Where and When: Predict Next POI and Its Explicit Timestamp in Sequential Recommendation

Yuanbo Xu[1] , **Hongxu Shen**[1] , **Yiheng Jiang**[1] , **En Wang**[1*]

[1]MIC Lab, College of Computer Science and Technology, Jilin University

{yuanbox, wangen}@jlu.edu.cn, {shenhx23, jiangyh22}@mails.jlu.edu.cn

Abstract

Sequential point-of-interest (POI) recommendation aims to recommend the next POI for users in accordance with their historical check-in information. However, few attempts treat *timestamps* of check-ins as a core factor for sequence models, leading to insufficient insight into user behavior and suboptimal recommendations. To address these limitations, we propose to assign equal importance to both POIs and their timestamps, shifting the point of view to recommend the next POI and predict the corresponding timestamp. Along these lines, we present the Time-Aware POI Recommender with Timestamp Prediction (TAPT), a multi-task learning framework for explainable POI recommendations. Specifically, we begin by decoupling timestamps into multi-dimensional vectors and propose a timestamp encoding module to encode these vectors explicitly. Additionally, we design a specialized timestamp prediction module built on the traditional sequence-based POI recommender backbone, effectively learning the strong correlation between POIs and their corresponding timestamps through these two modules. We evaluated the proposed model with three real-world LBSN datasets and demonstrated that TAPT achieves comparable or superior performance in POI recommendation compared to the baseline backbone. Besides, TAPT can not only recommend the next POI, but predict the corresponding timestamp in the future.

1 Introduction

The rapid growth of location-based social networks (LBSNs) has sparked significant interest in personalized location recommendation services [Wang *et al.*, 2019b]. One key approach within these services is sequential point-of-interest (POI) recommendation, which focuses on recommending the next POI that a user might explore based on his or her previous visit sequence [Manotumruksa *et al.*, 2018]. With advancements in information technology, researchers have developed numerous deep learning methods in this field [Wang *et al.*, 2022; Lian *et al.*, 2020; Xu *et al.*, 2025]. These methods include Multi-Layer Perceptron (MLP) [Zhou *et al.*, 2022; Jiang *et al.*, 2024], Recurrent Neural Networks (RNN) [Hidasi *et al.*, 2016a; Yang *et al.*, 2020], Convolutional Neural Networks (CNN) [Tang and Wang, 2018; Yuan *et al.*, 2019], and Transformer-based models [Lian *et al.*, 2020; Luo *et al.*, 2021; Devlin *et al.*, 2019]. Researchers have recognized the potential of these architectures and extended them to sequential POI recommendations, leveraging both spatial and temporal information to capture user preferences and predict the next POI based on previous visits.

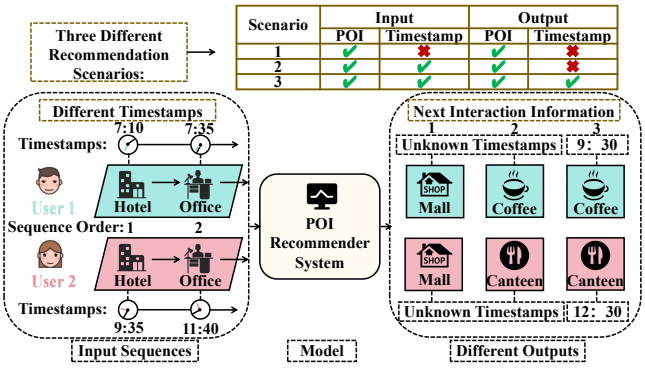


Figure 1: Illustration of three recommendation scenarios. Two identical POI sequences with different timestamps yield three distinct recommendation results. Scenario 1: Recommend the same POIs. Scenario 2: Take timestamps into account, recommending different POIs. Scenario 3: Explicitly utilize timestamps to recommend different POIs and their respective timestamps.

Despite the commendable performance achieved by existing methods, they exhibit two main limitations when modeling POI transitions, as shown in Scenario 1 and Scenario 2 in Figure 1. **Scenario 1: Recommend POIs based on sequential correlations.** By prioritizing sequential correlations, these methods overlook the specific timestamp information embedded within sequences and, therefore, lead to an incomplete modeling of user behavior, producing suboptimal recommendations [Rendle *et al.*, 2010; Kang and McAuley, 2018; Sun *et al.*, 2019; Xu *et al.*, 2024]. **Scenario 2: Recommend POIs with timestamps as auxiliary information.** Scenario 2 builds upon Scenario 1 by incorporating timestamps as auxiliary inputs, enabling the model to capture se-

* Corresponding author.

quential dependencies through temporal signals and produce personalized recommendations. Nevertheless, both scenarios concentrate exclusively on predicting the next check-in POI while overlooking the associated timestamp, thereby compromising recommendation timeliness and impeding the real-time deployment of LBSN services.

To resolve the bottlenecks in the aforesaid scenarios, we introduce a new scenario. **Scenario 3: Recommend the next check-in POI and predict the corresponding timestamp simultaneously.** In this scenario, the recommendations possess practical significance in terms of timeliness, facilitate timely individual travel planning, and enhance the effectiveness of LBSN applications. Since individuals tend to visit locations based on specific temporal and spatial contexts, accurately predicting their next destination at the most probable time will enhance the relevance and usability of such services. As shown in Figure 1, two users with the same POI sequence but different timestamp sequences are shown on the left, and the results of three scenarios are compared on the right. In **Scenario 1**, the system recommends identical POIs to both users due to identical input sequences. In **Scenario 2**, the timestamp information is an auxiliary input to the model, leading to different POI recommendations, but without corresponding time information. Our proposed **Scenario 3** recommends personalized POIs for users while predicting timestamps for their next interactions, reflecting when and where users are likely to go, and providing a holistic recommendation service.

To realize the above concept, we propose the Time-Aware POI recommender with Timestamp prediction (TAPT) as a multi-task learning framework. Our approach begins by decoupling timestamps into multidimensional vectors, which are then encoded into trainable embeddings via a timestamp encoder, which explicitly leverages check-in timestamps. Next, we design a connection layer to combine POI and timestamp embeddings into joint embeddings. We feed this joint embedding into a POI recommendation backbone and pass the output from the backbone into a designed timestamp predictor, enabling multitask learning through a unified loss function. To validate the effectiveness of TAPT, we conduct extensive experiments on three public LBSN datasets. The results demonstrate that TAPT achieves comparable or even superior POI recommendation performance compared to the baseline backbone. Besides, TAPT can recommend the next POI and predict the corresponding timestamp in the future. Our contributions are summarized as follows:

- This paper tackles the next check-in time prediction in the sequential POI recommendation scenario, which has been overlooked by previous works. It lights on a novel approach for meaningful and timely recommendations.
- This paper proposes an end-to-end multi-target learning framework, which attaches equal importance to both the next POI classification and time prediction.
- Experimental results on three real-world datasets validate its superior recommendation quality and temporal prediction precision compared to baseline models. The code for our work is available at: https://github.com/MICLab-Rec/TAPT.

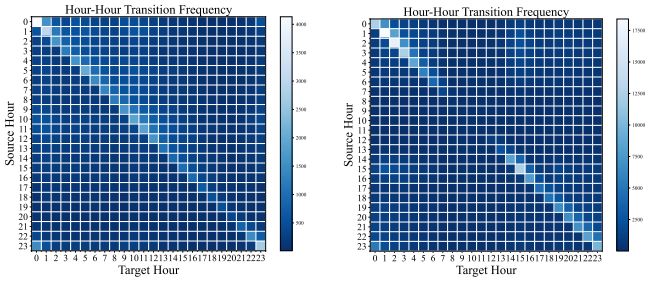
2 Related Work

2.1 Sequential POI Recommendation

Numerous excellent sequential POI recommendation methods have emerged with the advancement of information technology and the increasing availability of user check-in data. Traditional methods typically use Markov Chains [He and McAuley, 2016] and Matrix Factorization [Twardowski, 2016] to model the transition patterns between POIs [Liu *et al.*, 2016]. For instance, FPMC [Rendle *et al.*, 2010] linearly combines the two to model transitions between POIs across a series of baskets. With the advancement of deep learning technologies, RNN-based [Hidasi *et al.*, 2016b] and CNN-based [Yuan *et al.*, 2019] models have demonstrated high efficiency in sequence modeling. Among these, GRU4Rec [Hidasi *et al.*, 2016a] utilizes a modified gated recurrent unit to learn the patterns of users' dynamic preferences, while Caser [Tang and Wang, 2018] employs horizontal and vertical convolutional kernels to capture sequential dependencies from both local and global perspectives. Meanwhile, the success of Transformer-based models [Vaswani *et al.*, 2017] has also inspired the development of Transformer-based sequence encoders. For example, SASRec [Kang and McAuley, 2018] is the pioneering approach to incorporate self-attention networks into the sequential recommendation domain. Bert4Rec [Sun *et al.*, 2019] advances this by extending SASRec to capture bidirectional sequential dependencies. To account for temporal factors, TiSASRec [Li *et al.*, 2020] develops a time-aware self-attention mechanism to examine how varying time intervals impact predictions. Besides, the FMLP4Rec [Zhou *et al.*, 2022] method combines multi-head self-attention with frequency domain modeling to enhance sequence modeling capabilities in recommender systems. STAN [Luo *et al.*, 2021] is a transformer-based model that modifies the attention coefficients by incorporating temporal and spatial interval information to capture sequential dependencies more effectively. However, these methods either overlook or only implicitly consider the timestamp information of check-ins. GeoSAN [Lian *et al.*, 2020] introduces a novel self-attention-based geography encoder, demonstrating state-of-the-art performance in modeling precise locations.

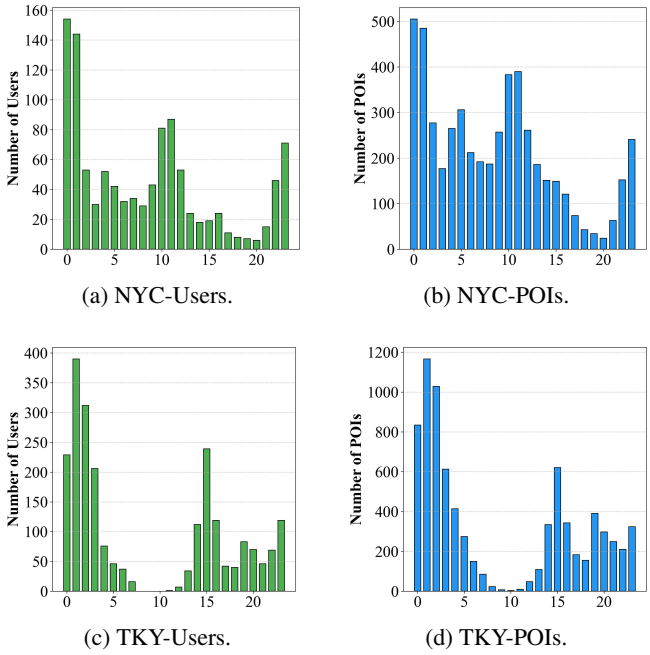
2.2 Timestamp Encoding

Timestamp encoding has been extensively researched in both academia and industry. Some previous works typically encode the time gaps in sequential models[Chang *et al.*, 2023], which captures the significance of sequential information but fail to directly model time information. Other works divide a day into morning, noon, evening, and night, applying different graph models to takeaway recommendations at different times of the day [Zhang *et al.*, 2023]. However, this method is clearly difficult to generalize to other tasks. Meanwhile, the timestamp encoding method [Li *et al.*, 2022] encodes the hours of the day as hour embeddings, discretely perceiving timestamp information. However, these methods merely use timestamp encoding as auxiliary information and do not leverage this encoding to predict the next interaction timestamp.



(a) NYC. (b) TKY.

Figure 2: The temporal distribution of sequential transitions on the NYC and TKY datasets.



(a) NYC-Users. (b) NYC-POIs.

(c) TKY-Users. (d) TKY-POIs.

Figure 3: The temporal distribution of users and POIs on the NYC and TKY datasets.

3 Preliminaries

3.1 Basic Definition

Definition 1: (Check-in) A check-in is represented as $c = (u, p, \ell, t)$, where user u visits POI p at timestamp t, and ℓ represents the geographical coordinates of p.

Definition 2: (Check-in Sequence) A check-in sequence consists of a chronologically ordered set of check-in records. Let $S^u = \{c_1, c_2, \cdots, c_m\}$ represent the check-in sequence of user u, where c_k denotes the k-th check-in in the sequence, and m denotes the length of the sequence.

Definition 3: (Sequential POI Recommendation) Given user u's check-in sequence, the goal of sequential POI recommendation is to recommend the Top-K POIs that the user will most likely be interested in at the next interaction.

Definition 4: (Sequential Timestamp Prediction) Given user u's check-in sequence, the objective of sequential timestamp prediction is to forecast the timestamp of the user's next visit to a specific POI.

3.2 Problem Definition

Given a user u's historical sequence S^u, the POI recommendation and timestamp prediction problem can be formalized as

$$\mathcal{F}(S^u) \rightarrow \mathcal{R}^u, \tilde{t}^u, \tag{1}$$

where $\mathcal{R}^u$ is the Top-K recommendation list and $\tilde{t}^u$ is the predicted timestamp for the next check-in. $\mathcal{F}(\cdot)$ represents the multi-task learning framework we aim to learn.

3.3 Data Observations and Analyses

To address the proposed problem, we conduct data analyses on the NYC and TKY datasets. We split all historical check-in sequences into consecutive pairs and treat the former and the latter check-in as source and target, respectively. Then, we statistically analyze the distribution of time transferring from sources to targets. Figure 2 shows these transitions, with the X-axis representing the target hour and the Y-axis the source hour. Each element indicates the frequency of transitions between the corresponding hours. We observe frequent transitions between adjacent hours, as shown by the diagonal dominance in the transition matrix. This indicates that timestamps play an important role in POI recommendation due to short-interval behavioral patterns. Secondly, we analyze the most active hour for each user and examine the peak hour for each POI. The temporal distributions are plotted in Figure 3, with the X-axis representing the hours of the day and the Y-axis showing the number of active users (or popular POIs) during each time period. We can conclude a pattern that the number of users performing check-in behaviors and the number of POIs being accessed exhibit synchronized changes over time. Specifically, both numbers achieve the peak point at midnight and reach the lowest point at 8 p.m. in NYC whereas the situation comes at 1 a.m. and 10 a.m. in TKY, respectively.

Motivated by these observations, predicting the next check-in time can be helpful in deploying LBSN services. This paper proposes a simple and effective method to establish the contact between POI recommendation and time prediction.

4 Methodology

4.1 Encoder Module

Figure 4 illustrates the overall architecture of TAPT. The encoder module of TAPT, consisting of the POI encoder and the timestamp encoder, encodes the POI and timestamp sequences derived from a user's check-in history. For the POI sequence, we transform it into $\mathbf{E}_p \in \mathbb{R}^{m \times d_1}$ through POI embedding techniques, where m represents the sequence length and d_1 represents the POI embedding dimension.

$$\mathbf{E}_p = [\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_m], \tag{2}$$

where $\mathbf{E}_p$ denotes the encoded POI embedding, $\mathbf{e}_i \in \mathbb{R}^{1 \times d_1}$ is the embedding vector of the i-th POI in the sequence. We split each timestamp into three components to capture daily behavior patterns: hour, minute, and second. These discrete values are transformed into embeddings, allowing the model

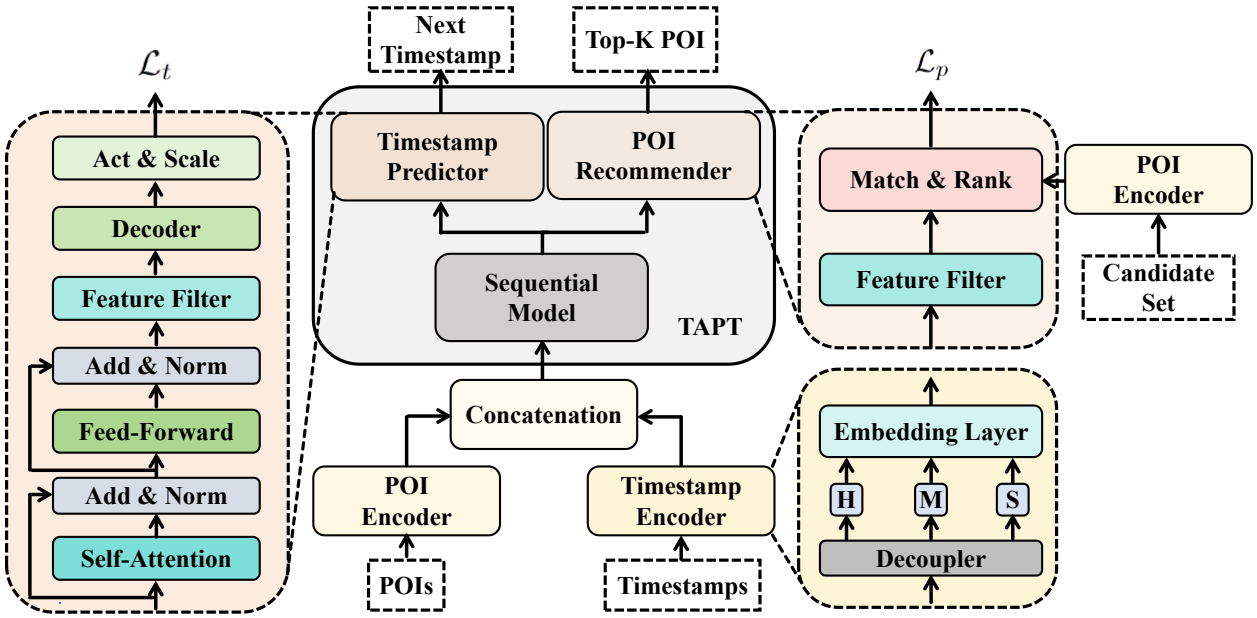


Figure 4: The overall framework of the proposed TAPT.

to capture the temporal aspects of each check-in with fine granularity. For instance, the hour component has 24 discrete values, ranging from 0 to 23.

$$\mathbf{T} = [\mathbf{T}_1, \mathbf{T}_2, \ldots, \mathbf{T}_m] = \begin{bmatrix} h_1 & m_1 & s_1 \\ h_2 & m_2 & s_2 \\ \vdots & \vdots & \vdots \\ h_m & m_m & s_m \end{bmatrix}, \quad (3)$$

where $\mathbf{T} \in \mathbb{R}^{\mathbf{m}\times\mathbf{3}}$ represents the decoupled timestamp sequence, and $\mathbf{T}_i \in \mathbb{R}^{1\times 3}$ denotes the timestamp at the i-th time step. Furthermore, h_i, m_i, and s_i represent the hour, minute, and second of the i-th time step to capture the user's behavior within a single day. Next, we encode $\mathbf{T}$ into timestamp embeddings as follows:

$$\mathbf{E}_t = [\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_m], \quad (4)$$

where $\mathbf{E}_t \in \mathbb{R}^{m\times d_2}$ is the timestamp embedding, with d_2 as the timestamp embedding dimension, and $\mathbf{t}_i \in \mathbb{R}^{1\times d_2}$ is the embedding of each timestamp $\mathbf{T}_i$ after encoding. We decompose timestamps as discrete hour, minute, and second indices, and embed them to high-dimensional representations $\mathbf{h}_i$, $\mathbf{m}_i$, and $\mathbf{s}_i$, where the vocabulary sizes are 24, 60 and 60, respectively.

$$\mathbf{t}_i = (\mathbf{h}_i \| \mathbf{m}_i \| \mathbf{s}_i), \quad (5)$$

where $\|$ denotes the concatenation operation. After obtaining the POI embeddings and timestamp embeddings, we input both into a concatenation layer to produce a joint embedding. This joint embedding is used to learn the strong correlations between POIs and timestamps and to facilitate subsequent multitask learning. The process is as follows:

$$\mathbf{E} = (\mathbf{E}_p \| \mathbf{E}_t), \quad (6)$$

where $\mathbf{E} \in \mathbb{R}^{m\times d}$ represents the joint embedding, with $d = d_1 + d_2$, and $\|$ denoting the concatenation operation.

4.2 POI Recommender Module

We feed $\mathbf{E}$ into the backbone of the POI recommendation model, allowing it to learn time-aware POI preferences and capture user behavior timestamp features.

$$\mathbf{Z} = \mathrm{Backbone}(\mathbf{E}), \quad (7)$$

where $\mathbf{Z} \in \mathbb{R}^{m\times d}$ represents the learned representation, $\mathrm{Backbone}(\cdot)$ denotes the backbone of the sequential POI recommendation model, and we extract the first d_1 elements from $\mathbf{Z}$ along the last dimension to form the POI-related representation matrix $\mathbf{Z}_p \in \mathbb{R}^{m\times d_1}$. Therefore, the preference score is calculated as follows:

$$y_{i,j} = f(\mathbf{Z}_{pi}, \mathbf{C}_j), \quad (8)$$

where $y_{i,j}$ is the preference score of the j-th POI in the candidate set, $f(\cdot)$ is the inner product function, $\mathbf{Z}_{pi} \in \mathbb{R}^{1\times d_1}$ is the user preference vector at the i-th position in $\mathbf{Z}_p$, and $\mathbf{C}_j \in \mathbb{R}^{1\times d_1}$ is the embedding vector of the j-th POI in the candidate set.

4.3 Timestamp Predictor Module

In the timestamp prediction branch, we employ a Transformer to model the transition pattern of temporal information and avoid the instability brought by various backbones. Given the input $\mathbf{Z}$, we calculate the linearly transformed features.

$$\mathbf{Q} = \mathbf{Z}\mathbf{W}_Q, \mathbf{K} = \mathbf{Z}\mathbf{W}_K, \mathbf{V} = \mathbf{Z}\mathbf{W}_V, \quad (9)$$

where $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V \in \mathbb{R}^{d\times d}$ are learnable parameters and $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{m\times d}$. We process the above features through a multi-head self-attention model:

$$\mathrm{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathrm{Concat}(\mathrm{head}_1, \ldots, \mathrm{head}_i)\mathbf{W}_O,$$

$$\text{head}_i = \text{Attention}(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V), \quad (10)$$

$$\text{Attention}\left(\mathbf{Q}, \mathbf{K}, \mathbf{V}\right) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}}\right)\mathbf{V}.$$

The outputs of the attention heads are merged via a linear transformation $\mathbf{W}_o \in \mathbb{R}^{d\times d}$, integrating information across attention spaces. Next, we apply a residual connection and layer normalization to the output.

$$\mathbf{Z}_1 = \text{LayerNorm}(\text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) + \mathbf{Z}). \quad (11)$$

We feed $\mathbf{Z}_1 \in \mathbb{R}^{m\times d}$ through the feed-forward network with residual connections and normalization to enhance feature transformation and capture nonlinear relationships.

$$\mathbf{Z}' = \left(\text{LayerNorm}\left(\text{ReLU}(\mathbf{Z}_1\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2\right) + \mathbf{Z}_1\right), \quad (12)$$

where $\mathbf{W}_1 \in \mathbb{R}^{d\times d_h}$ and $\mathbf{W}_2 \in \mathbb{R}^{d_h\times d}$ are the learnable weight matrices, and $\mathbf{b}_1 \in \mathbb{R}^{1\times d_h}$ and $\mathbf{b}_2 \in \mathbb{R}^{1\times d}$ are the bias terms, while $\mathbf{Z}' \in \mathbb{R}^{m\times d}$ is the further learned joint representation. Subsequently, we extract the last d_2 elements from $\mathbf{Z}'$ along the last dimension to form the timestamp-related features $\mathbf{Z}_t \in \mathbb{R}^{m\times d_2}$, where $d = d_1 + d_2$. Then, we decode $\mathbf{Z}_t$ through a decoder, translating the extracted features into meaningful timestamp predictions.

$$\mathbf{Z}_t' = \mathbf{Z}_t\mathbf{W} + \mathbf{b}, \quad (13)$$

where $\mathbf{Z}_t' \in \mathbb{R}^{m\times 3}$ is the decoded representation, and the parameters $\mathbf{W} \in \mathbb{R}^{\mathbf{d_2}\times \mathbf{3}}$ and $\mathbf{b} \in \mathbb{R}^{\mathbf{1}\times \mathbf{3}}$ are trainable parameters in the model. Finally, $\mathbf{Z}_t'$ is mapped back to discrete values that reflect timestamp information through an activation function and scaling operation.

$$\hat{\mathbf{T}}_\mathbf{i} = \text{Scale}(\text{Sigmoid}(\mathbf{Z}_t')), \quad (14)$$

where $\hat{\mathbf{T}}_\mathbf{i} \in \mathbb{R}^{m\times 3}$ is the final predicted timestamp information. $\text{Scale}(\cdot)$ and $\text{Sigmoid}(\cdot)$ represent the scaling and activation operations, respectively. Equation (14) is designed to predict the hour, minute, and second of the next check-in. Since $\mathbf{Sigmoid}(\cdot)$ restricts values in $[0, 1]$, the $\mathbf{Scale}(\cdot)$ function maps the results into three ranges, i.e., $[0, 23]$, $[0, 59]$ and $[0, 59]$, corresponding to the hour, minute, and second index, respectively. After the above steps, we predict the final hour, minute, and second to infer the timestamp of the user's next interaction.

4.4 Model Training

During the model training process, our proposed TAPT framework functions as a multi-task learning system, incorporating two distinct loss functions:

$$\mathcal{L} = \lambda\mathcal{L}_p + (1-\lambda)\mathcal{L}_t, \quad (15)$$

where $\mathcal{L}_p$ denotes the loss function for the POI recommendation, such as Binary Cross-Entropy (BCE) [Li *et al.*, 2020].

$$\mathcal{L}_p = -\sum_{i=1}^{m}\left(\log(\sigma(\mathrm{y}_{i,p})) + \log(1-\sigma(\mathrm{y}_{i,n}))\right), \quad (16)$$

where m denotes the length of the POI sequence, σ represents the sigmoid function, $\mathrm{y}_{i,p}$ is the logits for the i-th positive sample, and $\mathrm{y}_{i,n}$ is the logits for the i-th negative sample.

Dataset	TKY	NYC	Gowalla
#user	2,293	1,083	36,371
#POI	7,873	5,135	129,939
#check-in	447,570	147,938	3,136,496
sparsity	97.52%	97.33%	99.93%
avg. seq. length	193.19	134.60	84.24

Table 1: Statistics of the three datasets.

In equation (15), λ is a hyperparameter that balances the two loss functions. We use the Mean Squared Error (MSE) loss for optimizing $\mathcal{L}_t$.

$$\mathcal{L}_t = \frac{1}{m}\sum_{i=1}^{m}(T_i - \hat{T}_i)^2, \quad (17)$$

where T_i is the true timestamp of the i-th sample, and $\hat{T}_i$ is the predicted timestamp of the i-th sample.

5 Experiments

5.1 Datasets

We evaluate our framework using three publicly available LBSN datasets from Foursquare (Tokyo and New York City) [Yang *et al.*, 2015] and Gowalla (California and Nevada) [Cho *et al.*, 2011]. We filter out inactive users (fewer than 20 POI visits) and unpopular POIs (fewer than 10 interactions) to ensure data quality [Yang *et al.*, 2022]. The check-in records are organized in chronological order. The initial 80% of the data is allocated for training, the subsequent 10% for validation, and the final 10% for testing. Table 1 summarizes the details of the three processed datasets.

5.2 Baselines

We choose the following sequential models for POI recommendation as baseline backbones. 1) **GRU4Rec** [Hidasi *et al.*, 2016a]: Uses a Gated Recurrent Unit to model user interaction sequences for recommendations. 2) **Caser** [Tang and Wang, 2018]: Embeds POIs in user interaction history as images, using convolutional filters to capture both local and global sequential dependencies. 3) **SASRec** [Kang and McAuley, 2018]: Leverages users' long-term semantics and recent actions simultaneously for accurate POI recommendations. 4) **BERT4Rec** [Sun *et al.*, 2019]: Addresses the limitations of unidirectional sequential recommenders by modeling user behavior sequences under the BERT framework [Devlin *et al.*, 2019]. 5) **TiSASRec** [Li *et al.*, 2020]: Utilizes timestamps and time intervals between user-POI interactions to improve next-POI prediction. 6) **FMLP4Rec** [Zhou *et al.*, 2022]: Proposes a full MLP model that adaptively reduces noise information in the frequency domain through a learnable filter to perform POI recommendation. 7) **STAN** [Luo *et al.*, 2021]: Models relative spatial-temporal information among POIs with a bi-layer attention architecture, creating a state-of-the-art sequential POI recommender. 8) **GeoSAN** [Lian *et al.*, 2020]: Introduces a novel self-attention-based geography encoder, demonstrating state-of-the-art performance in modeling precise locations.

Dataset	TKY				NYC				Gowalla			
Metric	H@5	N@5	H@10	N@10	H@5	N@5	H@10	N@10	H@5	N@5	H@10	N@10
GRU4Rec	0.4588	0.3217	0.5717	0.3516	0.4903	0.3730	0.5688	0.3971	0.2445	0.1677	0.3355	0.1990
w. TAPT	**0.4666**	**0.3226**	**0.5931**	**0.3649**	0.4866	0.3707	**0.5771**	0.3944	**0.2610**	**0.1823**	**0.3516**	**0.2113**
Improv.	1.70%	0.27%	3.74%	3.78%	-0.75%	-0.61%	1.45%	-0.67%	6.74%	8.70%	4.76%	6.18%
Caser	0.3764	0.2626	0.5207	0.3078	0.4257	0.2908	0.5309	0.3260	0.2903	0.2072	0.3739	0.2358
w. TAPT	**0.3812**	**0.2636**	**0.5358**	**0.3154**	**0.4269**	0.2887	**0.5372**	0.3246	**0.2998**	**0.2148**	**0.3843**	**0.2428**
Improv.	1.27%	0.38%	2.89%	2.46%	0.28%	-0.72%	1.18%	-0.42%	3.27%	3.66%	2.78%	2.96%
SASRec	0.4610	0.3242	0.5844	0.3636	0.4691	0.3356	0.5660	0.3688	0.2433	0.1672	0.3283	0.1946
w. TAPT	**0.4719**	**0.3279**	**0.6066**	**0.3780**	**0.4875**	**0.3525**	**0.5873**	**0.3853**	**0.2565**	**0.1764**	**0.3435**	**0.2052**
Improv.	2.36%	1.14%	3.79%	3.96%	3.92%	5.03%	3.76%	4.47%	5.42%	5.50%	4.62%	5.44%
BERT4Rec	0.4374	0.3268	0.5238	0.3529	0.4746	0.3538	0.5439	0.3684	0.3200	0.2399	0.3743	0.2513
w. TAPT	**0.4416**	**0.3313**	**0.5312**	**0.3671**	**0.4820**	**0.3612**	**0.5568**	**0.3843**	**0.3276**	**0.2462**	**0.3857**	**0.2598**
Improv.	0.96%	1.37%	1.41%	4.02%	1.55%	2.09%	2.37%	4.31%	2.37%	2.62%	3.04%	3.38%
TiSASRec	0.4422	0.3095	0.5608	0.3452	0.5078	0.3649	0.5956	0.3899	0.2143	0.1510	0.2954	0.1813
w. TAPT	**0.4514**	**0.3119**	**0.5770**	**0.3562**	**0.5240**	**0.3791**	**0.6128**	**0.4035**	**0.2233**	**0.1576**	**0.3051**	**0.1883**
Improv.	2.08%	0.77%	2.88%	3.18%	3.79%	3.89%	2.88%	3.48%	4.19%	4.37%	3.28%	3.86%
FMLP4Rec	0.4278	0.3001	0.5203	0.3304	0.4220	0.3004	0.4995	0.3259	0.3347	0.2426	0.3988	0.2634
w. TAPT	**0.4396**	**0.3068**	**0.5255**	**0.3332**	**0.4303**	**0.3044**	**0.5014**	**0.3265**	**0.3376**	**0.2481**	**0.4083**	**0.2711**
Improv.	2.75%	2.23%	0.99%	0.84%	1.96%	1.33%	0.38%	0.18%	0.86%	2.26%	2.38%	2.92%
GeoSAN	0.4736	0.3328	0.5874	0.3711	0.5129	0.3736	0.5960	0.3844	0.3117	0.2432	0.4011	0.2724
w. TAPT	**0.4882**	**0.3419**	**0.6046**	**0.3816**	**0.5216**	**0.3772**	**0.6152**	**0.3937**	**0.3232**	**0.2486**	**0.4154**	**0.2830**
Improv.	3.10%	2.74%	2.93%	2.84%	1.71%	0.99%	3.23%	2.42%	3.71%	2.26%	3.58%	3.92%
STAN	0.4921	0.3575	0.6023	0.3826	0.5243	0.3891	0.6097	0.4178	0.3034	0.2635	0.4112	0.2914
w. TAPT	**0.5026**	**0.3594**	**0.6206**	**0.3914**	**0.5342**	**0.3933**	**0.6270**	**0.4324**	**0.3178**	**0.2799**	**0.4255**	**0.3062**
Improv.	2.15%	0.55%	3.05%	2.32%	1.90%	1.10%	2.85%	3.50%	4.75%	6.25%	3.50%	5.10%

Table 2: Performance comparison of sequential POI recommendation methods with the proposed TAPT method across three datasets. Results in bold indicate performance improvement, and "w. TAPT" represents TAPT using the respective baseline as the foundational framework. H@5 and H@10 denote HR@5 and HR@10, respectively, while N@5 and N@10 denote NDCG@5 and NDCG@10.

5.3 Evaluation Metrics

We evaluate the sequential POI recommendation task using two widely adopted Top-K metrics, Hit Ratio (HR@K) and NDCG@K, with $K \in \{5, 10\}$ [Wang *et al.*, 2020; Song *et al.*, 2021; Sun *et al.*, 2019]. Higher values indicate better performance. All POIs are considered as candidates during evaluation for stable and convincing results [Krichene and Rendle, 2020; Qin *et al.*, 2024].

For the timestamp prediction task, we select Mean Absolute Error (MAE) as the evaluation metric to intuitively measure the difference between the predicted timestamps and the actual values. A lower MAE indicates better predictive performance [Dong *et al.*, 2023].

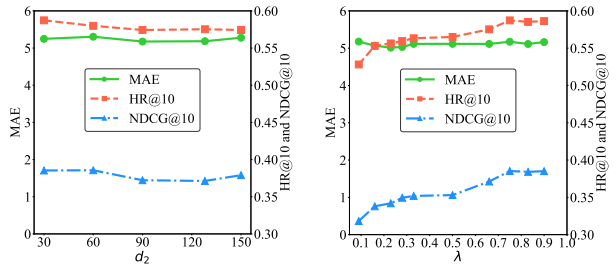
5.4 Experiment Settings

We implement TAPT and all baseline models in Pytorch. The POI and timestamp hidden dimensions are set to 128 and 30, respectively. The maximum sequence lengths for TKY, NYC, and Gowalla are 200, 150, and 100. We use the Adam optimizer with a learning rate of 0.001, a dropout rate of 0.2, and a batch size of 128. All baseline implementations are either provided by the original authors or based on the original papers. We set and tune the hyperparameters according to the instructions in the original papers [Du *et al.*, 2023]. TAPT and the baselines share the same backbone parameters. During training, we set the number of negative samples to 1, following the previous settings [Huang *et al.*, 2023; Wang *et al.*, 2020; Wang *et al.*, 2019a; Wang *et al.*, 2021].

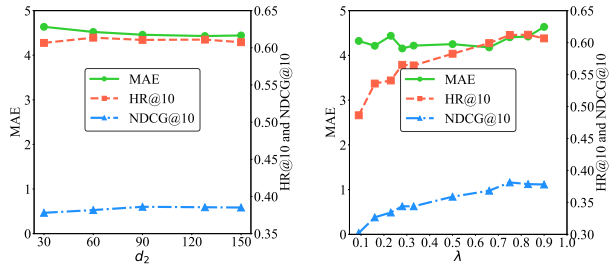
5.5 Recommendation Performance Comparison

We compare the recommendation performance of each baseline with TAPT, as shown in Table 2. The results demonstrate that TAPT achieves comparable or superior performance, which we attribute to the joint embeddings of POIs and timestamps, enabling richer contextual representations and capturing fine-grained POI preferences tied to timestamps.

Specifically, our TAPT model demonstrates notable improvements on the Gowalla dataset over the TKY and NYC datasets, attributed to the strong association between POIs and timestamps in Gowalla, facilitating effective learning of personalized POI preferences. Additionally, our TAPT achieves notable performance improvements across various sequential backbones, demonstrating the effectiveness of combining our timestamp encoding method with existing sequence recommendation models. This validates the advantages of TAPT in leveraging timestamp information to en-



(a) NYC. (b) NYC.



(c) TKY. (d) TKY.

Figure 5: Impact of parameter d_2 and parameter λ on dual-task performance for the NYC and TKY datasets, where we choose SASRec as the backbone to implement TAPT.

hance recommendation accuracy and shows the robustness of our approach. However, TAPT achieves a modest improvement over the next POI recommendation methods, which already incorporate rich spatial-temporal information.

5.6 Prediction Performance

Currently, no methods predict the timestamp information corresponding to recommended POIs. Therefore, we present our prediction results in Table 3 without comparisons, where each time unit represents 1 hour. While TAPT successfully predicts the timestamp for the user's next action, there remains a considerable gap in accuracy. In real LBSN datasets, user behavior exhibits significant randomness in the timestamp dimension, making it challenging to model.

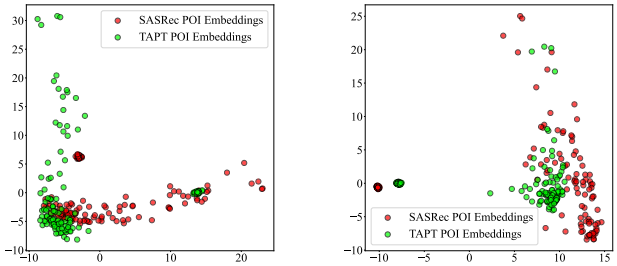
Specifically, our TAPT achieves a low prediction gap on the TKY dataset relative to the remaining datasets. We believe this dataset features regular and consistent check-in data in the timestamp dimension, enabling the model to capture timestamp variation patterns.

5.7 Parameter Sensitivity Analysis

We examine the effect of the timestamp embedding dimension d_2 and the weight coefficient λ on the performance of the dual-task model. As shown in Figures 5, the performance of POI recommendation and timestamp prediction remains stable across variations in the timestamp embedding dimension d_2, demonstrating the robustness of our model. The parameter λ is crucial for balancing the loss functions of the two tasks. When λ is small, POI recommendation performance suffers due to insufficient optimization. However, when λ is set to 0.75, TAPT achieves optimal and stable performance, while timestamp prediction remains stable across λ values.

Backbone	TKY	NYC	Gowalla
GRU4Rec	4.5033	5.1897	5.7957
Caser	4.5893	5.2072	5.7481
SASRec	4.6879	5.2472	5.7364
BERT4Rec	4.4372	5.0677	5.6935
TiSASRec	4.4536	5.1382	5.7245
FMLP4Rec	4.9103	5.3494	5.8527
GeoSAN	4.6274	5.2981	5.7556
STAN	4.5212	5.2138	5.7387

Table 3: The timestamp prediction results of TAPT with various baselines across the three datasets are evaluated using the MAE metric, with computation at the time-unit level.



(a) NYC. (b) TKY.

Figure 6: Visualization of the POI representations from model outputs on two datasets.

5.8 Visualization

To assess our method's effectiveness, we visualize the POI representations in Figure 6. The red points are scattered, indicating that the original model struggles to cluster similar POIs. In contrast, the green points show strong clustering, particularly in key areas, suggesting that our method effectively captures POI similarities for accurate recommendations. Additionally, TAPT exhibits low variance along the first principal component (horizontal axis), demonstrating effective separation of POIs in this dimension. Our method successfully learns user behavior patterns across timestamps to generate time-aware recommendations.

6 Conclusion

This paper proposes TAPT, a multi-task learning framework for interpretable POI recommendation. Specifically, we decouple timestamps into multi-dimensional vectors and design a timestamp encoding module to encode these vectors explicitly. Additionally, we design a specialized timestamp prediction module that builds on the sequence-based POI recommender backbone to address both the POI recommendation and timestamp prediction tasks. Extensive experiments conducted on three public LBSN datasets consistently demonstrate that TAPT achieves comparable or superior performance in POI recommendation compared to the baseline backbone.

7 Acknowledgement

This work is supported by the Natural Science Foundation of China No. 62472196, Jilin Science and Technology Research Project 20230101067JC, National Key R&D Program of China under Grant No. 2021ZD0112501 and 2021ZD0112502, National Natural Science Foundation of China under Grant No. 62272193, National Key R&D Program of China under Grant Nos. 2022YFB3103700 and 2022YFB3103702.

References

[Chang *et al.*, 2023] Jianxin Chang, Chenbin Zhang, Zhiyi Fu, Xiaoxue Zang, Lin Guan, Jing Lu, Yiqun Hui, Dewei Leng, Yanan Niu, Yang Song, and Kun Gai. TWIN: two-stage interest network for lifelong user behavior modeling in CTR prediction at kuaishou. In *KDD*, pages 3785–3794. ACM, 2023.

[Cho *et al.*, 2011] Eunjoon Cho, Seth A. Myers, and Jure Leskovec. Friendship and mobility: user movement in location-based social networks. In *KDD*, pages 1082–1090. ACM, 2011.

[Devlin *et al.*, 2019] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT (1)*, pages 4171–4186. Association for Computational Linguistics, 2019.

[Dong *et al.*, 2023] Jiaxiang Dong, Haixu Wu, Haoran Zhang, Li Zhang, Jianmin Wang, and Mingsheng Long. Simmtm: A simple pre-training framework for masked time-series modeling. In *NeurIPS*, 2023.

[Du *et al.*, 2023] Hanwen Du, Huanhuan Yuan, Pengpeng Zhao, Fuzhen Zhuang, Guanfeng Liu, Lei Zhao, Yanchi Liu, and Victor S. Sheng. Ensemble modeling with contrastive knowledge distillation for sequential recommendation. In *SIGIR*, pages 58–67. ACM, 2023.

[He and McAuley, 2016] Ruining He and Julian J. McAuley. Fusing similarity models with markov chains for sparse sequential recommendation. In *ICDM*, pages 191–200. IEEE Computer Society, 2016.

[Hidasi *et al.*, 2016a] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. In *ICLR (Poster)*, 2016.

[Hidasi *et al.*, 2016b] Balázs Hidasi, Massimo Quadrana, Alexandros Karatzoglou, and Domonkos Tikk. Parallel recurrent neural network architectures for feature-rich session-based recommendations. In *RecSys*, pages 241–248. ACM, 2016.

[Huang *et al.*, 2023] Chengkai Huang, Shoujin Wang, Xianzhi Wang, and Lina Yao. Dual contrastive transformer for hierarchical preference modeling in sequential recommendation. In *SIGIR*, pages 99–109. ACM, 2023.

[Jiang *et al.*, 2024] Yiheng Jiang, Yuanbo Xu, Yongjian Yang, Funing Yang, Pengyang Wang, Chaozhuo Li, Fuzhen Zhuang, and Hui Xiong. Trimlp: A foundational mlp-like architecture for sequential recommendation. *ACM Trans. Inf. Syst.*, 42(6):157:1–157:34, 2024.

[Kang and McAuley, 2018] Wang-Cheng Kang and Julian J. McAuley. Self-attentive sequential recommendation. In *ICDM*, pages 197–206. IEEE Computer Society, 2018.

[Krichene and Rendle, 2020] Walid Krichene and Steffen Rendle. On sampled metrics for item recommendation. In *KDD*, pages 1748–1757. ACM, 2020.

[Li *et al.*, 2020] Jiacheng Li, Yujie Wang, and Julian J. McAuley. Time interval aware self-attention for sequential recommendation. In *WSDM*, pages 322–330. ACM, 2020.

[Li *et al.*, 2022] Yinfeng Li, Chen Gao, Xiaoyi Du, Huazhou Wei, Hengliang Luo, Depeng Jin, and Yong Li. Automatically discovering user consumption intents in meituan. In *KDD*, pages 3259–3269. ACM, 2022.

[Lian *et al.*, 2020] Defu Lian, Yongji Wu, Yong Ge, Xing Xie, and Enhong Chen. Geography-aware sequential location recommendation. In *KDD*, pages 2009–2019. ACM, 2020.

[Liu *et al.*, 2016] Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. Predicting the next location: A recurrent model with spatial and temporal contexts. In *AAAI*, pages 194–200. AAAI Press, 2016.

[Luo *et al.*, 2021] Yingtao Luo, Qiang Liu, and Zhaocheng Liu. STAN: spatio-temporal attention network for next location recommendation. In *WWW*, pages 2177–2185. ACM / IW3C2, 2021.

[Manotumruksa *et al.*, 2018] Jarana Manotumruksa, Craig Macdonald, and Iadh Ounis. A contextual attention recurrent architecture for context-aware venue recommendation. In *SIGIR*, pages 555–564. ACM, 2018.

[Qin *et al.*, 2024] Yifang Qin, Hongjun Wu, Wei Ju, Xiao Luo, and Ming Zhang. A diffusion model for POI recommendation. *ACM Trans. Inf. Syst.*, 42(2):54:1–54:27, 2024.

[Rendle *et al.*, 2010] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. Factorizing personalized markov chains for next-basket recommendation. In *WWW*, pages 811–820. ACM, 2010.

[Song *et al.*, 2021] Wenzhuo Song, Shoujin Wang, Yan Wang, and Shengsheng Wang. Next-item recommendations in short sessions. In *RecSys*, pages 282–291. ACM, 2021.

[Sun *et al.*, 2019] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In *CIKM*, pages 1441–1450. ACM, 2019.

[Tang and Wang, 2018] Jiaxi Tang and Ke Wang. Personalized top-n sequential recommendation via convolutional sequence embedding. In *WSDM*, pages 565–573. ACM, 2018.

[Twardowski, 2016] Bartlomiej Twardowski. Modelling contextual information in session-aware recommender systems with neural networks. In *RecSys*, pages 273–276. ACM, 2016.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, pages 5998–6008, 2017.

[Wang *et al.*, 2019a] Chenyang Wang, Min Zhang, Weizhi Ma, Yiqun Liu, and Shaoping Ma. Modeling item-specific temporal dynamics of repeat consumption for recommender systems. In *WWW*, pages 1977–1987. ACM, 2019.

[Wang *et al.*, 2019b] Shoujin Wang, Liang Hu, Yan Wang, Longbing Cao, Quan Z. Sheng, and Mehmet A. Orgun. Sequential recommender systems: Challenges, progress and prospects. In *IJCAI*, pages 6332–6338. ijcai.org, 2019.

[Wang *et al.*, 2020] Chenyang Wang, Min Zhang, Weizhi Ma, Yiqun Liu, and Shaoping Ma. Make it a chorus: Knowledge- and time-aware item modeling for sequential recommendation. In *SIGIR*, pages 109–118. ACM, 2020.

[Wang *et al.*, 2021] Chenyang Wang, Weizhi Ma, Min Zhang, Chong Chen, Yiqun Liu, and Shaoping Ma. Toward dynamic user intention: Temporal evolutionary effects of item relations in sequential recommendation. *ACM Trans. Inf. Syst.*, 39(2):16:1–16:33, 2021.

[Wang *et al.*, 2022] En Wang, Yiheng Jiang, Yuanbo Xu, Liang Wang, and Yongjian Yang. Spatial-temporal interval aware sequential POI recommendation. In *ICDE*, pages 2086–2098. IEEE, 2022.

[Xu *et al.*, 2024] Hangtong Xu, Yuanbo Xu, and Yongjian Yang. Separating and learning latent confounders to enhancing user preferences modeling. In *International Conference on Database Systems for Advanced Applications*, pages 67–82. Springer, 2024.

[Xu *et al.*, 2025] Hangtong Xu, Yuanbo Xu, Chaozhuo Li, and Fuzhen Zhuang. Causal structure representation learning of unobserved confounders in latent space for recommendation. *ACM Trans. Inf. Syst.*, April 2025. Just Accepted.

[Yang *et al.*, 2015] Dingqi Yang, Daqing Zhang, Vincent W. Zheng, and Zhiyong Yu. Modeling user activity preference by leveraging user spatial temporal characteristics in lbsns. *IEEE Trans. Syst. Man Cybern. Syst.*, 45(1):129–142, 2015.

[Yang *et al.*, 2020] Dingqi Yang, Benjamin Fankhauser, Paolo Rosso, and Philippe Cudré-Mauroux. Location prediction over sparse user mobility traces using rnns: Flashback in hidden states! In *IJCAI*, pages 2184–2190. ijcai.org, 2020.

[Yang *et al.*, 2022] Song Yang, Jiamou Liu, and Kaiqi Zhao. Getnext: Trajectory flow map enhanced transformer for next POI recommendation. In *SIGIR*, pages 1144–1153. ACM, 2022.

[Yuan *et al.*, 2019] Fajie Yuan, Alexandros Karatzoglou, Ioannis Arapakis, Joemon M. Jose, and Xiangnan He. A simple convolutional generative network for next item recommendation. In *WSDM*, pages 582–590. ACM, 2019.

[Zhang *et al.*, 2023] Yuting Zhang, Yiqing Wu, Ran Le, Yongchun Zhu, Fuzhen Zhuang, Ruidong Han, Xiang Li, Wei Lin, Zhulin An, and Yongjun Xu. Modeling dual period-varying preferences for takeaway recommendation. In *KDD*, pages 5628–5638. ACM, 2023.

[Zhou *et al.*, 2022] Kun Zhou, Hui Yu, Wayne Xin Zhao, and Ji-Rong Wen. Filter-enhanced MLP is all you need for sequential recommendation. In *WWW*, pages 2388–2399. ACM, 2022.