Joint Attention Networks with Inherent and Contextual Preference-Awareness for Successive POI Recommendation

Haiting Zhong · Wei He · Lizhen Cui · Lei Liu · Zhongmin Yan · Kun Zhao

Received: 28 July 2022 / Revised: 22 September 2022 / Accepted: 16 October 2022 / Published online: 28 October 2022
© The Author(s) 2022

Abstract
Nowadays recording and sharing personal lives using mobile devices on the Internet is becoming increasingly popular, and successive POI recommendation is gaining growing attention from academia and industry. In mobile scenarios, multiple influencing factors including the diversity of user preferences, the changeability of user behavior and the dynamic of spatiotemporal context bring great challenges to the POI recommender system. In order to accurately capture both the stable and the contextual preferences of mobile users in dynamic contexts, we propose a fusion framework JANICP (Joint Attention Networks with Inherent and Contextual Preferences) for successive POI recommendation by jointly training an offline/nearline user inherent interest perception model and an online user contextual interest prediction model. The offline model is trained based on the global historical behavior data to achieve stable interest representation, while the online model is trained based on the instantly selected context-sensitive data to achieve dynamic interest perception. An attention aggregation and matching module is used to fully connect the two kinds of preference representations and generate the final POI recommendation. Extensive experiments were conducted on three real datasets and experimental results show that the proposed JANICP outperforms existing state-of-the-art methods.

Keywords Successive POI · Recommender system · Stable preferences · Contextual preferences · Self-attention · R-tree

1 Introduction

With the rapid development of location-based social networks and mobile devices, more and more people record and share their lives on different kinds of platforms, such as Yelp, Places, Foursquare etc. POI recommendation aims at recommending new POIs (Points-of-Interest) to users according to their personalized preferences, which is convenient for users to find new related places and explore the cities, while for advertisers to push advertisements to targeted users [1, 2]. In recent years, successive POI recommendation [2–4] is drawing attention from academia and industry. Compared with conventional POI recommendation [5] which only focuses on the user’s next visit, successive POI recommendation is also concerned about the expected time of the following visit (i.e. aiming to recommend potential POIs of a future visit to users). As successive POI recommendation can proactively help people plan their lives in a short period of future time, it can bring better experiences
for mobile users by enabling them to explore new interesting places, record and share their lives anywhere and anytime.

In mobile scenarios, each user’s behaviors such as visiting a particular POI are diversified due to many factors, including spatiotemporal context, geographical distance and user preferences etc. Also, even for the same user, they have different interests in different contexts. These influencing factors with their changeability and interaction bring considerable challenges to the POI recommender system. Early Markov chain-based models [6] focused on sequential transitions. After that many methods [7–9] propose various variants of recurrent neural networks to better extract long-term and short-term features of user check-ins. The recent state-of-the-art models on POI recommendation mainly focus on capturing the users’ sequential patterns from their historical check-in or trajectory data by training different types of models, with consideration of spatial-temporal features or time influences [10–12]. Essentially, there are still two vital issues that have not been solved particularly well, and consequently the performance of recommender systems in mobile scenarios cannot always be satisfactory.

1. How to accurately capture user preferences with diversity and variability in mobile environments? The sequence patterns of user-POI interaction captured from historical data are not always reliable due to the following two aspects. Firstly, the user-expected behaviors vary in different contexts, such as going out for a trip on weekends but staying at home on weekdays. Secondly, the users’ preferences may not be continuously stable. Some recent research used self-attention networks to capture short-term interests by embedding the latest interacted POIs and combining users regular representations of long-term preferences in connect layer [13, 14]. However, simply applying attention/self-attention mechanism on contextual features such as time or location will lead to overfitting or sub-optimal solutions, as well as being difficult to perceive users’ sustained interests.

2. How does a model-based approach meet the demand for real-time response of the POI recommender system? The valuable results of a successive POI recommendation should be generated from an effective model trained from both user historical data and latest interacted POIs. On the other hand, user preferences prediction models require offline training and periodical updates because the behavioral data of mobile users emerge quickly anytime and anywhere. Neither offline nor nearline model training-based approach can adapt to the demand for real-time response with models trained using the latest user interacted data in mobile scenarios where users’ contexts are changing at any time.

In this paper we investigate the above issues in successive POI recommender systems, that is, accurate preferences perception for mobile users and real-time response of the POI recommender system with models trained using the latest user interacted data. The basic assumption here is that user preferences remain basically stable over time while showing dynamicity and diversity affected by different contexts. In brief, the choices of mobile users are not only affected by their inherent factors such as personalities or likes, but also by variable external factors. Different from recent approaches, we model user behavioral preferences as a combination of constant interests and dynamic interests, which are referred to as User Inherent preferences and User Contextual Preferences, respectively, in this paper. According to the conflict between real-time response demand and offline/nearline model training, we combine user constant interest mining (using an offline model) and dynamic interest perception (directly using a memory-based model) together. To capture user’s real-time contextual behavioral intentions, we emphatically investigate the strategy of selection and quick retrieval for a small amount of appropriate user interactive POI data from the extensive historical data.

To this end, we propose a fusion framework JANICP (Joint Attention Networks with Inherent and Contextual Preferences) by integrating a user inherent interest mining model which is periodically trained offline/nearline and a user contextual interest perception model which is applied online. The offline model is trained and updated periodically based on historical data to achieve user stable interest perception, and the online model is trained based on the latest context-related data to achieve dynamic interest perception. Specifically, the self-attention layer of the offline model is used to learn the inherent preferences of users by assigning different weights to each visit, and solve the long-term dependency problem of recurrent neural networks. For real-time contextual intent capture of users, a self-attention layer is required to model contexts such as current user location and trajectory. Different from current works which focus on the importance of user interaction time in training data, our proposed approach believes that contexts (such as location) have a greater impact on users’ intention. Thus, an R-tree-based POI index structure is designed to generate a candidate set of POIs according to users’ current location, which can quickly retrieve a small amount of valid user POI interaction data from extensive historical data and finally achieve online recommendation efficiency.

In short, our contributions are summarized as follows:

- We propose JANICP, an attention model based on users’ inherent and contextual preferences, which fully considers the stability of users’ inherent preferences and the dynamics of contextual preferences.
We design an R-tree-based index structure for global POIs to reduce the computational space and improve the fast response capability of the model.

Extensive experiments were conducted on three real datasets. Experimental results show that our JANICP performs better compared to state-of-the-art models.

The rest of this article is organized as follows. Section 2 introduces relevant work. Section 3 defines the problem and related terms. Section 4 introduces our successive POI recommendation model based on self-attention mechanism in detail. Section 5 reports the experiment. Section 6 summarizes the paper.

2 Related Work

In this section, we briefly review some works on successive POI recommendations. Successive POI recommendation pays more attention to the most recent check-in(s) than conventional POI recommendation.

2.1 Conventional POI Recommendation

POI recommendation has been an important and popular service in location-based social networks. Conventional POI recommendation is mainly developed based on temporal influence [15–17] and geographic information [18–21]. Modeling such user-specific spatial temporal activity preferences (STAP) needs to tackle high-dimensional data, i.e., user-location-time-activity quadruples, which is complicated and usually suffers from a data sparsity problem. In order to address this problem, Yang et al. [15] put forward a context-aware fusion framework to combine the spatial and temporal activity preferences models for preferences inference. In spite of evolving from tensor factorization to RNN-based neural networks, existing methods did not make effective use of geographical information and suffered from the sparsity issue. To this end, Lian et al. [20] proposed a geography-aware recommender model based on the self-attention network (GeoSAN for short) for location recommendation. While recent works have explored the idea of adopting collaborative ranking (CR) for recommendation, there have been a few attempts to incorporate temporal information for POI recommendation using CR. Hence, Aliannejadi et al. [17] proposed a two-phase CR algorithm that combined the geographic influence of POIs based on the variance of POI popularity and user activities over time.

2.2 Successive POI Recommendation

The successive POI recommendation was first proposed by Cheng et al. [2]. Their previous work ignored the temporal relational of check-ins and only recommended the POI globally. In the past, many models were proposed based on Markov chain and RNN for POI recommendation. Cheng et al. [2] proposed a matrix factorization method (FPMC-LR) by embedding personalized Markov chains and localized regions. They not only exploited the personalized Markov chain in the check-in sequence, but also took into account users’ movement constraints. Wang et al. [22] proposed the SPENT method which used similarity tree to organize all POIs and applied Word2Vec to perform POI embedding, and then used a recurrent neural network (RNN) to model users’ successive transition behaviors. Similarly, Lu et al. [23] proposed a latent-factor and RNN-based successive POI recommendation method, named PEU-RNN, to integrate the sequential visits of POIs and user preferences to recommend POIs.

Different from the previous works that model users’ successive transition through various methods, our proposed solution believes that the behavior of a user is mainly determined by her inherent preferences which are relatively stable and invariant, at the same time the current contexts (e.g., location, time, etc.) also have impacts.

2.3 Attention Mechanisms

Nowadays, the attention mechanisms have been widely used in various fields, such as natural language processing [24], computer vision [25], recommender systems [10, 11, 18] and so on. The core of the attention mechanisms is to assign different weights to inputs, concentrating more on relevant information and ignoring irrelevant sections. Recently, the transform has achieved the best performance in machine translation, which completely eliminated recurrence and convolutions [26].

The self-attention module of transform has been widely used in recommendation systems and has achieved very good performance. SASRec [27], a sequence model based on self-attention, can not only capture long-term semantics, but also make predictions based on relatively few actions. TiSASRec [12] uses self-attention to model the absolute positions of items in the sequence and their time intervals. SAE-NAD [28] uses a multi-dimensional attention mechanism to adaptively differentiate the user preferences degrees in multiple aspects.

3 Problem Definition

In this section, the symbolic representation and problem definition are given. The user set is expressed as \( U = \{ u_1, u_2, \ldots, u_U \} \), the POI set is expressed as \( V = \{ v_1, v_2, \ldots, v_V \} \), the POI category set is expressed as \( C = \{ c_1, c_2, \ldots, c_C \} \), and the time set is expressed as
Joint Attention Networks with Inherent and Contextual Preference-Awareness for Successive...

$T = \{t_1, t_2, \ldots, t_T\}$. Each POI has its longitude and latitude and is associated with a POI category. In addition, $|U|, |V|, |C|$ represent the number of users, POIs, and POI categories, respectively. Our notation is summarized in Table 1.

### Check-in Records
The set of check-ins is denoted as $CH = \{ch_1, ch_2, \ldots\}$. Each check-in record $ch_i$ is represented as a quaternion $(u, v_i, c_j, t_i)$, representing that user $u$ checked in at POI $v_i$ at time $t_i$, and POI $v_i$ is associated with POI category $c_j$.

**Top-N Successive POI Recommendation.** Given a user $u \in U$, the users’ check-in records $CH$, the current POI $v_i \in V$ of the user $u$, the POI category $c_j \in C$ of the current POI $v_i$, and the current time $t$, recommend $N$ POIs to the user $u$ that he or she is likely to visit in the next few hours.

### 4 The JANICP Framework

In this section, we describe the proposed JANICP framework in detail. We first demonstrate how to construct indexes for time stamps and locations in our model. Then we depict the architecture of the JANICP model, which is demonstrated in Fig. 1. In order to comprehensively consider the inherent and contextual preferences of users, we propose a new method based on self-attention mechanisms. In general, the proposed model consists of three modules: (1) an inherent preferences mining module, which is used to learn users’ inherent preferences, including an embedding layer and a self-attention aggregation layer. The inherent preferences embedding layer is used to learn dense representations of user and POI category, and the self-attention aggregation layer is used to generate the inherent preferences representation by connecting the important related items in the historical check-in records and then updating the representation of each user-visit. (2)
a contextual preferences mining module, which is used to learn the users’ contextual preferences, including an embedding layer and a self-attention aggregation layer. The contextual preferences embedding layer is designed to learn the dense representation of user, POI, POI category and contexts. The contextual preferences self-attention aggregation layer is used to connect the important related items in contextual check-in records, and update the representation of each visit as the contextual preferences representation. (3) a recommender module, which includes an R-tree-based POI index structure and an attention matching layer. The POI indexing module uses R-tree to store all POIs, and retrieves the POIs near users’ current POIs to generate candidate set. Then the attention matching layer combines the inherent preferences with the contextual preferences, and calculates the softmax probability to obtain the probability that each POI in the candidate set is visited. Finally, we demonstrate how to make the model inference.

4.1 Indexing Schema

4.1.1 Time Indexing Schema

The time indexing schema is defined similar to [29], in which a day is divided into eight time slots by hours \{h_1, h_2, \ldots, h_8\}, i.e., each of which is 3 00:00:00-02:59:59, 03:00:00-05:59:59, 06:00:00-08:59:59, 09:00:00-11:59:59, 12:00:00-14:59:59, 15:00-17:59:59, 18:00:00-20:59:59, 21:00:00-23:59:59. Also, two day types \{w_1, w_2\} are used to represent weekday and non-working days, respectively. Generally speaking, the users’ behaviors present preferences variances [1]. For example, the behaviors of a user will be different at different times of the day, as well as on workdays, weekends and vacations. In addition, according to some historical research [30, 31], users’ behaviors show periodicity, and many users often visit specific POIs at specific times. Therefore, in order to capture the influence of time on user preferences, both the time slot and day type of each check-in record are extracted and used to represent the temporal context.

4.1.2 Location Indexing Schema

Based on the structure of minimum bounding rectangle, all POIs are mapped in the R-tree [32] using the POI location information with longitude and latitude. According to some studies, the geographical location of both POIs and users have an impact on their behaviors, and another evidence indicates that more users tend to visit POIs in the nearby areas [33]. Therefore, we propose an R-tree-based POI index structure, which is described in detail in Sect. 4.4.1.

4.2 Inherent Preferences Capture

As a carrier of information, vector is very important to the model. However, when one-hot encoding is used to represent each user, POI, POI category and time, it is difficult to capture user preferences due to its sparsity. Therefore, the user, POI, POI category and time are encoded into latent vectors. Latent vectors \(U_u \in \mathbb{R}^d\) represent the latent features of the user, latent vectors \(V_c \in \mathbb{R}^d\) represent the latent features of the POI, \(C_c \in \mathbb{R}^d\) represent the latent features of the POI category, and \(T_t \in \mathbb{R}^d\) represent the latent features of the time. Among them, the index size of time embedding is 16 \((2 \times 8 = 16)\), and the specific index scheme is in Sect. 4.1.1. The index sizes embedded in the user, POI, and POI categories are \(|U|, |V|, |C|\), respectively. In order to make the learned inherent preferences more stable, here we use the POI category. The output of embedding layer for each check-in \(j\) is the sum \(H_j = U_j + C_c \in \mathbb{R}^d\). For each user’s check-in record \(CH = \{cH_1, cH_2, \ldots\}\), we only consider \((n + m)\) items. If the number of check-in records \(cH_i\) of the user \(i\) are greater than \((n + m)\), the most recent \((n + m)\) records are considered. If the user’s check-in records \(cH_i\) are less than \((n + m)\), then zero is used to fill up to \((n + m)\) at the right end, and mask off the padding items during calculation. The most recent \(m\) check-in records are used to learn the user’s contextual preferences, and the earlier \(n\) check-in records are used to learn the user’s inherent preferences. For the embedding of each user’s earlier check-in, we express it as \(E(u_i) = \{H_1, H_2, \ldots, H_n\} \in \mathbb{R}^{n \times d}\).

In the successive POI recommendation, we argue that the user’s next visit is mainly affected by two aspects: inherent preferences and contextual preferences. Since inherent preferences are generally relatively stable, they need to learn from more historical check-in records of users. In addition, the same POI may have different effects on different users. For example, some people go to the cinema to watch a movie because of interest, and some people go to the cinema to watch a movie to accompany their friends. In this case, the same POI should have different weights for different users.

In order to meet the above requirements, we use self-attention mechanisms that have been successfully applied in many fields, such as natural language processing (NLP), computer vision (CV) and speech processing [34]. Let \(E(u)\) with non-padding length \(n’\) represent the embedding
matrix, that is \( E(u) = \{H_1, H_2, \ldots, H_n\} \in \mathbb{R}^{n \times d} \), where \( H_i = U_i + C_i \in \mathbb{R}^d \). First, we construct the mask matrix as \( M \in \mathbb{R}^{m \times n} \) with each element \( M_{ij} \) satisfying:

\[
M_{ij} = \begin{cases} 
1, & \text{if } i, j \leq n' \\
0, & \text{otherwise} 
\end{cases}
\]

\( e.g., M = \begin{bmatrix} 1 & 1 & 0 & \cdots & 0 \\
1 & 1 & 0 & \cdots & 0 \\
0 & 0 & 0 & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \cdots & 0 \end{bmatrix} \) with \( n' = 2 \).

And then the new check-in records are calculated through different parameter matrices \( W_1^Q, W_1^K, W_1^V \in \mathbb{R}^{d \times d} \) as:

\[
I(u) = \text{Attention}(E(u)W_1^Q, E(u)W_1^K, E(u)W_1^V, M)
\]

with:

\[
\text{Attention}(Q, K, V, M) = \left( M \ast \text{softmax} \left( \frac{QK^T}{\sqrt{d}} \right) \right)V
\]

We input \( E(u) \) as query, key and value of self-attention, respectively. First, we project query, key and value to the same space through nonlinear transformation with shared parameters. Here, the mask and softmax attention are multiplied element by element and other elements use matrix multiplication. In order to avoid the small gradient of the softmax function when \( d \) is large, we scale the dot products by \( \frac{1}{\sqrt{d}} \). We compute the potential correlation between different visits in the check-in record via the scaled dot product and assign a different weight to each visit. When predicting the \( (n' + m + 1) \)-st visit, we only take the first \( (n' + m) \in [1, c] \) check-in records as input. During training, we control the check-in records used to learn user inherent preferences by adjusting the labels of the mask matrix \( M \). Finally, we get \( I(u) \in \mathbb{R}^{m \times d} \) to represent the user’s inherent preferences. In addition, to improve the real-time responsiveness of the model, the acquiring of user stability preferences can be learned offline.

### 4.3 Contextual Preferences Capture

User’s next visit will be more easily influenced by contextual factors, such as time, weather, location, etc. For each user, only the latest \( m \) check-ins are considered as contextual trajectories. Similar to the embedding layer in the inherent preferences module, the output of embedding layer for each check-in \( j \) is the sum \( H_j' = U_j + V_j + C_j + T_j \in \mathbb{R}^d \). For the embedding of contextual check-ins, we express it as \( E'(u_i) = \{H'_1, H'_2, \ldots, H'_m\} \in \mathbb{R}^{m \times d} \).

Similar to the inherent preferences module, we still use the self-attention mechanism. Let \( E'(u) \) represent the embedding matrix, that is \( E'(u_i) = \{H'_1, H'_2, \ldots, H'_m\} \in \mathbb{R}^{m \times d} \), where \( H'_i = U_i + V_i + C_i + T_i \). Then the new contextual check-in records are calculated through different parameter matrices \( W_2^Q, W_2^K, W_2^V \in \mathbb{R}^{d \times d} \) as:

\[
S(u) = \text{Attention}(E'(u)W_2^Q, E'(u)W_2^K, E'(u)W_2^V)
\]

with:

\[
\text{Attention}(Q, K, V) = \left( \text{softmax} \left( \frac{QK^T}{\sqrt{d}} \right) \right)V
\]

Similarly, we assign a different weight to each contextual visit by scaling the dot product calculation. We can get \( S(u) \in \mathbb{R}^{m \times d} \) as a representation of the user’s contextual preferences.

### 4.4 Recommended Module

#### 4.4.1 R-Tree-Based POI Index

Previous research shows that the location of the user’s next visit is often not very far from the current location. In order to reduce the search space, R-tree [32] is used to quickly locate the area where the user is currently located, and the POIs in this area are used as a candidate set. In this way, computing efficiency is achieved through responding to user requests faster and recommending POIs to users in real time.

An R-tree [32] is a height-balanced tree data structure. Leaf node in an R-tree has entries of the form (ObjPtr, MBR), where ObjPtr identifier refers to a POI in the database and MBR is a minimum bounding rectangle which is the bounding box of POI. A non-leaf node has entries of the form (NodePtr, MBR), where NodePtr is the address of a lower node in the R-tree, and the MBR covers all rectangles of the POIs in the lower node. Figure 2b is the concrete form of R-tree nodes example according to the 8 POIs in Fig. 2a.

First, we store all POIs in an R-tree in the form of a minimum bounding area according to the spatial information. Among the basic operation algorithms of R-tree, range search is the most commonly used. The classic search operation needs to traverse all leaf nodes to determine whether the requirements are met, and its time complexity is \( O(N) \) (N
represents the number of leaf nodes). The range search algorithm of R-tree is a depth-first tree search algorithm with an average time complexity of only $O(\log(N))$. If MBRs do not overlap on $r$, the complexity is $O(\log_m N)$ where $m \in [0, M/2]$. $M$ is the maximum number of entries in a node. If MBRs overlap on $r$, it may not be logarithmic, in the worst case when all MBRs overlap on $r$, it is $O(N)$. The detailed R-tree range search algorithm is given in Algorithm 1. Concretely, given an R-tree, the search process starts from the root node $n$. If $n$ is a non-leaf node, then the entries $c$ are traversed, and its children are recursively searched if $c$ intersects the range $r$. If $n$ is a leaf node, then the process searches the object $o$ in it one by one, and once $o$ is included in the range $r$, the object $o$ is returned as the result.
Algorithm 1 Range Search Algorithm

Require: R-tree, range $r$
Ensure: candidateSet
1: candidateSet $\leftarrow$ null
2: function SEARCH(range $r$, R-tree node $n$)
3: if $n$ is a non-leaf node then
4: for each child node $c$ in $n$ do
5: if $c$ intersects $r$ then
6: SEARCH($r$, $c$)
7: end if
8: end for
9: else
10: if $n$ intersects $r$ then
11: for each object node $o$ in $n$ do
12: if $o$ inside $r$ then
13: candidateSet.add($o$)
14: end if
15: end for
16: end if
17: end if
18: return candidateSet
19: end function

4.4.2 Attention Matching Layer

This module combines users’ inherent preferences with contextual preferences, and recalls the $N$ candidate POIs that the user is most likely to visit next from the candidate set. We modify the scaled dot product attention [26] to calculate the similarity between the POI candidate set and users’ comprehensive preferences. The candidate set of $N$ POIs can be expressed as $K = \{K_1, K_2, \ldots, K_N\} \in R^{N \times d}$. This layer calculates the probability that each POI in the POI candidate set will be visited in the future:

$$S_u = \text{Concat}(I(u), S(u))$$  \hspace{0.5cm} (5)

$$P(u) = \text{Matching}(K, S_u) = \text{Sum} \left( \text{softmax} \left( \frac{KS_u^T}{\sqrt{d}} \right) \right)$$  \hspace{0.5cm} (6)

Here, $S_u = \text{Concat}(I(u), S(u)) \in R^{(n+m) \times d}$, which represents a comprehensive representation of users’ inherent preferences and contextual preferences. Calculate the attention score of $K$ and $S_u$ by scaling the dot product, and use softmax on it to get the attention weight. Finally, the Sum operation computes the weighted sum of the last dimension of the attention weights, transforming a two-dimensional matrix into an N-dimensional vector, $P(u) \in R^N$. The $N$ values in $P(u)$ respectively represent the visited probability of $N$ POIs in the candidate set. As is shown in Eq. (6), we comprehensively consider the user’s inherent preferences and contextual preferences, that is, take into account the updated representations of all the user’s check-ins, and at the same time, do not treat them equally.

4.5 Model Inference and Learning

Given the user $i$’s check-in records, the matching probability of each candidate POI $p_j \in P(u_i)$ for $j \in [1, N]$, and the label $v_k$ with number of order $k$ in the candidate set $K$, the binary cross entropy loss is adopted as the objective function:

$$L = -\sum_i \sum_j [\log(p_{ki}) + \sum_{j=1}^N \log(1 - \sigma(p_{j}))]$$  \hspace{0.5cm} (7)

where $\sigma$ is the sigmoid function. Moreover, for every positive sample $p_k$, we need to compute $(N-1)$ negative samples in the meantime. We use the Adam optimizer to train the model, and the detailed learning algorithm is shown in Algorithm 2. Among them, $\Theta = \{\Theta_1, \Theta_2\}$ is the set of model parameters. $\Theta_1 = \{W^Q_1, W^K_1, W^V_1, W^Q_2, W^K_2, W^V_2\}$ is the parameters set of attention networks. $\Theta_2 = \{U_u, V_v, C_c, T_t\}$, which represents the embedding set of users, POIs, POI categories, and time, respectively.

Algorithm 2 JANICP Algorithm

Require: check-in records $CH$, embedding dimension $d$, learning rate $lr$, dropout rate $dr$
Ensure: model parameters $\Theta$
1: Draw $\Theta_2$ from Normal Distribution $N(0, 0.01)$;
2: Initialize parameters $\Theta_1$;
3: repeat
4: shuffle the check-in records $\{(u, v_1, c_1, t_1)\}$
5: for each record $\{(u, v_1, c_1, t_1)\}$ do
6: compute $I(u)$ according to Equation (1)-(2)
7: compute $S(u)$ according to Equation (3)-(4)
8: compute $P(u)$ according to Equation (5)-(6)
9: update model parameters with Adam optimizer
10: end for
11: until converge
12: return $\Theta$

5 Experiments

5.1 Datasets

5.1.1 Data Description

We evaluated the model on three real data sets: Weeplaces, NYC, and TKY.4 Weeplaces dataset is collected from Weeplaces, a website that aims to visualize users’ check-in activities in location-based social networks (LBSN). The

4 http://www.yongliu.org/datasets.
5 http://www-public.it-sudparis.eu/~zhang_da/pub/dataset_tsmc2014.zip.
NYC and TKY dataset include long-term (approximately 10 months) check-in data for New York City and Tokyo from April 12, 2012 to February 16, 2013 collected from Four-square [15]. We preprocess these datasets by deleting users with fewer than 100 check-in records and POIs with fewer than 10 check-in records considering that they are outliers in the data. The number of users, POIs, POI categories and check-ins of each data set after preprocessing are shown in Table 2.

5.1.2 Successive Check-in Analysis

We analyzed and counted user check-in records in the three datasets. Figure 3a shows that the longer the distance between POIs, the smaller the probability of successive check-ins. Figure 3b shows that in NYC and TKY, the distance between two successive check-ins is less than 15 km, which accounts for more than 90%, and the Weeplaces takes up more than 80%. Figure 4 is the distribution of POIs in latitude and longitude in NYC. For example, if a user is in the red triangle position, their next check-in is generally in the gray area. Therefore, it is reasonable for us to filter the POI candidate set through the region query of the R-tree.

5.2 Baseline Models

We compare our JANICP with the following baselines:

- **STRNN** [8] an invariant RNN model that incorporates spatio-temporal features between consecutive visits.
- **FPMC** [6] a model that subsumes both a common Markov chain and the normal matrix factorization.
- **SHAN** [14] a novel two-layer hierarchical attention network that combines user’s long- and short-term preferences.
- **SAE-NAD** [28] a novel autoencoder-based model to learn the complex user-POI relations, which consists of a self-attentive encoder and a neighbor-aware decoder.

|          | NYC | TKY | Weeplaces |
|----------|-----|-----|-----------|
| #users   | 1083| 2293| 1467      |
| #POIs    | 5135| 7873| 8161      |
| #POI categories | 209 | 292 | 481       |
| #check-ins | 147938 | 447570 | 134762   |

Table 2 Basic dataset statistics after preprocessing
Joint Attention Networks with Inherent and Contextual Preference-Awareness for Successive POI Recommendations

**TiSASRec** [12] a method which models both the absolute positions of items as well as the time intervals between them in a sequence.

**STAN** [11] a bi-layer attention architecture that firstly aggregates spatiotemporal correlation within user trajectory.

## 5.3 Evaluation Matrices and Settings

In order to evaluate the performance of successive POI recommendations, we use two commonly used performance metrics, the top-k precision rates and recall rates. In general, the higher the recall and precision, the better the recommendation performance of the model.

Here, we give the hyperparameters used in the experiments. The embedding dimension is 50, the training epoch of 100, the learning rate of 0.005, and the dropout rate of 0.3.

## 5.4 Experimental Results on the Comparisons to Prior Methods

Tables 3 and 4 show the recommendation performance of our JANICP and baselines on the three datasets. It is clear that our JANICP outperforms all other baselines. Among all the baselines, STRNN performs the worst because RNN cannot solve the problem of long-term dependence. The poor performance of FPMC may be due to the fact that it only captures sequential effects and ignores spatiotemporal influences. The performance of SHAN is better than STRNN, FPMC and SAE-NAD. It uses a hierarchical attention network that combines the user’s dynamic long-term and short-term preferences. STAN and TiSASRec outperform the other methods significantly, both of which take the time interval into account. Only JANICP fully considers the user’s inherent preferences and contextual preferences, and fully considers the POI category, time and geographic influence. In addition, the six baselines did not consider the impact of POI categories, and FPMC, SHAN did not consider the impact of temporal and spatial relationships, which may be the reason why the performance is slightly worse than JANICP.

### 5.5 Experimental Results on Different Versions of JANICP

In order to verify the effectiveness of several key modules designed in our model, we conducted more experiments to evaluate whether there are model variants of this type of design.

**JANICP-inherent** Users’ stable inherent preferences and dynamic contextual preferences simultaneously affect the users’ behavior. To verify the importance of the two preferences, we designed variants JANICP-inherent and JANICP-contextual. JANICP-inherent only considers users’ inherent preferences, i.e. not including the users’ contextual preferences.
Users’ next visit will be largely influenced by contextual factors (e.g., location, time, etc.). Therefore, we design variant JANICP-contextual, which only considers the users’ contextual preferences and does not consider the inherent preferences.

**JANICP-R – tree** Use the R-tree to filter the POI candidate set. Here the query range of latitude and longitude is set to 0.3.

Figure 5 shows the experimental results of these JANICP variants. In general, JANICP performs better than its variants on the three data sets. This shows that every part of the design plays an important role. The performance difference between JANICP-inherent and JANICP-contextual in the three datasets is fairly small. JANICP-R – tree performs similarly on TKY and NYC, but differs greatly on Weeplaces. This is caused by the different sparsity of the dataset in space.

**5.6 Effect of Query Range**

We conducted a series of experiments on different query ranges. Set the R-tree query latitude and longitude range to 0.1, 0.2, 0.3, and without R-tree. Figure 6 shows that in the three data sets, the performance without R-tree is the best. The experimental results on NYC and TKY are similar, the query range increases, and the recall rate also increases. This is due to the similarity of the data sets. It can be seen from Fig. 3 that the probability statistics and cumulative probability distributions of successive check-in distances in the two data sets are similar. The Weeplaces is slightly different because the distance between successive check-ins is greater and the time interval between successive check-ins is longer.

**5.7 Effect of Embedding Dimensions**

In our model, we change the embedding dimension from 10 to 70 with a step size of 10. We use Rec@N as the evaluation criterion. Figure 7 shows that various embedding
dimensions $d$ lead to some differences in the experimental results. Figure 7a shows that $d=20$ is the best dimension. Figure 7b shows that $d=50$ is the best dimension of NYC and TKY, and $d=30$ is the best dimension of Weeplaces. Figure 7c shows that $d=30$ is the best dimension of NYC and TKY, and $d=40$ is the best dimension of Weeplaces. In general, our model is relatively stable and is not significantly affected by the hyperparameter $d$.

6 Conclusion

In this paper, we proposed a model based on the self-attention mechanism, abbreviated as JANICP, for successive POI recommendation. JANICP combines users’ inherent preferences and contextual preferences. In order to improve the real-time performance of the recommendation, users’ inherent preferences learning can be performed offline. In addition, in order to improve the response speed of the model, we recommend using R-tree to filter the POI candidate set. We conducted a lot of experiments on three real data sets. Experimental results showed that JANICP outperformed other state-of-the-art models in terms of precision and recall.

Acknowledgements  This work was supported by National Key R &D Program of China (No.2021YFF0900802), Natural Science Foundation of China (No.61972230) and Natural Science Foundation of Shandong Province (No.ZR2019LZ0008).

Author contributions  HZ contributed to methodology and writing; WH contributed to methodology evaluation, writing, supervision, and funding acquisition; LC contributed to evaluation and funding acquisition; LL contributed to evaluation and writing; ZY contributed to writing review and editing; KZ contributed to evaluation.

Availability of data and materials  http://www.yongliu.org/datasets, http://www-public.it-sudparis.eu/~zhang_da/pub/dataset_tsmc2014.zip.

Declarations

Conflict of interest  The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Open Access  This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article’s Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article’s Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

1. Gao H, Tang J, Hu X, Liu H (2013) Exploring temporal effects for location recommendation on location-based social networks. In: Proceedings of the 7th ACM conference on recommender systems, pp. 93–100
2. Cheng C, Yang H, Lyu MR, King I (2013) Where you like to go next: successive point-of-interest recommendation. In: Twenty-Third International Joint Conference on Artificial Intelligence
3. Yu F, Cui L, Guo W, Lu X, Li Q, Lu H (2020) A category-aware deep model for successive poi recommendation on sparse check-in data. In: Proceedings of the Web Conference 2020, pp. 1264–1274
4. Lu Y-S, Huang J-L (2020) Glr: a graph-based latent representation model for successive poi recommendation. Future Gener Comput Syst 102:230–244
5. Cheng C, Yang H, King I, Lyu M (2012) Fused matrix factorization with geographical and social influence in location-based social networks. In: Proceedings of the AAAI Conference on Artificial Intelligence 26:17–23
6. Rendle S, Freudenthaler C, Schmidt-Thieme L (2010) Factorizing personalized markov chains for next-basket recommendation. In: Proceedings of the 19th International Conference on World Wide Web, pp. 811–820
7. Feng J, Li Y, Zhang C, Sun F, Meng F, Guo A, Jin D (2018) Deepmove: predicting human mobility with attentional recurrent
networks. In: Proceedings of the 2018 World Wide Web Conference, pp. 1459–1468
8. Liu Q, Wu S, Wang L, Tan T (2016) Predicting the next location: a recurrent model with spatial and temporal contexts. In: Thirtieth AAAI Conference on Artificial Intelligence
9. Sun K, Qian T, Chen T, Liang Y, Nguyen QVH, Yin H (2020) Where to go next: modeling long-and short-term user preferences for point-of-interest recommendation. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, pp. 214–221
10. Ying H, Wu J, Xu G, Liu Y, Li Q, Wang J, Zhang X, Xiong H (2019) Time-aware metric embedding with asymmetric projection for successive poi recommendation. World Wide Web 22(5):2209–2224
11. Luo Y, Liu Q, Liu Z (2021) Stan: Spatio-temporal attention network for next location recommendation. In: Proceedings of the Web Conference 2021, pp. 2177–2185
12. Li J, Wang Y, McAuley J (2020) Time interval aware self-attention for sequential recommendation. In: Proceedings of the 13th International Conference on Web Search and Data Mining, pp. 322–330
13. Zhang S, Tay Y, Yao L, Sun A (2018) Next item recommendation with self-attention. arXiv preprint arXiv:1808.06414
14. Ying H, Zhuang F, Zhang F, Liu Y, Xu G, Xie X, Xiong H, Wu J (2018) Sequential recommender system based on hierarchical attention network. In: IJCAI International Joint Conference on Artificial Intelligence
15. Yang D, Zhang D, Zheng VW, Yu Z (2014) Modeling user activity preference by leveraging user spatial temporal characteristics in LBSNs. IEEE Trans Syst Man Cybern Syst 45(1):129–142
16. Zhao S, Zhao T, King I, Lyu MR (2017) Geo-teaser: Geo-temporal sequential embedding rank for point-of-interest recommendation. In: Proceedings of the 26th International Conference on World Wide Web Companion, pp. 153–162
17. Aliannejadi M, Rafailidis D, Crestani F (2019) A joint two-phase time-sensitive regularized collaborative ranking model for point of interest recommendation. IEEE Trans Knowl Data Eng 32(6):1050–1063
18. Yin J, Li Y, Liu Z, Xu J, Xia B, Li Q (2019) Adpr: an attention-based deep learning point-of-interest recommendation framework. In: 2019 International Joint Conference on Neural Networks (IJCNN), pp. 1–8 . IEEE
19. Feng S, Li X, Zeng Y, Cong G, Chee YM, Yuan Q (2015) Personalized ranking metric embedding for next new poi recommendation. In: Twenty-Fourth International Joint Conference on Artificial Intelligence
20. Lian D, Wu Y, Ge Y, Xie X, Chen E (2020) Geography-aware sequential location recommendation. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 2009–2019
21. Baral R,伊想 S, Li T, Balakrishnan N (2018) Close: contextualized location sequence recommender. In: Proceedings of the 12th ACM Conference on Recommender Systems, pp. 470–474
22. Wang MF, Lu YS, Huang JL (2019) Spent: a successive poi recommendation method using similarity-based poi embedding and recurrent neural network with temporal influence. In: 2019 IEEE International Conference on Big Data and Smart Computing (BigComp)
23. Lu Y S, Shi w WY, Gau H Y, Chung K-C, Huang J-L (2019) On successive point-of-interest recommendation. World Wide Web 22(3):1151–1173
24. Galassi A, Lippi M, Torroni P (2020) Attention in natural language processing. IEEE Trans Neural Networks Learn Syst 32(10):4291–4308
25. Parmar N, Vaswani A, Uszkoreit J, Kaiser L, Shazeer N, Ku A, Tran D (2018) Image transformer. In: International Conference on Machine Learning, pp. 4055–4064 . PMLR
26. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser L, Polosukhin I (2017) Attention is all you need. Advances in neural information processing systems 30
27. Kang W-C, McAuley J (2018) Self-attentive sequential recommendation. In: 2018 IEEE International Conference on Data Mining (ICDM), pp. 197–206 . IEEE
28. Ma C, Zhang Y, Wang Q, Liu X (2018) Point-of-interest recommendation: Exploiting self-attentive autoencoders with neighbor-aware influence. In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management, pp. 697–706
29. Zhao S, Zhao T, Yang H, Lyu MR, King I (2016) Stellar: Spatial-temporal latent ranking for successive point-of-interest recommendation. In: Thirtieth AAAI Conference on Artificial Intelligence
30. Yuan Q, Cong G, Ma Z, Sun A, Thalmann NM (2013) Time-aware point-of-interest recommendation. In: Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 363–372
31. Cho E, Myers SA, Leskovec J (2011) Friendship and mobility: user movement in location-based social networks. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1082–1090
32. Guttmann A (1984) R-trees: A dynamic index structure for spatial searching. In: Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data, pp. 47–57
33. Gau HY, Lu Y-S, Huang J-L (2017) A grid-based successive point-of-interest recommendation method. In: 2017 10th International Conference on Ubi-media Computing and Workshops (Ubi-Media), pp. 1–6 . IEEE
34. Lin T, Wang Y, Liu X, Qiu X (2021) A survey of transformers. arXiv preprint arXiv:2106.04554