TSNE: Trajectory Similarity Network Embedding

Jiaxin Ding*
jiaxinding@sjtu.edu.cn
Shanghai Jiao Tong University

Bowen Zhang*
zzljoy3091498@sjtu.edu.cn
Shanghai Jiao Tong University

Xinbing Wang
xwang8@sjtu.edu.cn
Shanghai Jiao Tong University

Chenghu Zhou
zhouch@lreis.ac.cn
Chinese Academy of Sciences

ABSTRACT

Trajectory representation learning studies the problem of embedding trajectories into low-dimensional vectors, while preserving mutual similarity for the convenience of downstream tasks, such as nearest neighbor search, clustering, classification, etc. In this work, we propose the Trajectory Similarity Network Embedding (TSNE) which exploits representation learning on the k-nearest neighbor partial similarity graph to generate trajectory embeddings, that preserve different similarity efficiently. In theory, we prove that TSNE is equivalent to factorizing the similarity graph, while in practice, TSNE achieves better performance. In the experiment, we show that TSNE outperforms the state-of-the-art baselines, including matrix factorization approaches and RNN based models in terms of similarity preserving and dimension reduction.

CCS CONCEPTS

• **Computing methodologies** → **Learning latent representations**.

KEYWORDS

Spatio-temporal data embedding, network embedding, similarity computation

ACM Reference Format:
Jiaxin Ding, Bowen Zhang, Xinbing Wang, and Chenghu Zhou. 2022. TSNE: Trajectory Similarity Network Embedding. In *The 30th International Conference on Advances in Geographic Information Systems (SIGSPATIAL '22), November 1–4, 2022, Seattle, WA, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3557915.3561022

1 INTRODUCTION

With the proliferation of the Internet of Things and mobile networks, high volume of spatio-temporal trajectory data is generated and collected, nowadays. It provides us unprecedented opportunities to study human mobility patterns to improve our daily lives, such as traffic monitoring [3], route planning [10], tourism planning [9], etc.

However, trajectories are typically collected heterogeneously, with nonuniform sampling rates and different sampling lengths commonly with data missing and errors, and usually organized in various heterogeneous structures, which adds difficulties to the downstream tasks. Besides, the similarities or distances between trajectories, the key trajectory features, mostly require quadratic computational complexity, including Hausdorff distance, Fréchet distance, Dynamic Time Warping (DTW), Edit distance with Real Penalty (ERP) [1], Edit Distance on Real sequence (EDR) [2], etc. Such computational demands make it a hurdle to process large scale trajectory data. Furthermore, most of the trajectory distances are sensitive to sampling rates and outliers. For example, Hausdorff distance is measured by the maximum of all pairwise distances on two trajectories, Fréchet distance is measured by the min-max distance of the traversal along two trajectories, while Dynamic Time Warping is measured by the min-sum distance of the traversal. Such distances are *extreme* measures and capture the worst-case scenario. Given two real world continuous trajectories, these distances will change, and even become totally different, when the sampling rates, lengths of the two trajectories are changed or errors, sometimes with extreme values, are introduced. In summary, measuring distances over trajectories with heterogeneity is challenging for large scale trajectory datasets.

In face of the above challenges, trajectory representation learning is proposed and developed, which embeds trajectories into low-dimensional latent vector space and the similarity between trajectories is thereby measured with the Euclidean distance or cosine distance between the embedding vectors, reducing the quadratic computation complexity to linear. Besides, learning representation vectors of input trajectories can be leveraged to facilitating downstream data mining and machine learning tasks, such as nearest neighbor search, clustering, prediction and so on [14, 5], making trajectory representation learning an important research area in spatio-temporal data mining. In the trajectory representation learning for similarity computation, Recurrent Neural Network (RNN) based approaches [6, 12, 13] are proposed to approximate the traditional distance metrics. However, the RNN based approaches encode sequences with different lengths into the same pre-defined low-dimensional latent space, which results in accuracy loss accumulation as sequences grow longer. Moreover, the similarity between all pairs of trajectories in the training data must be computed as supervised labels in the training process, which is time consuming and meanwhile only a small fraction of distances among the similarity pairs of trajectories are actually taken into account.

All above, in this work, we propose trajectory similarity network embedding (TSNE), to embed trajectories into low-dimensional

*Both authors contributed equally to this research.

SIGSPATIAL '22, November 1–4, 2022, Seattle, WA, USA

ACM ISBN 978-1-4503-9529-8/22/11.
https://doi.org/10.1145/3557915.3561022

vectors with graph representation learning on k-nearest neighbor graph (k-NNG) defined by our proposed partial similarity measures. k-NNG is a graph structure that keeps the top-k nearest neighbors by the given metric for each node. With the principle of homophily, information more than the k-nearest neighbors can be inferred. The similarity for k-NNG is our proposed partial similarity, which measures the proportion of locations within the same region between two trajectories in both ordered and unordered ways. Based on the k-NNG, we propose our TSNE and prove its equivalence to implicitly factorizing the k-nearest neighbor similarity graph matrix, which interprets the feature representations extracted by TSNE. Through the learnable parameters, our approach can automatically adjust the importance of neighbor embedding and infer the similarity of non-neighbor nodes.

Our contributions can be summarized as follows:

- To the best of our knowledge, we are the first to study the trajectory representation with the mobility similarity graph. We introduce the k-NNG to further reduce the computation cost to make our approach efficient in the training process. The graph can be easily applied to different metrics by changing the edges, our proposed framework can be applied to any similarity measure.
- We propose TSNE, an efficient trajectory representation model. TSNE has a random-walk based representation learning with better interpretability.
- We have conducted extensive experiments over proposed measure and real-world datasets, demonstrating that our TSNE outperforms the state-of-the-art techniques in terms of preserving similarity and dimension reduction.

2 PRELIMINARY

Given a trajectory dataset $\mathcal{T}$, each trajectory $T \in \mathcal{T}$ is a sequence of location points, represented as

$$T = \{P_1, P_2, \cdots, P_m\}, \tag{1}$$

where $P_i \in \mathbb{R}^2$ is the coordinates of the location sampled on the trajectory. The location points in T are sorted in chronological order. For any trajectory pair (T_i, T_j), we apply a function $f(T_i, T_j)$ to measure the similarity between T_i and T_j, and the problem that we deal with can be formulated as follows.

2.1 Problem Definition

Given a trajectory dataset $\mathcal{T}$ and a similarity function $f(\cdot,\cdot)$, our objective is to obtain an embedding $\Phi : T \to \mathbb{R}^d$ that embeds trajectory T into a d-dimensional vector, such that

$$\min_{\Phi} \left| g\big(\Phi(T_i), \Phi(T_j)\big) - f(T_i, T_j) \right|,$$

for any T_i and T_j, where $g(\cdot,\cdot)$ is a similarity that measures the similarity between two d-dimensional vectors.

3 PARTIAL SIMILARITY

In this section, we propose a new category of trajectory similarity named *partial similarity*. The motivation for partial similarity is to avoid the measure being overly sensitive to outliers. We mainly discuss two types of partial similarity: **Ordered partial similarity** and **Unordered partial similarity**.

First, consider two distinct trajectories T_1 and T_2. We define Ordered partial similarity based on the length of common subsequence $|CS(T_1, T_2)|$.

Definition 1 (Ordered partial similarity). *Given two trajectories T_1 and T_2, the Ordered partial similarity $s_o(T_1, T_2) = \frac{2|CS(T_1,T_2)|}{l_1+l_2}$, where l_1 and l_2 is the length of T_1 and T_2, respectively.*

Next, we relax the sequential order to consider only the points that both trajectories arrived, termed Unordered partial similarity. We denote the number of the common points as $|CP(T_1, T_2)|$.

Definition 2 (Unordered partial similarity). *Given two trajectories T_1 and T_2, the Unordered partial similarity $s_u(T_1, T_2) = \frac{2|CP(T_1,T_2)|}{l_1+l_2}$, where l_1 and l_2 is the length of T_1 and T_2, respectively.*

To improve the tolerance of partial similarity to noise and reduce computational cost, we convert the original trajectories into sequences of grid cells by discretizing the search area. The location points in the same grid is considered to be geographical close.

Further, in the sensing process, it is ordinary to miss some points or have some noise points in the trajectory. Thus, we propose a random downsampling sketch to simulate the practical situation and improve the robustness.

4 TRAJECTORY SIMILARITY NETWORK EMBEDDING

In this section, we present an overview of our model, Trajectory Similarity Network Embedding (TSNE). Figure 1 illustrates the framework of TSNE. We first construct k-NNG for the trajectory dataset, where each trajectory is represented as a vertex in the graph, and each edge reflects the k-nearest neighbor (k-NN) relationship between the corresponding trajectories. Thus, we bridge the gap between trajectory similarity and graph topology, and convert the trajectory representation problem into the task of graph embedding. Thereafter, we adapt graph embedding methods to trajectory representation learning.

4.1 k-Nearest Neighbor Graph Construction

Taking advantage of methods in [4] with near linear computation complexity, we construct the k-NNG, denoted as $G(V, E, S)$, which contains three components:

- V is the vertex set, where each vertex $v_i \in V$ represents the trajectory $T_i \in \mathcal{T}$;
- E is the edge set, if there is an edge $e_{ij} \in E$, T_j is one of the k-nearest neighbors of T_i;
- $S = (s_{ij})_{|V|\times|V|}$ is the weighted adjacent matrix of the k-NNG, where the weight s_{ij} is computed by Equation 2, if T_j the k-nearest neighbor of T_i.

We denote $\mathcal{K}_{T_i}$ to be the k-nearest neighbors of T_i.

$$s_{ij} = \frac{\exp(s(T_i, T_j))}{\sum_{T_{j'} \in \mathcal{K}_{T_i}} \exp(s(T_i, T_{j'}))}, \tag{2}$$

After constructing k-NNG, we will learn a representation model that map each vertex $v \in V$ into a d-dimensional representation vector $h \in \mathbb{R}^d$. We propose a random-walk based graph embedding model: *TSNE*.

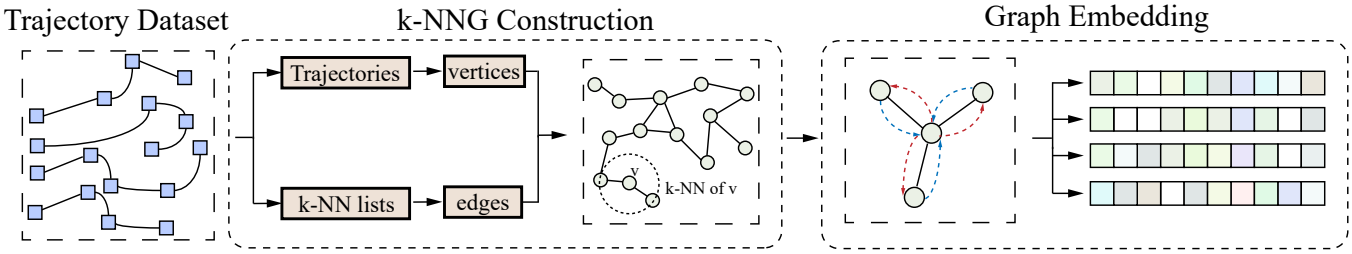

Figure 1: The framework of TSNE.

4.2 TSNE Algorithm

We propose a trajectory embedding model *TSNE*, which exploits random walk to explore the topology of k-NNG, and apply Skip-Gram with Negative Sampling (SGNS) [7] on the random walk sequences to obtain the embeddings. We also prove that the TSNE is equivalent to a matrix factorization of the similarity adjacent matrix.

As is proposed in [7], we can randomly sample b nodes as"negative" samples, denoted as $\mathcal{D}$, to normalize the probability computation. Considering all the vertices, the overall objective becomes finding a set of model parameters Φ that minimizes

$$J(\Phi) = \sum_{v \in V} \sum_{u \in \mathcal{N}_v} [-\log C_b + \Phi(u)^T \Phi(v)]. \tag{3}$$

Where $C_b = \sum_{r \in \mathcal{D}} \exp[\Phi(r)^T \Phi(v)]$, related to negative sampling strategy.

4.2.1 Interpretation of TSNE. Now we give the theorem that the TSNE is equivalent to factorizing the similarity adjacent matrix S.

Theorem 1. *Assume that we perform unbiased random walk on an undirected weighted graph $G = (V, E, S)$ to generate the neighbor sets of vertices and run SGNS to obtain the embedding. When the length of the random walk tends to infinity , it is equivalent to the following factorization:*

$$XY^T = \log \frac{vol(G)(\frac{1}{w}\sum_{r=1}^{w}(D^{-1}S)^r)D^{-1})}{b},$$

where $X, Y \in \mathbb{R}^{n \times d}$, $\log(\cdot)$ represents the element-wise logarithmic operation in the matrix, b is the number of negative samples, D is a diagonal matrix, $D_{ii} = \sum_{j=1}^{|V|} s_{ij}$, $vol(G) = \sum_{i=1}^{|V|}\sum_{j=1}^{|V|} s_{ij}$, w is the window size for two vertices on the random walk to be considered as neighbors.

Theorem 2. *Assume that we perform TSNE on $G(V, E, S)$, where S is a normalized similarity matrix computed by Equation 2. When the length of the random walk tends to infinity and the window size $w = 1$, TSNE is equivalent to the following factorization:*

$$XY^T = \log \frac{|V|}{b} S,$$

where $X, Y \in \mathbb{R}^{n \times d}$, b is the number of negative samples.

Theorem 2 addresses that TSNE is implicitly factorizing the logarithmic of the similarity adjacent matrix S. Since the positive entries of S are exponents of the similarities, after taking the logarithmic back, TSNE is actually factorizing the similarity matrix of k-NNG, while the computation complexity of TSNE is far lower than the traditional factorization methods, such as SVD. Therefore, the above theorem provides an interpretation of TSNE and implies the effectiveness and efficiency of our approach.

5 EXPERIMENT

In this section, we conduct extensive experiments to demonstrate the effectiveness and efficiency of the proposed model TSNE.

5.1 Experimental Settings

5.1.1 Dataset. Our experiments are based on a real-world open-source trajectory dataset **Geolife** [15]. It consists of 17, 621 trajectories, collected by 182 users, from 2007 to 2012 in Beijing, China.

We choose trajectories in the center area of the city and discretize the area into 200m × 200m grid cells. We remove all trajectories that are too sparse with less than 10 points, and obtain 10, 504 trajectories in Geolife.

5.1.2 Evaluation Settings. Following the state-of-the-art method [11], we evaluate the performance on the top-N similarity search problem, with two types of partial similarity. The top-N similarity search problem measures the success rate that we find top-N most similar trajectories with trajectory embeddings. We compute hitting ratio HR@10, HR@20, HR@50, HR@100 and recall R10@50 for performance evaluation, same as the setting in [11].

5.1.3 Baselines. We compare TSNE with five baselines, which can be divided into two categories:

- *Matrix factorization based methods*: **SVD** and **MDS**, which take the distance matrix as input and apply matrix factorization to construct low-dimensional trajectory embeddings.
- *RNN based methods*: **Siamese** [8], **NEUTRAJ** [12] and **T3S** [11] (state-of-the-art), which exploit RNN to capture the latent sequential information of trajectories We use the default configurations of these methods in the training.

5.1.4 Parameter Settings. We set $k = 20$ for the k-NNG. For **TSNE**, we conduct unbiased random walk with fixed length $l = 100$. The window size of SGNS is $w = 2$, and the number of negative samples is $b = 5$. The default setting of embedding dimension for all baseline methods is $d = 128$ to achieve the best performance, while we can achieve better performance with only $d = 16$ for TSNE.

	Ordered					Unordered				
Method	HR@10	HR@20	HR@50	HR@100	R10@50	HR@10	HR@20	HR@50	HR@100	R10@50
SVD	0.3466	0.3732	0.4457	0.4421	0.8991	0.3710	0.4002	0.4926	0.4878	0.8625
MDS	0.2799	0.3750	0.4025	0.4117	0.8660	0.3132	0.3936	0.4666	0.4710	0.8417
Siamese	0.2929	0.3698	0.4132	0.4296	0.7963	0.2933	0.3873	0.4219	0.4775	0.7766
NEUTRAJ	0.3118	0.3644	0.4449	0.4643	0.8300	0.3223	0.3879	0.4761	0.5081	0.8561
T3S	0.3512	0.4112	0.4781	0.5018	0.8677	0.3609	0.4365	0.4882	0.5343	0.8776
TSNE	**0.5241**	**0.6813**	**0.6379**	**0.5234**	**0.9578**	**0.5532**	**0.7307**	**0.6761**	**0.5966**	**0.9792**

Table 1: Performance comparison for different methods on Ordered and Unordered partial similarity.

5.2 Performance Comparison

Table 1 shows the performance of different methods for the top-N similarity search. Overall, TSNE model outperforms all the baselines. It shows that the topology of k-NNG retains the majority of original similarity information of trajectories, and it is effective to capture the topology information of k-NNG with graph embedding methods. Lastly, the embedding dimension of TSNE is much lower than the baselines (16 vs 128).

5.3 Parameter Sensitivity Study

We evaluate the sensitivity of TSNE on three parameters: the number of the utilized nearest neighbors k, the window size w, and the embedding dimension d. Firstly, to study the impact of true k-NN information on the performance of TSNE, we choose $k = \{10, 20, 50, 100\}$, and $HR@k$ with the same k as k-NNG achieves the best performance; the performance of $HR@2k$ only slightly declines, since the neighbors of a neighbor is still near. Secondly, the window size of SGNS w in TSNE affects the exploring field of the embedding process. The result shows that $w = 2$ achieves the best performance, which is much better than $w = 1$ just considering nearest neighbors; $w = 3, 4, 5, 6$ achieves slightly declined performance, which means that the neighbors of a neighbor is near, but the error also increases as the hop length of the path grows. Therefore, $w = 2$ is a balance between the similarity information explored and error introduced. Lastly, We set $d = \{8, 16, 32, 64, 128, 256\}$ to explore the impact of embedding dimension d on the performance of TSNE in the top-N similarity search. When d increases from 8 to 16, the performance increases greatly; after that, the performance remains almost the same or slightly declines as d increases. Intuitively, when the embedding dimension increases, the expressive capacity of the embedding space rises, allowing the embedding vectors to capture more latent features. When the dimension is too large, the model is likely to encounter an "overfitting" obstacle, resulting in performance loss. Therefore, we set the embedding dimension of TSNE in the experiment as $d = 16$.

6 CONCLUSION

We proposed a trajectory similarity network embedding (TSNE) model to embed trajectories into low-dimensional representation vectors, preserving similarity information. Specifically, we proposed to construct k-NNG to preserve partial similarity information of the k-nearest neighbors, and thus converted the trajectory embedding task into graph embedding task. Experiments on a real-world dataset show that TSNE achieves state-of-the-art performance in terms of similarity preserving, dimension reduction and efficiency improvement.

ACKNOWLEDGMENT

We would like to acknowledge supports from NSF China under Grant (No. 42050104, 62202299, 62020106005, 62041205, 61960206002, 61829201, 62041205), Shanghai Sailing Program 20YF1421300, Natural Science Foundation of Shanghai No. 22ZR1429100.

REFERENCES

[1] Lei Chen and Raymond Ng. 2004. On the marriage of lp-norms and edit distance. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, 792–803.

[2] Lei Chen, M Tamer Özsu, and Vincent Oria. 2005. Robust and fast similarity search for moving object trajectories. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, 491–502.

[3] Jiaxin Ding, Chien-Chun Ni, Mengyu Zhou, and Jie Gao. 2017. Minhash hierarchy for privacy preserving trajectory sensing and query. In *2017 16th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*. IEEE, 17–28.

[4] Wei Dong, Charikar Moses, and Kai Li. 2011. Efficient k-nearest neighbor graph construction for generic similarity measures. In *Proceedings of the 20th international conference on World wide web*, 577–586.

[5] Qiang Gao, Fan Zhou, Kunpeng Zhang, Goce Trajcevski, Xucheng Luo, and Fengli Zhang. 2017. Identifying human mobility via trajectory embeddings. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence* (IJCAI'17). AAAI Press, Melbourne, Australia, 1689–1695. ISBN: 9780999241103.

[6] Xiucheng Li, Kaiqi Zhao, Gao Cong, Christian S Jensen, and Wei Wei. 2018. Deep representation learning for trajectory similarity computation. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*. IEEE, 617–628.

[7] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, 3111–3119.

[8] Wenjie Pei, David MJ Tax, and Laurens van der Maaten. 2016. Modeling time series similarity with siamese recurrent networks. *arXiv preprint arXiv:1603.04713*.

[9] Sheng Wang, Mingzhao Li, Yipeng Zhang, Zhifeng Bao, David Alexander Tedjopurnomo, and Xiaolin Qin. 2018. Trip planning by an integrated search paradigm. In *Proceedings of the 2018 International Conference on Management of Data*, 1673–1676.

[10] Yong Wang, Guoliang Li, and Nan Tang. 2019. Querying shortest paths on time dependent road networks. *Proceedings of the VLDB Endowment*, 12, 11, 1249–1261.

[11] Peilun Yang, Hanchen Wang, Ying Zhang, Lu Qin, Wenjie Zhang, and Xuemin Lin. 2021. T3s: effective representation learning for trajectory similarity computation. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 2183–2188.

[12] Di Yao, Gao Cong, Chao Zhang, and Jingping Bi. 2019. Computing trajectory similarity in linear time: a generic seed-guided neural metric learning approach. In *2019 IEEE 35th international conference on data engineering (ICDE)*. IEEE, 1358–1369.

[13] Hanyuan Zhang, Xingyu Zhang, Qize Jiang, Baihua Zheng, Zhenbang Sun, and Weiwei Sun. 2020. Trajectory similarity learning with auxiliary supervision and optimal matching.(2020). In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, Yokohama, Japan*, 11–17.

[14] Yifan Zhang, An Liu, Guanfeng Liu, Zhixu Li, and Qing Li. 2019. Deep representation learning of activity trajectory similarity computation. In *2019 IEEE International Conference on Web Services (ICWS)*, 312–319. DOI: 10.1109/ICWS.2019.00059.

[15] Yu Zheng, Xing Xie, Wei-Ying Ma, et al. 2010. Geolife: a collaborative social networking service among user, location and trajectory. *IEEE Data Eng. Bull.*, 33, 2, 32–39.