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 [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 [17]. During the training stages, the rotation parameters are updated iteratively and ensured 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]. 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.