GeoInformatica (2024) 28:631–648
https://doi.org/10.1007/s10707-024-00515-x

RESEARCH

Transfer-learning-based representation learning for trajectory similarity search

Danling Lai[1] · Jianfeng Qu[1] · Yu Sang[2] · Xi Chen[1]

Received: 31 October 2023 / Revised: 29 February 2024 / Accepted: 26 March 2024 /
Published online: 13 April 2024

Abstract
Trajectory similarity search is one of the most fundamental tasks in spatial-temporal data analysis. Classical methods are based on predefined trajectory similarity measures, consuming high time and space costs. To accelerate similarity computation, some deep metric learning methods have recently been proposed to approximate predefined measures based on the learned representation of trajectories. However, instead of predefined measures, real applications may require personalized measures, which cannot be effectively learned by existing models due to insufficient labels. Thus, this paper proposes a transfer-learning-based model FTL- TRAJ, which addresses this problem by effectively transferring knowledge from several existing measures as source measures. Particularly, a ProbSparse self-attention-based GRU unit is designed to extract the spatial and structural information of each trajectory. Confronted with diverse source measures, the priority modeling assists the model for the rational ensemble. Then, sparse labels are enriched with rank knowledge and collaboration knowledge via transfer learning. Extensive experiments on two real-world datasets demonstrate the superiority of our model.

Keywords Trajectory similarity · Top k similarity search · Transfer learning

1 Introduction

The utilization of GPS devices has revolutionized our ability to precisely track and document the movement of objects. There has been an increasing number of research focused on

✉ Danling Lai
dllai@stu.suda.edu.cn

✉ Jianfeng Qu
jfqu@suda.edu.cn

Yu Sang
7213107001@stu.jiangnan.edu.cn

Xi Chen
Chenxi10@suda.edu.cn

1 School of Computer Science and Technology, Soochow University, Suzhou, Jiangsu, China

2 School of Artificial Intelligence and Computer Science, Jiangnan University, Wuxi, Jiangsu, China

spatial-temporal data mining [1, 2]. This wealth of recorded information has given rise to a rich source of data known as trajectories, which have found diverse applications across numerous industries, such as trajectory entity linking [3], route recommendation [4] and anomaly detection [5]. Trajectory similarity search is crucial for various spatial-temporal data mining tasks. It's targeted at identifying the Top *k* most similar trajectories corresponding to a given query trajectory.

Numerous classical approaches have been developed to assess trajectory similarity by employing point-based comparisons using predefined metrics like Dynamic Time Warping (DTW), Edit Distance with Real Penalty (ERP), and the Hausdorff distance. Nonetheless, these methods tend to demand significant computational resources and are plagued by substantial time complexity. For efficiency concerns, recent studies have been dedicated to accelerating the computation. Some of them design approximating algorithms [6–8], sacrificing acceptable accuracy in order to alleviate the time complexity. There are also some methods to perform pruning strategies [9, 10] to speed up the computational process. However, these methods are specific to a certain measure and lack of generalization.

To enable a generic, efficient, and accurate trajectory similarity search. An increasing number of learning-based methods [11–14] are proposed. With a deep learning model, these methods extract the spatial information and structural knowledge of trajectories, in order to map trajectories into the low dimensional space and reduce space consumption. They significantly accelerate the similarity computation and achieve high accuracy. However, these methods suffer from some limitations:

First, it is tough to determine an appropriate measure for some real-world applications, since a predefined measure that accommodates a certain scenario may not be suitable for another [15, 16]. The target measure needed by a specific case may not be fully supported by any predefined measure. It thus requires the full utilization of trajectory similarity labels (e.g. the pair-wise distance between trajectories, the similarity order of query trajectory on target trajectories) on target measure. However, these labels are hard to get due to the expensive cost of manual annotation, which leads to data scarcity on target measures. Hence, it calls for a few-shot trajectory metric learning strategy that can not only quickly learn from insufficient target measure labels, but also make meaningful use of predefined measures.

Second, since trajectories are sequential data, existing models [12] often employ RNN modules to extract spatial features of trajectories. While these methods can effectively capture knowledge in an informative way to some extent, they only consider LSTM [13] and ignore other RNN cells, which may lead to redundant parameters. What is more, they exhibit sub-optimal GPU utilization. Attention-based modules are then used to extract structural information. However, according to recent research [17], the self-attention mechanism suffers from high time complexity and consumes high memory usage. It thus requires efficiently combining RNN and attention mechanism, in order to effectively extract spatial and structural knowledge of long trajectories while making full use of GPU and reducing memory usage.

To overcome the above limitations of trajectory similarity learning, we propose a novel deep-learning-based model FTL- TRAJ, which is designed for **F**ew-shot **T**ransfer-**L**earning-based **Traj**ectory similarity search. It employs ProbSparse self-attention-based Gated Recurrent Unit(GRU) to obtain trajectory embeddings. Then, metric knowledge from data-rich source measures is transferred to the target measure, including the relative ranking orders and common similarity in source measures. Our contributions can be summarized as below:

- We propose an innovative representation learning method via ProbSparse self-attention-based GRU to extract spatial and structural knowledge of trajectories, which reduces memory usage and fully utilizes the GPU resources.

- We effectively transfer rank knowledge and collaborative knowledge from data-rich measures by developing the triplet-loss-based transfer loss, which mitigates the performance drop due to sparse labels.
- Experimental results on real-world trajectory datasets demonstrate the effectiveness of our approach.

2 Related work

2.1 Trajectory similarity search

A trajectory, generated by a moving object, is composed of a series of ordered GPS points. The advancement of LBS (Location-Based Services) devices has generated large-scale trajectory data, driving progress across multiple disciplines, such as geography and computer science. Trajectories contain rich information besides location information, such as dwell time [18], mode of transportation [19], and waypoints, which can be combined with map and road networks to achieve further research [20–24]. In our paper, we mainly focus on trajectory similarity search.

For a query trajectory in the dataset $\mathcal{T}$, trajectory Similarity Search calculates the pair-wise trajectory distance under the specific measure, to obtain the Top k most similar ones in $\mathcal{T}$. The search process analyzes the spatial and structural relationship between trajectories and is fundamental for further data mining. Efficiency is crucial when conducting spatial queries [25–27] such as Top k similar trajectory search in massive datasets. Existing trajectory similarity search relies on measures. They can be broadly divided into two types: Heuristic Measures and Learning-based Measures.

2.1.1 Heuristic measures

Heuristic measures follow predefined formulas to find the optimal matches of points in trajectories, in order to calculate the pair-wise distance. For example, Hausdorff [15] measures the greatest distance among all distance values between trajectories. EDR [28] and LCSS [29] are edit-distance-based measures, which calculate the number of operations that transform one trajectory into another one. The time complexity of calculating the similarity between T_a and T_b via heuristic measures is usually $O(l_a l_b)$ where l_a is the length of T_a, and l_b denotes that of T_b.

To reduce the time consumption during the computation, there have been solutions to perform pruning via the filter-and-refinement framework [9, 10]. Some methods utilize approximate algorithms to accelerate the computation [6–8]. Despite their ability to speed up the computation to some extent, they may still be inadequate to meet the need of the large-scale trajectory similarity search.

2.1.2 Learning-based measures

With the development of deep learning, learning-based trajectory similarity computation models are designed to map trajectories into a d-dimensional embedding representation, thus reducing the complexity of similarity evaluation to $O(d)$. For example, NeuTraj [11] proposes a spatial memory unit to model the correlation between trajectories. Traj2SimVec [12] exploits the distance knowledge between sub-trajectories and learns the optimal matching relationship between points with a point-matching function. T3S [13] applies LSTM and global self-attention network to capture the spatial and structural information of the trajectories. TrajGAT [14] models each trajectory as a graph and captures long-term dependency based on graph attention modules. These methods are generic and can support the existing measures with annotated labels (e.g. the pair-wise distance, and the relative similarity rank orders). However, they suffer from performance drops on data-sparse measures.

2.2 Transfer learning for trajectory data mining

When employing machine learning methods or deep learning approaches in some real-world scenarios, there are no abundant labels to support effective training, which leads to sub-optimal performance. Recently, transfer learning has proven to be a successful approach to few-shot learning [30, 31]. They focused on transferring useful knowledge from source domains to the target domain [32, 33].

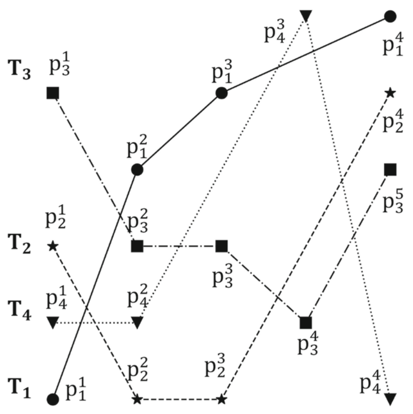
When performing transfer learning for trajectory data mining, especially for similarity search, how to identify useful source measures and transfer useful information remains a hard nut to crack. Different measures focus on different aspects of trajectories, some of them focus on the points, while others focus on the segment [15]. The measure of diversity leads to different search results under different measures. As shown in Fig. 1, the retrieved results between the target measure and source measure conflicts. Directly transferring knowledge will interfere with the learning of the target measure. What is more, compared with the ERP distance, the DTW distance seems more consistent with the Hausdorff distance according to the search results. If such a consistent relationship still holds for the majority of trajectories, more attention should be paid to the DTW distance. It thus requires effective priority modeling on source measures.

3 Preliminary

In this section, we describe the related concepts and definitions involved in our work. Based on the definitions, the problem statement is formulated.

3.1 Definitions

Definition 1 (Trajectory) A trajectory T records the movement of entities via a series of GPS points with timestamps. Usually, it is denoted by $T = [c_1, c_2, ..., c_l]$, where l is the length



$\mathbf{T_1}$ = [(1, 1), (2, 4), (3, 5), (5, 6)],
$\mathbf{T_2}$ = [(1, 3), (2, 1), (3, 1), (5, 5)],
$\mathbf{T_3}$ = [(1, 5), (2, 3), (3, 3), (4, 2), (5, 4)],
$\mathbf{T_4}$ = [(1, 2), (2, 2), (4, 6), (5, 1)]

	Hausdorff (Target)	**DTW (Source1)**	**ERP (Source2)**
$\mathbf{d(T_1, T_2)}$	1.90 ①	9.16 ①	10.00 ②
$\mathbf{d(T_1, T_3)}$	2.53 ②	11.58	13.47
$\mathbf{d(T_1, T_4)}$	3.79	9.41 ②	9.41 ①

Fig. 1 The Similarity Search Diversity under different measures

of the trajectory, and $c_t = (lat_t, lng_t)$ describes coordinate (i.e. the latitude and longitude) at the t-th point of the trajectory.

Definition 2 (Grid Cells) The whole map of each dataset is divided into $m \times n$ grids. Here, m and n are determined by the size of grids, which are configurable to different scenarios. Then, each trajectory can be mapped into a grid sequence $T^g = [g_1, g_2, ..., g_l]$, where $g_t = (x_t, y_t)$ denotes the grid indices of t-th points.

Definition 3 (Trajectory Similarity Measure) Given a pair of trajectories T_a and T_b, a measure $d_M(T_a, T_b)$ calculates the pair-wise distance to describe the similarity between them. The smaller the distance is, the higher the similarity is.

Definition 4 (Top k Trajectory Similarity Search) Given a query trajectory T, trajectory similarity search employs a specific measure to calculate the pair-wise distance between query trajectory and other trajectories in the given dataset $\mathcal{T}$. Then it retrieves the Top k similar trajectories according to the calculated distance.

Definition 5 (Target Measure and Source Measure) In many real-world scenarios, predefined measures may be unable to assess the similarity between trajectories precisely. Therefore, additional data annotation is required, which is both time-consuming and labor-intensive. We refer to the sparsely annotated measures involved in these scenarios as the target measure, which is the primary focus of our research. Meanwhile, other well-annotated measures can provide us with supplementary information, and we refer to them as source measures.

3.2 Problem statement

Given a set of trajectories in the dataset $\mathcal{T}$, we aim at learning a function $f = T \rightarrow E \in \mathbb{R}^d$ for each trajectory, such that a trajectory T can be mapped into d-dimensional embedding e which describe the similarity relationship between trajectory in an informative way. Our problem is to obtain the representations so that the distance between e_i, e_j can be as close as possible with the target measure M, i.e. minimizing $L_{sim} = |d(e_i, e_j) - d_M(T_i, T_j)|$.

4 Methodology

In this section, we will first present the overall architecture of our FTL- Traj for the few-shot trajectory similarity search. As illustrated in Fig. 2, FTL- Traj utilizes a ProbSparse self-attention-based GRU cell to model trajectories in both the spatial level and structural level, so that we can capture the similarity relationship effectively. Then, transfer-learning-based metric learning enriches labels on the data-sparse target measure.

4.1 Overview of FTL-Traj

We scheme the architecture of FTL- Traj with a neural metric learning framework. It selects annotated trajectories from the dataset $\mathcal{T}$ as anchor trajectory (i.e. the query trajectory) and samples several target trajectories to evaluate the performance of the model, in order to achieve efficient optimization. Then, these trajectories are mapped into low-dimensional vectors. After the learned mapping, the distances between vectors should closely resemble the actual distances between trajectories. To capture the spatial and structural information of trajectories, we adopt GRU and ProbSparse self-attention-based mechanisms. To effectively

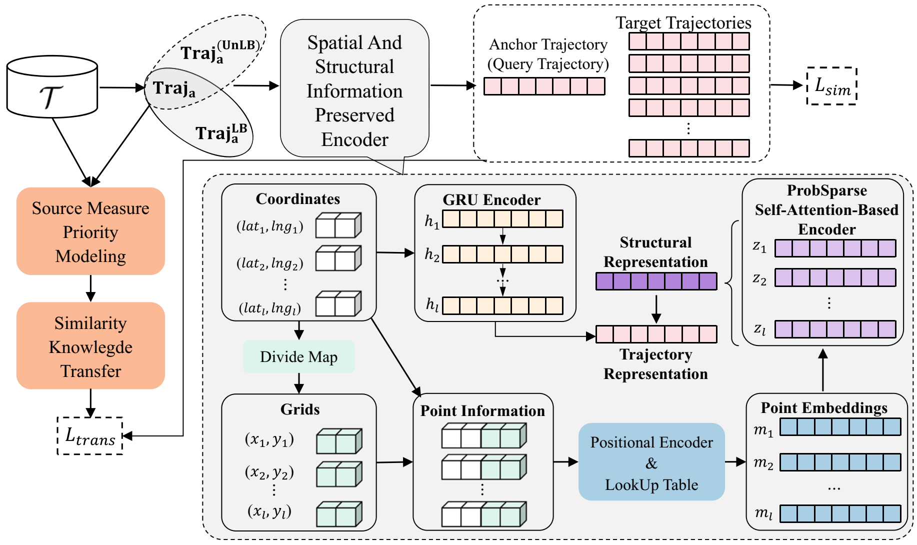


Fig. 2 Architecture of FTL- TRAJ

transfer knowledge from domains with rich data annotation, we design a source measure priority modeling method and enrich labels on data-sparse target domains with the transferred knowledge from source measures.

4.2 Spatial and structural knowledge preserved representation learning

4.2.1 GRU-based spatial information extraction

Real-world trajectories are typically described as sequences of geographic coordinate pairs and are initially modeled using RNNs. Nonetheless, basic RNNs are susceptible to information loss when handling long trajectories. Recent methods [11–13] employs the LSTM network to process the coordinate tuples. Compared with LSTM, which has three gates and employs more parameters, GRU only contains the reset gate and the update gate, which simplifies the structure and reduces the number of parameters, which contributes to better generalization. What is more, in few-shot learning, utilizing LSTM to model the sparse training data may lead to over-fitting problems. Hence, we replace LSTM with GRU to achieve better efficiency.

In the context of the GRU (Gated Recurrent Unit), when focusing on the forward direction, sequence information is managed through the use of gate mechanisms. For a given input series represented as $T^c = [c_1, c_2, ..., c_l]$, where $c_i = (lat_i, lng_i) \in \mathbb{R}^{2\times 1}$, and l is the number of points, and d is the dimension of embedding vectors. The gate mechanisms are characterized by their respective mathematical formulations:

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t]) \tag{1}$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t]) \tag{2}$$

$$\hat{h}_t = \tanh(W \cdot [r_t \times h_{t-1}, x_t]) \tag{3}$$

$$h_t = (1 - z_t) \times h_{t-1} + z_t \times \hat{h}_t \tag{4}$$

where all the gates share the shame shape $\mathbb{R}^{d \times 1}$. In the end, the hidden state at the final step l, which contains all the coordinate information, will be used as the spatial representation of the given trajectory.

4.2.2 ProbSparse self-attention mechanism for structural information extraction

In order to capture the structural information of each trajectory, grid cells and coordinate cells should both be taken into consideration. However, the contribution of each point to the structural information varies, necessitating the model to focus on the points that are more critical for structural information. Self-attention modules are able to model the dependencies within sequential data while remaining unaffected by prior interference in sequence distances. Hence, it will be conducted to extract structural information of trajectories.

In this study, we first concatenate the coordinate inputs and grid inputs of each trajectory as point inputs. Then, we follow [13] to initialize the grid embedding matrix:

$$m_i = \text{lookup}(p_i) + pos_i \tag{5}$$

$$\text{lookup}(p_i) = p_i \cdot W_l + b_l \tag{6}$$

where $p_i = (x_i, y_i, lat_i, lng_i)$ is the concatenated input at i-th step. The lookup function consisting of $W_l \in \mathbb{R}^{4 \times d}$, $b_l \in \mathbb{R}^{1 \times d}$ serves as a projection matrix to project the inputs into d-dimensional vectors. Then, we combine the vector with positional encoding result pos_i. The positional encoding process follows a learned or pre-defined pattern, such as cosine and sine positional functions, that specify the positions of individual elements. By combining the lookup representation and positional representation together, we derive the input embedding m_i. We use the m_i of each step as the query vectors for self-attention modules. Then, we perform the dot-product process.

$$(Q, K, V) = \mathcal{M}_i \times (W_Q, W_K, W_V) \tag{7}$$

$$A(Q, K, V) = \text{softmax}(\frac{Q \times K^{\mathrm{T}}}{\sqrt{d}}) \tag{8}$$

$$Z = A \times V \tag{9}$$

The computation process of self-attention modules can be summarized as above, where $\mathcal{M}_i \in \mathbb{R}^{b \times l \times d}$ denotes the input matrix and b denotes the batch size. Q, K, V are query matrices, key matrices, and value matrices respectively. After the module, the output result Z is used as the representation of a grid. To further improve the robustness of our model, multi-head self-attention is employed in our model. Each of these *head* learns different attention patterns, enabling the model to capture various relationships and features within the input data simultaneously. Finally, we utilize the mean value of vectors in Z as the final structural representation of the trajectory.

$$(Q_i, K_i, V_i) = \mathcal{M}_i \times (W_{i,Q}, W_{i,K}, W_{i,V}) \tag{10}$$

$$A_i(Q_i, K_i, V_i) = \text{softmax}(\frac{Q_i \times K_i^{\mathrm{T}}}{\sqrt{d}}) \tag{11}$$

$$Z_i = A_i \times V_i \tag{12}$$

$$Z = \text{concatenate}\,[Z_1, Z_2, ..., Z_h] \tag{13}$$

$$h_s = \text{mean}(z_1, z_2, ..., z_l) \tag{14}$$

According to previous studies [34], the distribution of self-attention probability has potential sparsity. Therefore, only a few dot-product pairs in the self-attention modules contribute to the attention, which forms a long-tail distribution. It thus requires to effectively distinguish useful pairs from trivial ones. Inspired by [17], we utilize ProbSparse self-attention to improve vanilla self-attention modules. First, we employ Kullback-Leibler divergence to evaluate the likeness between a pair of distributions. Then, we formulate the i-th query's sparsity measurement as the formula (15):

$$QSM(q_i, K) = \max_j \frac{q_i k_j^{\mathrm{T}}}{\sqrt{d}} - \frac{1}{l}\sum_j \frac{q_i k_j^{\mathrm{t}}}{\sqrt{D}} \tag{15}$$

Then, the Top k query in QSM result will be selected as $\hat{Q}$. The rest query will not be calculated. With ProbSparse self-attention, the time and space complexity of the self-attention module are both reduced to $O(L \ln L)$.

Based on the extracted spatial information h_l and structural information h_s, we use a learnable parameter α to balance their weights. Finally, the final representation of each trajectory $e = \alpha \cdot h_l + (1 - \alpha) \cdot h_s$.

4.3 Deep metric learning via transfer learning

4.3.1 Source measures priority modeling

While different measures may have distinct emphases, their computed results still exhibit significant overlap. Taking Fig. 3 as an example, given a query trajectory, we calculate the pair-wise distance between it and 1000 target trajectories under different measures. It can be concluded from the figure that the Top k search results under various measures still show a substantial degree of overlap. This provides us with valuable insights: we can learn from measures with rich annotations and transfer useful knowledge to data-sparse target measures.

The difficulty of making full use of the trajectories is the limited labels on the target measure. For a given trajectory query, it would be great to provide predictions on source measures as soft labels in our problems. However, directly transferring knowledge from source domains without considering the divergence between source domains and the target domain is harmful to the performance of models. More attention should be paid to high-

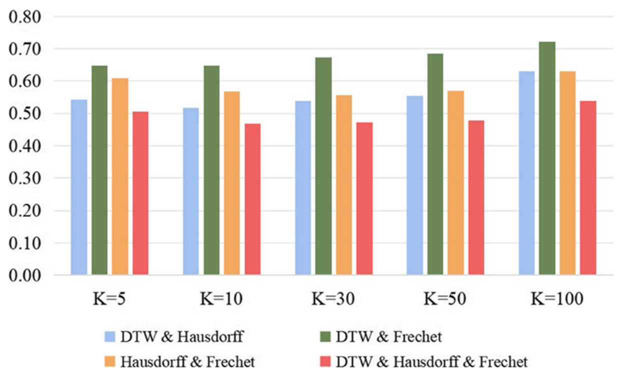


Fig. 3 The Overlap Ratio under Different Measures

quality source domains, where source measures are more consistent with the target measure. It thus requires effective consistency evaluation on each source measure.

Different measures focus on different aspects of trajectories. In order to better approximate the target measure, it is preferable for the model to learn more from source measures that are more consistent with the target measure on ranking and distance. Hence, it calls for reasonable methods to evaluate between measures. We adopt the hitting ratio as our evaluation method.

Specifically, for each annotated trajectory, we regard the predicted ranking results on the target measure as ideal ranking, and the calculated ranking results on source measures computed by teacher models on source measure will be referred to as optional ranking. For the target measure, suppose each annotated trajectory is provided with Top u trajectories. We perform The Top v search on each source measure, in order to calculate the hitting ratio of the Top u result retrieved. The more the Top v search results on a source measure overlap with the Top u results on the target measure, the higher score, the source measure will be assigned. Then, the weight of each source measure is derived.

$$w_i = \frac{\mathrm{HR}_i}{\sum_{j \in \text{source measures}} score_j} \tag{16}$$

4.3.2 Label enrichment from source domains

With the importance weight of each source measure, we are enabled to distinguish more valuable source measures. Then, to effectively transfer the knowledge, we further assign each trajectory with soft labels. It contains rank-based information and collaboration-based information:

Rank-based score When predicting top-K similar trajectory of a given T_i, trajectories at Top positions are more relevant to T_i [35]. Hence, treating each trajectory equally will mislead the model. We predefined s^r as follows. As K increases, the correlation between trajectory and T_i decreases. We utilize an empirical weight following the geometric distribution [36] to calculate the rank-based score, where j denotes the rank position, and ρ, r are dataset-dependent hyper-parameters.

$$s_j^r \propto \rho(1-\rho)^r \tag{17}$$

Collaboration-based score Despite the diversity of measures, some trajectories can be top-K similar to a given T_i on different measures. Compared with those trajectories that are only top-K on one source measure, the wrong estimation of ones that are top-K on several source measures will leave more significant impacts on the target measure. Hence, we define collaboration-based weight s^c as follows:

$$s^c \propto \frac{\mathrm{count}(T_j)}{q \cdot n_s} \tag{18}$$

where n_s denotes the number of source measures. $\mathrm{count}(T_j)$ denotes how many times the trajectory T_j is included in Top q trajectory query under different source measures.

Given Fig. 4 as an example. Trajectory T_2 appears in the Top q for all sources, making it get the highest collaboration score. similarly, T_3 and T_4 appear twice, while T_1 and T_5 appear once, thus their collaboration scores are ranked as shown in the figure. As mentioned earlier, we are only concerned with the Top q trajectories under the source measure, so rankings beyond this threshold will not be considered.

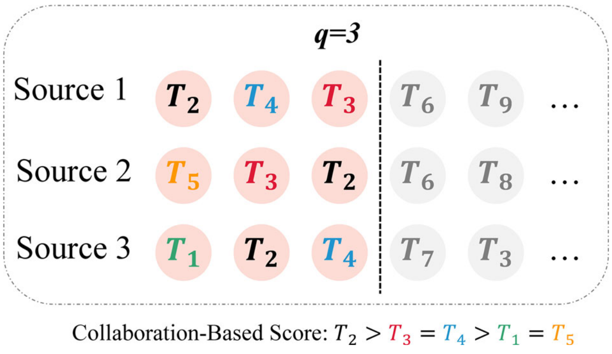

q=3
Source 1
Source 2
Source 3
T_2 T_4 T_3 T_6 T_9 ...
T_5 T_3 T_2 T_6 T_8 ...
T_1 T_2 T_4 T_7 T_3 ...
Collaboration-Based Score: $T_2 > T_3 = T_4 > T_1 = T_5$

Fig. 4 An example of Collaboration-based score

Given a query trajectory, with the rank-based score and collaboration-based score, we obtain the consensual rank-based score of the target trajectory T_j as s_{T_j} according to the formula (19).

$$s_{T_j} = \sum_{i \in \text{source measures}} w_i \cdot (s_T^r \cdot s_T^c) \tag{19}$$

Then, we rank the target trajectories according to their calculated score. We define the Top ϕ trajectories as *transfer-positive* ones, and the rest of the trajectories are regarded as *transfer-negative* ones. Then, we use random sampling to sample r transfer-positive trajectories and r transfer-negative ones. Then we obtain r triples $< a_1, p_1, n_1 >$, $< a_2, p_2, n_2 >$, ..., $< a_r, p_r, n_r >$. The representations of the anchor trajectories are supposed as similar as possible to the representation of transfer-positive trajectories, and conversely, those between anchor trajectory and transfer-negative trajectories should be dissimilar enough. Thereafter, we define the transfer loss as the formula (20), where ϵ is the margin value. Finally, the loss function for the training set consisting of L_{sim} and L_{trans}.

$$L_{trans} = \frac{1}{r}\sum_{t=1}^{r} \max(\gamma \cdot d(e_{a_t}, e_{p_t}) - \gamma \cdot d(e_{a_t}, e_{n_t}) + \epsilon, 0) \tag{20}$$

$$L = L_{sim} + \beta \cdot L_{trans} \tag{21}$$

4.4 Experimental settings

In this section, we present our experimental setup and empirical results. We conduct extensive experiments on real-world datasets to validate our model's effectiveness.

4.4.1 Datasets

Our experiments are based on two real-world public trajectory datasets. The first one [37], named *Porto*, consists of over 1.7 million vehicle trajectories in Porto from 2013 to 2014. The second dataset referred to as*Chengdu*, contains over 1.4 billion taxi GPS data points generated by more than 14,000 taxis between August 3, 2014, and August 30, 2014 in Chengdu, China.

Following [11, 13], we choose trajectories in the city center. Then we remove the trajectories of less than 10 records and randomly select 10k trajectories as our dataset. To evaluate the effectiveness of our methods on trajectories with different lengths, for Porto, we excluded trajectories that exceeded a length of 150. As for Chengdu, we excluded trajectories that exceeded a length of 250.

4.4.2 Experimental guidance

We compute the similarity on measures including ERP, DTW, SSPD, Hausdorff, and Fréchet distance. During the training, we selected DTW, Hausdorff, and SSPD as target measures. When training the model on a specific target measure, the rest of the provided measures are regarded as source measures.

To simulate the data-sparsity problem in the real application, we randomly partition our dataset into the training set and test set following a ratio of 1:19. We use 20% trajectories in the training set. Given a trajectory, only the pair-wise distance of top-15 trajectories is available in training. The margin value ϵ of point matching loss is 0.01. We set the batch size as 10, and the learning rate as $1e-3$. The trajectory representation vector is 128-dimensional (i.e., d = 128).

4.4.3 Evaluation metrics

As in [11, 12, 35], two different evaluation metrics are used:

- **Hitting Ratio**@k calculates the overlap percentage of Top k results based on the predicted Top k and ground truth. A higher hitting rate suggests more accuracy.
- **Recall** k**@R** evaluates how many of the Top k ground-truth trajectories are recovered by the top-R results. Higher recall indicates better recovering capability.

4.5 Compared methods

We select our base model among several learning-based trajectory similarity computation methods, including NeuTraj [11], Traj2SimVec [12], T3S [13], TrajCL [38] and variants of them. The source code of NeuTraj, TrajCL is available online, so we directly use its implementation. For other methods, we follow the settings in the paper and implement these methods on our own. To implement the unsupervised method TrajCL for our similarity search task, we follow the paper to utilize a two-layer MLP to connect the model, optimizing with MSE loss.

Additionally, some modules in student models are invalid due to the data sparsity and should be removed during the training, such as sub-trajectory distance in Traj2SimVec, and dissimilar sampling loss in NeuTraj. Hence, we change these methods to accommodate our problem. As shown in Tables 1 and 2, data sparsity leads to the performance drop on each measure.

4.6 Performance study

In this section, we present the evaluation outcomes for the top-k similarity search task of the compared methods, considering the SSPD, Hausdorff, DTW, and discrete Fréchet distance measures. The table 1 and 2 shows the performance evaluation for different methods, with the bolded entries indicating the best results under different measures on the specific dataset. As shown in the tabels, our model FTL- Traj surpasses other methods under most evaluation metrics. For the Porto dataset, the recall rate keeps up to 70

Table 1 Performance Evaluation for Methods Under Different Distance Measures on Porto Dataset

Approach	SSPD				Hausdorff			
	HR@10	HR@50	R5@50	R10@50	HR@10	HR@50	R5@50	R10@50
RNN	0.2805	0.3773	0.6987	0.6218	0.3299	0.4752	0.7655	0.6875
NeuTraj-NoSam	0.3219	0.4077	0.7267	0.6552	0.3707	0.4911	0.7779	0.7264
NeuTraj	0.3244	0.4127	0.7333	0.6595	0.3862	0.5095	0.7901	0.7316
Traj2SimVec	0.3358	0.4208	0.7441	0.6699	0.3864	0.5169	0.7951	0.7413
T3S	0.3428	0.4305	0.7495	0.6776	0.3916	0.5222	0.8006	0.7469
TrajCL	0.2887	0.4061	0.6757	0.6186	0.3521	0.4416	0.6624	0.6002
Ours-No L_{trans}	**0.3623**	**0.4322**	0.7728	0.6996	**0.4179**	0.5451	0.8707	0.8161
Ours	0.3559	0.4295	**0.7831**	**0.7072**	0.4171	**0.5495**	**0.8788**	**0.8165**
Approach	DTW				Discret Frechet			
	HR@10	HR@50	R5@50	R10@50	HR@10	HR@50	R5@50	R10@50
RNN	0.3767	0.4759	0.7836	0.7502	0.3794	0.4934	0.7400	0.6795
NeuTraj-NoSam	0.4272	0.5041	0.8110	0.7618	0.3932	0.5173	0.7653	0.7267
NeuTraj	0.4208	0.5187	0.8313	0.7712	0.4143	0.5301	0.7826	0.7243
Traj2SimVec	0.4325	0.5300	0.8303	0.7782	0.4157	0.5262	0.7821	0.7362
T3S	0.4377	0.5361	0.8384	0.7867	0.4208	0.5353	0.7907	0.7439
TrajCL	0.4010	0.4813	0.7331	0.6794	0.3601	0.4280	0.6609	0.6082
Ours-No L_{trans}	0.5471	0.6472	0.9346	0.9043	0.5654	0.6676	0.9228	0.8937
Ours	**0.5519**	**0.6548**	**0.9432**	**0.9139**	**0.5661**	**0.6715**	**0.9313**	**0.9040**

Table 2 Performance Evaluation for Methods Under Different Distance Measures on Chengdu Dataset

Approach	SSPD				Hausdorff			
	HR@10	HR@50	R5@50	R10@50	HR@10	HR@50	R5@50	R10@50
RNN	0.2498	0.3704	0.6683	0.6214	0.2376	0.3717	0.6608	0.5716
NeuTraj-NoSam	0.2764	0.3885	0.6978	0.6259	0.2617	0.3972	0.6840	0.5796
NeuTraj	0.2853	0.3989	0.6966	0.6309	0.2681	0.3978	0.6891	0.5914
Traj2SimVec	0.2907	0.3987	0.7125	0.6352	0.2737	0.4036	0.6972	0.5951
T3S	0.2937	0.4084	0.7115	0.6416	0.2825	0.4080	0.6988	0.5987
TrajCL	0.2328	0.3716	0.6227	0.5709	0.2426	0.3771	0.5789	0.5302
Ours-No L_{trans}	0.3475	0.4190	0.7770	0.7050	0.4115	0.5355	0.8582	0.7974
Ours	**0.3599**	**0.4339**	**0.7832**	**0.7104**	**0.4127**	**0.5433**	**0.8659**	**0.8103**
Approach	DTW				Discret Frechet			
	HR@10	HR@50	R5@50	R10@50	HR@10	HR@50	R5@50	R10@50
RNN	0.3152	0.4259	0.6997	0.6501	0.2533	0.3496	0.5329	0.5749
NeuTraj-NoSam	0.3363	0.4302	0.7112	0.6586	0.2742	0.3592	0.5377	0.5857
NeuTraj	0.3409	0.4430	0.7254	0.6684	0.2841	0.3714	0.5510	0.5924
Traj2SimVec	0.3390	0.4489	0.7298	0.6724	0.2956	0.3820	0.5630	0.6058
T3S	0.3490	0.4504	0.7326	0.6768	0.2964	0.3819	0.5590	0.6055
TrajCL	0.3272	0.3812	0.6317	0.5986	0.2678	0.3444	0.5414	0.5022
Ours-No L_{trans}	**0.5503**	0.6421	0.9287	0.9006	0.5558	0.6638	0.9255	0.8916
Ours	0.5489	**0.6533**	**0.9410**	**0.9110**	**0.5660**	**0.6691**	**0.9314**	**0.9023**

Furthermore, we analyzed the performance of the model on the Chengdu dataset. Due to the increased length of trajectory data, previous methods were unable to effectively model longer trajectories, resulting in a decline in performance. The hitting ratio of FTL- TRAJ is even 10% higher than simple RNN on each measure, with some measures even showing an improvement of up to 20%.

Overall, our training time is a bit longer than other supervised methods, with each epoch taking around 20-40 seconds. Despite introducing new knowledge from other sources and employing transfer learning, the use of GRU and ProbSparse Self-attention mechanisms has improved the efficiency of the model. Additionally, all supervised methods have significantly shorter training times compared to TrajCL, as training TrajCL models requires obtaining vertex embeddings using node2vec and has higher GPU requirements.

4.7 Ablation study

Next, we conduct the ablation study to evaluate the usefulness of key components. First, we can conclude from the Tables 1 and 2 that the transfer knowledge introduces useful knowledge from source domains to the target domain and contributes to the model accuracy.

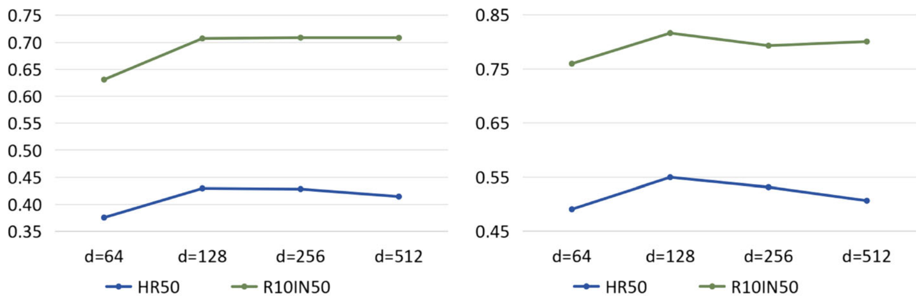
Second, T3S can be viewed as the variant of FTL- TRAJ, as it employs LSTM and a self-attention mechanism to extract spatial and structural information of trajectories. Compared with T3S, FTL- TRAJ performs much better. It achieves at least 7

4.8 Parameter sensitivity study

In this research, apart from the standard hyperparameters found in the deep learning model (e.g., the learning rate), we will provide additional vital parameters associated with the training and testing procedures as follows.

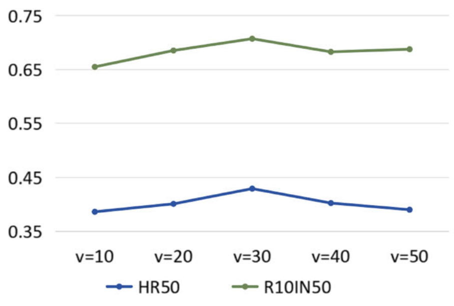
4.8.1 Impact of embedding dimension *d*

The embedding dimension d is employed to define the latent embedding space for representing trajectories. It plays a key role in determining the effectiveness of our deep learning approach since it is interconnected with both training duration and processing time. Smaller d may reduce the processing duration, but it will sacrifice important knowledge as well. According to the Fig. 5, increasing d can improve the model performance. However, when

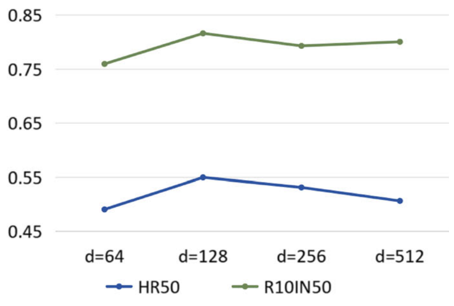


(a) Performance Under SSPD (b) Performance Under Hausdorff

Fig. 5 Performance with different d in metrics HR50, R10@50



(a) Performance Under SSPD



(b) Performance Under Hausdorff

Fig. 6 Performance with different v in metrics HR50, R10@50

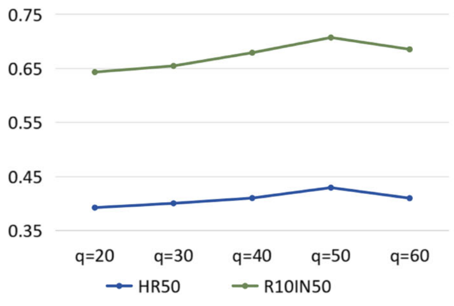
d is too large, the model fails to keep complex information. Hence, d is set to 128 in our experiments.

4.8.2 Impact of *v* for source measure priority modeling

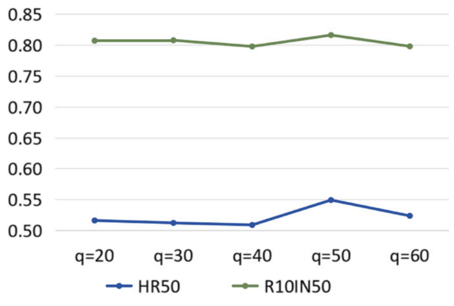
When modeling source measure priority, we perform a search on the target measure to determine how many of the Top v search results from the source measure contain similar trajectories that are annotated *similar* on the target measure. Taking the divergence between source measures and target measures into account, their ranking results are not entirely identical. A small v may lead to missing high-quality source measures, while an excessively large v may introduce noise and affect the accuracy of priority assessment. Hence, we set the v to 30 according to Fig. 6.

4.8.3 Impact of *q* for transfer-positive trajectories division

When enriching trajectory labels with the information from source measures, we generated the score of target trajectories. Here, trajectories that are ranked among the Top q in terms of scores are considered transfer-positive. When q is too small, some potentially similar trajectories might be mistakenly classified. Conversely, when q is too large, the quality of negative samples deteriorates, affecting the effectiveness of the model. According to Fig. 7, we set the q as 50.



(a) Performance Under SSPD



(b) Performance Under Hausdorff

Fig. 7 Performance with different q in metrics HR50, R10@50

5 Conclusion

In this paper, we discuss the few-shot trajectory similarity computation. Existing methods suffer from performance decline when computing data-sparse measures in real applications. To this end, we propose a transfer-learning-based model FTL- Traj. FTL- Traj overcomes the hurdle of learning personalized measures by transferring knowledge from data-rich source measures. Notably, it employs a ProbSparse self-attention-based GRU unit to capture spatial and structural insights from trajectories. Given the diversity of source measures, a priority modeling approach guides the model in crafting a well-reasoned ensemble. Furthermore, transfer learning enriches the model with knowledge regarding ranking and collaboration, building upon sparse labels. Extensive experimentation on two real-world datasets serves as compelling evidence for the effectiveness and superiority of our proposed model.

Author Contributions Danling Lai wrote the main manuscript text. Jianfeng Qu, Yu Sang and Xi Chen participated in model design and technical discussion. All contributing authors reviewed the manuscript.

Funding This work was supported by the National Natural Science Foundation of China (Grant No. 62102276), the Natural Science Foundation of Jiangsu Province (Grant No. BK20210705), China Postdoctoral Science Foundation(Grant No. 2023M732563)and the Natural Science Foundation of Educational Commission of Jiangsu Province, China (Grant No. 21KJD520005)

Availability of Data and Materials Both datasets analyzed during the current study are available in the following resources: https://drive.google.com/drive/folders/1wGDt81o80yfbWx3Dmb66Q9VgQdP2-mqr?usp=drive_link

Declarations

Competing interests The authors declared no potential conflicts of interest with respect to the research, authorship, or publication of this article.

Ethical approval Not applicable.

References

1. Shuo S, Chen L, Jensen CS, Wen J-R, Panos (2017) Searching trajectories by regions of interest. TKDE 29(7):1549–1562
2. Chen L, Shang S, Yang C, Li J (2019) Spatial keyword search: a survey. GeoInformatica 24(3)
3. Jin F, Hua W, Zhou T, Xu J, Francia M, Orlowska ME, Zhou X (2022) Trajectory-based spatiotemporal entity linking. IEEE Trans Knowl Data Eng 9:34
4. Yang S, Liu J, Zhao K (2022) Getnext: Trajectory flow map enhanced transformer for next POI recommendation. In: Amigó E, Castells P, Gonzalo J, Carterette B, Culpepper JS, Kazai G (eds) SIGIR, pp 1144–1153
5. Wang C, Erfani SM, Alpcan T, Leckie C (2023) Online trajectory anomaly detection based on intention orientation. In: IJCNN, pp 1–8
6. Colombe C, Fox K (2021) Approximating the (continuous) fréchet distance. SoCG 189:26–12614
7. Xi Z, Kuszmaul W (2022) Approximating dynamic time warping distance between run-length encoded strings. ESA 244:90–19019
8. Backurs A, Sidiropoulos A (2016) Constant-distortion embeddings of hausdorff metrics into constant-dimensional l_p spaces. APPROX/RANDOM 60:1–1115
9. Gong X, Xiong Y, Huang W, Chen L, Lu Q, Hu Y (2015) Fast similarity search of multi-dimensional time series via segment rotation. DASFAA 9049:108–124
10. Sakurai Y, Yoshikawa M, Faloutsos C (2005) FTW: fast similarity search under the time warping distance. In: ACM SIGACT-SIGMOD-SIGART, pp 326–337

11. Yao D, Cong G, Zhang C, Bi J (2019) Computing trajectory similarity in linear time: A generic seed-guided neural metric learning approach. In: ICDE, pp 1358–1369
12. Zhang H, Zhang X, Jiang Q, Zheng B, Sun Z, Sun W, Wang C (2020) Trajectory similarity learning with auxiliary supervision and optimal matching. In: IJCAI, pp 3209–3215
13. Yang P, Wang H, Zhang Y, Qin L, Zhang W, Lin X (2021) T3S: effective representation learning for trajectory similarity computation. In: ICDE, pp 2183–2188
14. Yao D, Hu H, Du L, Cong G, Han S, Bi J (2022) Trajgat: A graph-based long-term dependency modeling approach for trajectory similarity computation. In: SIGKDD, pp 2275–2285
15. Hu D, Chen L, Fang H, Fang Z, Li T, Gao Y (2023) Spatio-temporal trajectory similarity measures: A comprehensive survey and quantitative study. CoRR arXiv:2303.05012
16. Toohey K, Duckham M (2015) Trajectory similarity measures. ACM SIGSPATIAL Special 7(1):43–50
17. Zhou H, Zhang S, Peng J, Zhang S, Li J, Xiong H, Zhang W (2021) Informer: Beyond efficient transformer for long sequence time-series forecasting. In: AAAI, pp 11106–11115
18. Fan Y, Xu J, Zhou R, Li J, Zheng K, Chen L, Liu C (2022) Metaer-tte: An adaptive meta-learning model for en route travel time estimation. In: Raedt LD (ed) IJCAI. Main Track, pp 2023–2029
19. Sun J, Xu J, Zhou R, Zheng K, Liu C (2018) Discovering expert drivers from trajectories. In: ICDE, pp 1332–1335
20. Shang S, Lu H, Pedersen TB, Xie X (2013) Finding traffic-aware fastest paths in spatial networks. In: SSTD. SSTD 2013, pp 128–145. Springer, Berlin, Heidelberg
21. Shi W, Xu J, Fang J, Chao P, Liu A, Zhou X (2023) Lhmm: A learning enhanced hmm model for cellular trajectory map matching. In: ICDE, pp 2429–2442
22. Xu Y, Xu J, Zhao J, Zheng K, Liu A, Zhao L, Zhou X (2022) Metaptp: An adaptive meta-optimized model for personalized spatial trajectory prediction. SIGKDD. NY, USA, New York, pp 2151–2159
23. Zhao J, Xu J, Zhou R, Zhao P, Liu C, Zhu F (2018) On prediction of user destination by sub-trajectory understanding: A deep learning based approach. In: CIKM. CIKM ’18, pp 1413–1422. Association for Computing Machinery, New York, NY, USA
24. Shang S, Chen L, Zheng K, Jensen CS, Wei Z, Kalnis P (2019) Parallel trajectory-to-location join. TKDE 31(6):1194–1207
25. Chen L, Shang S (2019) Approximate spatio-temporal top-k publish/subscribe. World Wide Web
26. Zheng K, Su H, Zheng B, Shang S, Xu J, Liu J, Zhou X (2015) Interactive top-k spatial keyword queries. In: ICDE, pp 423–434
27. Chen L, Shang S, Jensen CS, Xu J, Kalnis P, Yao B, Shao L (2020) Top-k term publish/subscribe for geo-textual data streams. The VLDB Journal 29:1101–1128
28. Chen L, Özsu MT, Oria V (2005) Robust and fast similarity search for moving object trajectories. In: SIGMOD, pp 491–502
29. Vlachos M, Gunopulos D, Kollios G (2002) Discovering similar multidimensional trajectories. In: Agrawal R, Dittrich KR (eds) ICDE, pp 673–684
30. Gerschner F, Paul J, Schmid L, Barthel N, Gouromichos V, Schmid F, Atzmueller M, Theissler A (2023) Domain transfer for surface defect detection using few-shot learning on scarce data. In: INDIN, pp 1–7
31. Zhang Q, Wu X, Yang Q, Zhang C, Zhang X (2022) Few-shot heterogeneous graph learning via cross-domain knowledge transfer. In: Zhang A, Rangwala H (eds) KDD, pp 2450–2460
32. Lin J, Wang Y, Chen Z, He T (2020) Learning to transfer: Unsupervised domain translation via meta-learning. In: AAAI, pp 11507–11514
33. Zhang W, Zhang P, Zhang B, Wang X, Wang D (2023) A collaborative transfer learning framework for cross-domain recommendation. In: Singh AK, Sun Y, Akoglu L, Gunopulos D, Yan X, Kumar R, Ozcan F, Ye J (eds) KDD, pp 5576–5585
34. Tay Y, Dehghani M, Bahri D, Metzler D (2023) Efficient transformers: A survey. ACM Comput Surv 55(6):109–110928
35. Tang J, Wang K (2018) Ranking distillation: Learning compact ranking models with high performance for recommender system. In: SIGKDD, pp 2289–2298
36. Rendle S, Freudenthaler C (2014) Improving pairwise learning for item recommendation from implicit feedback. In: Carterette B, Diaz F, Castillo C, Metzler D (eds) WSDM, pp 273–282
37. Moreira-Matias L, Gama JM, Ferreira M, Mendes-Moreira J, Damas L (2016) Time-evolving o-d matrix estimation using high-speed gps data streams. Expert Syst Appl Int J 44:275–288
38. Chang Y, Qi J, Liang Y, Tanin E (2023) Contrastive trajectory similarity learning with dual-feature attention. In: 2023 IEEE 39th International conference on data engineering (ICDE), pp 2933–2945. IEEE

Danling Lai received a BS degree from the School of Computer Science and Technology at Soochow University in 2020. She is currently a master's student at the Advanced Data Analytics Laboratory of Soochow University. Her research interests include trajectory data mining.

Jianfeng Qu received the BS, MS, and PhD degrees in computer science from Jilin University, Changchun, China, in 2013, 2016, and 2019, respectively. He is currently an associate professor with the School of Computer Science and Technology, Soochow University. His research interests include information extraction, data mining, natural language processing, and deep learning.

Yu Sang received the M.S degree from the Beijing City University, China, in 2018. From 2018 to 2021, she worked as a Research Assistant in the Advanced Data Analytics Laboratory of Soochow University. She is currently a doctoral student majoring in light industry information technology in Jiangnan University. Her research interest is recommender system.

Xi Chen received the bachelor's in computer sci ence from Northeastern University, China, in 2012, the master's degree in software engineer ing from Harbin Engineering University, China,in 2015, and the PhD degree in computer science from RMIT University, Australia, in 2022. She is a lecturer with the School of Computer Science and Technology, Soochow University, China. Her research interest lies in social media analysis and mining.