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 [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 [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 the 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 the 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 the 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 utilizing hypergraph transformers for the 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