatDM$_{NF}$ show the importance of the filter. In addition to reducing search space, it helps CatDM to exploit user preferences accurately. This also proves that generating candidate set reasonably does not impair the model performance.

5.6 Effect of the Filter

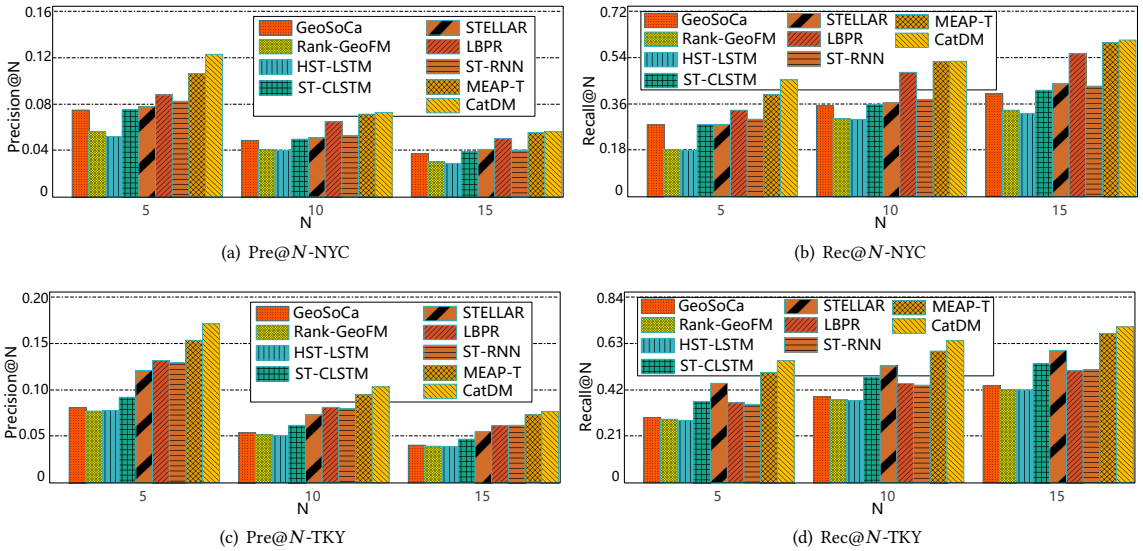
To further show the importance and rationality of the filter, we add the filter to other approaches. Since HST-LSTM, ST-CLSTM and ST-RNN are implemented based on RNN, we only evaluate ST-RNN as it achieves good performance on successive POI recommendation. Figure 9 depicts the performance results with and without filtering POIs on NYC. We can observe that after filtering POIs, ST-RNN and MEAP-T improve by about 21% and 14% on *Pre*@5, respectively. Moreover, although our proposed CatDM still outperforms MEAP-T, the gap is significantly reduced.



(a) Pre@N-NYC (b) Rec@N-NYC

(c) Pre@N-TKY (d) Rec@N-TKY

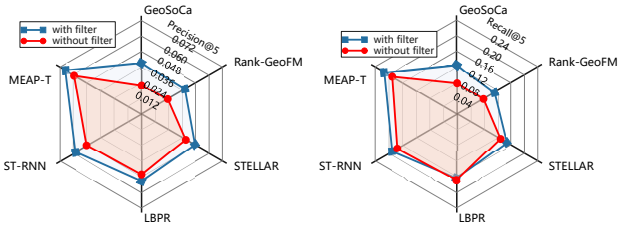
Figure 7: Comparison of successive POI recommendation performance on sparse dataset.



(a) Pre@N-NYC (b) Rec@N-NYC

(c) Pre@N-TKY (d) Rec@N-TKY

Figure 8: Successive POI category recommendation.



(a) Pre@5-NYC (b) Rec@5-NYC

Figure 9: Performance of approaches with Filter.

6 CONCLUSIONS AND FUTURE WORK

We formulate and study top-N successive POI recommendation on sparse check-in data. To overcome data sparsity and recommend a sequence of POIs for a user to visit in a future period, we propose a POI category-aware deep model (CatDM). It constructs binary classifiers in a two-layer filter architecture. The two layers filter POI categories and POIs, respectively, when searching for POIs to recommend. Moreover, we design two deep encoders based on LSTM to capture user preferences and temporal patterns. We design a sequential loss function for the second LSTM encoder to address the issue of data sparsity. We conduct extensive experiments on real datasets. The results demonstrate that our CatDM is effective on sparse data, and it outperforms the state-of-the-art methods in terms of multiple performance metrics.

For future work, it is interesting to go beyond time windows and capture time patterns in a hierarchical structure, e.g, a tree. More complex time patterns can even better model user behaviors at

different times. It is also relevant to locate a user's activity regions at different times to filter out POIs more accurately .

ACKNOWLEDGMENTS

This work is partially supported by the National Key R&D Program No.2017YFB1400100, the NSFC No.91846205, the Innovation Method Fund of China No.2018IM020200, the Shandong Key R&D Program No.2018YFJH0506 and 2019JZZY011007. We also thank Kangning Huang for his efforts in a preliminary version of this work.

REFERENCES

[1] Jie Bao, Yu Zheng, and Mohamed F. Mokbel. 2012. Location-based and preference-aware recommendation using sparse geo-social networking data. In *SIGSPATIAL 2012 International Conference on Advances in Geographic Information Systems (formerly known as GIS), SIGSPATIAL'12, Redondo Beach, CA, USA, November 7-9, 2012*. 199–208.

[2] 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 Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada.*

[3] Chen Cheng, Haiqin Yang, Michael R. Lyu, and Irwin King. 2013. Where You Like to Go Next: Successive Point-of-Interest Recommendation. In *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*. 2605–2611.

[4] Shanshan Feng, Gao Cong, Bo An, and Yeow Meng Chee. 2017. POI2Vec: Geographical Latent Representation for Predicting Future Visitors. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*. 102–108.

[5] Huiji Gao, Jiliang Tang, Xia Hu, and Huan Liu. 2013. Exploring temporal effects for location recommendation on location-based social networks. In *Seventh ACM Conference on Recommender Systems, RecSys '13, Hong Kong, China, October 12-16, 2013*. 93–100.

[6] Huiji Gao, Jiliang Tang, Xia Hu, and Huan Liu. 2015. Content-Aware Point of Interest Recommendation on Location-Based Social Networks. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*. 1721–1727.

[7] Jing He, Xin Li, and Lejian Liao. 2017. Category-aware Next Point-of-Interest Recommendation via Listwise Bayesian Personalized Ranking. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*. 1837–1843.

[8] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9, 8 (1997), 1735–1780.

[9] Longke Hu, Aixin Sun, and Yong Liu. 2014. Your neighbors affect your ratings: on geographical neighborhood influence to rating prediction. In *The 37th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '14, Gold Coast , QLD, Australia - July 06 - 11, 2014*. 345–354.

[10] Dejiang Kong and Fei Wu. 2018. HST-LSTM: A Hierarchical Spatial-Temporal Long-Short Term Memory Network for Location Prediction. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*. 2341–2347.

[11] Xutao Li, Gao Cong, Xiaoli Li, Tuan-Anh Nguyen Pham, and Shonali Krishnaswamy. 2015. Rank-GeoFM: A Ranking based Geographical Factorization Method for Point of Interest Recommendation. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, Santiago, Chile, August 9-13, 2015*. 433–442.

[12] Bin Liu and Hui Xiong. 2013. Point-of-Interest Recommendation in Location Based Social Networks with Topic and Location Awareness. In *Proceedings of the 13th SIAM International Conference on Data Mining, May 2-4, 2013. Austin, Texas, USA*. 396–404.

[13] 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 Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*. 194–200.

[14] Xin Liu and Karl Aberer. 2013. SoCo: a social network aided context-aware recommender system. In *22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, May 13-17, 2013*. 781–802.

[15] Xin Liu, Yong Liu, Karl Aberer, and Chunyan Miao. 2013. Personalized point-of-interest recommendation by mining users' preference transition. In *22nd ACM International Conference on Information and Knowledge Management, CIKM'13, San Francisco, CA, USA, October 27 - November 1, 2013*. 733–738.

[16] Xin Liu, Yong Liu, and Xiaoli Li. 2016. Exploring the Context of Locations for Personalized Location Recommendations. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*. 1188–1194.

[17] Yong Liu, Wei Wei, Aixin Sun, and Chunyan Miao. 2014. Exploiting Geographical Neighborhood Characteristics for Location Recommendation. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM 2014, Shanghai, China, November 3-7, 2014*. 739–748.

[18] Yong Liu, Lifan Zhao, Guimei Liu, Xinyan Lu, Peng Gao, Xiao-Li Li, and Zhihui Jin. 2018. Dynamic Bayesian Logistic Matrix Factorization for Recommendation with Implicit Feedback. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*. 3463–3469.

[19] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *UAI 2009, Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, Montreal, QC, Canada, June 18-21, 2009*. 452–461.

[20] 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, WWW 2010, Raleigh, North Carolina, USA, April 26-30, 2010*. 811–820.

[21] Dingqi Yang, Daqing Zhang, Zhiyong Yu, and Zhu Wang. 2013. A sentiment-enhanced personalized location recommendation system. In *24th ACM Conference on Hypertext and Social Media (part of ECRC), HT '13, Paris, France - May 02 - 04, 2013*. 119–128.

[22] Dingqi Yang, Daqing Zhang, Vincent W. Zheng, and Zhiyong Yu. 2015. Modeling User Activity Preference by Leveraging User Spatial Temporal Characteristics in LBSNs. *IEEE Trans. Systems, Man, and Cybernetics: Systems* 45, 1 (2015), 129–142.

[23] Mao Ye, Peifeng Yin, and Wang-Chien Lee. 2010. Location recommendation for location-based social networks. In *18th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems, ACM-GIS 2010, November 3-5, 2010, San Jose, CA, USA, Proceedings*. 458–461.

[24] Mao Ye, Peifeng Yin, Wang-Chien Lee, and Dik Lun Lee. 2011. Exploiting geographical influence for collaborative point-of-interest recommendation. In *Proceeding of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2011, Beijing, China, July 25-29, 2011*. 325–334.

[25] 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.

[26] Quan Yuan, Gao Cong, Zongyang Ma, Aixin Sun, and Nadia Magnenat-Thalmann. 2013. Time-aware point-of-interest recommendation. In *The 36th International ACM SIGIR conference on research and development in Information Retrieval, SIGIR '13, Dublin, Ireland - July 28 - August 01, 2013*. 363–372.

[27] Jia-Dong Zhang and Chi-Yin Chow. 2015. GeoSoCa: Exploiting Geographical, Social and Categorical Correlations for Point-of-Interest Recommendations. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, Santiago, Chile, August 9-13, 2015*. 443–452.

[28] 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, Dallas/Fort Worth, TX, USA, November 4-7, 2014*. 103–112.

[29] Wei Zhang and Jianyong Wang. 2015. Location and Time Aware Social Collaborative Retrieval for New Successive Point-of-Interest Recommendation. In *Proceedings of the 24th ACM International Conference on Information and Knowledge Management, CIKM 2015, Melbourne, VIC, Australia, October 19 - 23, 2015*. 1221–1230.

[30] Yuyu Zhang, Hanjun Dai, Chang Xu, Jun Feng, Taifeng Wang, Jiang Bian, Bin Wang, and Tie-Yan Liu. 2014. Sequential Click Prediction for Sponsored Search with Recurrent Neural Networks. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada*. 1369–1375.

[31] Pengpeng Zhao, Haifeng Zhu, Yanchi Liu, Zhixu Li, Jiajie Xu, and Victor S. Sheng. 2018. Where to Go Next: A Spatio-temporal LSTM model for Next POI Recommendation. *CoRR* abs/1806.06671 (2018). arXiv:1806.06671

[32] Shenglin Zhao, Tong Zhao, Haiqin Yang, Michael R. Lyu, and Irwin King. 2016. STELLAR: Spatial-Temporal Latent Ranking for Successive Point-of-Interest Recommendation. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*. 315–322.

[33] Han Zhu, Xiang Li, Pengye Zhang, Guozheng Li, Jie He, Han Li, and Kun Gai. 2018. Learning Tree-based Deep Model for Recommender Systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*. 1079–1088.