770-ROTAN

R1- Reviewer UXgD

Con1:

Lacks analysis on the differentiation between the new proposed tasks and the similar task proposed in TPG[25].

Response-to-Con1:

The research task in TPG and our task are highly related. In our work, based on the observations and analyses from real-world datasets, we formally define the Time-Specific Next POI Recommendation (TS-NPR). In [25], a Temporal Prompt-based and Geography-aware (TPG) framework is developed. In the TPG [25], the research task is not formally defined. TPG performs the next location recommendation and interval predictions. The research problem of our work and TPG are similar. However, the techniques are distinct. It proposes a temporal prompt-based decoder, using timestamps as prompts and queries for the decoder. The temporal information is incorporated via the traditional concatenation approach. We also conduct extensive experiments to evaluate and compare its performance, which demonstrate the advantages of our solution.

Con2:

The advantage of the T2R technique in mining periodic patterns is not clearly compared with existing approaches[5,10,40,51].

Response-to-Con2:

Thanks for the suggestions. In fact, periodic patterns and temporal information are widely used factors for the next POI recommendation. There are extensive research studies, including [5, 10, 40, 51], that have considered this factor. However, existing studies do not explicitly incorporate the target time when predicting the next POIs. They are developed for the conventional next-POI recommendation task. Additionally, [5, 10, 40, 51] were published 3-6 years ago, which are not the latest methods. Consequently, we do not compare them in our work. Instead, we compare our model with the more recent and stronger baselines in this paper.

To address the reviewer's concern, we have conducted experiments on three datasets to compare with these mentioned methods. Please refer to the results in the Response-to-Q5.

Con3:

The paper claims that studying temporal patterns can help solve the TS-NPR problem, but there is no comparison with methods utilizing temporal patterns in the experiments.

Response-to-Con3:

The temporal information is widely used in the POI recommendation task. Most recent methods have considered temporal patterns. In fact, the baseline models (STRNN, STGN, PLSPL, STAN, GETNext, CLSPRec, AGRAN, and TPG) compared in Section 5.1.2 have utilized various temporal patterns. For example, TPG [25] uses temporal information as prompts and GETNext [49] concatenates time embeddings into the input of the sequence modeling method (i.e., Transformer). The major differences lie in using temporal patterns for what task and how they are exploited. Most existing methods do not consider temporal patterns for the time-specific next POI recommendation problem. In addition, we propose a novel strategy to incorporate the temporal patterns.

Q1:

In Section 2.2, it is mentioned that similar work TPG can generate POI recommendations for a specific target time, but the analysis only focuses on methodological differences. It is unclear how the problem proposed in TPG differs from the one addressed in this paper, which raises doubts about the main contribution stated in the introduction - proposing a new POI recommendation task.

Response-to-Q1:

The problem proposed in TPG is similar to our task. Based on the data observations on real-world datasets, we formally defined the time-specific next-POI recommendation problem in our work. We find the asymmetric temporal influence and different temporal distribution of users and POIs. Then we design a novel Time2Rotation technique to capture temporal information. Please refer to the **Response-to-Con1**.

Q2:

T2R utilizes the concept of rotation to preserve embedding distribution. However, the impact of distribution changes due to temporal information encoding through concatenation or addition in existing work on recommendation performance is not analyzed or experimentally demonstrated.

Response-to-Q2:

In Table 3 (Section 5.3), we have presented the results of ablation studies. The variant of "(2) w/0 T2R" exploits the concatenation to replace the Time2Rotation technique in our framework, which is widely used in previous methods (e.g., TPG[25]). The concatanation of time embedding would increase the number of dimension of the learned representations, which alters the space of learned representations. Moreover, by comparing the "Original" and "(2) w/0 T2R", we can observe that the performance of rotation is much better than the concatenation.

Concatenation, addition, and rotation are three methods of utilizing time embeddings. We further investigate these methods on the datasets, and the results are shown in the following table. According to the results, we can observe that Rotation outperforms the traditional methods (concatenation and addition), which demosntrates the benefits of rotation techniques for temporal information.

Table1. The results of different methods of incorporating temporal information.

	Models	**Acc@1**	**Acc@5**	**Acc@10**	**MRR**
NYC	Concatenation	0.2562	0.4927	0.5881	0.3642
	Addition	0.2872	0.4833	0.5566	0.3792
	Rotation	0.3115	0.5259	0.6085	0.4105
TKY	Concatenation	0.2326	0.4607	0.5599	0.3402
	Addition	0.2292	0.4614	0.5492	0.3363
	Rotation	0.2493	0.4631	0.5421	0.3504
CA	Concatenation	0.1851	0.3724	0.4467	0.2750
	Addition	0.1807	0.3634	0.4444	0.2705
	Rotation	0.2173	0.3707	0.4411	0.2921

Q3:

In Section 4.2, the authors mention the construction of a collaborative POI-POI transition graph without providing details on its components, such as the representation of nodes and edges. The relationship between the quadruplet and the POI-POI graph is also not explained. It is suggested to provide an illustration within Figure 4 to clarify this aspect.

Response-to-Q3:

Thanks for your suggestion. Due to the space limit, we don't present the details of the POI-POI transition graph in this manuscript. We have introduced the main idea of the collaborative

POI-POI transition graph in Section 4.2, which has been used in previous studies [43, 49, 50]. The POI-POI transition relations (e.g., $p_i \to p_{i+1}$) are extracted from the POI trajectory. Different from existing methods, we additionally consider the temporal information (t_i, t_{i+1}) for each transition (e.g., $p_i \to p_{i+1}$). In this way, each quadruplet denotes a sequential transition relation in the POI-POI graph, shown in Equation 2. As illustrated on the left side of Figure 4, based on the given user check-in trajectory, three quadruplets can be extracted. After that, we can learn node embeddings and timeslots' rotations. Note that to capture the asymmetric POI-POI transition relations, we utilize source rotation and target rotation for the temporal information. Overall, the process is well-defined, but details are not explained due to the space limit. We will provide more explanations and enhance the illustrations within Figure 4.

Q4:

The experimental setup fails to discuss how the next POI prediction method without using the target time is applicable to the problem proposed in the paper.

Response-to-Q4:

The proposed TS-NPR problem is essentially a variant of traditional next POI prediction. Our experimental setup is the same as the conventional next POI recommendation, except for the inclusion of target time information. It incorporates the target time as input when making recommendations. TS-NPR thus combines next POI prediction with target time modeling, allowing methods designed for next POI prediction to be applicable in this paper's setup.

In Table 3 (Section 5.3), we present the results of ablation studies. The variant "(3) w/o Tgt" denotes that target time is not used, representing the same setting as the traditional next POI recommendation task. By comparing "Original" with "(2) w/o Tgt", we observe a significant performance decrease upon eliminating target time information, highlighting its pivotal role in the next POI recommendation task.

To further examine performance without using target time, we provide additional experimental results. Here, we remove the target time and evaluate our model in a next POI prediction setting. The results are shown in the following tables. Based on these tables, we observe that even without target time information, our proposed ROTAN method still achieves remarkable performance, outperforming various next-POI recommendation methods. Notably, the task in TPG is highly similar to ours, representing a recent and strong method. We compare its performance (either with or without target time) to ROTAN's performance. The experimental results indicate the benefits of the proposed ROTAN method.

Table2. The results of without using the target time on NYC.

Models	**Acc@1**	**Acc@5**	**Acc@10**	**MRR**

STAN	0.2231	0.4582	0.5734	0.3253
GETNext	0.2406	0.4815	0.5811	0.3528
CLSPRec	0.1784	0.3830	0.4591	0.2691
AGRAN	0.2121	0.4519	0.5529	0.3179
TPG(w/o Target Time)	0.2592	0.4928	0.5854	0.3635
ROTAN (w/o Target Time)	0.2499	0.5007	0.5943	0.3634
TPG	0.2555	0.5005	0.5932	0.3669
ROTAN	0.3106	0.5281	0.6131	0.4104

Table3. The results of without using the target time on TKY.

Models	**Acc@1**	**Acc@5**	**Acc@10**	**MRR**
STAN	0.1963	0.3798	0.4464	0.2852
GETNext	0.1829	0.4045	0.4961	0.2853
CLSPRec	0.1453	0.3394	0.4106	0.2340
AGRAN	0.1428	0.3737	0.4605	0.2471
TPG(w/o Target Time)	0.1770	0.3915	0.4748	0.2767
ROTAN (w/o Target Time)	0.2291	0.4531	0.5469	0.3345
TPG	0.1420	0.3631	0.4492	0.2436
ROTAN	0.2458	0.4626	0.5392	0.3475

Table4. The results of without using the target time on CA.

Models	**Acc@1**	**Acc@5**	**Acc@10**	**MRR**
STAN	0.1104	0.2348	0.3018	0.1869
GETNext	0.1526	0.3278	0.3946	0.2364
CLSPRec	0.0891	0.1815	0.2013	0.1302

AGRAN	0.1199	0.3148	0.4017	0.2140
TPG(w/o Target Time)	0.1661	0.3366	0.4019	0.2456
ROTAN (w/o Target Time)	0.1685	0.3523	0.4381	0.2535
TPG	0.1749	0.3285	0.3860	0.2479
ROTAN	0.2199	0.3718	0.4334	0.2931

Q5:

While the paper argues for the importance of temporal patterns in solving the TS-NPR problem, the comparison methods do not utilize periodic pattern methods, such as [5,10,40,51] in experiments.

Response-to-Q5:

Thank you for your comment. As mentioned in Response-to-Con2, the papers [5,10,40,51] were published 3-6 years ago and do not represent the latest methods. Instead, we employ more recent and robust baselines in our experiments. All evaluated baseline methods in our study incorporate temporal information, which is commonly used for POI recommendation.

However, to address the reviewer's concern, we made efforts to evaluate the performance of [5,10,40,51]. These methods do not explicitly consider the target time when predicting next locations. Due to limited response time and unavailability of the source code, we could not obtain results for [40] (Time-aware User Modeling with Check-in Time Prediction for Next POI Recommendation, In 2021 IEEE International Conference on Web Services (ICWS), 125–134). Their experimental results on three datasets are reported as follows. Based on results, we observe that our proposed ROTAN (with or without target time) outperforms these methods, demonstrating the superiority of the ROTAN method.

Table5. The results of the above four baselines and ROTAN on NYC.

	NYC			
	Acc@1	**Acc@5**	**Acc@10**	**MRR**
[5] (CIKM-21)	0.2477	0.4709	0.5525	0.3449
[10] (WWW-18)	0.1499	0.3661	0.4262	0.2433
[40] (ICWS-21)	0.1678	0.3581	0.4218	0.2539
[51] (WWW journal-19)	0.1996	0.4018	0.4738	0.2924

ROTAN (W/o Target Time)	0.2499	0.5007	0.5943	0.3634
ROTAN	0.3106	0.5281	0.6131	0.4104

Table6. The results of the above four baselines and ROTAN on TKY.

	TKY			
	Acc@1	**Acc@5**	**Acc@10**	**MRR**
[5]（CIKM-21）	0.1972	0.4230	0.5139	0.3034
[10] (WWW-18)	0.1168	0.2939	0.3813	0.2032
[40] (ICWS-21)	0.1649	0.3405	0.4248	0.2502
[51] (WWW journal-19)	0.1832	0.3698	0.4422	0.2707
ROTAN (W/o Target Time)	0.2291	0.4531	0.5469	0.3345
ROTAN	0.2458	0.4626	0.5392	0.3475

Table7. The results of the above four baselines and ROTAN on CA.

	CA			
	Acc@1	**Acc@5**	**Acc@10**	**MRR**
[5]（CIKM-21）	0.1654	0.3161	0.3913	0.2411
[10] (WWW-18)	0.1115	0.2336	0.2798	0.1708
[40] (ICWS-21)	0.1119	0.2472	0.2884	0.1763
[51] (WWW journal-19)	0.1292	0.2771	0.3469	0.2034
ROTAN (W/o Target Time)	0.1685	0.3523	0.4381	0.2535
ROTAN	0.2199	0.3718	0.4334	0.2931

R2 Reviewer SBto

Q1:

While must is stated about rotation vector , I do not see how this is generated within the scope of this paper. Is a learnable rotation vector, or is it generated via the Euler's identity? If that's the case, how do we ensure that is a valid rotation?

Response-to-Q1:

The rotation vector is a learnable parameter representing rotation. Following the original RotatE technique (Reference 35), we define it using Euler's identity. In practice, it can be implemented using various equivalent methods, as described in references [1, 11, 35]. The rotation operation has linear time complexity. Overall, implementing rotation vectors is straightforward. The initial rotation vectors can be randomly initialized or generated using the time2vector method (Reference 17). During the training stages, the rotation parameters are updated iteratively and ensure to represent valid rotations.

Q2:

The collaborative transition graph learning module is a bit confusing. Here, the authors propose some form of negative sampling and L1 distance optimization. What is this for? Is it a pre-training of the POI embeddings, or a pre-training of time slot embeddings? It is not clear how this graph eventually ends up with the main methods.

Response-to-Q2:

The collaborative transition graph learning module mentioned in Section 4.2 learns POI embeddings and timeslot rotations. Negative sampling is a commonly used approach for graph learning, such as in knowledge graph embedding [35] and nodevec. The main idea is a kind of contrastive learning strategy: the positive relation should have a smaller distance (the L1-norm distance defined in Equation (3)) than the sampled negative relation. Note that the L1-norm distance reflects our core idea of incorporating the temporal information for the POI-POI transition graph. Equation (3) is different from existing methods. Due to space constraints, we omit the technical details of these common techniques in this manuscript, which are widely used in knowledge graph embedding methods.

As shown in Figure 4, our ROTAN model consists of three modules: Firstly, we pre-train a collaborative transition graph mentioned in Section 4.2 to obtain the POI embeddings and timeslot rotations; Secondly, we utilize the pre-trained POI embeddings and timeslot rotations

in the trajectory learning module mentioned in Section 4.3; Finally, we recommend the time-specific next POIs by using the trajectory representations and input target time. In this framework, the learned POI embeddings and timeslot rotations, extracted from the transition graph, are used in the following two modules.

R3 Reviewer yDFY

Con1:

C1. The paper exhibits limited technical novelty. Its primary contribution lies in applying an existing technology—rotation operations—to the field of Points of Interest (POI) recommendations, specifically to capture temporal information. However, this application of this approach has already been explored in prior works. Moreover, the task of time-specific next POI recommendation has been previously addressed (e.g., TGP, CIKM2023), diminishing the uniqueness of this contribution. The remainder of the work appears to be a compilation of existing methodologies, raising questions about the paper's innovative value and its advancement of the current state of research in the field.

Response-to-Con1:

Thanks for your comments.

We cannot fully agree with the first point. To the best of our knowledge, this is the first work to explore the rotation technique to incorporate temporal information for the POI recommendation tasks. No previous study has considered the temporal influence via rotation operations in general recommendation tasks. Existing approaches commonly utilize addition or concatenation to incorporate temporal information. As we discussed in the last paragraph of Section 2.3 (lines 311-319), two papers [38, 50] utilize the rotation technique for POI recommendation. However, they only use the RotatE technique to extract embeddings of entities and relations, which is different from our Time2Rotation technique. They cannot capture the temporal information.

For the second concern, previous work (e.g., TPG (Reference 25)) treats the target time as a prompt for the next POI recommendation. In this work, we formally define the time-specific next POI recommendation task, based on extensive data observations and analyses on real-world datasets. We find that the temporal patterns of POI and user are significantly different, and analyze the asymmetric sequential transitions. Consequently, we conduct user temporal sequence modeling and POI sequence modeling separately. We propose a novel Time2Rotation technique to incorporate temporal information, which is different from the commonly used concatenation operation in TPG (Reference 25). Unlike existing approaches that assign a single embedding to each timeslot, we adopt a novel strategy by representing each timeslot with a pair of rotations to capture asymmetric temporal relations: one for the

source and another for the target. The experimental results indicate that our proposed ROTAN significantly outperforms the TPG method.

Overall, in this work, we mainly developed a novel Time2Rotation technique for incorporating temporal information, which has not been studied before. It offers several benefits: it naturally captures periodicities, does not alter the original embedding space, and can capture asymmetric temporal relations in user trajectories. Extensive experiments on real-world datasets also demonstrate the superiority of the proposed ROTAN method.

Con2:

C2. The experimental is insufficient, particularly in its omission of recent significant baselines [1 ~ 3].

[1] Successive POI Recommendation via Braininspired Spatiotemporal Aware Representation, AAAI 2024

[2] Spatio-Temporal Hypergraph Learning for Next POI Recommendation , SIGIR2024

[3] Next POI recommendation with dynamic graph and explicit dependency, AAAI2023

Response-to-Con2:

Thanks for your comment. The mentioned baselines are excellent methods.

The paper [1] (AAAI-2024) was published after our submission. Therefore, it is impractical for us to compare with it. Its source code is not publicly available. Despite our efforts to obtain it by contacting the authors or attempting to implement it ourselves, we were unsuccessful in the short response period.

While paper [2] serves as a strong baseline, it employs spatiotemporal hypergraph learning for next POI recommendation. It heavily relies on higher-order information in user trajectories and collaborative relations among trajectories. It calculates the similarity and relations between trajectories and then builds the hypergraphs. This process is computationally expensive. Note that our work and all the compared baselines do not utilize trajectory-level information. To make a fair comparison, we do not compare with paper [2].

The paper [3] has been introduced in our related work (line 312 in Section 2.3). It uses the RotatE technique for learning the POI transition graph and multi-step dependency with RNN structures.

However, to address the reviewer's concern, we provide the empirical results in the following tables. We observe that ROTAN significantly outperforms paper [3], demonstrating the benefits of the Time2Rotation technique. Our ROTAN performance is comparable with paper [2], which additionally utilizes trajectory-level information. In fact, the performance of paper [2] is significantly better than the RNN/transformer-based baselines (Section 5.1.2). This is because a trajectory-level hypergraph is first constructed, and then a hypergraph transformer

is utilized to make the next POI recommendation. However, there is no free lunch. The process in paper [2] is relatively more complex and time-consuming.

Table1. The results of above three baselines and ROTAN on NYC.

Models	Acc@1	Acc@5	Acc@10	MRR
[1] (AAAI-2024)	-	-	-	-
[2] (SIGIR-2023)	0.2702	0.5337	0.6119	0.3889
[3] (AAAI-2023)	0.1286	0.2838	0.3538	0.2029
ROTAN(w/o Target-Time)	0.2499	0.5007	0.5943	0.3634
ROTAN	0.3106	0.5281	0.6131	0.4104

Table2. The results of above three baselines and ROTAN on TKY.

Models	Acc@1	Acc@5	Acc@10	MRR
[1] (AAAI-2024)	-	-	-	-
[2] (SIGIR-2023)	0.2947	0.5161	0.5922	0.3964
[3] (AAAI-2023)	0.1601	0.3517	0.4274	0.2496
ROTAN(w/o Target-Time)	0.2291	0.4531	0.5469	0.3345
ROTAN	0.2458	0.4626	0.5392	0.3475

Table3. The results of above three baselines and ROTAN on CA.

Models	Acc@1	Acc@5	Acc@10	MRR
[1] (AAAI-2024)	-	-	-	-
[2] (SIGIR-2023)	0.1683	0.3406	0.4179	0.2512

[3] (AAAI-2023)	0.1114	0.2601	0.3334	0.1847
ROTAN(w/o Target-Time)	0.1685	0.3523	0.4381	0.2535
ROTAN	0.2199	0.3718	0.4334	0.2931

Table4. The running time of STHGCN and ROTAN on three datasets.

Models	**NYC**	**TKY**	**CA**
STHGCN [2] (SIGIR-2023)	3min24s	59min31s	15min40s
ROTAN	1min33s	5min43s	3min40s

Con3:

C3. The detail provided in the illustrations, such as Figure 4, is insufficient and overly simplistic, failing to convey the intricacies of your method's workflow. The lacking of granularity prevents a thorough understanding of the procedural steps and the methodological nuances, crucial for assessing the technique's validity and applicability. A more detailed depiction of the process would significantly enhance comprehension and evaluation of the method's innovative aspects.

Response-to-Con3:

Thank you very much for your comments and suggestions. In Figure 4, we present an overview of ROTAN. To ensure clarity and conciseness, we focus on illustrating the workflow from user check-in trajectories to final recommendations. Technical details are provided within the context, such as equations and descriptions. As depicted in Figure 4, ROTAN mainly consists of three connected components. Firstly, we utilize the collaborative transition graph mentioned in Section 4.2 to obtain POI embeddings and temporal rotations. Secondly, we employ the extracted embeddings and temporal rotations in the two blocks (user temporal sequence and POI temporal sequence) to reflect the different temporal patterns of users and POIs. Lastly, we recommend time-specific next POIs by utilizing trajectory representations and input target time. We will provide more nuanced details in Figure 4 to enhance its depiction.

Con4:

C4. The equations presented in the paper suffer from a lack of clarity, with issues stemming from redundant parameters and a lack of essential explanations. Specifically, Equation (4) introduces parameters 'p', 'u', and 't' without detailing their transformation into dense vectors or their initialization process. Furthermore, Equation (9) mentions the spatial embedding 's' without prior introduction or explanation, an oversight that could be rectified by elaborating on this symbol when discussing the GeoEncoder. Additionally, Section 4.2 omits crucial details on the construction of the collaborative POI-POI transition graph, including the specific graph representation learning method utilized. Addressing these gaps by providing the missing details and explanations would greatly enhance the paper's clarity and comprehensiveness.

Response-to-Con4:

Thanks very much for your comments. Due to space constraints, we primarily focus on conveying the meaning and logical relationships of these symbols, without delving into detailed transformation and initialization processes. The construction of the collaborative POI-POI transition graph follows the same approach as previous studies [43, 49, 50]. The main difference lies in our incorporation of temporal information into the POI-POI transition graph. We will provide more comprehensive details and explanations in our future manuscript.

Con5:

C5. The paper does not include an analysis of time complexity or efficiency testing, despite the significant time consumption implied by the use of rotation operations across various modules. This omission limits the understanding of the method's practicality and scalability, especially concerning computational resources and processing time. Incorporating efficiency experiments would provide valuable insights into the method's performance, particularly in real-world applications where computational efficiency is critical. Such analysis is essential for a comprehensive evaluation of the proposed approach.

Response-to-Con5:

Thank you very much for your comments. It's worth noting that the proposed Time2Rotation technique primarily leverages the rotation operation [35]. The time complexity of applying rotation is O(d), where d is the number of dimensions, similar to commonly used addition and multiplication operations.

ROTAN comprises primarily three components: collaborative POI transition graph learning,

individual check-in trajectory modeling, and time-specific next POI recommendation.

The time complexity of the first component is O(|E| * Neg * d * I), where |E| denotes the size of the edge set in the POI-POI transition graph, Neg is the number of negative samples for each edge (typically 5-10), d is the number of dimensions for the embeddings, and I is the number of iterations until convergence.

The second component shares the same time complexity as widely used transformer architectures, O(|T| * h * (L^2 * d + L * d^2)), where |T| represents the number of trajectories, h denotes the number of heads in the multi-head transformer, |L| is the average length of the trajectories, and d is the number of dimensions.

The time complexity of the last component is O(d * N), where d is the number of dimensions and N denotes the number of POIs in the dataset.

Additionally, we evaluate the running time of various methods for one epoch on three datasets. AGRAN, GetNext, and ROTAN utilize the POI-POI graph and transformer structure, while TPG only employs the transformer structure for the sequence. Spatiotemporal Hypergraph Convolutional Networks (STHGCN) primarily employ spatiotemporal hypergraph learning by constructing trajectory-level hypergraphs and utilize hypergraph transformers for next POI recommendation.

Theoretically, the time complexity of ROTAN is similar to GetNext [49] and AGRAN [43]. Empirically, we find that the running time of ROTAN is at a similar level to TPG, GetNext, and AGRAN. However, the running time of STHGCN is larger than other methods, as it employs more sophisticated transformer architectures.

Table5. The running time of some baselines and ROTAN on three datasets.

Models	NYC	TKY	CA
GETNext [49]	1min48s	11min52s	7min53s
STHGCN [46]	3min24s	59min31s	15min40s
TPG [25]	1min18s	5min38s	3min4s
AGRAN [43]	1min16s	5min20s	4min23s
ROTAN	1min33s	5min43s	3min40s

R4 Reviewer N4rj

W1:

w1: Some descriptions are not clear enough in this paper, see D1.

Response-to-W1:

Thanks for your comment. Please find the corresponding response to D1.

W2:

W2: There are some concerns about the experiments in this paper, see D2, D3.

Response-to-W2:

Thanks for your comment. Please find the corresponding responses to D2 and D3.

W3:

W3: The experiments of this paper can be improved, see D4, D5.

Response-to-W3:

Thanks for your comment. Please find the corresponding responses to D4 and D5.

Q1:

D1: The authors introduce their graph learning module in Section 4.2. However, they do not provide clear instructions on how to integrate the outputs with the trajectory learning in Section 4.3.

Response-to-Q1:

The collaborative transition graph learning module, as mentioned in Section 4.2, learns POI embeddings and timeslot rotations. Our ROTAN model, illustrated in Figure 4, comprises three modules: Firstly, we pre-train a collaborative transition graph mentioned in Section 4.2 to obtain the POI embeddings and timeslot rotations. Secondly, we utilize the pre-trained POI embeddings and timeslot rotations in the trajectory learning module mentioned in Section 4.3. Finally, we recommend time-specific next POIs by using the trajectory representations and input target time in Section 4.4. In this framework, the learned POI embeddings and timeslot rotations, extracted from the transition graph, are used in the following trajectory learning module. In the ROTAN framework, we utilize two independent transformers to capture the different temporal patterns for users and POIs (i.e., Section 4.3.1 and Section 4.3.2). Hence, the embeddings are used in Eq. (4) and Eq. (9), respectively. Note that their parameters will be updated during the training phases.

Q2:

D2: In Section 5.1.1, the authors propose to use the preprocessing method of [46], but the statistical information presented in this paper is inconsistent with that of [46].

Response-to-Q2:

Yes, we use the dataset and preprocessing method described in [46]. The statistical analysis presented in [46] was conducted on the raw data. However, our statistical analysis is based on the preprocessed data, which may result in some differences. We follow the preprocessing methods outlined in [46] to obtain our preprocessed data. Hence, the statistical information obtained from our preprocessed data is accurate and aligns with the paper [46].

Table1. The raw data and preprocessed data statistics of datasets in [46].

		NYC	TKY	CA
Raw data	#Users	1,048	2,282	3,957
	#POIs	4,981	7,833	9,690
	#Check-ins	103,941	405,000	238,369
Preprocessed data	#Users	1,047	2,281	3,951
	#POIs	4,937	7,821	9,670
	#Check-ins	80,166	306,345	168,922

Q3:

D3: In Section 5.3, the results of the ablation experiments indicate a significant decline in performance when ROTAN does not consider the target time. These results are lower than those reported in [46]. This may suggest that the model's performance heavily relies on the target time, but it may not be effective enough when only learning from historical information.

Response-to-Q3:

Different from conventional methods (e.g., GetNext and AGRAN), STHGCN uses the hypergraph to capture trajectory similarities and correlations. Note that our work and all the compared baselines do not utilize trajectory-level information. To make a fair comparison, we do not include paper [2] in our manuscript.

To address the reviewer's concern, we provide the performance of ROTAN without using target time in the following tables. We observe that ROTAN (w/o Target Time) still

significantly outperforms all evaluated baselines (in Section 5.1.2), indicating the effectiveness of our model when only learning from historical information. Moreover, while the performance of ROTAN (w/o Target Time) is worse than STHGCN on NYC and TKY datasets, its performance on CA is comparable to STHGCN. This is because STHGCN additionally exploits trajectory similarity information.

While paper [2] serves as a strong baseline, it employs spatiotemporal hypergraph learning for next POI recommendation. It heavily relies on higher-order information in user trajectories and collaborative relations among trajectories. In fact, the performance of paper [2] is significantly better than the RNN/transformer-based baselines (Section 5.1.2), since it additionally utilizes trajectory-level information. A trajectory-level hypergraph is first constructed, and then a hypergraph transformer is utilized to make the next POI recommendation. Although the performance is excellent, the process is complex and computationally expensive.

Table2. The results of baselines and ROTAN (with/without target time) on NYC.

Models	**Acc@1**	**Acc@5**	**Acc@10**	**MRR**
STAN	0.2231	0.4582	0.5734	0.3253
GETNext	0.2406	0.4815	0.5811	0.3528
CLSPRec	0.1784	0.3830	0.4591	0.2691
AGRAN	0.2121	0.4519	0.5529	0.3179
STHGCN[46]	0.2702	0.5337	0.6119	0.3889
ROTAN(w/o target time)	0.2499	0.5007	0.5943	0.3634
TPG	0.2555	0.5005	0.5932	0.3669
ROTAN	0.3106	0.5281	0.6131	0.4104

Table3. The results of baselines and ROTAN (with/without target time) on TKY.

Models	**Acc@1**	**Acc@5**	**Acc@10**	**MRR**
STAN	0.1963	0.3798	0.4464	0.2852
GETNext	0.1829	0.4045	0.4961	0.2853
CLSPRec	0.1453	0.3394	0.4106	0.2340
AGRAN	0.1428	0.3737	0.4605	0.2471
STHGCN[46]	0.2947	0.5161	0.5922	0.3964
ROTAN(w/o target tIme)	0.2291	0.4531	0.5469	0.3345
TPG	0.1420	0.3631	0.4492	0.2436

ROTAN	0.2458	0.4626	0.5392	0.3475

Table4. The results of baselines and ROTAN (with/without target time) on CA.

Models	**Acc@1**	**Acc@5**	**Acc@10**	**MRR**
STAN	0.1104	0.2348	0.3018	0.1869
GETNext	0.1526	0.3278	0.3946	0.2364
CLSPRec	0.0891	0.1815	0.2013	0.1302
AGRAN	0.1199	0.3148	0.4017	0.2140
STHGCN[46]	0.1683	0.3406	0.4179	0.2512
ROTAN(w/o target tIme)	0.1685	0.3523	0.4381	0.2535
TPG	0.1749	0.3285	0.3860	0.2479
ROTAN	0.2199	0.3718	0.4334	0.2931

Q4:

D4: As mentioned in D2, this paper employs the preprocessing method from [46], so [46] should also be used as a baseline method for experiments. Additionally, this paper lacks sufficient experiments. To address this, the authors could refer to the experiments conducted in [46, 49], e.g., they can conduct experiments on users with varying levels of activity and trajectory lengths.

Response-to-Q4:

Different from conventional methods (e.g., GetNext and AGRAN), STHGCN [46] uses hypergraphs to capture trajectory similarities and correlations. Note that our work and all compared baselines do not utilize trajectory-level information. To ensure a fair comparison, we do not include paper [2] in our manuscript. Additionally, STHGCN [46] is complex and computationally expensive, with a running time much larger than previous RNN/Transformer methods.

We provide additional experimental results for users with varying levels of activity and trajectory lengths in the following tables. Following [46, 49], we divided users/trajectories into different groups and evaluated their performances. Furthermore, we reproduce TPG [25] under the same experimental setup, and the results are presented in the tables. We can see that the performance of ROTAN is significantly better than GetNext [49], STHGCN [46]. and

TPG in most cases. These experimental results indicate that ROTAN performs well for both active and normal user groups. It outperforms other baselines for different kinds of trajectories.

Table5. The results of users with varying levels of activity on NYC.

User Groups	Models	Acc@1	Acc@5	Acc@10	MRR
Very active	GETNext	0.2692	0.5639	0.6995	0.3933
Normal	GETNext	0.2421	0.4739	0.5422	0.4254
Inactive	GETNext	0.1224	0.3471	0.4394	0.2319
Very active	STHGCN	0.3085	-	-	0.4402
Normal	STHGCN	0.3050	-	-	0.4265
Inactive	STHGCN	0.1460	-	-	0.3933
Very active	TPG	0.3129	0.5798	0.6791	0.4315
Normal	TPG	0.2722	0.5264	0.5825	0.3767
Inactive	TPG	0.0388	0.0780	0.1024	0.0626
Very active	ROTAN	0.3822	0.6244	0.7264	0.4961
Normal	ROTAN	0.3078	0.5276	0.5949	0.4056
Inactive	ROTAN	0.0102	0.0286	0.0494	0.0293

Table6. The results of trajectory lengths on NYC.

Trajectory	Models	Acc@1	Acc@5	Acc@10	MRR
Long trajs	GETNext	0.2452	0.5378	0.6698	0.3695
Middle trajs	GETNext	0.2441	0.4927	0.5881	0.3732
Short trajs	GETNext	0.2186	0.4561	0.5269	0.3570
Long trajs	STHGCN	0.3184	-	-	0.4401
Middle trajs	STHGCN	0.2545	-	-	0.3795
Short trajs	STHGCN	0.2703	-	-	0.3783
Long trajs	TPG	0.3048	0.5482	0.6270	0.4042
Middle trajs	TPG	0.2681	0.5127	0.5824	0.3749
Short trajs	TPG	0.2396	0.4082	0.4881	0.3191
Long trajs	ROTAN	0.3432	0.5664	0.6307	0.4449

Middle trajs	ROTAN	0.3012	0.5282	0.6011	0.4057
Short trajs	ROTAN	0.2781	0.4403	0.5059	0.3555

Q5:

D5: Unlike most previous POI recommendation works, this paper utilizes the next temporal information as a known condition for predicting POIs. In the existing baseline methods, only a portion of experiments in TPG aligns with this paper. However, TPG also conducts experiments on conventional POI recommendation. To better demonstrate the effectiveness of ROTAN, the authors should consider conducting similar experiments with TPG.

Response-to-Q5:

Thanks for the suggestion. We have carefully checked TPG[25], and it doesn't conduct extensive experiments on conventional POI recommendation. Similarly, it conducts an ablation study as our work by removing the temporal-based prompt (i.e., remove TP).

To make thorough comparisons, we provide extensive results in the following tables. We can learn that our ROTAN outperforms TPG in various experimental settings, including without target time, without both time embeddings and target time, and the default setting. The results demonstrate the superiority of the proposed ROTAN over the latest TPG method.

Table7. The results of different experiment settings of TPG and ROTAN on NYC.

	Acc@1	**Acc@5**	**Acc@10**	**MRR**
TPG(w/o Target time)	0.2592	0.4928	0.5854	0.3635
TPG(w/o sequence Time and Target time)	0.2533	0.4991	0.5737	0.3613
TPG	0.2555	0.5005	0.5932	0.3669
ROTAN(w/o Target time)	0.2499	0.5007	0.5943	0.3634
ROTAN(w/o sequence Time and Target time)	0.2628	0.5024	0.6096	0.3754
ROTAN	0.3106	0.5281	0.6131	0.4104

Table8. The results of different experiment settings of TPG and ROTAN on TKY.

	Acc@1	**Acc@5**	**Acc@10**	**MRR**
TPG(w/o Target time)	0.1770	0.3915	0.4748	0.2767
TPG(w/o sequence Time and Target time)	0.1588	0.3689	0.4583	0.2577
TPG	0.1420	0.3631	0.4492	0.2436
ROTAN(w/o Target time)	0.2291	0.4531	0.5469	0.3345
ROTAN(w/o sequence Time and Target time)	0.2139	0.4636	0.5539	0.3276
ROTAN	0.2458	0.4626	0.5392	0.3475

Table9. The results of different experiment settings of TPG and ROTAN on CA

	Acc@1	**Acc@5**	**Acc@10**	**MRR**
TPG(w/o Target time)	0.1661	0.3366	0.4019	0.2456
TPG(w/o sequence Time and Target time)	0.1612	0.3394	0.4105	0.2441
TPG	0.1749	0.3285	0.3860	0.2479
ROTAN(w/o Target time)	0.1685	0.3523	0.4381	0.2535
ROTAN(w/o sequence Time and Target time)	0.1832	0.3671	0.4524	0.2697
ROTAN	0.2199	0.3718	0.4334	0.2931

R5 Reviewer rJau

Con1:

The proposed solution is originally proposed for time-specific next POI recommendation task. Better to analyze its capability for general time-aware recommendation tasks.

Response-to-Con1:

The proposed solution is not limited to the POI recommendation task. Similar to commonly used addition or concatenation operations, the time2rotation technique incorporates temporal influence through rotation operations, making it a model-agnostic general technique. It can be readily applied to other time-aware recommendation tasks. We plan to further explore the time2rotation technique for general temporal studies in the future.

We provide the experimental results of the latest method (TPG [25]) and ROTAN in the following table. In this experimental setting, we only consider the time-aware POI recommendation task, i.e., predicting the POIs at specific times. It doesn't utilize the context information in the trajectories. We can see that ROTAN significantly outperforms TPG, indicating the advantages of the Time2rotation technique.

Table1. The results of TPG and ROTAN for general time-aware recommendation task on three datasets.

Datasets	**Models**	**Acc@1**	**Acc@5**	**Acc@10**	**MRR**
NYC	TPG time-aware	0.1812	0.3623	0.4387	0.2665
	ROTAN time-aware	0.2247	0.4084	0.4893	0.3128
TKY	TPG time-aware	0.0819	0.2142	0.2723	0.1448
	ROTAN time-aware	0.1071	0.2256	0.2745	0.1656
CA	TPG time-aware	0.0838	0.1548	0.1889	0.1198
	ROTAN time-aware	0.1015	0.1925	0.2315	0.1446

Con2:

The differences between Time2rotation and the time2vec technique [17] can be further elaborated.

Response-to-Con2:

Time2Rotation is a method of utilizing temporal information, while time2vector [17] is a method of representing temporal information, i.e., learning a vector representation of time. In time2vec [17], it mainly utilizes periodic activation functions for time representation. Conventionally, timeslots are represented with time embedding vectors, and then these vectors are integrated using concatenation or addition operations. In this work, we propose an innovative approach to integrate temporal information using rotation techniques. Instead of

conventional embedding vectors, we represent timeslots with rotation vectors. Moreover, we incorporate temporal influence via rotation operations, which differs from existing methods.

These two techniques can be jointly utilized. For instance, we could use the time2vec technique to initialize the rotation vectors, which are learnable and can be updated during the training phases.

Q1:

As far as I know, the rotation technique has been used for periodical time series predictions [2]. What is the difference between time2rotation and learning-to-rotate [2]? The authors should provide more explanations regarding this issue.

Response-to-Q1:

The learning-to-rotate [2] is also proposed for temporal information, mainly designed for complicated periodical time series forecasting. It primarily utilizes Quaternion Transformer to capture multiple periods, variable periods, and phase shifts in real-world datasets. Learning-to-rotate [2] indicates that the rotation operation can effectively incorporate temporal information, especially periodical patterns. However, our work differs from learning-to-rotate in three aspects. (1) They address different tasks. Our ROTAN solves the time-specific recommendation task, while [2] focuses on time series forecasting. The proposed solution of [2] cannot directly address our task. (2) The approach to rotation is different. In [2], quaternions are used for the attention module. Quaternions are extensions of complex numbers, represented as □ = □ + □i + □j + □k, where □, □, □, □ are real numbers and i, j, k are imaginary units. It exploits rotation with unit quaternions. In our work, the rotation is based on complex space, which is relatively simpler. (3) The strategy for exploiting rotation to incorporate temporal information is different. To measure similarity in time series, [2] designs a novel Learning-to-Rotate Attention mechanism, which replaces the canonical attention mechanism. Our work applies rotation operations to the extracted representations, inspired by RotatE [35].

Q2:

Figure 3 is difficult to interpret. It requires some effort to get the different patterns from the hour distributions of users/POIs.

Response-to-Q2:

To investigate the temporal information of check-in data, we conduct data observations based

on real-world datasets. Specifically, we study the distributions of the number of hours for users and POIs in Figure 3. Figure 3(a) shows the statistics of the number of hours that users are active. We can see that most users have check-ins between 10 and 18 hours, indicating their activity throughout various periods. Meanwhile, as shown in Figure 3(b), most POIs were visited between 2 and 8 hours within the 24-hour period, which is narrower than the distribution of Figure 3(a). This distribution suggests that most POIs are accessible by users during specific hours. For instance, restaurants may primarily be visited during lunch or dinner times. The distinct temporal distributions of users and POIs necessitate separate modeling approaches. Since users and POIs exhibit distinct temporal patterns, we consider them independently in Section 4.3: User Temporal Sequence Modeling and POI Temporal Sequence Modeling.

R6 Reviewer h4BY

Con1:

The time complexity and running time of the ROTAN is not analysed. I am wondering the additional time complexity introduced by the Time2Rotation technique.

Response-to-Con1:

Thanks for your comments.

The proposed Time2Rotation technique has linear complexity, primarily leveraging the rotation operation [35]. The time complexity of applying rotation is O(d), where d is the number of dimensions, similar to commonly used addition and multiplication operations.

ROTAN primarily comprises three components: collaborative POI transition graph learning, individual check-in trajectory modeling, and time-specific next POI recommendation.

The time complexity of the first component is O(|E| * Neg * d * I), where |E| denotes the size of the edge set in the POI-POI transition graph, Neg is the number of negative samples for each edge (typically 5-10), d is the number of dimensions for the embedding, and I is the number of iterations until convergence.

The second component shares the same time complexity as widely used transformer architectures, O(|T| * h * (L^2 * d + L * d^2)), where |T| represents the number of trajectories, h denotes the number of heads in the multi-head transformer, |L| is the average length of the trajectories, and d is the number of dimensions.

The time complexity of the last component is O(d * N), where d is the number of dimensions

and N denotes the number of POIs in the dataset.

Theoretically, the time complexity of ROTAN is close to GetNext[49] and AGRAN[43]. Empirically, we find that the running time of ROTAN is at a similar level with TPG, GetNext, and AGRAN. However, the running time of STHGCN is larger than other methods, as it employs more sophisticated transformer architectures.

Table1. The running time of baselines and ROTAN on three datasets.

Models	**NYC**	**TKY**	**CA**
GETNext [49]	1min48s	11min52s	7min53s
STHGCN [46]	3min24s	59min31s	15min40s
TPG [25]	1min18s	5min38s	3min4s
AGRAN [43]	1min16s	5min20s	4min23s
ROTAN	1min33s	5min43s	3min40s

Con2:

A few typos exist in this manuscript, which can be further improved. For instance, line 202-203 (should in one line) and line 383 (should be figure 3).

Response-to-Con2:

Thanks for your comments. We will thoroughly proofread to address those typos.

Q1:

This work divides each day into 48 timeslots. It also examines the granularity of time slots in Figure 5. However, it still relies on discrete time representations. Is it possible to consider the continuous temporal information?

Response-to-Q1:

This is a very good question. The current ROTAN framework cannot be directly used for continuous temporal information. This is because we first split the temporal information into time slots and then learn the rotations for these time slots. This manner is commonly adopted in previous methods (e.g., GETNext[49] and TPG[25]). Additionally, as shown in Figure 5(c),

finer time granularity of time slots does not lead to performance enhancement. Although more accurate, very fine granularity or continuous temporal information is not always helpful due to data noise, uncertainty, relatively sparse check-in data, and complex user behaviors, etc. A trade-off is required. In fact, representation learning of continuous temporal information is a very challenging task in many domains, such as time series prediction problems. We will explore continuous temporal information for next POI recommendation in future work.

Q2:

This paper is inspired by the RotatE strategy. As described in Section 2.3, there are some works that also utilize RotatE-relevant techniques (references [50,38]) for next POI recommendation. Better to discuss their performances compare to ROTAN.

Response-to-Q2:

SNPM[50] and ToP[38] are two effective methods for next-POI recommendation. Both [38] and [50] are inspired by RotatE[35]. Based on the extracted POI relevant graphs, they use the rotation technique to learn representations of entities and relations. The rotations in [38, 50] are mainly used for the relations between two entities, while we explore rotations for temporal information and asymmetrical sequential transitions, which distinguishes it from these methods..

To enhance the credibility of our work, we tried to compare with them. Since the source code of ToP[38] is not publicly available and very complex to implement, we cannot obtain its results in the short term. For SNPM[50], we use the source code and the recommended parameter settings. We compare the SNPM[50] and our ROTAN method on the three datasets. Based on the empirical results, we can observe that ROTAN significantly outperforms the SNPM method.

Table2. The results of above two baselines and ROTAN on three datasets.

Datasets	Models	Acc@1	Acc@5	Acc@10	MRR
NYC	ToP[38]	-	-	-	-
	SNPM[50]	0.1286	0.2838	0.3538	0.2029
	ROTAN	0.3106	0.5281	0.6131	0.4104
TKY	ToP[38]	-	-	-	-
	SNPM[50]	0.1601	0.3517	0.4274	0.2496
	ROTAN	0.2458	0.4626	0.5392	0.3475

CA	ToP[38]	-	-	-	-
	SNPM[50]	0.1114	0.2601	0.3334	0.1847
	ROTAN	0.2199	0.3718	0.4334	0.2931