KGTS: Contrastive Trajectory Similarity Learning over Prompt Knowledge Graph Embedding

Zhen Chen[1], Dalin Zhang[2], Shanshan Feng[3, 4], Kaixuan Chen[2], Lisi Chen[1], Peng Han[1*], Shuo Shang[1*]

[1]University of Electronic Science and Technology of China
[2]Aalborg University, Denmark
[3]Centre for Frontier AI Research, A*STAR, Singapore
[4]Institute of High-Performance Computing, A*STAR, Singapore
{chenzhen059, jedi.shang}@gmail.com, {dalinz, kchen}@cs.aau.dk,
{victor_fengss, penghan_study}@foxmail.com, lchen012@e.ntu.edu.sg

Abstract

Trajectory similarity computation serves as a fundamental functionality of various spatial information applications. Although existing deep learning similarity computation methods offer better efficiency and accuracy than non-learning solutions, they are still immature in trajectory embedding and suffer from poor generality and heavy preprocessing for training. Targeting these limitations, we propose a novel framework named KGTS based on knowledge graph grid embedding, prompt trajectory embedding, and unsupervised contrastive learning for improved trajectory similarity computation. Specifically, we first embed map grids with a GRot embedding method to vigorously grasp the neighbouring relations of grids. Then, a prompt trajectory embedding network incorporates the resulting grid embedding and extracts trajectory structure and point order information. It is trained by unsupervised contrastive learning, which not only alleviates the heavy preprocessing burden but also provides exceptional generality with creatively designed strategies for positive sample generation. The prompt trajectory embedding adopts a customized prompt paradigm to mitigate the gap between the grid embedding and the trajectory embedding. Extensive experiments on two real-world trajectory datasets demonstrate the superior performance of KGTS over state-of-the-art methods.

Introduction

GPS sensors emit continued points of location data, the sequence of which assembles a *trajectory* that describes the spatial path of a moving object over time. The similarity between two trajectories of different objects or different time segments of the same object is a fundamental measure for many real-world applications, such as animal migration analysis (Li et al. 2011), transportation optimization (Song et al. 2014), route retrieval (Ranu et al. 2015), and traffic prediction (Li et al. 2023; Lin et al. 2023; Chang et al. 2023). Therefore, the *trajectory similarity computation* has always been a research hotspot for decades (Yang et al. 2021; Yao et al. 2019; Evans et al. 2013; van Kreveld and Luo 2007).

Various trajectory similarity computation approaches have been investigated such as dynamic time warping (DTW) (Yi, Jagadish, and Faloutsos 1998), edit distance on real sequences (EDR) (Chen, Özsu, and Oria 2005), and Hausdorff distance (Atev, Miller, and Papanikolopoulos 2010). Despite some success, they suffer from a common deficiency of high computation complexity (Li et al. 2018). Specifically, since two trajectories need to be aligned point by point, the computation complexity attains a quadratic form $\mathcal{O}(l^2)$, where l is the mean length of the trajectories. This flaw hinders their practical applications to long trajectories and large datasets.

Although previous methods have achieved good description about similarity, there are still some problems in certain aspects. First, existing deep learning similarity computation work requires heavy preprocessing. Specifically, these studies generally rely on supervised learning, which requires the similarity measure of each pair of trajectories as the supervision label (Yang et al. 2021; Han et al. 2021; Yang et al. 2022b). However, the similarity measures are not directly available in the raw dataset and thus have to be calculated during the dataset preprocessing, requiring extensive computation. Assuming there are q trajectories in the dataset, each of which has l points, the complexity of computing the similarities of all trajectory pairs is thus quadratic as $\mathcal{O}(q^2l^2)$. Therefore, it is costly to achieve a ready-to-use dataset when the raw dataset has numerous long trajectories.

Second, a realistic dataset cannot contain trajectory patterns exhaustively. The supervision based methods find positive samples in the dataset, however, there are still some trajectories whose positive samples are not really similar to themselves. In addition, distance of location is considered more than similarity of structure in previous methods, and having high similarity of structure is in a same paragraph. Therefore, a large dataset with diverse trajectory relationships is essential to a model with high generality. However, it is unrealistic to handle a real-world dataset with exhaustive relationships of trajectories.

In addition to the limitations, there is still room to improve the performance of existing solutions. Some studies (Li et al. 2018; Yao et al. 2019) do not use dedicated modules to embed three sorts of information of a trajectory, namely loca-

*Shuo Shang and Peng Han are corresponding authors.

tion, structure, and the order of points, while others rely on out-of-dated modules or ineffective frameworks.

In light of the above considerations, we attempt to address the above issues. For the first problem, we adopt an unsupervised method to reduce the burden of computing labels, which is achieved through contrastive learning. For the second matter, just ordinary data augmentation (Deng et al. 2022) has great limitations and is not sufficient for similarity calculation. We propose a new method to alleviate the shortcomings of unsupervised methods. Specifically, this enhancement method aims at generating positive samples which are similar to the original trajectories. To extract spatial information accurately, we consider the relationship between grids and attempt to acquire better grid representations through knowledge graph embedding methods.

In this paper, we propose KGTS, a brand-new framework for improved trajectory similarity computation. Specifically, based on the grid-based approach (Li et al. 2018; Zhang et al. 2011), we first innovatively employ a novel relation model GRot (Grid RotateE) to embed the grids of the entire space to encourage neighbouring grids to have similar embedding. Next, to alleviate the incompatibility from pretrain to fintune, we propose a prompt trajectory embedding module that has a novel attentive prompt scheme to effectively incorporate the grid embedding into the final trajectory embedding. The trajectory embedding module has a GCN to model the trajectory structure and a GRU to extract the order of points in a trajectory. We train the prompt trajectory embedding module with unsupervised contrastive learning so that supervision labels and consequently the costly preprocessing are not required. Besides, three novel strategies of positive sample generation for contrastive learning are devised to simulate diverse cases of highly similar trajectories thus enhancing the model's generality.

The main contributions are summarized as follows:

- First, we propose a framework KGTS including grid embedding and prompt trajectory embedding through unsupervised training scheme.
- Second, we propose the GRot for trajectory grid embedding so that spatially neighbouring grids are encouraged to have similar embedding, and a prompt trajectory embedding module for trajectory embedding that properly grasps the location, structure and points order information of trajectories.
- Third, we train the prompt trajectory embedding module using unsupervised contrastive learning with newly designed positive sample generation strategies.
- Finally, we conduct extensive experiments on two large benchmark datasets to justify our design and its superior performance to state-of-the-art studies.

Related Work

Trajectory Similarity Computation Trajectory similarity computation methods can roughly be divided into two categories, namely, knowledge-based methods and learning-based methods. The computational complexity of knowledge-based methods, such as longest common subsequence (LCSS) (Vlachos, Gunopulos, and Kollios 2002), Dynamic Time Warping (DTW) (Yi, Jagadish, and Faloutsos 1998), edit distance with real penalty (ERP) (Chen and Ng 2004), and edit distance on real sequences (Chen, Özsu, and Oria 2005) (EDR), heavily depends on the length of trajectories. With the rapid development of machine learning especially the deep learning technique in various areas (LeCun, Bengio, and Hinton 2015), an early study is t2vec (Li et al. 2018; Yao et al. 2019), which employs a seq2seq model to consider the order of points for trajectory embedding. T3S (Yang et al. 2021) is a combination of the grid-based method and the coordinate-based method that alleviates the impact of noisy points and measurement errors. A recent work GTS (Han et al. 2021) represents the trajectories with a two-step process, i.e., a skip-gram point embedding step and a GNN-based trajectory embedding step for good performance.

Prompt Learning In this work, we adopt a two-step process: the first step embedding trajectory points and the second step embedding the whole trajectories. Since these two steps are separately trained and the first step does not comply with the final trajectory embedding objective, we adopt the prompt learning (Liu et al. 2021) scheme to facilitate the second-step learning process. Prompt learning is a technique that autonomously tunes the downstream learning process to fit a pre-trained model through prompts. In this light, recent studies propose to learn the prompt together with the downstream model (Gao, Fisch, and Chen 2021; Jiang et al. 2020), which are revised from early studies (Brown et al. 2020; Raffel et al. 2020).

Contrastive Learning Contrastive learning has recently shown profound performance and influence on various tasks (Rethmeier and Augenstein 2021; Yu et al. 2022; Yang et al. 2022a). A key aspect of contrastive learning is to create positive and negative samples. SimCSE (Gao, Yao, and Chen 2021) utilizes the smile dropout technique to generate positive samples from input sentences. Deep InfoMax (Hjelm et al. 2019) creates positive and negative samples by manipulating local and global features of images. CBT (Sun et al. 2019a) masks a part of a video clip and uses the masked videos as positive samples.

Preliminary

Problem Statement

DEFINITION 1 (Trajectory)**.** *A trajectory T is formed as a sequence of points, i.e., $T = \langle p_1, p_2, ..., p_n \rangle$, where $p_i = (lat_i, lon_i)$ is the i-th point associated with a tuple of latitude lat_i and longitude lon_i, and n is the length of the trajectory.*

DEFINITION 2 (Trajectory Similarity)**.** *Given two trajectories T_i and T_j, their distance $dist(T_i, T_j)$ is used as the similarity measure $sim_*(T_i, T_j)$ to assess the similarity between T_i and T_j. Then, T_j is similar to T_i, if $sim_*(T_i, T_j) < \epsilon$ or $\forall\, T_p \in D$, $sim_*(T_i, T_j) \leq sim_*(T_i, T_p)$, where D is a set of trajectories.*

Given a specific similarity measure $sim_*(\cdot, \cdot)$, our goal is to find an optimal trajectory embedding function $f_e^*(\cdot)$:

$$f_e^* = \underset{f_e}{\operatorname{argmin}} \underset{D}{\mathbb{E}} |sim_*(f_e(T_i), f_e(T_j)) - sim_*(T_i, T_j)|, \tag{1}$$

where $T_i, T_j \in D$. Note that $sim_*(\cdot,\cdot)$ can be any distance metric such as Euclidean distance for trajectory computing or Cosine distance for representation computing.

Grid-based Trajectory Embedding Realistic trajectories often have non-uniform sampling rates and noisy points. For example, GPS receivers may miss data or have erroneous recordings due to poor satellite visibility. One of the problems with trajectory embedding is how to embed points. Following previous work (Yang et al. 2021; Li et al. 2018), we adopt the grid-based trajectory embedding for handling varying sampling rates and noise in trajectories. Specifically, the space (e.g., a map) is partitioned into m grids of equal size (see the map in Figure 1), and a trajectory point falling into a grid is represented by the grid entity g_i. Next, a trajectory T can be represented by a sequence of grids:

$$T = \langle p_1, p_2, ..., p_n \rangle \Rightarrow T \approx \langle g_1, g_2, ..., g_n \rangle, \tag{2}$$

where $g_i \in [1, m]$ is the gird ID of the i-th trajectory grid. Then, the trajectory embedding problem is converted to embed the grid sequence of a trajectory.

The Proposed KGTS Method

As shown in Figure 1, our KGTS has two main modules: a grid embedding module that embeds all grids in the entire space and a prompt trajectory embedding module that embeds specific trajectories. In addition, we propose an unsupervised contrastive learning scheme for efficient trajectory similarity learning.

GRot Gird Embedding

Knowledge graph embedding is the task of learning representations of graph nodes considering both their entities and relations. Recent research has well demonstrated its success for various downstream tasks (Ji et al. 2021; Huang et al. 2019; Sun et al. 2018). The task of grid embedding is to embed space grids about their locations and relations, so knowledge graph embedding naturally fits this goal. Specifically, we regard the entire space as a graph and its grids as nodes. We only consider one relation between grids, that is *direct connection*; One grid is considered to have direct connections to its eight immediately neighbouring grids in the space. The notable RotatE model (Sun et al. 2019b) is chosen and further modified to better adapt to our case.

The RotatE model originates from the Euler's identity and represents the head $\mathbf{h}$ and tail $\mathbf{t}$ entity of an edge/relation in a graph with embedding in the complex space. The mapping from $\mathbf{h}$ to $\mathbf{t}$ induced by relation $\mathbf{r}$ is realised by an element-wise rotation:

$$\mathbf{t} = \mathbf{h} \circ \mathbf{r}, \tag{3}$$

where $\mathbf{h}$, $\mathbf{t} \in \mathbb{C}^k$ are embeddings in the complex space of the head and tail entities, $\mathbf{r} \in \mathbb{C}^k$ is the embedding of the relation between $\mathbf{h}$ and $\mathbf{t}$, and $\circ$ is the Hadamard (or element-wise) product. The r_j is of the form $e^{i\Theta_{r,j}} = \cos\Theta_{r,j} + i\sin\Theta_{r,j}$ and thus $|r_j| = 1$. Therefore, the original RotatE score function that measures how distant two nodes $\mathbf{h}$, $\mathbf{t}$ are relative to their relation $\mathbf{r}$ is defined as:

$$d(\mathbf{h}, \mathbf{t}) = \|\mathbf{h} \circ \mathbf{r} - \mathbf{t}\|. \tag{4}$$

In the context of knowledge graph for grid embedding, $\mathbf{h}$ and $\mathbf{t}$ are grid embeddings in the complex space, i.e., $\mathbf{h}, \mathbf{t} = \cos\Phi_j + i\sin\Phi_j$, where Φ_j is randomly initialized and learnable hidden embedding for grid g_j. Unlike conventional knowledge graph embedding problems, we argue that the graph of the map space additionally requires that the neighbouring grids in terms of geographical locations have similar embedding. It is thus expected that $\mathbf{h} \circ \mathbf{r} = \mathbf{t} \circ \mathbf{r}$ and consequently the score function in Eq. 4 is modified as:

$$d_r(\mathbf{h}, \mathbf{t}) = \|\mathbf{h} \circ \mathbf{r} - \mathbf{t} \circ \mathbf{r}\|, \tag{5}$$

where $\mathbf{r} = \cos\Theta + i\sin\Theta$ parameterized by Θ is the relation of direct connection of two grids. With this customized RotatE score function, both grids and grid relations can be well embedded. Same as the original RotatE model, $\Theta \in \mathbb{R}^k$ is randomly initialized and learned together with Φ_j using the self-adversarial negative sampling approach (Sun et al. 2019b) and the following loss function (Sun et al. 2019b):

$$\begin{aligned} L = &-\log\sigma(\gamma - d_r(\mathbf{h}, \mathbf{t})) \\ &- \sum_{j=1}^{o} p(h_j^{'}, r, t_j^{'}) \log\sigma(d_r(\mathbf{h}_j^{'}, \mathbf{t}_j^{'}) - \gamma), \end{aligned} \tag{6}$$

where γ is a fixed margin and $\sigma(\cdot)$ is the sigmoid function. $p(h_j^{'}, r, t_j^{'})$ is the negative sampling probability of the j-th negative sample $(h_j^{'}, r, t_j^{'})$, which is the grid embedding pair that does not have the relation r, i.e., the two grids are not directly connected.

After training the GRot, we achieve the embedding of each grid as the Hadamard product between $\mathbf{h}_i$ and $\mathbf{r}$:

$$\tilde{\Phi}_i = \mathbf{h}_i \circ \mathbf{r}, \text{ where } \tilde{\Phi}_i \in \mathbb{R}^k. \tag{7}$$

Prompt Trajectory Embedding

We now present how to use the achieved grid embedding $\tilde{\Phi}_i$ to produce the final trajectory embedding. The grid embedding embeds all grid entities regarding the grid locations in the entire map space without considering trajectory information. However, our goal is to embed specific trajectories. Thus, there is a gap between these two embedding objectives. Inspired by the recent success of prompt learning (Liu et al. 2021), we employ a prompt to instruct the trajectory embedding module to unify the two embedding objectives.

Prompt Design Different from the original prompt learning for NLP, which designs human-understandable prompts, we propose to learn the prompt together with the trajectory embedding network. Moreover, we propose an attentive prompt concatenation scheme that concatenates prompt and grid embedding (Eq. 7) with attentive weights. Then, the prompt grid embedding P_i is formally denoted as:

$$\begin{aligned} P_i &= [\ \alpha_1^i U\ ;\ \alpha_2^i \tilde{\Phi}_i\ ] \\ \alpha_1^i &= UW_1,\ \alpha_2^i = \tilde{\Phi}_i W_2, \end{aligned} \tag{8}$$

where α_*^i is the attentive concatenation coefficient, $U \in \mathbb{R}^{1\times u}$ is the prompt vector, and $W_1 \in \mathbb{R}^{u\times 1}$ and $W_2 \in \mathbb{R}^{k\times 1}$ are the learnable weights to achieve α_*^i. Thus, $P_i \in \mathbb{R}^{u+k}$. Through this design, the subsequent trajectory embedding can properly incorporate grid embedding under the instruction of a learnable attentive prompt.

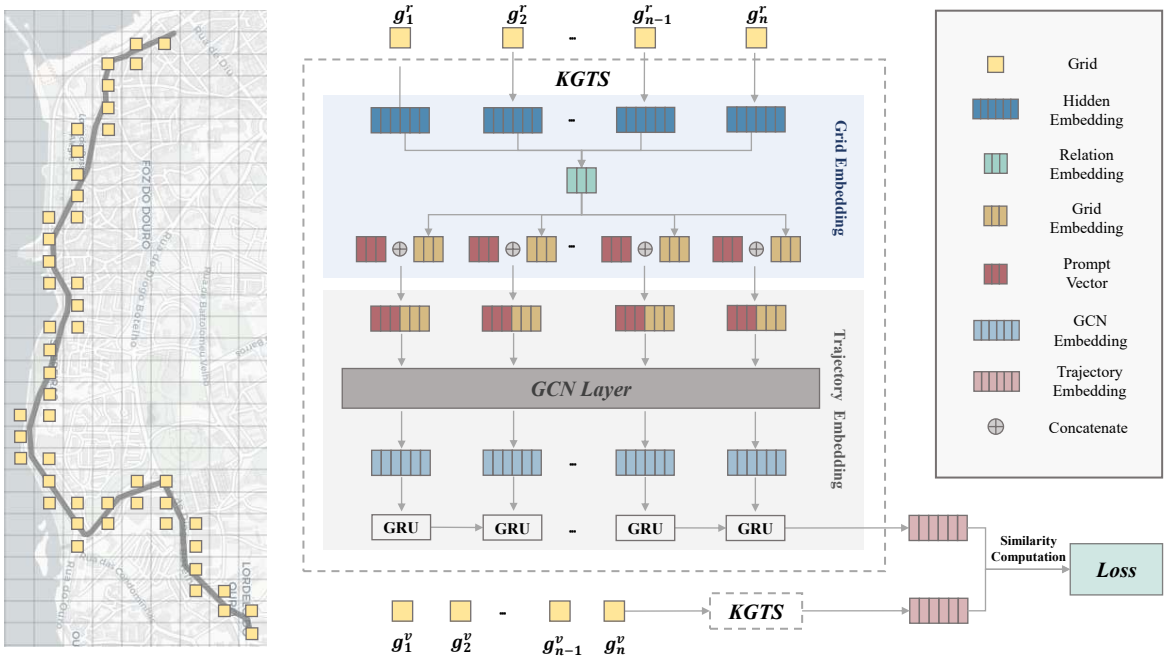


Figure 1: Overall framework of KGTS. Left: a trajectory and its grid-based representation; Right: the KGTS network structure. $[g_1^r, ..., g_n^r]$ is the grid sequence of a trajectory T_i; $[g_1^v, ..., g_n^v]$ is the grid sequence of a negative/positive sample for T_i.

Trajectory Embedding Besides the gird embedding, it is also critical to embed the structure (or the shape) of trajectories to enhance the spatial connections of grids. We appeal to the graph convolutional network (GCN) (Kipf and Welling 2017) for this task, which is known for its extraordinary capability to embed spatial structures. We consider all grids in the map as nodes v_i to construct a graph $\mathtt{G}$ for GCN. In addition, two grids are regarded as adjacent, if they are directly connected in any trajectories in the dataset. Specifically, the edge set of $\mathtt{G}$ is $E = \{(v_i, v_j)\}$, where v_i and v_j (i.e., g_i and g_j) are directly connected in any trajectories.

With the graph $\mathtt{G}$ and the prompt grid embedding $P = [P_1, P_2, ..., P_m]$ from Eq. 8 for nodes, we have one GCN layer embedding as:

$$\begin{aligned} H &= \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}PW) \\ \tilde{A} &= A + I, \tilde{D}_{ii} = \sum\nolimits_j \tilde{A}_{ij}, \end{aligned} \tag{9}$$

where $A \in \mathbb{R}^{m\times m}$ is the adjacency matrix, I is an identity matrix, $W \in \mathbb{R}^{(u+k)\times(u+k)}$ is the weight matrix, and $\sigma(\cdot)$ is a nonlinear activation function.

The element of the adjacency matrix A is:

$$A_{ij} = \begin{cases} 1 & \text{if } (v_i, v_j) \in E, \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

Through GCN embedding and graph $\mathtt{G}$, the grid connection patterns in trajectories can be well embedded.

We finally proceed to embed the order of grids in trajectories. Since a trajectory is generated by a moving object, the order of its locations is a critical characteristic for matching trajectories. Among many existing sequence order embedding models, the gated recurrent unit (GRU) (Cho et al. 2014) is widely adopted due to its superiority. We thus utilize it to embed the trajectory grid order:

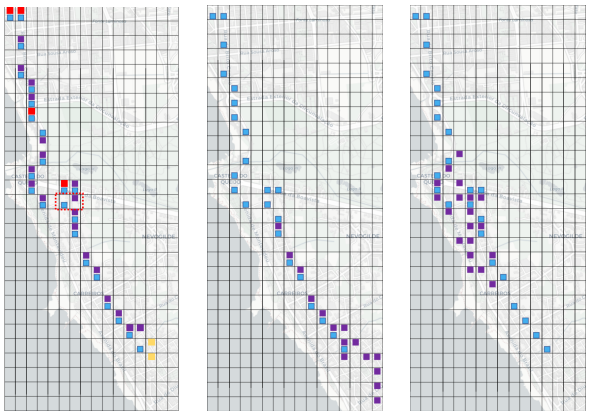
$$z = \mathtt{GRU}(H(T)|\Psi), \tag{11}$$

where $z \in \mathbb{R}^d$ is the last step of the GRU, d is the GRU embedding size and Ψ is the parameters of GRU. We take z as the final trajectory embedding.

Unsupervised Contrastive Similarity Learning

Given the trajectory embedding network, two trajectories are fed to achieve their embeddings, which are then used to calculate a particular similarity measure. Existing approaches need to compute labels to conduct supervised learning during preprocessing. This learning scheme suffers from two limitations: (1) the preprocessing for generating supervision labels is costly (the computation complexity is quadratic to the number and length of trajectories); (2) the training set cannot well contain adequate cases where two trajectories are highly similar, so the resulting similarity computation model has poor generality.

In light of the above limitations and the recent success of the SimCSE model (Gao, Yao, and Chen 2021), we employ unsupervised contrastive learning for training the prompt trajectory embedding module. The basic idea of contrastive learning is to minimize the distance between similar samples while maximizing the distance between dissimilar samples.



(a) (b) (c)

Figure 2: Examples of the positive sample generation strategies. (a) whole trajectory strategy; (b) partial trajectory strategy-end; (c) partial trajectory strategy-mid.

We adopt the following infoNCE loss (Oord, Li, and Vinyals 2018) for training:

$$\mathcal{L}_i = -\mathop{\mathbb{E}}_{M}\left[\log \frac{e^{sim(z_i,z_i^+)/\tau}}{e^{sim(z_i,z_i^+)/\tau}+\sum_{j=1}^{N} e^{sim(z_i,z_j^-)/\tau}}\right], \quad (12)$$

where $\mathcal{L}_i$ is the loss for the trajectory T_i, z_i is the embedding of the trajectory T_i, z_i^+ and z_i^- are embeddings of the *positive samples* T_i^+ (i.e., similar trajectories to T_i) and the *negative samples* T_i^- (i.e., dissimilar trajectories to T_i), respectively, τ is a hyperparameter that controls the convergence speed, and $sim(\cdot,\cdot)$ is the similarity computation function (e.g., Euclidean distance computation). There are N negative samples and M positive samples for each trajectory.

As shown in Figure 1, a trajectory and its positive/negative samples are fed into the KGTS encoder to compute their similarities and thus the loss in Eq. 12. We use the other trajectories in a training batch other than T_i as its negative samples; we argue that the trajectories in the training set are not able to properly cover enough cases where two trajectories are highly similar, so we suggest creative strategies (as detailed in next section) to generate positive samples from the original trajectories.

Positive Sample Generation This task is to create similar trajectories to a base trajectory. Existing studies only use simple operations, such as randomly dropping grids from the base trajectory (Li et al. 2018), while the positive samples generated by these simple operations are nearly identical to the base trajectory, narrowing down the possible cases of similar trajectories. For example, two trajectories are actually similar when they have identical structures but are on adjacent paths; however, this case cannot be simulated and identified by existing solutions.

Before introducing our new strategies of positive sample generation, we first present the basic operations used to implement the strategies. These operations include *copy* which duplicates a grid at the same position, *add* which creates a new grid to one of the eight immediately neighbouring positions of a base grid, *delete* which deletes a grid, and *move* which moves a grid to one of its eight immediate neighbours.

With the above basic operations, we now present the proposed positive sample generation strategies as follows. The basic operations are denoted as *italic*. We present exemplar trajectories generated by the strategies in Figure 2.

- **Whole trajectory strategy.** We use the whole trajectory as the base to generate positive samples. There are three steps: (1) we first *copy* base grids to create a new grid sequence (purple grids in Figure 2a) which completely overlaps the original one (the blue grids in Figure 2a); (2) we then *delete* a random amount of consecutive grids (red grids in the upper left corner in Figure 2a) on one end of the new grid sequence and *add* the same number of consecutive grids (yellow grids in the lower right corner in Figure 2a) on the other end; (3) finally, we randomly *move* (grids in the dashed rectangle in Figure 2a) and *delete* several grids (the red grids in the middle of the trajectory in Figure 2a). This strategy can generate positive samples with minimized dissimilarities to the original trajectory.
- **Partial trajectory strategy-end.** In addition to generating positive samples that are similar to the entire base trajectory, we also generate positive samples that are only similar to part of the base trajectory. As shown in Figure 2b, a base sequence is first divided into three equal parts, then we apply the **whole trajectory strategy** to one of the two end parts to generate positive samples, but there are no *delete* operations in the second step to keep the positive samples having a comparable length to the base trajectory.
- **Partial trajectory strategy-mid.** To create the positive samples that are on adjacent paths of the base trajectory, we apply the *move* operation to the grids of the middle one of the three aliquots of the base trajectory (Figure 2c). Then, we apply the **whole trajectory strategy** without the *delete* operation in the second step to the new sequence of grids to enhance the diversity of positive samples.

Overall Training Process

There are two modules in KGTS that are trained in two unsupervised learning phases, respectively. The first module is the GRot for grid embedding with trainable hidden vectors $\{\Phi_i\}_{i=1}^m$ and relation embedding parameter Θ. These parameters are trained using loss function Eq. 6 and are fixed to produce grid embedding $\{\tilde{\Phi}_i\}_{i=1}^m$ for the subsequent trajectory embedding. Next, we train the prompt trajectory embedding module using the loss function Eq. 12 to learn the prompt vector U, weight matrix W_*, and GRU parameters Ψ. After fully training KGTS, the embedding of a trajectory can be achieved through Eq. 11. Note that the trajectory embedding can be used for different similarity measures and we choose cosine similarity in this work.

Experiments

Experimental Setup

Dataset We adopt two popular large benchmark datasets for trajectory analysis in our experiments, namely GeoLife

and Porto. (1) The GeoLife dataset (Zheng, Xie, and Ma 2010) contains trajectories recorded over five years from 182 users in the city of Beijing, China. The time interval between two successive points is about five seconds. We discard trajectories under 2km and cut the trajectories above 5km as done for the Porto dataset. (2) The Porto dataset (Moreira-Matias et al. 2016) contains trajectories recorded over one year from 442 taxis in the city of Porto, Portugal. The time interval between two consecutive points is 15 seconds. Following conventional research (Yao et al. 2019), we discard trajectories under 2km and randomly cut the trajectories above 5km into trajectories between 2km and 5km long. For both datasets, we randomly chose trajectories to keep the training, validation, and test ratio approximately as 1:1:1.

Implementation Details We divide the geographical space into 1000×1300 grids for the GeoLife dataset and 140×280 grids for the Porto dataset. The interval of the latitude and longitude are both 0.001. We first train the GRot module using the loss function in Eq. 6 to obtain the grid embedding. The margin γ in Eq. 6 is set to 12. We then train the trajectory embedding module using unsupervised contrastive learning with the loss function in Eq. 12. The hyperparameter τ in Eq. 12 is set to 0.05. Both phases are trained with the Adam optimizer and a learning rate of 0.0001. All experiments are conducted with GeForce RTX 3090 GPU.

Evaluation Metrics Following existing studies (Han et al. 2021), we use the Top-k hitting ratio to measure the performance of trajectory similarity computation,

$$HR@K = \frac{1}{\mid D^t \mid} \sum_{\tau \in D^t} \frac{\mid L_\tau^P@K \bigcap L_\tau^R \mid}{\mid L_\tau^R \mid} \tag{13}$$

where τ represents a specific trajectory; D^t is the test set; L_τ^R is the set of the Top-k similar trajectories in the training set for the given trajectory τ, and $L_\tau^P@K$ is the set of the Top-k similar trajectories predicted by our approach for τ.

Baselines We compare our method with the following approaches:(1) SRN (Pei, Tax, and van der Maaten 2016) uses a Siamese recurrent network for sequence similarity computation. (2) t2vec (Li et al. 2018) is also an unsupervised approach, which adopts a seq2seq model and a customized positive sample generation method for unsupervised training. (3) NeuTraj (Yao et al. 2019) uses a spatial attention memory unit to train the similarity computation model. (4) T3S (Yang et al. 2021) jointly considers both coordinates and grid entities to train the trajectory embedding model. (5) GTS (Han et al. 2021) uses a modified skip-gram module and a GNN for improved POI embedding; an LSTM is also used to embed the trajectory points order information. (6) CL-TSim (Deng et al. 2022) also adopts the skip-gram module and LSTM to gather grid sequences' information with contrastive learning to narrow similar trajectories' representation. (7) TMN (Yang et al. 2022b) combines points across trajectories and considers not only individual trajectory sequence information, but also interaction between trajectories to reach state-of-the-art.

Dataset	Method	HR@1	HR@5	HR@10
GeoLife	SRN	0.3363	0.4257	0.4624
	t2vec	0.363	0.3761	0.3184
	NeuTraj	0.4113	0.53	0.5823
	T3S	0.4273	<u>0.5362</u>	<u>0.5843</u>
	GTS	<u>0.4327</u>	0.4962	0.5102
	CL-TSim	0.3233	0.3448	0.3027
	TMN	0.4290	0.5092	0.5517
	KGTS (Ours)	**0.5123**	**0.579**	**0.5942**
Porto	SRN	0.3503	0.5079	0.5606
	t2vec	<u>0.4105</u>	0.5056	0.5138
	NeuTraj	0.4103	0.5465	0.591
	T3S	0.3923	<u>0.5506</u>	<u>0.6109</u>
	GTS	0.3987	0.5269	0.578
	CL-TSim	0.2973	0.3347	0.3379
	TMN	0.401	0.5263	0.586
	KGTS (Ours)	**0.5244**	**0.6277**	**0.6542**

Table 1: Comparison results on the GeoLife and Porto datasets. HR@k denotes the Top-k hitting rate.

Experimental Results

Overal Performance The comparison results are summarized in Table 1 with the best results shown in **bold** and the second best results shown in <u>underline</u>. It is demonstrated that our KGTS remarkably outperforms all baselines. We note that KGTS outperforms the baselines more significantly on smaller Top-k hit rates. This is due to the fact that our positive sample generation strategies for contrastive learning can properly cover enough cases of highly similar trajectories, while existing approaches can only learn from datasets that do not contain diverse cases of similar trajectories.

Compared with other models, we find that although some baselines (i.e., SRN, NeuTraj, T3S, and TMN) use powerful modules to capture the trajectory structures, they do not have specific modules for trajectory point embedding. By contrast, our KGTS has the GRot to well embed the trajectory grids and uses a prompt trajectory embedding to properly incorporate grid embedding into the subsequent trajectory structure and the grid order embedding.

Similarly, the t2vec and CL-TSim also use the same unsupervised learning scheme as KGTS. However, their positive sample generation strategy only considers the cases of mostly overlap. This strategy is quite limited, as it is common that trajectories are partially similar in real datasets. In contrast, we suggest three positive sample generation strategies that cover more general cases of similar trajectories.

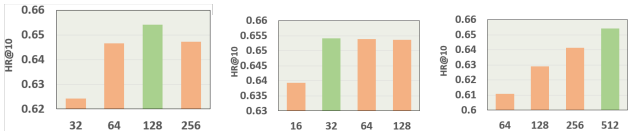
Besides the better performance of trajectory similarity computation, our method uses zero time to prepare supervision labels, while supervised learning baselines take around one hour per 5,000 trajectories and the cost increases quadratically with the number and length of trajectories.

Ablation Study We perform an ablation study to justify our design choices for KGTS. In particular, we compare KGTS with the following variants: (1) w/o GRot: it does not use the proposed GRot model for grid embedding. (2) w/ original RotatE: it utilizes the original RotatE instead of

Figure 3: Visualization of the case studies. The red trajectories are the query trajectories; the blue trajectories are the Top-3 ground truth similar trajectories; the black trajectories are the Top-3 similar trajectories found by KGTS.

Method	HR@1	HR@5	HR@10
w/o GRot	0.3243	0.4211	0.4468
w/ original RotatE	0.5067	0.6151	0.6445
w/o prompt	0.5096	0.6227	0.6514
w/o GCN	0.4853	0.5951	0.6304
w/o GRU	0.152	0.2231	0.2553
KGTS (ours)	**0.5244**	**0.6277**	**0.6542**

Table 2: Ablation study on the Porto dataset.



(a) dimension of k (b) dimension of u (c) dimension of d

Figure 4: Hyperparameter sensitivity analysis.

our proposed modified one to study its effectiveness. (3) w/o prompt: it does not employ a prompt to instruct the trajectory embedding to incorporate grid embedding. The grid embedding $\tilde{\Phi}_i$ is directly input into the trajectory embedding module. (4) w/o GCN: it does not apply the GCN to trajectory structure embedding, and the prompt grid embedding is directly fed into the GRU. (5) w/o GRU: it does not consider the order of points in a trajectory. The embeddings from the GCN are summed up to form a single trajectory embedding.

Table 2 reports the ablation study results. We make the following observations: (1) The *w/o GRot* variant is lack of calculation of inter grid relationships, so its effectiveness is relatively low. (2) KGTS is better than the *w/ original RotatE* variant demonstrating the superior capability of the GRot. (3) Prompt learning is originally proposed to adapt a pre-trained network to downstream tasks with the help of a prompt. We adopt this idea to better utilize grid embedding for the final trajectory embedding, and the results that $w/oprompt$ is inferior to KGTS well justify our design. (4) The *w/o GCN* variant shows superior performance to the *w/o GRot* variant. This suggests that properly embedding the grids in the space is more important. Thus, our GRot module for grid embedding is an essential innovation. (5) The *w/o GRU* variant is the worst variant with considerably lower hitting rates. This draws two conclusions: the trajectory point order information is critical to the trajectory embedding; simply adding up the embedding of each grid in a trajectory into a single embedding is far from enough to produce a favourable trajectory embedding.

Case Study We randomly choose one query trajectory from the test set and find its similar trajectories from the training set using our KGTS. We visualize the obtained Top-3 trajectories from KGTS and ground truth Top-3 similar trajectories in Figure 3. For the query trajectory T_{3111} (in Figure 3), our approach successfully finds all the Top-3 similar trajectories, although there is a slight deviation between the order of the three trajectories and the ground truth. This still indicates that under unsupervised conditions, the similarity between trajectories can be accurately calculated.

Hyperparameter Sensitivity Analysis Three hyperparameters influence our KGTS performance the most: grid embedding dimension k, prompt dimension u, and trajectory embedding dimension d. We show the parameter sensitivity analysis on the Proto dataset in Figure 4. We observe that the hitting rate increases as the grid embedding dimension k grows from 32 to 128 and decreases at 256. A larger embedding dimension can embed more information, while a too-large value would make it overfitting and thus poorly generalised. The hitting rate rises sharply as prompt dimension u doubles from 16 to 32 and decreases slightly afterwards. This manifests the importance of the prompt scheme for trajectory embedding. For the trajectory embedding dimension d, the hitting rate increases straightly as it climbs from 64 to 512. We do not try larger dimensions due to exponentially increasing computational costs. The best trajectory embedding dimension (i.e., 512) is larger than the best grid embedding dimension (i.e., 128). This is due to the fact that the trajectory embedding needs to encompass richer information than the grid embedding.

Conclusion

In this paper, we target the trajectory similarity computation task and manage to mitigate the limitations of current deep learning solutions. We propose KGTS which has a modified RotatE module for grid embedding and a prompt trajectory embedding module for the final trajectory embedding. Furthermore, we present three novel positive sample generation strategies for unsupervised contrastive trajectory embedding learning, which can well cover extensive cases of highly similar trajectories. Thus, the proposed approach manifests improved generality and does not require the costly preprocessing for generating supervision labels as done by existing deep learning solutions. Extensive experiments on two benchmark datasets well demonstrate the effectiveness of each KGTS component.

Acknowledgments

This work was supported by the NSFC (U2001212, U22B2037, U21B2046, 62032001, and 61932004).

References

Atev, S.; Miller, G.; and Papanikolopoulos, N. P. 2010. Clustering of Vehicle Trajectories. *IEEE Transactions on Intelligent Transportation Systems*, 11(3): 647–657.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901.

Chang, S. Y.; Wu, H.-C.; Kuan, Y.-C.; and Wu, Y. 2023. Tensor Levenberg-Marquardt Algorithm for Multi-Relational Traffic Prediction. *IEEE Transactions on Vehicular Technology*, 1–17.

Chen, L.; and Ng, R. T. 2004. On The Marriage of Lp-norms and Edit Distance. In *(e)Proceedings of the Thirtieth International Conference on Very Large Data Bases, VLDB 2004, Toronto, Canada, August 31 - September 3 2004*, 792–803. Morgan Kaufmann.

Chen, L.; Özsu, M. T.; and Oria, V. 2005. Robust and Fast Similarity Search for Moving Object Trajectories. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, Baltimore, Maryland, USA, June 14-16, 2005*, 491–502. ACM.

Cho, K.; van Merrienboer, B.; Gülçehre, Ç.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, 1724–1734. ACL.

Deng, L.; Zhao, Y.; Fu, Z.; Sun, H.; Liu, S.; and Zheng, K. 2022. Efficient Trajectory Similarity Computation with Contrastive Learning. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, CIKM '22, 365–374. New York, NY, USA: Association for Computing Machinery. ISBN 9781450392365.

Evans, M. R.; Oliver, D.; Shekhar, S.; and Harvey, F. 2013. Fast and exact network trajectory similarity computation: a case-study on bicycle corridor planning. In *Proceedings of the 2nd ACM SIGKDD International Workshop on Urban Computing, UrbComp@KDD 2013, Chicago, Illinois, USA, August 11, 2013*, 9:1–9:8. ACM.

Gao, T.; Fisch, A.; and Chen, D. 2021. Making Pre-trained Language Models Better Few-shot Learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP*, 3816–3830. Association for Computational Linguistics.

Gao, T.; Yao, X.; and Chen, D. 2021. SimCSE: Simple Contrastive Learning of Sentence Embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, 6894–6910. Association for Computational Linguistics.

Han, P.; Wang, J.; Yao, D.; Shang, S.; and Zhang, X. 2021. A Graph-based Approach for Trajectory Similarity Computation in Spatial Networks. In *KDD '21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Singapore, August 14-18, 2021*, 556–564. ACM.

Hjelm, R. D.; Fedorov, A.; Lavoie-Marchildon, S.; Grewal, K.; Bachman, P.; Trischler, A.; and Bengio, Y. 2019. Learning deep representations by mutual information estimation and maximization.

Huang, X.; Zhang, J.; Li, D.; and Li, P. 2019. Knowledge Graph Embedding Based Question Answering. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, WSDM 2019, Melbourne, VIC, Australia, February 11-15, 2019*, 105–113. ACM.

Ji, S.; Pan, S.; Cambria, E.; Marttinen, P.; and Philip, S. Y. 2021. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Transactions on Neural Networks and Learning Systems*, 33(2): 494–514.

Jiang, Z.; Xu, F. F.; Araki, J.; and Neubig, G. 2020. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8: 423–438.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks.

LeCun, Y.; Bengio, Y.; and Hinton, G. 2015. Deep learning. *nature*, 521(7553): 436–444.

Li, F.; Feng, J.; Yan, H.; Jin, G.; Yang, F.; Sun, F.; Jin, D.; and Li, Y. 2023. Dynamic Graph Convolutional Recurrent Network for Traffic Prediction: Benchmark and Solution. *ACM Trans. Knowl. Discov. Data*, 17(1).

Li, X.; Zhao, K.; Cong, G.; Jensen, C. S.; and Wei, W. 2018. Deep Representation Learning for Trajectory Similarity Computation. In *34th IEEE International Conference on Data Engineering, ICDE 2018, Paris, France, April 16-19, 2018*, 617–628. IEEE Computer Society.

Li, Z.; Han, J.; Ji, M.; Tang, L.-A.; Yu, Y.; Ding, B.; Lee, J.-G.; and Kays, R. 2011. Movemine: Mining Moving Object Data for Discovery of Animal Movement Patterns. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(4): 1–32.

Lin, J.; Li, Z.; Li, Z.; Bai, L.; Zhao, R.; and Zhang, C. 2023. Dynamic Causal Graph Convolutional Network for Traffic Prediction. arXiv:2306.07019.

Liu, P.; Yuan, W.; Fu, J.; Jiang, Z.; Hayashi, H.; and Neubig, G. 2021. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing.

Moreira-Matias, L.; Gama, J.; Ferreira, M.; Mendes-Moreira, J.; and Damas, L. 2016. Time-evolving O-D matrix estimation using high-speed GPS data streams. *Expert Systems and Applications*, 44: 275–288.

Oord, A. v. d.; Li, Y.; and Vinyals, O. 2018. Representation Learning with Contrastive Predictive Coding.

Pei, W.; Tax, D. M.; and van der Maaten, L. 2016. Modeling Time Series Similarity with Siamese Recurrent Networks.

Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; Liu, P. J.; et al. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140): 1–67.

Ranu, S.; P, D.; Telang, A. D.; Deshpande, P.; and Raghavan, S. 2015. Indexing and matching trajectories under inconsistent sampling rates. In *31st IEEE International Conference on Data Engineering, ICDE 2015, Seoul, South Korea, April 13-17, 2015*, 999–1010. IEEE Computer Society.

Rethmeier, N.; and Augenstein, I. 2021. A Primer on Contrastive Pretraining in Language Processing: Methods, Lessons Learned & Perspectives.

Song, R.; Sun, W.; Zheng, B.; and Zheng, Y. 2014. PRESS: A Novel Framework of Trajectory Compression in Road Networks. *Proceedings of the VLDB Endowment: 40th VLDB*, 7: 661–672.

Sun, C.; Baradel, F.; Murphy, K.; and Schmid, C. 2019a. Learning video representations using contrastive bidirectional transformer.

Sun, Z.; Deng, Z.; Nie, J.; and Tang, J. 2019b. RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space.

Sun, Z.; Yang, J.; Zhang, J.; Bozzon, A.; Huang, L.; and Xu, C. 2018. Recurrent knowledge graph embedding for effective recommendation. In *Proceedings of the 12th ACM Conference on Recommender Systems, RecSys 2018, Vancouver, BC, Canada, October 2-7, 2018*, 297–305. ACM.

van Kreveld, M. J.; and Luo, J. 2007. The definition and computation of trajectory and subtrajectory similarity. In *15th ACM International Symposium on Geographic Information Systems, ACM-GIS 2007, November 7-9, 2007, Seattle, Washington, USA, Proceedings*, 44. ACM.

Vlachos, M.; Gunopulos, D.; and Kollios, G. 2002. Discovering Similar Multidimensional Trajectories. In *Proceedings of the 18th International Conference on Data Engineering, San Jose, CA, USA, February 26 - March 1, 2002*, 673–684. IEEE Computer Society.

Yang, J.; Li, C.; Zhang, P.; Xiao, B.; Liu, C.; Yuan, L.; and Gao, J. 2022a. Unified Contrastive Learning in Image-Text-Label Space. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, 19141–19151. IEEE.

Yang, P.; Wang, H.; Lian, D.; Zhang, Y.; Qin, L.; and Zhang, W. 2022b. TMN: Trajectory Matching Networks for Predicting Similarity. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, 1700–1713.

Yang, P.; Wang, H.; Zhang, Y.; Qin, L.; Zhang, W.; and Lin, X. 2021. T3S: Effective Representation Learning for Trajectory Similarity Computation. In *37th IEEE International Conference on Data Engineering, ICDE 2021, Chania, Greece, April 19-22, 2021*, 2183–2188. IEEE.

Yao, D.; Cong, G.; Zhang, C.; and Bi, J. 2019. Computing Trajectory Similarity in Linear Time: A Generic Seed-Guided Neural Metric Learning Approach. In *35th IEEE International Conference on Data Engineering, ICDE 2019, Macao, China, April 8-11, 2019*, 1358–1369. IEEE.

Yi, B.; Jagadish, H. V.; and Faloutsos, C. 1998. Efficient Retrieval of Similar Time Sequences Under Time Warping. In *Proceedings of the Fourteenth International Conference on Data Engineering, Orlando, Florida, USA, February 23-27, 1998*, 201–208. IEEE Computer Society.

Yu, J.; Yin, H.; Xia, X.; Chen, T.; Cui, L.; and Nguyen, Q. V. H. 2022. Are Graph Augmentations Necessary?: Simple Graph Contrastive Learning for Recommendation. In *SIGIR '22: The 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, Madrid, Spain, July 11 - 15, 2022*, 1294–1303. ACM.

Zhang, D.; Li, N.; Zhou, Z.; Chen, C.; Sun, L.; and Li, S. 2011. iBAT: detecting anomalous taxi trajectories from GPS traces. In *UbiComp 2011: Ubiquitous Computing, 13th International Conference, UbiComp 2011, Beijing, China, September 17-21, 2011, Proceedings*, 99–108. ACM.

Zheng, Y.; Xie, X.; and Ma, W. 2010. GeoLife: A Collaborative Social Networking Service among User, Location, and Trajectory. *IEEE Data Engineering Bulletin*, 33(2): 32–39.