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 to 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, 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 the 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 the 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