Is the Last Check-In All You Need? Next POI Recommendation: Recall and Rerank

Zhengjia Xu
zhengjia.xu@hdr.mq.edu.au
School of Computing
Macquarie University
Sydney, NSW, Australia

Dingyang Lyu
dingyang.lyu@student.unimelb.edu.au
School of Computing and Information Systems
University of Melbourne
Melbourne, VIC, Australia

Zitai Qiu
zitai.qiu@hdr.mq.edu.au
School of Computing
Macquarie University
Sydney, NSW, Australia

Shan Xue
emma.xue@mq.edu.au
School of Computing
Macquarie University
Sydney, NSW, Australia

Jian Yang
jian.yang@mq.edu.au
School of Computing
Macquarie University
Sydney, NSW, Australia

Jia Wu
jia.wu@mq.edu.au
School of Computing
Macquarie University
Sydney, NSW, Australia

Abstract

Next Point-of-Interest (POI) recommendation aims to predict the next POI a user will visit based on the historical check-ins, but widely used datasets suffer from severe sparsity, making long-range transition modeling difficult to learn reliably. In this work, we revisit next-POI prediction from a minimalist perspective and point out an extreme short-horizon dominance in these datasets: the most predictive signal often collapses to the last check-in, while higher-order transition patterns quickly become too sparse to exploit. Based on this, we propose **K1-POI**, a concise recall-and-rerank framework that formulates sparse next-POI prediction as candidate generation and calibration. In the recall stage, we design a window-input self-attention backbone that uses only the most recent check-in ($k = 1$) to retrieve a high-quality top-K candidate set; we further introduce a lightweight contrastive objective to align the backbone representation with the embedding of the ground-truth next POI within the candidate set. In the rerank stage, we propose an efficient Spatio-Temporal Prior Reranker (ST-Reranker) that calibrates the ordering within the top-K candidates using additive, model-agnostic spatio-temporal priors computed from the training split. Experiments on three public datasets show that K1-POI achieves strong performance across all datasets. Beyond performance, we propose a set of simple statistics-based "guess" baselines and conduct extensive analyses, showing that benchmark performance is largely explained by user preference, one-step transition routines, and spatio-temporal habits. These findings suggest that mining long-range check-in transition patterns may be a particularly challenging direction under current sparse benchmark settings.

CCS Concepts

• **Information systems** → *Data mining*; • **Networks** → **Location based services**; • **Human-centered computing** → Ubiquitous and mobile computing design and evaluation methods.

KDD '26, Jeju Island, Republic of Korea

ACM ISBN 979-8-4007-2259-2/2026/08
https://doi.org/10.1145/3770855.3817836

Keywords

Point of Interest, Next POI Recommendation, Next POI Prediction

ACM Reference Format:
Zhengjia Xu, Dingyang Lyu, Zitai Qiu, Shan Xue, Jian Yang, and Jia Wu. 2026. Is the Last Check-In All You Need? Next POI Recommendation: Recall and Rerank. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2 (KDD '26), August 09–13, 2026, Jeju Island, Republic of Korea.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3770855.3817836

Resource Availability:
The source code of this paper has been archived at https://doi.org/10.5281/zenodo.20365763. The GitHub repository is available at https://github.com/XZJIsme/k1-poi.

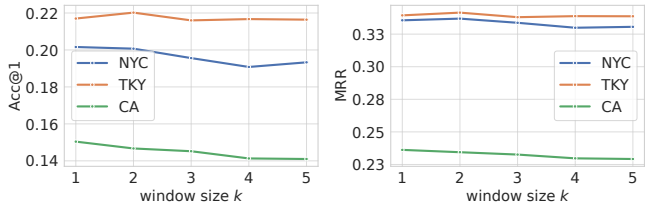

Figure 1: Results on three commonly used next-POI prediction datasets as the amount of historical information increases. As the window size increases, the overall performance shows a decreasing trend.

1 Introduction

Next Point-of-Interest (POI) recommendation aims to predict the next place a user will visit based on their historical check-ins, supporting a wide range of location-based services such as POI recommendation, route planning, and urban mobility analytics [19, 20, 24]. Despite rapid progress, a persistent challenge in widely used Location-based Social Networking (LBSN) benchmarks is severe data sparsity: trajectories exhibit low continuity, POIs follow a long-tailed distribution, and user-specific transition patterns are observed only a few times. In recent years, many methods

have adopted sequential models (e.g., RNN [19], LSTM [6], and Transformer [21]) and injected additional relational structures (e.g., multi-modality graphs [7, 16], transition graphs [21], or hypergraphs [8, 10]), or by explicitly querying historical trajectories [3, 19]. However, on sparse LBSN benchmarks, the prevailing training and inference paradigm remains largely *single-stage*: the model outputs scores over the entire POI set, and performance improvements often rely on increasingly sophisticated representations of the recent context. This raises a natural question: *what effective signal do these benchmarks actually provide, and how should we best exploit it?*

As shown in Fig. 1, a small motivating experiment is conducted by feeding only simple user embeddings and POI embeddings, with positional embedding, into an attention model. On commonly used next-POI benchmarks, increasing the window size not only fails to improve performance, but even leads to a slight degradation. Therefore, we point out a key yet under-explored property of standard next-POI benchmarks, namely a strong *short-horizon dominance*. Concretely, when trajectories are extremely sparse, the most predictive evidence often collapses to the *last check-in*, while higher-order transition patterns (window size $k > 1$) quickly become too sparse. Although prior work has recognized the strong reliance on short-term signals [1, 4, 19, 21, 23], most existing methods attempt to seek additional help from richer historical trajectories and complex global transition graphs. **However, they rarely consider that historical check-ins themselves may act as noise: while making such noisy history useful is one direction, simply discarding history may also be a viable alternative**.

This observation motivates a different view of sparse next-POI prediction: rather than treating it as a one-shot classification problem over the full POI space and addressing sparsity by adding more historical information and graph relations, we take a subtractive rather than additive approach, and view it as a *candidate generation and calibration* problem that mainly relies on the last check-in and its immediate context (time and location). Based on this view, we propose a concise recall-and-rerank framework, named **K1-POI**. In the **recall** stage, a window-input self-attention backbone takes only the most recent check-in (i.e., window size k=1) together with a user token, and produces base logits over all POIs to retrieve a high-quality top-K candidate set. The check-in embedding is constructed by combining POI embeddings with multi-scale Time-of-Day (ToD) embeddings and multi-scale geographic cell embeddings, allowing the backbone to capture fine-to-coarse spatio-temporal signals with minimal sequential context. To further improve the representation, we add a lightweight contrastive objective that aligns the backbone representation with the embedding of the ground-truth next POI within the candidate set. In the **rerank** stage, we design a Spatio-Temporal Prior Reranker (ST-Reranker) that calibrates the ordering *within* the top-K candidate set using spatio-temporal priors computed solely from the training split. Specifically, the reranker adds: (i) a POI-specific time prior, (ii) a user-and-time-bin activity-center affinity, and (iii) a last-step distance penalty. These priors are additive, model-agnostic, and inexpensive: after caching candidate scores on the validation split, reranker hyperparameters can be selected via a validation-only search protocol. Besides, we reuse the similarity in the contrastive space as an additional reranking feature.

We evaluate our framework on three public benchmarks, and compared with strong baselines, it achieves consistently stable and strong performance across all datasets. Ablation studies further show that the backbone is able to recall the ground-truth POI into a short candidate list, while the ST-Reranker significantly improves early ranks by calibrating candidates with spatio-temporal priors. Beyond performance gains, we conduct extensive statistical analyses and propose a set of simple statistics-based "guess" baselines. Together with the results of our K1-POI, these analyses provide a deeper characterization of the intrinsic sparsity of higher-order transition patterns in these datasets, offering practical insights into the causes of extreme short-horizon dominance in the benchmarks. Our findings suggest that mining long-range check-in transition patterns is not only difficult, but may even be largely infeasible, and we further provide suggestions for future research. Our contributions are summarized as follows:

- We introduce a recall-and-rerank method K1-POI for sparse next-POI recommendation and show that it is a strong alternative to the prevailing sophisticated single-stage scoring setup.
- We develop a concise yet effective self-attention recall backbone with a dedicated user token and multi-scale spatio-temporal embeddings, together with a candidate-set contrastive alignment objective to enhance representation quality, achieving reliable and high-quality recall with window size $k = 1$.
- We propose a lightweight ST-Reranker that exploits training-split spatio-temporal priors to calibrate top-K candidates and substantially improves candidate ranking quality.
- We propose simple statistics-based "guess" baselines built from the training split, and through their results, provide detailed dataset-driven analyses that explain the extreme short-horizon dominance in standard next-POI benchmarks. Our findings reveal intrinsic benchmark properties and offer interesting insights, suggesting that mining long-range check-in transition patterns may not only be hard, but potentially almost absent in practice, and we provide suggestions for future work.

2 Current Attempts

In recent years, a number of works have been proposed for next POI prediction or recommendation. Neural sequential models encode check-in sequences to learn transition patterns. DeepMove [5] and PLSPL [15] leverage attention mechanisms to capture mobility patterns and historical preferences. Methods such as GETNext [21], AGRAN [14], STHGCN [18], SNPM [22], and DePOI [25] further enrich relational information by modeling spatio-temporal POI relations with graphs. In addition, MCLP [12], Flashback [19], and REPLAY [3] attempt to retrieve relevant historical information to enhance accuracy. These methods have made significant contributions to the field and, to varying degrees, have recognized the importance of short-term signals. They typically enrich the prediction context by incorporating graph modeling and leveraging more historical information. However, they rarely consider the opposite and more radical direction—discarding most of the context—to explicitly examine whether learning trajectory transition patterns is

truly feasible under such sparsity. More detailed related work can be found in Appendix.

3 Methodology

As shown in Fig. 2, our K1-POI follows a **recall-and-rerank** paradigm. A window-input self-attention backbone produces base logits over all POIs, and a lightweight ST-Reranker rescales and reorders the top-K candidates. In addition, we design a simple contrastive objective that aligns the self-attention output representation with the next POI embedding.

3.1 Preliminaries

Let $\mathcal{U} = \{u_1, \ldots, u_{|\mathcal{U}|}\}$ be the user set and $\mathcal{P} = \{p_1, \ldots, p_{|\mathcal{P}|}\}$ be the POI set. Each POI $p \in \mathcal{P}$ has a category attribute $c_p \in \mathcal{C}^{cat}$ and a geographic coordinate $\mathrm{loc}(p) = (lat_p, lon_p)$. A check-in is a triple (u, p, t), meaning user $u \in \mathcal{U}$ visits POI $p \in \mathcal{P}$ at local timestamp t. For each user u, we sort check-ins chronologically to form a sequence $\mathbf{S}_u = \langle s_{u,1}, \ldots, s_{u,L_u} \rangle$, where $s_{u,i} = (p_{u,i}, t_{u,i})$ and $t_{u,1} < t_{u,2} < \cdots < t_{u,L_u}$.

Next-POI prediction. The goal of next-POI prediction is to predict the POI of the next check-in $p_{u,n}$, given the historical prefix $\mathbf{S}_{u,<n} = \langle s_{u,1}, \ldots, s_{u,n-1} \rangle$, where $n > 1$.

3.2 Window-Input Self-Attention Backbone

3.2.1 Token Construction and Embedding Composition. We use a window-input setting with input window size k. For each prediction target $p_{u,n}$ (with $n > 1$), we take the most recent k historical check-ins. We present the ideal case with at least k past check-ins; when the available history is shorter than k, we simply use all past check-ins. Let the POI window be

$$\mathbf{p}_{u,n}^{(k)} = \left[p_{u,n-k}, \ldots, p_{u,n-1}\right] \in \mathcal{P}^k. \tag{1}$$

Aligned windows are similarly formed for categories and other per-check-in fields.

The backbone input sequence is:

$$\mathbf{x}_{u,n} = \left[\mathbf{x}_u^{usr} \;;\; \mathbf{x}_{u,n-k}^{chk}, \ldots, \mathbf{x}_{u,n-1}^{chk}\right] \in \mathbb{R}^{(1+k)\times d}, \tag{2}$$

where $\mathbf{x}_u^{usr}$ is the user token and $\mathbf{x}_{u,i}^{chk}$ is the check-in token, with $i \in \{n-k, \ldots, n-1\}$. We define $\mathbf{x}_u^{usr}$ and $\mathbf{x}_{u,i}^{chk}$ as follows:

User token. We embed the user ID and project it:

$$\mathbf{x}_u^{usr} = \mathbf{W}_{usr}\, \mathbf{e}^{usr}(u) \in \mathbb{R}^d. \tag{3}$$

Here $\mathbf{e}^{usr}(\cdot)$ is a learnable embedding, and $\mathbf{e}^{usr}(u) \in \mathbb{R}^{d_{usr}}$. The projection matrix $\mathbf{W}_{usr} \in \mathbb{R}^{d\times d_{usr}}$ maps it to the model width d.

For each check-in token, besides the simple embeddings of POI ID and category ID, we introduce multi-scale embeddings to better leverage temporal and spatial signals, as described below.

Multi-scale ToD slots. We discretize each check-in time into multiple granularities. Let $t_{u,i}$ denote the local timestamp in epoch seconds of check-in $s_{u,i}$. We define $sec_{u,i} = t_{u,i} \bmod 86400$ as the number of seconds since local midnight. We use a set of ToD granularities $\mathcal{S}$, where each $s \in \mathcal{S}$ specifies the number of slots per day. The ToD slot ID of check-in $s_{u,i}$ under scale s is

$$slot_{u,i}^{(s)} = \left\lfloor \frac{sec_{u,i} \cdot s}{86400} \right\rfloor + 1 \in \{1, \ldots, s\}. \tag{4}$$

Multi-scale geo-cells. We discretize the coordinate of each check-in $s_{u,i}$ at POI $p_{u,i}$ into multi-scale grid buckets. We use a set of cell sizes $\mathcal{M}$, where each $m \in \mathcal{M}$ is a cell size in meters. Latitude and longitude are given in degrees and are converted to radians:

$$\phi_{u,i} = lat_{p_{u,i}} \cdot \frac{\pi}{180},\ \lambda_{u,i} = lon_{p_{u,i}} \cdot \frac{\pi}{180}. \tag{5}$$

From the training split, reference values ϕ_0 and λ_0 are computed as the mean latitude and mean longitude in radians. We set $c_0 = \cos\phi_0$ and apply a local equirectangular projection to obtain planar coordinates:

$$x_{u,i} = R\, c_0 (\lambda_{u,i} - \lambda_0),\ y_{u,i} = R(\phi_{u,i} - \phi_0), \tag{6}$$

where $R = 6{,}371{,}000$ is the Earth radius in meters. For each cell size $m \in \mathcal{M}$, we bucketize (x, y) into grid indices

$$g_{x,u,i}^{(m)} = \left\lfloor \frac{x_{u,i} - x_{\min}}{m} \right\rfloor + 1,\ g_{y,u,i}^{(m)} = \left\lfloor \frac{y_{u,i} - y_{\min}}{m} \right\rfloor + 1. \tag{7}$$

where $x_{\min}$ and $y_{\min}$ are the minimum projected coordinates on the training split.

Check-in token. For a historical check-in $s_{u,i}$, we construct multi-scale ToD and geo-cell embeddings:

$$\begin{aligned} \mathbf{e}_{u,i}^{tod} &= \left[\mathbf{e}_s^{tod}\big(slot_{u,i}^{(s)}\big)\right]_{s\in\mathcal{S}} \in \mathbb{R}^{|\mathcal{S}|\cdot d_{tod}}, \\ \mathbf{e}_{u,i}^{geo} &= \left[\mathbf{e}_m^{x}\big(g_{x,u,i}^{(m)}\big) + \mathbf{e}_m^{y}\big(g_{y,u,i}^{(m)}\big)\right]_{m\in\mathcal{M}} \in \mathbb{R}^{|\mathcal{M}|\cdot d_{geo}}. \end{aligned} \tag{8}$$

where $[\cdot]_{s\in\mathcal{S}}$ and $[\cdot]_{m\in\mathcal{M}}$ denote concatenation over scales using a fixed ordering of $\mathcal{S}$ and $\mathcal{M}$. All embedding components are concatenated into a check-in feature vector:

$$\mathbf{f}_{u,i} = \left[\mathbf{e}^{poi}(p_{u,i}) \,\|\, \mathbf{e}^{cat}(c_{p_{u,i}}) \,\|\, \mathbf{e}_{u,i}^{tod} \,\|\, \mathbf{e}_{u,i}^{geo}\right] \in \mathbb{R}^{d_{chk}}. \tag{9}$$

Finally, it is projected to the model width:

$$\mathbf{x}_{u,i}^{chk} = \mathbf{W}_{chk}\, \mathbf{f}_{u,i} \in \mathbb{R}^d, \tag{10}$$

where $\mathbf{W}_{chk} \in \mathbb{R}^{d\times d_{chk}}$.

In our implementation, we use standard embedding lookups: $\mathbf{e}^{poi}(p) \in \mathbb{R}^{d_{poi}}$, $\mathbf{e}^{cat}(c) \in \mathbb{R}^{d_{cat}}$, $\mathbf{e}_s^{tod}(\cdot) \in \mathbb{R}^{d_{tod}}$, and $\mathbf{e}_m^{x}(\cdot), \mathbf{e}_m^{y}(\cdot) \in \mathbb{R}^{d_{geo}}$.

Positional embedding. We use learnable positional embeddings $\mathbf{p}_j \in \mathbb{R}^d$:

$$\mathbf{x}_{u,n}[j] \leftarrow \mathbf{x}_{u,n}[j] + \mathbf{p}_j, j = 0, \ldots, k. \tag{11}$$

Note that the positional index j differs from the check-in index i. Our backbone input is a token sequence consisting of a user token followed by k check-in tokens, so positional embeddings are defined over token positions $j = 0, \ldots, k$. In some prior work [2, 21], the user embedding is concatenated to every check-in token. We instead place the user embedding as a single dedicated token, which avoids repeating identical user information across the whole window. In our practice, this design leads to better performance.

3.2.2 Multi-Head Self-Attention Layers. L layers of multi-head self-attention are stacked. Let $\mathbf{H}^{(0)} = \mathbf{x}_{u,n}$. For $\ell = 1, \ldots, L$:

$$\begin{aligned} \tilde{\mathbf{H}}^{(\ell-1)} &= \mathrm{LayerNorm}\big(\mathbf{H}^{(\ell-1)}\big), \\ \mathbf{A}^{(\ell)} &= \mathrm{MHA}\big(\tilde{\mathbf{H}}^{(\ell-1)}, \tilde{\mathbf{H}}^{(\ell-1)}, \tilde{\mathbf{H}}^{(\ell-1)}\big), \\ \mathbf{H}^{(\ell)} &= \mathbf{H}^{(\ell-1)} + \mathrm{Dropout}\big(\mathbf{A}^{(\ell)}\big). \end{aligned} \tag{12}$$

where $\mathrm{MHA}(\cdot)$ is standard multi-head self-attention.

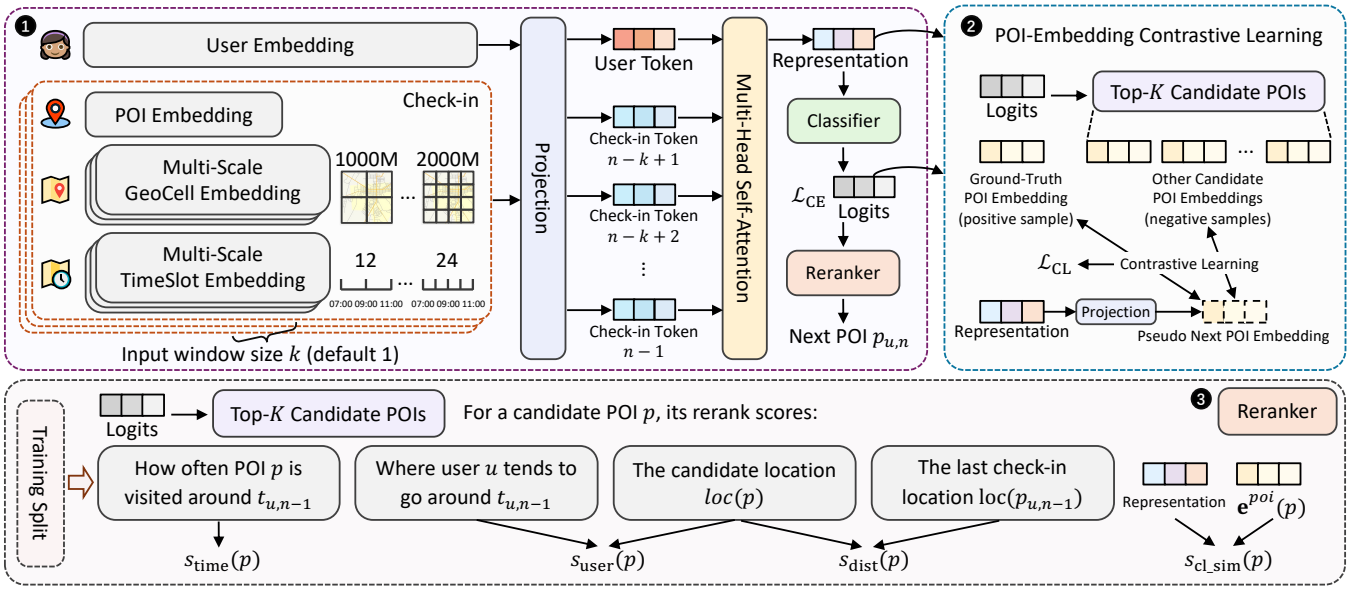

Figure 2: (1) A token sequence is formed by combining a user embedding and check-in embeddings, and is fed into multi-head self-attention to recall a candidate set of POIs. (2) The multi-head self-attention output representation is aligned with the ground-truth next-POI embedding. (3) The candidate POIs are reranked using spatio-temporal priors.

3.2.3 Output Representation and Next-POI Loss. We use the output at the user-token position as the instance representation:

$$\mathbf{h}_{u,n} = \mathbf{H}^{(L)}[0] \in \mathbb{R}^d. \tag{13}$$

Logits are then computed over all POIs:

$$\mathbf{z}_{u,n} = \mathbf{W}_{out}\mathbf{h}_{u,n} + \mathbf{b}_{out} \in \mathbb{R}^{|\mathcal{P}|}, \tag{14}$$

and train with the standard cross-entropy loss

$$\mathcal{L}_{\mathrm{CE}} = -\log \frac{\exp\left(\mathbf{z}_{u,n}[p_{u,n}]\right)}{\sum_{p' \in \mathcal{P}} \exp\left(\mathbf{z}_{u,n}[p']\right)}. \tag{15}$$

3.2.4 POI-Embedding Contrastive Learning. To further shape representations, we add a POI-embedding contrastive objective on a candidate set using an InfoNCE-style loss. For each instance, a candidate set $C_{u,n} = \{q_1, \ldots, q_K\}$ is formed by selecting the top-K POIs according to the backbone logits $\mathbf{z}_{u,n}$. Here K is a hyperparameter controlling the candidate-set size. If the ground-truth POI is not in $C_{u,n}$, it is inserted by replacing the lowest-ranked candidate to ensure a positive sample for contrastive learning.

The backbone representation and candidate POI embeddings are projected into a shared contrastive space of dimension d_{CL} using linear projections:

$$\mathbf{u} = \mathbf{W}^{\mathrm{CL}}_{usr}\mathbf{h}_{u,n} + \mathbf{b}^{\mathrm{CL}}_{usr} \in \mathbb{R}^{d_{\mathrm{CL}}},\ \mathbf{v}_j = \mathbf{W}^{\mathrm{CL}}_{poi}\mathbf{e}^{poi}(q_j) + \mathbf{b}^{\mathrm{CL}}_{poi} \in \mathbb{R}^{d_{\mathrm{CL}}}, \tag{16}$$

where $\mathbf{u}$ is the next-POI query embedding, and $\mathbf{v}_j$ is the contrastive embedding of candidate POI q_j. The projection parameters satisfy $\mathbf{W}^{\mathrm{CL}}_{usr} \in \mathbb{R}^{d_{\mathrm{CL}} \times d}$, $\mathbf{W}^{\mathrm{CL}}_{poi} \in \mathbb{R}^{d_{\mathrm{CL}} \times d_{poi}}$, and $\mathbf{b}^{\mathrm{CL}}_{usr}, \mathbf{b}^{\mathrm{CL}}_{poi} \in \mathbb{R}^{d_{\mathrm{CL}}}$.

We apply ℓ_2 normalization and use cosine similarity. Let $\hat{\mathbf{u}} = \mathbf{u}/\|\mathbf{u}\|_2$ and $\hat{\mathbf{v}}_j = \mathbf{v}_j/\|\mathbf{v}_j\|_2$. The contrastive logits are

$$\ell_j = \frac{\langle \hat{\mathbf{u}}, \hat{\mathbf{v}}_j \rangle}{\tau}, j = 1, \ldots, K, \tag{17}$$

where $\tau > 0$ is the temperature. Let j^+ be the index of the positive POI in $C_{u,n}$. The candidate-set InfoNCE loss is a cross-entropy over K candidates:

$$\mathcal{L}_{\mathrm{CL}} = -\log \frac{\exp(\ell_{j^+})}{\sum_{j=1}^{K} \exp(\ell_j)}. \tag{18}$$

3.2.5 Overall Training Objective. The final training loss combines next-POI classification and contrastive learning:

$$\mathcal{L} = \mathcal{L}_{\mathrm{CE}} + w_{\mathrm{CL}}\,\mathcal{L}_{\mathrm{CL}}, \tag{19}$$

where $w_{\mathrm{CL}} \geq 0$ is a hyperparameter controlling the strength of the contrastive loss (and setting $w_{\mathrm{CL}} = 0$ disables contrastive learning).

3.3 ST-Reranker

The backbone produces base logits over all POIs, but it may ignore simple yet strong spatio-temporal regularities. We therefore apply an **additive prior reranker** on the top-K candidates, using only statistics computed from the training split.

3.3.1 Spatial-Temporal Prior Statistics from the Training Split. We divide a day into B time bins. For an instance with $tod \in [0, 1)$, its bin is

$$b = \lfloor tod \cdot B \rfloor \in \{0, \ldots, B-1\}. \tag{20}$$

where tod is the ToD ratio within a day. We use $b_{u,j}$ to denote the bin index of check-in $s_{u,j}$.

From the training split, we build spatial-temporal priori:

POI coordinates. Each POI p has a $\mathrm{loc}(p) = (lat_p, lon_p)$.

POI time prior. Smoothed log-probability of a time bin given POI p:

$$s_{\mathrm{time}}(p) = \log P(b \mid p),\ P(b \mid p) = \frac{N(p,b) + \alpha}{\sum_{b'} N(p,b') + \alpha B}. \tag{21}$$

where $N(p, b)$ counts check-ins in the training split that are associated with POI p and assigned to time bin b, and $\alpha > 0$ is the smoothing coefficient.

User-time spatial center. For each user u and bin b, the center $\boldsymbol{\mu}(u, b)$ is the mean coordinate of check-ins assigned to bin b; if the count is too small, we back off to the all-time center $\boldsymbol{\mu}_{all}(u)$:

$$\boldsymbol{\mu}(u,b) = \begin{cases} \text{mean}_{j:b_{u,j}=b}\big[\text{loc}(p_{u,j})\big] & \text{if } \big|\{j : b_{u,j} = b\}\big| \geq c_{\min} \\ \boldsymbol{\mu}_{all}(u) & \text{otherwise.} \end{cases} \tag{22}$$

where $c_{\min}$ is a minimum-count threshold for deciding whether the bin-specific center is used. We also compute a per-bin radius $\sigma_{\text{raw}}(u, b)$ as the mean distance from check-ins to the empirical center in bin b, and use an all-time estimate when the bin is empty.

Global distance scale. By default, we set τ_{dist} to the 75-th percentile of consecutive check-in distances in the training split.

3.3.2 Candidate-Set Reranker Score. Let $C_{u,n} \subseteq \mathcal{P}$ denote the top-K candidate POIs selected by the backbone scores. Specifically, let

$$C_{u,n} = \text{Top-}K\big(\mathbf{z}_{u,n}, K\big), \tag{23}$$

and for each candidate $p \in C_{u,n}$, define its base score as the corresponding logit produced by the self-attention backbone

$$s_{\text{base}}(u, n, p) = \mathbf{z}_{u,n}[p]. \tag{24}$$

Then three sptial-temporal prior terms are computed for each candidate $p \in C_{u,n}$:

Time prior: $s_{\text{time}}(p) = \log P(b \mid p)$.

User-center affinity:

$$s_{\text{user}}(u, p) = -\frac{d\big(\text{loc}(p), \boldsymbol{\mu}(u, b)\big)}{\max(\sigma_{\text{raw}}(u, b), \sigma_{\min})}, \tag{25}$$

where $d(\cdot, \cdot)$ denotes the haversine distance in kilometers and $\sigma_{\min} > 0$ prevents overly sharp penalties.

Last-step distance penalty: let loc_{last} be the location of the most recent historical check-in, and

$$s_{\text{dist}}(p) = -\frac{d\big(\text{loc}(p), \text{loc}_{last}\big)}{\tau_{dist}}. \tag{26}$$

When the backbone is trained with the contrastive objective, the cosine similarity in the contrastive space is reused as an additional term:

$$s_{\text{cl\_sim}}(u, n, p) = \left\langle \frac{\mathbf{u}}{\|\mathbf{u}\|_2}, \frac{\mathbf{v}(p)}{\|\mathbf{v}(p)\|_2} \right\rangle, \tag{27}$$

where $\mathbf{u} = \mathbf{W}^{\text{CL}}_{usr}\mathbf{h}_{u,n} + \mathbf{b}^{\text{CL}}_{usr}$ is the next-POI query embedding and $\mathbf{v}(p) = \mathbf{W}^{\text{CL}}_{poi}\mathbf{e}^{poi}(p) + \mathbf{b}^{\text{CL}}_{poi}$ is the contrastive embedding of POI p.

The final reranker score is computed additively within the candidate set:

$$\begin{aligned} s_{\text{rerank}}(u, n, p) &= \lambda_{\text{time}}\, s_{\text{time}}(p) \\ &\quad + \lambda_{\text{user}}\, s_{\text{user}}(u, p) + \lambda_{\text{dist}}\, s_{\text{dist}}(p) + \lambda_{\text{cl}}\, s_{\text{cl\_sim}}(u, n, p) \\ \text{score}(u, n, p) &= s_{\text{base}}(u, n, p) + s_{\text{rerank}}(u, n, p), \end{aligned} \tag{28}$$

where $\lambda_{\text{cl}} = 0$ if $s_{\text{cl\_sim}}$ is not used. The final ranked candidates for next-POI prediction are obtained by $\text{score}(u, n, p)$.

3.3.3 Intuition of the Reranker Terms. The reranker combines four terms, all designed such that a larger value indicates a POI is more likely to be the next visited location. Intuitively, the time prior s_{time} assigns a higher score to a POI if it is frequently visited around the current ToD in the training split. The user-center affinity s_{user} favors candidate POIs that are closer to where user u typically visits at this ToD, while those farther away are penalized. The last-step distance penalty s_{dist} prefers candidate POIs that are geographically close to the last check-in location, and penalizes long jumps from the last location. Finally, the contrastive similarity term $s_{\text{cl\_sim}}$ favors POIs whose embeddings are more similar to the next-POI query embedding produced by the backbone, reusing the aligned embedding space learned by the contrastive objective.

3.3.4 Val-Only Hyperparameter Search. The reranker hyperparameters ($\lambda_{\text{time}}, \lambda_{\text{user}}, \lambda_{\text{dist}}, \lambda_{\text{cl}}, \sigma_{\min}, \tau_{dist}$), are selected using a validation-only protocol. The backbone is trained once and then kept fixed. To enable efficient search, top-K candidates together with their base scores and prior terms are cached on the validation split so that each hyperparameter trial reduces to a fast vectorized rescoring computation. A two-stage random search is adopted: many trials are first screened on a subset of the validation instances, and the best M trials are then re-evaluated on the full validation split to select the final configuration. The selected configuration is evaluated once on the test split. No statistics are computed from the validation/test splits when constructing the priors. This follows standard practice in deep learning, where model selection and hyperparameter tuning are performed on validation data while the test split is used only for final reporting.

4 Experiments

4.1 Experimental Settings

Following prior works [18, 21, 25], we evaluate on public datasets NYC, TKY, and CA. In our evaluation protocol, each check-in except the first one of a user is treated as a prediction target. The baselines are modified to fit our evaluation protocol. The details of the datasets, evaluation metrics, and the hyperparameter settings are in Appendix.

4.2 RQ1: Overall Performance Comparison

The overall performance comparisons are in Tab. 1. Our recall-and-rerank framework achieves the best results on NYC and CA, and remains competitive on TKY.

In baselines, a common approach is to improve performance by mining richer historical context beyond the raw sequence, such as modeling collaborative transition structures (e.g., adaptive graphs and spatio-temporal hypergraphs), leveraging longer check-in histories ($k > 1$), or injecting explicit temporal cues. Our framework takes a complementary route: instead of constructing a POI graph or querying historical data, we rely only on the last check-in as the historical input (i.e., window size $k = 1$). We then use a self-attention backbone to recall a high-quality candidate set, and a lightweight reranker further refines the ordering using simple spatio-temporal priors computed from the training split. Notably, DPRL is evaluated both with and without future-time input, and our method still performs competitively even against the future-time variant.

Table 1: Performance comparison in Acc@k and MRR on three datasets.

	NYC					TKY					CA				
	Acc@1	Acc@5	Acc@10	Acc@20	MRR	Acc@1	Acc@5	Acc@10	Acc@20	MRR	Acc@1	Acc@5	Acc@10	Acc@20	MRR
DeepMove$_{(WWW'2018)}$	0.1721	0.4342	0.5478	0.6292	0.2899	0.1738	0.4058	0.5060	0.5897	0.2817	0.0938	0.2108	0.2679	0.3294	0.1536
PLSPL$_{(TKDE'2020)}$	0.1978	0.5112	0.6368	0.7282	0.3371	0.2173	0.4757	0.5744	0.6552	0.3353	0.1333	0.3084	0.3812	0.4549	0.2170
GETNext$_{(SIGIR'2022)}$	0.1846	0.4516	0.5716	0.6742	0.3074	0.1408	0.3177	0.3909	0.4539	0.2247	0.1079	0.2451	0.3034	0.3568	0.1731
AGRAN$_{(SIGIR'2023)}$	0.1468	0.4940	0.6262	0.7062	0.2971	0.1379	0.4343	0.5452	0.6319	0.2714	0.1018	0.3233	0.4139	0.4969	0.2069
STHGCN$_{(SIGIR'2023)}$	0.2071	0.5077	0.6300	0.7078	0.3420	**0.2524**	0.5075	0.6050	0.6777	0.3694	0.1450	0.3163	0.3924	0.4700	0.2291
SNPM$_{(AAAI'2023)}$	0.1795	0.4447	0.5620	0.6513	0.3019	0.2089	0.4639	0.5636	0.6395	0.3265	0.1308	0.3014	0.3796	0.4550	0.2142
MCLP$_{(KDD'2024)}$	0.1877	0.3776	0.4761	0.6001	0.3179	0.2329	0.4002	0.4788	0.5755	0.3456	0.1340	0.2480	0.3033	0.3722	0.2153
TAPT$_{(IJCAI'2025)}$	0.1211	0.4104	0.5691	0.6908	0.2552	0.1049	0.3340	0.4655	0.5841	0.2161	0.0544	0.1910	0.2790	0.3819	0.1261
DePOI$_{(SIGIR'2025)}$	0.1499	0.4630	0.6233	0.7204	0.2910	0.1528	0.4370	0.5489	0.6389	0.2812	0.1040	0.2658	0.3424	0.4265	0.1858
DPRL$_{(IJCAI'2025)}$	0.2086	0.4375	0.5166	0.5771	0.3106	0.2402	0.4859	0.5812	0.6546	0.3534	0.1465	0.2971	0.3669	0.4352	0.2209
DPRL (w/o Future Time)	0.1677	0.4099	0.5017	0.5651	0.2776	0.2223	0.4737	0.5694	0.6489	0.3374	0.1224	0.2799	0.3508	0.4229	0.1995
REPLAY$_{(TMC'2025)}$	0.2033	0.4385	0.5268	0.6049	0.3103	0.1884	0.3898	0.4707	0.5399	0.2837	0.1221	0.2402	0.2942	0.3550	0.1818
Ours w/o CL	0.2264	0.5203	0.6370	0.7225	0.3583	0.2477	0.5112	0.6056	0.6833	0.3674	**0.1763**	0.3547	0.4379	0.5174	0.2630
Ours w/o Reranker	0.2129	0.5148	0.6397	**0.7303**	0.3496	0.2393	0.5010	0.6044	0.6842	0.3586	0.1539	0.3465	0.4342	0.5178	0.2472
Ours w/ $\mathbf{e}^{cat}$	0.2271	0.5210	0.6399	0.7259	0.3606	0.2513	**0.5131**	**0.6109**	0.6862	**0.3698**	0.1754	**0.3593**	0.4403	0.5216	**0.2637**
Ours	**0.2289**	**0.5247**	**0.6415**	0.7288	**0.3626**	0.2496	0.5121	0.6100	**0.6872**	0.3682	0.1739	0.3576	**0.4442**	**0.5242**	0.2632

On the NYC dataset, our method achieves the best overall performance. Compared with the strongest baselines, it consistently improves both Acc@1 and MRR, and also yields higher Acc@K at larger K. These gains over strong structure-based and time-aware baselines suggest that a strong neural recall model combined with candidate-set reranking can capture an effective next-POI distribution without explicitly constructing historical check-in sequences or POI graphs. On the TKY dataset, the strongest hypergraph-enhanced baseline remains slightly better than our method on Acc@1 and MRR, while our method stays very close and tends to perform better at larger K. This indicates that our reranker mainly improves the ranking quality within the POI candidate set. The most significant improvements are observed on CA. Our method outperforms the strongest baselines by a clear margin on both Acc@1 and MRR, and it also consistently lifts Acc@K. We attribute this to the fact that CA has shorter user histories and sparser personalized transitions, where simple spatio-temporal regularities (e.g., time preference, user-center affinity, and last-step distance) become more informative, making candidate-set reranking particularly effective.

4.3 RQ2: Ablation Analysis

To understand the contribution of each component, we compare the full model with three variants: **w/o CL**, which removes the POI embedding contrastive learning; **w/o Rerank**, which disables the ST-Reranker and uses only the backbone logits; and **w/** $\mathbf{e}^{cat}$, which augments each check-in token with a category embedding.

4.3.1 Contrastive Learning Ablation. As shown in Tab. 1, we evaluate the effect of contrastive learning on POI embeddings. The ablation results support our design choice of using contrastive learning as an auxiliary signal: although removing the contrastive objective appears to have only a minor impact, the performance drop is consistently observed across all datasets. By aligning the next-POI query embedding with POI embeddings, contrastive learning slightly improves representation quality and ranking stability, while most of the gains still come from the overall recall-and-rerank framework.

Table 2: Reranker term ablation on three datasets.

	NYC		TKY		CA	
	Acc@1	MRR	Acc@1	MRR	Acc@1	MRR
w/o $s_{\text{cl\_sim}}$	**0.2270**	**0.3616**	0.2494	0.3682	0.1745	0.2636
w/o s_{time}	0.2195	0.3530	0.2412	0.3603	0.1534	0.2469
w/o s_{user}	0.2257	0.3607	0.2491	0.3673	**0.1750**	**0.2645**
w/o s_{dist}	0.2247	0.3602	**0.2497**	**0.3683**	**0.1750**	**0.2645**
w/o $s_{\text{st\_priori}}$	0.2140	0.3498	0.2400	0.3588	0.1537	0.2472

Table 3: Evaluation results without base logits, using only reranker scores.

	NYC		TKY		CA	
	Acc@1	MRR	Acc@1	MRR	Acc@1	MRR
w/ $s_{\text{cl\_sim}}$	0.1907	0.3002	0.2283	0.3426	0.1395	0.2257
w/ $s_{\text{st\_priori}}$	0.0295	0.0893	0.0297	0.0750	0.0351	0.0799
w/ s_{all}	0.0424	0.1165	0.0541	0.1161	0.0711	0.1274

4.3.2 The Effect of Category Embedding. We also observe that incorporating category embeddings leads to only marginal and sometimes inconsistent changes. Since our experiments show that naive concatenation of category embeddings does not reliably improve performance, we do not use category embeddings in our default

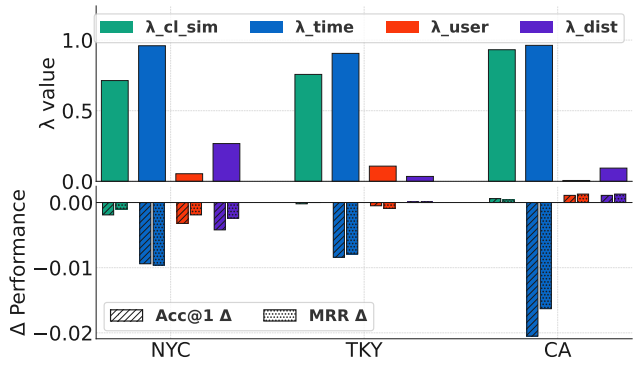

Figure 3: Hyperparameter weights (top) and the performance drops after removing each term (bottom).

setting and report this variant only as a reference. How to better leverage category information is left for future work.

4.3.3 Effect of the ST-Reranker. Disabling the ST-Reranker causes the largest performance drop across all datasets, while Acc@K at larger K changes much less. This suggests that the backbone already recalls the ground-truth POI into a short candidate list, and the reranker mainly improves early ranks by recalibrating the within-candidate ordering. The effect is most significant on CA, where trajectories and personalized transitions are sparser [18, 21], making simple spatio-temporal regularities more informative. To further understand each design in the ST-Reranker and reveal dataset characteristics, we conduct the following analyses:

Reranker Term Ablation. In Eq. 28, the ST-Reranker score consists of four reranking terms. To investigate the contribution of each term, Tab. 2 reports the results after removing each term, where w/o $s_{\text{st\_priori}}$ denotes keeping only $s_{\text{cl\_sim}}$. Overall, $s_{\text{st\_priori}}$ contribute the most and are far more important than $s_{\text{cl\_sim}}$. Within $s_{\text{st\_priori}}$, we find that the time preference term s_{time} is the most influential across all datasets. In contrast, the two spatial terms s_{user} and s_{dist} have smaller and more dataset-dependent effects, and can be neutral or even slightly noisy in some settings. The contrastive similarity term $s_{\text{cl\_sim}}$ provides only a minor marginal gain, which is expected since it is computed from the same backbone representation that produces s_{base}.

Can prediction be made using only rule-based priors? In Tab. 3, the backbone base scores s_{base} are discarded and candidates are ranked only by the reranker terms. It can be observed that relying solely on the spatio-temporal prior scores is far from sufficient for effective prediction, while $s_{\text{cl\_sim}}$ provides a much stronger stand-alone signal. However, unlike the spatio-temporal priors, $s_{\text{cl\_sim}}$ does not bring a large gain when combined with s_{base}, as shown in Tab. 2. The explanation for this can be referred to the discussion in the previous paragraph. Together, these results clarify the division of labor in our framework: the backbone logits provide the primary retrieval signal, and the reranker terms mainly act as calibration features that make subtle but impactful adjustments within an already plausible candidate set.

Validation-selected reranker weights. In Fig. 3, we report the searched hyperparameter weights for each term in s_{rerank}, along with the corresponding performance changes after removing each term. The search consistently assigns the largest weight to the λ_{time}, while the spatial priors are down-weighted and vary by dataset. Although the search may assign a relatively large weight to $s_{\text{cl\_sim}}$, its marginal contribution on top of base logits remains limited due to redundancy with s_{base}. Moreover, the results to some extent reflect the intrinsic differences among the datasets. Transitions in NYC and TKY are both more sensitive to distance, but in different ways. NYC has a larger λ_{dist}, indicating that it is more sensitive to the travel distance of each movement, whereas TKY assigns a larger λ_{user}, suggesting that users tend to be centered around their activity centers. In contrast, due to its sparsity, CA places less emphasis on both terms. As shown in Fig. 3, removing s_{dist} and s_{user} even leads to a slight improvement (The average transition distances are 3.94 km for NYC, 3.96 km for TKY, and 16.60 km for CA).

Table 4: Simple "guess" baselines on three datasets.

	NYC		TKY		CA	
	Acc@1	MRR	Acc@1	MRR	Acc@1	MRR
Global Popularity	0.0058	0.0162	0.0296	0.0690	0.0132	0.0239
User Popularity	**0.1704**	**0.2973**	0.1623	0.2837	0.1114	**0.1972**
Global Markov-1	0.1438	0.2334	0.1451	0.2368	0.1184	0.1810
User Markov-1	0.1662	0.2575	**0.2052**	**0.3061**	**0.1323**	0.1950

4.4 RQ3: Is really the last check-in all you need?

In RQ3, we first use simple statistics-based baselines to examine what distributions are actually learned from the datasets. We then analyze the information content of historical check-ins using Negative Log-Likelihood (NLL). Finally, through simple empirical ablations, we show that transition patterns are hard to learn, explaining why the subtractive design of our K1-POI is reasonable.

Simple Baselines Based on "guess". We introduce several statistics-based baselines computed from the training split to help reveal key characteristics of the datasets: **Global Popularity:** rank POIs by their visit counts. **User Popularity:** rank POIs by the user's visit counts. **Global Markov-1:** rank POIs by transition counts conditioned on the last visited POI. **User Markov-1:** rank POIs by user-specific transition counts conditioned on the last visited POI, with backoff to Global Markov-1 and then Global Popularity when needed. As shown in Tab. 4, somewhat surprisingly but reasonably, it indicates that the data contain substantial statistical regularities. By leveraging only simple statistics, these simple baselines achieve performance comparable to many carefully designed methods. This also suggests that many methods, despite their relatively strong performance, have not gone beyond simple statistical regularities, and may not have successfully captured complex transition patterns from the datasets. Even if such patterns are learned, additional noise is likely introduced during the modeling process.

The results also reveal some characteristics of the three datasets: Global Popularity performs poorly, whereas User Popularity and User Markov-1, yield more accurate "guesses". This indicates that there are very few globally popular POIs visited by the vast majority of users, and personalized user preferences play a dominant role.

Table 5: Window size ablation experiment results, showing that larger k can improve when $\mathbf{e}^{usr}$ is not enabled, but bringing performance degration when using $\mathbf{e}^{usr}$.

	NYC		TKY		CA	
	Acc@1	MRR	Acc@1	MRR	Acc@1	MRR
w/o $\mathbf{e}^{usr}$ (k=1)	0.1425	0.2401	0.1424	0.2369	0.1181	0.1892
w/ $\mathbf{e}^{usr}$ (k=1)	0.2016	0.3356	0.2170	0.3394	0.1504	0.2362
w/o $\mathbf{e}^{usr}$ (k=2)	0.1610	0.2670	0.1576	0.2575	0.1184	0.1904
w/ $\mathbf{e}^{usr}$ (k=2)	0.2007	0.3369	0.2202	0.3415	0.1467	0.2344
w/o $\mathbf{e}^{usr}$ (k=3)	0.1597	0.2735	0.1631	0.2684	0.1091	0.1885
w/ $\mathbf{e}^{usr}$ (k=3)	0.1956	0.3337	0.2160	0.3380	0.1452	0.2326
w/o $\mathbf{e}^{usr}$ (k=5)	0.1525	0.2744	0.1706	0.2812	0.1127	0.1940
w/ $\mathbf{e}^{usr}$ (k=5)	0.1933	0.3306	0.2164	0.3387	0.1410	0.2292

On NYC, User Popularity is stronger than User Markov-1, implying that stable user preference dominates and that sparse user-specific transition counts may not always provide additional benefits. On TKY, User Markov-1 is the best, indicating that TKY may exhibit more consistent one-step POI transition routines conditioned on the last check-in. On CA, User Markov-1 performs better on Acc@1, while User Popularity is slightly better on MRR, suggesting that one-step transition habits help identify the most likely next POI, whereas preference-based statistics provide a stable ranking signal under sparse user-specific transitions.

Although K1-POI still outperforms these simple baselines, the absolute performance remains far from a significant leap. The improvements on these sparse benchmarks are not from learning complex transition patterns. Several components of our model can be interpreted as learning and combining some distributions, and to some extent, essentially performing a role similar to these simple "guess" baselines. The user embedding captures a user-specific preference distribution that is closely related to what User Popularity exploits. Moreover, our user embedding together with POI embeddings can be viewed as a learned counterpart of User Markov-1: it models a user-specific one-step transition tendency conditioned on the last visited POI. In addition, the temporal and geographic embeddings, as well as the ST-Reranker terms, can be viewed as injecting conditional biases that shape the next-POI distribution by time and location. From this perspective, the trajectory flow map in GETNext [21] is also a structured representation of a Global Markov-1 distribution.

A Cross-Entropy View on Long-Range Dependencies and Global/User Markov-k. Specifically, in Fig. 4, we compute the average NLL of a count-based Markov-k backoff estimator:

$$\mathrm{NLL}(k) = \frac{1}{|\mathcal{D}_{\mathrm{eval}}|} \sum_{(u,i)\in\mathcal{D}_{\mathrm{eval}}} -\log P_k\left(p_{u,i} \mid \mathrm{S}^{(k)}_{u,<i}\right), \tag{29}$$

where $\mathrm{S}^{(k)}_{u,<i}$ denotes the most recent k POIs before position i (truncated at the beginning of the sequence), and $P_k(\cdot \mid \cdot)$ is estimated from the training split with standard backoff. Empirically, for both the Global and User variants, $\mathrm{NLL}(k)$ increases as k grows. This

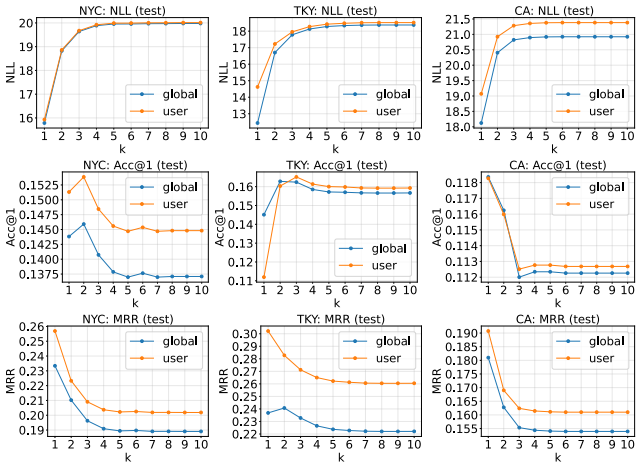

Figure 4: $\mathrm{NLL}(k)$ statistics and Acc@1/MRR of User Markov-k with different window size k.

suggests that higher-order ($k > 1$) contexts are extremely sparse: conditioning on longer histories produces overly specific and unreliable next-POI distributions, so longer dependencies do not reduce uncertainty in practice.

Moreover, we extend the above User Markov-1 baseline to User Markov-k. We build user-conditioned k-gram transition count statistics from the training split. At evaluation time, we take the most recent k POIs as the context and rank candidate POIs by the empirical next-POI counts under the longest context that is observed. If this context is unseen, we back off to shorter contexts, then to global k-gram statistics, and finally to Global Popularity. Empirically, the trend is not strictly monotonic: on NYC and TKY, increasing k from 1 to 2 improves Acc@1, while MRR already drops at $k = 2$ and keeps decreasing as k grows; on CA, both Acc@1 and MRR drop once k increases. Moreover, as shown in Tab. 5, on TKY, when user embeddings are disabled, using $k = 2$ indeed yields a noticeable improvement in Acc@1 compared to $k = 1$. CA is the sparsest dataset among the three, which has also been acknowledged in prior work. In this statistical experiment, CA exhibits a much faster drop in both Acc@1 and MRR as k increases, which is consistent with its higher sparsity. This pattern is consistent with sparsity: a slightly longer context can help make a sharper top-1 "guess" for a subset of frequent bigram routines, but higher-order contexts are quickly too sparse to estimate reliably, which makes the resulting rankings less stable and can push the true POI much lower, hurting both Acc@1 and MRR.

User embedding ablation with different windows size k. The last evidence comes from a user-embedding ablation. We let a simple attention model use only $\mathbf{e}^{usr}$ and $\mathbf{e}^{poi}$ as inputs, and compare different window sizes. The results are shown in Tab. 5. As expected, w/o $\mathbf{e}^{usr}$ is the worst, confirming that static user preference is crucial. More importantly, when $\mathbf{e}^{usr}$ is enabled, increasing the window size brings no benefit and even hurts performance; in contrast, without $\mathbf{e}^{usr}$, enlarging k gives some gains, but this "gain" does not mean long-range dependencies are learnt. Enlarging k mainly serves as a substitute for user preference. If larger k

Table 6: MRR results under different input window sizes on Foursquare and Gowalla.

Dataset	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$
Foursquare	0.3613	0.3639	0.3666	0.3676	**0.3689**
Gowalla	**0.2094**	0.2085	0.2075	0.2075	0.2053

Table 7: Performance on Foursquare and Gowalla.

Dataset	Method	Acc@1	Acc@5	Acc@10	Acc@20	MRR
Foursquare	w/o Reranker	0.2464	0.5320	0.6165	0.6874	0.3766
	w/ Reranker	**0.2930**	**0.5455**	**0.6256**	**0.6938**	**0.4091**
Gowalla	w/o Reranker	0.1701	0.3706	0.4502	0.5217	0.2651
	w/ Reranker	**0.1925**	**0.3847**	**0.4608**	**0.5321**	**0.2835**

truly provided useful transition patterns, we should also see improvements when user embeddings are enabled. Nevertheless, to be fair, $k=2$ can still provide some additional transition-pattern information compared to $k=1$. For NYC, Acc@1 exhibits a similar rise-then-fall trend in both Fig. 4 and Tab. 5 (w/o $\mathbf{e}^{usr}$). However, in Fig. 4, the performance of NYC drops below that of $k=1$ starting from $k=3$, whereas in Tab. 5 (w/o $\mathbf{e}^{usr}$), NYC still remains higher than $k=1$ at $k=3$. This is because increasing k to 2 indeed provides some additional predictive signal, while further enlarging k allows the model to partially treat the history window as a substitute for user embeddings. As a result, the degradation is not as drastic as that observed in the purely statistics-based "guess" User Markov-k.

4.5 Additional Experiments of K1-POI on the Foursquare and Gowalla datasets

In Tab. 6 and Tab. 7, we additionally evaluate our K1-POI on the Foursquare and Gowalla datasets from REPLAY [3], which are larger in scale. The results show that the proposed reranker achieves significantly improved performance. This further indicates that our method can reliably improve performance. Considering that these two datasets are larger, it can be assumed that they provide more stable data quality. The ablation results on these datasets show that the proposed Reranker yields a more pronounced performance improvement, suggesting that the Reranker can achieve better results on more robust datasets. Conversely, this may also indicate that these two datasets are of higher quality. Therefore, future work could conduct more in-depth analysis and research on these two datasets.

Consistent with the findings of the motivation experiment shown in Fig. 1, Tab. 6 reveals a similar phenomenon on Foursquare and Gowalla (notably, this result is not obtained using our K1-POI, but rather a simple model similar to that in Fig. 1); in other words, our observation that short-term information dominates is not limited to the three datasets primarily employed in this paper.

5 Conclusion

Based on our study, we argue that these three datasets are highly sparse and their trajectories lack strong semantic coherence, making it difficult for models to capture long-range transition patterns. As a result, our method achieves strong performance using only the last check-in. However, this success mainly comes from modeling several distributions rather than truly learning meaningful trajectory transitions. In the future, we still hope to learn genuine long-range mobility patterns. We believe future research can move forward in two directions: (1) constructing check-in datasets with stronger spatio-temporal semantic coherence in trajectories, and (2) investigating how to capture transition patterns beyond one-step dynamics in sparse trajectories.

Acknowledgments

This work was supported by the Macquarie University Graph Science Network. Corresponding author: Jia Wu.

References

[1] Jiajie Chen, Yu Sang, Peng-Fei Zhang, Jiaan Wang, Jianfeng Qu, and Zhixu Li. 2025. Enhancing long-and short-term representations for next POI recommendations via frequency and hierarchical contrastive learning. In *Proceedings of the Thirty-Ninth AAAI Conference on Artificial Intelligence and Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence and Fifteenth Symposium on Educational Advances in Artificial Intelligence (AAAI'25/IAAI'25/EAAI'25)*. AAAI Press, Article 1275, 9 pages. doi:10.1609/aaai.v39i11.33248

[2] Shaojie Dai, Yanwei Yu, Hao Fan, and Junyu Dong. 2022. Spatio-temporal representation learning with social tie for personalized poi recommendation. *Data Science and Engineering* 7, 1 (2022), 44–56.

[3] Bangchao Deng, Bingqing Qu, Pengyang Wang, Dingqi Yang, Benjamin Fankhauser, and Philippe Cudre-Mauroux. 2025. REPLAY: Modeling Time-Varying Temporal Regularities of Human Mobility for Location Prediction Over Sparse Trajectories. *IEEE Transactions on Mobile Computing* 24, 10 (2025), 9428–9440. doi:10.1109/TMC.2025.3562669

[4] Chenghua Duan, Wei Fan, Wei Zhou, Hu Liu, and Junhao Wen. 2023. CLSPRec: Contrastive Learning of Long and Short-term Preferences for Next POI Recommendation. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management* (Birmingham, United Kingdom) *(CIKM '23)*. Association for Computing Machinery, New York, NY, USA, 473–482. doi:10.1145/3583780.3614813

[5] Jie Feng, Yong Li, Chao Zhang, Funing Sun, Fanchao Meng, Ang Guo, and Depeng Jin. 2018. DeepMove: Predicting Human Mobility with Attentional Recurrent Networks. In *Proceedings of the 2018 World Wide Web Conference* (Lyon, France) *(WWW '18)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 1459–1468. doi:10.1145/3178876.3186058

[6] Liwei Huang, Yutao Ma, Shibo Wang, and Yanbo Liu. 2021. An Attention-Based Spatiotemporal LSTM Network for Next POI Recommendation. *IEEE Transactions on Services Computing* 14, 6 (2021), 1585–1597. doi:10.1109/TSC.2019.2918310

[7] Siyuan Huang, Jiahui Jin, Xin Lin, Xigang Sun, and Yukun Ban. 2025. IM-POI: Bridging ID and Multi-modal Gaps in Next POI Recommendation. In *Proceedings of the 33rd ACM International Conference on Multimedia* (Dublin, Ireland) *(MM '25)*. Association for Computing Machinery, New York, NY, USA, 5979–5987. doi:10.1145/3746027.3754937

[8] Yantong Lai, Yijun Su, Lingwei Wei, Tianqi He, Haitao Wang, Gaode Chen, Daren Zha, Qiang Liu, and Xingxing Wang. 2024. Disentangled Contrastive Hypergraph Learning for Next POI Recommendation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Washington DC, USA) *(SIGIR '24)*. Association for Computing Machinery, New York, NY, USA, 1452–1462. doi:10.1145/3626772.3657726

[9] Peibo Li, Maarten de Rijke, Hao Xue, Shuang Ao, Yang Song, and Flora D. Salim. 2024. Large Language Models for Next Point-of-Interest Recommendation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Washington DC, USA) *(SIGIR '24)*. Association for Computing Machinery, New York, NY, USA, 1463–1472. doi:10.1145/3626772.3657840

[10] Xixi Li, Zhuo Gu, Rui Yao, Yong Zhou, Hancheng Zhu, Jiaqi Zhao, and Wenliang Du. 2025. Beyond individual and point: next POI recommendation via region-aware dynamic hypergraph with dual-level modeling. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence* (Montreal, Canada) *(IJCAI '25)*. Article 343, 9 pages. doi:10.24963/ijcai.2025/343

[11] Xuan Rao, Shuo Shang, Lisi Chen, Renhe Jiang, and Peng Han. 2025. Disentangled and personalized representation learning for next point-of-interest recommendation. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence* (Montreal, Canada) *(IJCAI '25)*. Article 856, 9 pages. doi:10.24963/ijcai.2025/856

[12] Tianao Sun, Ke Fu, Weiming Huang, Kai Zhao, Yongshun Gong, and Meng Chen. 2024. Going Where, by Whom, and at What Time: Next Location Prediction Considering User Preference and Temporal Regularity. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (Barcelona, Spain) *(KDD '24)*. Association for Computing Machinery, New York, NY, USA, 2784–2793. doi:10.1145/3637528.3671916

[13] Dongsheng Wang, Yuxi Huang, Shen Gao, Yifan Wang, Chengrui Huang, and Shuo Shang. 2025. Generative Next POI Recommendation with Semantic ID. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2* (Toronto ON, Canada) *(KDD '25)*. Association for Computing Machinery, New York, NY, USA, 2904–2914. doi:10.1145/3711896.3736981

[14] Zhaobo Wang, Yanmin Zhu, Chunyang Wang, Wenze Ma, Bo Li, and Jiadi Yu. 2023. Adaptive Graph Representation Learning for Next POI Recommendation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Taipei, Taiwan) *(SIGIR '23)*. Association for Computing Machinery, New York, NY, USA, 393–402. doi:10.1145/3539618.3591634

[15] Yuxia Wu, Ke Li, Guoshuai Zhao, and Xueming Qian. 2022. Personalized Long- and Short-term Preference Learning for Next POI Recommendation. *IEEE Transactions on Knowledge and Data Engineering* 34, 4 (2022), 1944–1957. doi:10.1109/TKDE.2020.3002531

[16] Yang Xu, Gao Cong, Lei Zhu, and Lizhen Cui. 2024. MMPOI: A Multi-Modal Content-Aware Framework for POI Recommendations. In *Proceedings of the ACM Web Conference 2024* (Singapore, Singapore) *(WWW '24)*. Association for Computing Machinery, New York, NY, USA, 3454–3463. doi:10.1145/3589334.3645449

[17] Yuanbo Xu, Hongxu Shen, Yiheng Jiang, and En Wang. 2025. Where and when: predict next POI and its explicit timestamp in sequential recommendation. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence* (Montreal, Canada) *(IJCAI '25)*. Article 390, 9 pages. doi:10.24963/ijcai.2025/390

[18] Xiaodong Yan, Tengwei Song, Yifeng Jiao, Jianshan He, Jiaotuan Wang, Ruopeng Li, and Wei Chu. 2023. Spatio-Temporal Hypergraph Learning for Next POI Recommendation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Taipei, Taiwan) *(SIGIR '23)*. Association for Computing Machinery, New York, NY, USA, 403–412. doi:10.1145/3539618.3591770

[19] Dingqi Yang, Benjamin Fankhauser, Paolo Rosso, and Philippe Cudre-Mauroux. 2021. Location prediction over sparse user mobility traces using RNNs: flashback in hidden states!. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence* (Yokohama, Yokohama, Japan) *(IJCAI'20)*. Article 302, 7 pages.

[20] Dingqi Yang, Daqing Zhang, Vincent W. Zheng, and Zhiyong Yu. 2015. Modeling User Activity Preference by Leveraging User Spatial Temporal Characteristics in LBSNs. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 45, 1 (2015), 129–142. doi:10.1109/TSMC.2014.2327053

[21] Song Yang, Jiamou Liu, and Kaiqi Zhao. 2022. GETNext: Trajectory Flow Map Enhanced Transformer for Next POI Recommendation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Madrid, Spain) *(SIGIR '22)*. Association for Computing Machinery, New York, NY, USA, 1144–1153. doi:10.1145/3477495.3531983

[22] Feiyu Yin, Yong Liu, Zhiqi Shen, Lisi Chen, Shuo Shang, and Peng Han. 2023. Next POI recommendation with dynamic graph and explicit dependency. In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence (AAAI'23/IAAI'23/EAAI'23)*. AAAI Press, Article 539, 8 pages. doi:10.1609/aaai.v37i4.25608

[23] Lu Zhang, Zhu Sun, Ziqing Wu, Jie Zhang, Yew Soon Ong, and Xinghua Qu. 2022. Next Point-of-Interest Recommendation with Inferring Multi-step Future Preferences.. In *IJCAI*. 3751–3757.

[24] Qianru Zhang, Peng Yang, Junliang Yu, Haixin Wang, Xingwei He, Siu-Ming Yiu, and Hongzhi Yin. 2025. A survey on point-of-interest recommendation: Models, architectures, and security. *IEEE Transactions on Knowledge and Data Engineering* (2025).

[25] Hailun Zhou, Jiajie Xu, Qiaoming Zhu, and Chengfei Liu. 2025. Disentangled Graph Debiasing for Next POI Recommendation. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Padua, Italy) *(SIGIR '25)*. Association for Computing Machinery, New York, NY, USA, 1779–1788. doi:10.1145/3726302.3729952

A Appendix

A.1 Datasets

Following GETNext [21], we evaluate on three public datasets: NYC, TKY, and CA. Each check-in record includes a user ID, a POI ID, a POI category, GPS coordinates, and a timestamp. All check-ins are globally ordered by timestamp and split chronologically: the earliest 80% are used for training, the middle 10% for validation, and the latest 10% for testing. This dataset-level temporal split avoids time travel issue. In evaluation, validation and test check-ins are discarded if their user or POI never appears in the training split. The dataset statistics after conversion are in Tab. 8.

Table 8: Dataset statistics.

Dataset	#user	#poi	#categary	#check-in
NYC	1,047	4,980	318	101,760
TKY	2,262	7,793	290	358,750
CA	3,652	9,756	298	221,400

A.2 Evaluation Metrics

Following common practice in next-POI recommendation, we report Acc@K and Mean Reciprocal Rank (MRR), where each prediction instance has a single ground-truth next POI. Let N be the total number of prediction targets. For the i-th target, let $rank_i$ be the rank (1 is best) of the ground-truth POI in the predicted list. We report Acc@K for $K \in \{1, 5, 10, 20\}$ and MRR. In our evaluation protocol, each check-in except the first one of a user is treated as a prediction target. We report Acc@K for $K \in \{1, 5, 10, 20\}$ and MRR. Acc@K and MRR are computed as:

$$\begin{aligned} \text{Acc@}K &= \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}\left[rank_i \le K\right], \\ \text{MRR} &= \frac{1}{N}\sum_{i=1}^{N} \frac{1}{rank_i}. \end{aligned} \tag{30}$$

A.3 Experimental Settings and Hyperparameters

Our models are implemented in PyTorch and optimized with Adam; we use an input window size of $k = 1$, train for 100 epochs with batch size 1024, learning rate 3×10^{-4}, and weight decay 5×10^{-6}. The backbone uses model width $d = 128$ with $L = 2$ self-attention layers and 4 attention heads. Category embeddings are disabled in our default setting. We set $d_{poi} = 128$ and $d_{usr} = 256$; $\mathcal{S} = \{6, 12, 24\}$ with $d_{tod} = 64$. $\mathcal{M} = \{500, 1000, 2000\}$ meters with $d_{geo} = 64$. For contrastive learning, we set $d_{\text{CL}} = 128$, $\tau = 0.07$, and $w_{\text{CL}} = 1.0$, with negatives drawn from the backbone top-200 candidate set. The ST-Reranker operates within the same top-200 candidate set with $B = 24$ time bins, $\alpha = 1.0$, and $c_{\min} = 5$. All baselines are adapted from their official implementations and modified to fit our evaluation protocol.

A.4 Related Work

POI recommendation and prediction is a core task in location-based personalized services, aiming to infer the POIs that a user is likely

Table 9: Geo-cell size and ToD slot scale ablation on three datasets without ST-Reranker.

	NYC		TKY		CA	
	Acc@1	MRR	Acc@1	MRR	Acc@1	MRR
500	0.2105	0.3483	**0.2403**	**0.3605**	0.1544	0.2460
1000	0.2107	0.3469	0.2387	0.3582	0.1527	0.2457
2000	0.2160	0.3502	0.2364	0.3561	0.1552	0.2467
w/o $\mathbf{e}^{geo}$	0.2097	0.3430	0.2342	0.3542	0.1500	0.2396
6	**0.2195**	**0.3530**	0.2363	0.3566	0.1566	0.2480
12	0.2136	0.3496	0.2392	0.3590	**0.1576**	**0.2499**
24	0.2113	0.3480	0.2385	0.3580	0.1550	0.2463
w/o $\mathbf{e}^{tod}$	0.2093	0.3474	0.2198	0.3453	0.1514	0.2466
Full	0.2129	0.3496	0.2393	0.3586	0.1539	0.2472

Table 10: Geo-cell size and ToD slot scale ablation with ST-Reranker on three datasets with ST-Reranker.

	NYC		TKY		CA	
	Acc@1	MRR	Acc@1	MRR	Acc@1	MRR
500	0.2277	0.3606	0.2499	**0.3693**	0.1746	0.2622
1000	0.2195	0.3562	0.2506	0.3687	0.1762	0.2642
2000	0.2283	0.3602	0.2466	0.3658	0.1760	0.2626
w/o $\mathbf{e}^{geo}$	0.2242	0.3561	0.2473	0.3651	0.1662	0.2550
6	**0.2301**	**0.3627**	0.2472	0.3667	0.1754	0.2634
12	0.2249	0.3592	0.2499	0.3688	**0.1776**	**0.2659**
24	0.2251	0.3602	**0.2509**	0.3690	0.1757	0.2622
w/o $\mathbf{e}^{tod}$	0.2239	0.3598	0.2353	0.3586	0.1747	0.2646
Full	0.2289	0.3626	0.2496	0.3682	0.1739	0.2632

to visit in the future based on their historical visits. In recent years, neural sequence models strengthen short-term trajectory signals by encoding check-ins with RNN and attention. DeepMove combines recurrent modeling of sequential transitions with an attention mechanism to capture periodic mobility patterns [5]. Flashback targets sparse mobility traces by revisiting past hidden states that are temporally relevant to the current context, improving next-location prediction when trajectories are short or irregular [19]. Preference decomposition is further explored by PLSPL [15], which learns long-term preference from a longer history (via attention) and short-term preference from recent check-ins (via an LSTM), and fuses them with user-specific weights. These methods highlight the central role of short-term evidence in next POI prediction, yet they typically implement inference as producing a distribution over the full POI set in a single step. Recent work increases model capacity and incorporates richer relational/contextual signals using Transformers and graph structured learning. GETNext constructs a user-agnostic global trajectory flow map to encode collaborative transition patterns and injects it into a Transformer framework with time-aware category context [21]. AGRAN builds a directed trajectory graph with spatio-temporal edge weights and applies recurrent graph representation learning to capture adaptive dependencies [14]. STHGCN models higher-order trajectory-level relations via hypergraphs spanning intra-user and inter-user trajectories, coupled with a hypergraph Transformer to integrate spatio-temporal signals [18], while SNPM introduces dynamic graphs and explicit dependency modeling for next POI recommendation [22]. Beyond representation learning, MCLP explicitly models user preference and temporal regularity by combining topic-based preference mining, arrival-time estimation, and Transformer-based sequential pattern mining [12], and TAPT treats POI and timestamp as equally important through multi-task learning that predicts both the next POI and its timestamp [17]. Robustness and distribution shift are studied through debiasing on spatio-temporal graphs, where causal and bias signals are disentangled to improve bias-robust recommendation [25]. Generative and foundation-model based approaches have also emerged, including semantic-ID based generative next POI modeling [13] and LLM-based frameworks that preserve heterogeneous LBSN context in natural form to mitigate cold-start and short-trajectory issues [9]. For sparse trajectories and evolving temporal patterns, REPLAY models time-varying temporal regularities to improve location prediction robustness [3]. Disentangled personalization has been explored as well, though we could only access metadata-level details for DPRL [11].

These works have made significant contributions to the field and provided valuable insights. They also recognize the strong reliance of next-POI prediction on short-term signals, and thus enhance the context by modeling historical information more elaborately and incorporating graph structures. However, they rarely consider that even short-term history can introduce substantial noise, and that besides adding more information, removing noisy context can also be an effective direction.

A.5 Multi-Scale Geo-Cell Embedding and Multi-Scale ToD Embedding Ablation Study

In Tab. 9 and Tab. 10, we compare single-scale ToD embeddings, single-scale geo-cell embeddings, and the full multi-scale setting, both with and without the ST-Reranker. We observe that removing the multi-scale embeddings generally leads to performance degradation to varying degrees. However, in many cases, a single-scale setting can perform better, and the preferred scale varies across datasets. For example, on NYC, using only ToD scale 6 achieves the best performance regardless of whether the ST-Reranker is applied, while on CA, ToD scale 12 works best. In addition, TKY shows a preference for the geo-cell scale of 500. These results indicate that the optimal scale is dataset-dependent and needs to be tuned accordingly. They also suggest that future work could design more flexible multi-scale spatio-temporal embeddings to better capture and fully integrate information across different scales.

Finally, by comparing the results in Tab. 9 and Tab. 10, we can see that the performance gains brought by the ST-Reranker are highly stable and effective.