Towards Efficient Selection of Activity Trajectories Based on Diversity and Coverage

Chengcheng Yang[1], Lisi Chen[2], Hao Wang*[3], Shuo Shang*[2]
[1]East China Normal University
[2]University of Electronic Science and Technology of China
[3]Nanjing University of Information Science and Technology
ychengc@mail.ustc.edu.cn, {chenlisi.cs, haowang.paper, jedi.shang}@gmail.com

Abstract

With the prevalence of location based services, activity trajectories are being generated at a rapid pace. The activity trajectory data enriches traditional trajectory data with semantic activities of users, which not only shows where the users have been, but also the preference of users. However, the large volume of data is expensive for people to explore. To address this issue, we study the problem of Diversity-aware Activity Trajectory Selection (DaATS). Given a region of interest for a user, it finds a small number of representative activity trajectories that can provide the user with a broad coverage of different aspects of the region. The problem is challenging in both the efficiency of trajectory similarity computation and subset selection. To tackle the two challenges, we propose a novel solution by: (1) exploiting a deep metric learning method to speedup the similarity computation; and (2) proving that DaATS is an NP-hard problem, and developing an efficient approximation algorithm with performance guarantees. Experiments on two real-world datasets show that our proposal significantly outperforms state-of-the-art baselines.

Introduction

Numerous applications continuously generate massive amounts of activity trajectory data [Zheng 2015; Zhang et al. 2019; Chen et al. 2020], which augments traditional trajectory data with "activity" features. In this kind of data, each location point is associated with a keyword that semantically describes the venue of a performed activity, e.g., shop, restaurant, bank. From activity trajectory data, we can know not only where users have been, but also the main preferences of users by looking over the semantic descriptions. However, the availability of such large scale data makes the information prohibitively expensive to explore. In many real-world applications, it is of great significance to provide support for users to perform data exploration on their interested regions. For instance, users would like to browse a small number of representative tourist routes when planning trips to some tourism attractions.

Another example is the online map system [Ward, Grinstein, and Keim 2010]. For specified geographical area, users want to get intuitive information on various type of activity patterns based on data visualization functions. They would prefer receiving a small set of representative trajectories rather than being overwhelmed by a great many ones.

Based on above applications, we consider the problem of Diversity-aware Activity Trajectory Selection (DaATS), which aims to select a small set (denoted as k) of representative and diverse trajectories from the region that satisfies user-specified conditions. The trajectories in the result set are dissimilar to each other and should cover a maximum number of other ones that locate in the region. Result diversification has been considered as a good way to increase users' satisfaction in web search and recommendation systems [Vee et al. 2008; He et al. 2012; Ashkan et al. 2015; Parambath, Usunier, and Grandvalet 2016]. However, most of previous approaches pick objects based on scoring functions that are designed specifically for the underlying applications, and therefore do not apply to the DaATS problem. In this paper, we propose to address diversity and coverage in the perspective of similarity threshold. We consider two trajectories τ_1 and τ_2 to be dissimilar if their similarity is lower than a threshold θ. On the contrary, τ_1 covers τ_2 (and vice versa) if their similarity exceeds θ.

Solving the DaATS problem faces two challenges. The first is the high computation cost of trajectory similarity. Existing similarity measures [Shang et al. 2017; Chen et al. 2020] usually utilize a scan-and-align method to match the sampled points in two trajectories, which incurs quadratic time complexity. The second challenge is the efficiency of the subset selection. One naive idea is to enumerate all their subsets of size k and take the subset with maximum coverage as the result. However, there is a huge number of subsets and it is prohibitively expensive to exhaustively compute the coverage of each subset.

In this paper, we present a novel solution for the DaATS problem. To address the first challenge, we leverage deep metric learning to embed arbitrary-length trajectories into fixed-size vectors, which is capable of computing the similarity in linear time complexity. We propose to adopt the bidirectional LSTMs (BiLSTM) with mean-pooling and inter-trajectory attention (ITA) module to produce high-quality embeddings. Specifically, the BiLSTM structure captures the dependencies between adjacent points in each trajectory. The ITA module utilizes the memory network, trajectory clustering and attention mechanisms to acquire

*Hao Wang and Shuo Shang are corresponding authors.

the correlations between trajectories. To address the second challenge, we define the problem formally and prove that the problem is NP-hard. It indicates that obtaining the result in polynomial time is non-trivial. Because of the hardness of the problem, we propose a greedy based approximation method with a suite of optimization techniques. We also provide lower bounds of the approximation ratio, which compares the coverage of result produced by our method with the optimal coverage. In summary, we make the following contributions:

- We study the DaATS problem, and propose an efficient and effective approximation solution by means of deep metric learning and heuristic optimizations.
- We develop a deep metric learning method that considers both intra- and inter-trajectory correlations.
- We devise an efficient algorithm for the subset selection. It has theoretical guarantees on the approximation ratio.
- We conduct experiments on two real-world datasets. The experimental results demonstrate the efficiency and effectiveness of our proposal.

Related Work

Deep Metric Learning. Deep metric learning has been successful in many applications. Mueller et al. [Mueller and Thyagarajan 2016], Pei et al. [Pei, Tax, and Maaten 2016], Tolosana et al. [Tolosana et al. 2018], Coskun et al. [Coskun et al. 2018], and Yao et al. [Yao et al. 2019] utilize the Siamese recurrent networks to learn similarity metrics for different tasks, e.g., sequence classifications and spatial distance estimations. Liu et al. [Liu, Zhao, and Cong 2018] develop a metric learning method for comparing the spatial-textual relevance between two regions. Li et al. [Li et al. 2018] and Zhang et al. [Zhang et al. 2019] utilize the Seq2Seq model to learn similarity metrics for trajectories with low data quality. Yao et al. [Yao et al. 2019] add a spatial gate to the LSTM cell for spatial trajectory distance estimations. Wang et al. [Wang et al. 2019] employ metric learning to accelerate multi-trajectory similarity computation.

Result Diversification. The MaxMin and MaxSum diversification [Drosou and Pitoura 2014] select k out of a set of n items such that the minimum or the average distance between the items are maximized. Drosou et al. [Drosou and Pitoura 2012] introduce the dynamic diversification to support interactive operations. Guo et al. [Guo et al. 2018] considers the visibility for visualized explorations on geospatial data. Besides focusing on the diversity aspect, some studies find a subset of items based on scoring functions that consider both relevance and diversity [Angel and Koudas 2011; Ashkan et al. 2015; Chen and Cong 2015].

Preliminaries

Activity Trajectory. The activity trajectory $\tau = \langle p_1, \cdots, p_n \rangle$ is defined as a sequence of geo-textual points, where $p_i = (l_i, \psi_i)$, with $l_i = (x_i, y_i)$ being the location of ith point and ψ_i being the keyword that describes the activity venue.

Similarity Measures. We consider both spatial proximity and semantic similarity when measuring trajectory similarities, and use the symmetrical method proposed in [Shang et al. 2017; Chen et al. 2020] to match point pairs continuously. Given a geo-textual point p and an activity trajectory τ, the spatio-textual relevance between them are defined as:

$$r(p, \tau) = \max_{p_i \in \tau} \{ST(p, p_i)\}$$

where $ST(p, p_i)$ denotes the similarity between p and p_i, which is computed by a linear combination of their spatial proximity and textual (semantic) similarity [Lu et al. 2014]:

$$ST(p_i, p_j) = \alpha \cdot S(l_i, l_j) + (1 - \alpha) \cdot T(\psi_i, \psi_j).$$

where $S(l_i, l_j)$ denotes the normalized spatial proximity [Shang et al. 2017] between l_i and l_j, which is inversely proportional to their distance, and $T(\psi_i, \psi_j)$ denotes the normalized textual similarity between ψ_i and ψ_j. In this paper, we use the cosine similarity of Glove word vectors to measure the textual similarity. Finally, we define the similarity between two trajectories τ_1 and τ_2 as follows:

$$Sim(\tau_1, \tau_2) = \frac{1}{2}\left(\frac{\sum_{p_i \in \tau_1} r(p_i, \tau_2)}{|\tau_1|} + \frac{\sum_{p_j \in \tau_2} r(p_j, \tau_1)}{|\tau_2|}\right)$$

Problem Definition. Given a user-specified region ζ, a similarity threshold θ, and an integer k of the result size, let S be a set of trajectories located in ζ. The DaATS problem finds a subset R of S such that: (1) $|R| = k$; (2) $\forall \tau_i, \tau_j \in R$, $Sim(\tau_i, \tau_j) \leq \theta$; and (3) the number of trajectories in S covered by R is maximized. For any $\tau \in S$, we consider R covers τ if there exists a $\tau' \in R$ such that $Sim(\tau, \tau') > \theta$. Formally, we define the coverage of R for S as follows:

$$\mathcal{C}(R, S) = |\{\tau | \tau \in S \wedge \exists \tau' \in R \; s.t. \; Sim(\tau, \tau') > \theta\}|$$

Trajectory Similarity Learning

In this section, we aim to learn a neural network that (1) creates simple vector representations for arbitrary-length trajectories; and (2) well approximates the similarity measures.

Model

An overview of our proposed model is shown in Fig. 1. It relies on two parts to model a trajectory: (1) *Bidirectional recurrent neural network*. In many practical scenarios, objects usually move in spatial networks with certain structural characteristics. As a result, there exists some dependency between adjacent points within one trajectory due to the constraints of networks. To capture this dependency, we propose to use the bidirectional LSTM (BiLSTM) structure because it has a good grasp on the contextual information; (2) *Inter-trajectory attention (ITA) module*. We cluster trajectories according to their similarities and use memory tensors to store their summary information over two directions. Then, we exploit an attention based method to get the correlations between trajectories.

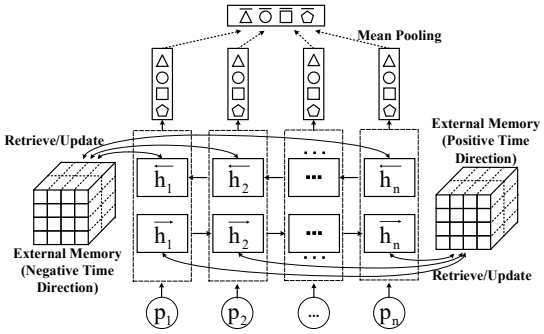


Figure 1: Structure of the trajectory embedding model.

Given a trajectory $\tau = \langle p_1, \cdots, p_n \rangle$, we use an ITA-augmented BiLSTM to process it in two opposite directions:

$$\overrightarrow{h_t} = \overrightarrow{ITALSTM}(p_t, \overrightarrow{h_{t-1}});$$
$$\overleftarrow{h_t} = \overleftarrow{ITALSTM}(p_t, \overleftarrow{h_{t+1}}).$$

We concatenate $\overrightarrow{h_t}$ and $\overleftarrow{h_t}$ to obtain the hidden state $h_t = [\overrightarrow{h_t}, \overleftarrow{h_t}]$. To produce a fixed-size trajectory embedding, we consider four ways of combing various number of $\{h_t\}_{t=1,\cdots,n}$, namely max-pooling, mean-pooling, self-attention, and using the last hidden state of both directions. The max-pooling selects the max value for each dimension of $\{h_t\}_t$, while the mean-pooling summarizes the average presence of the hidden states. The self-attention method extracts multiple views of the input trajectory, which represent different moving behaviors contained in it. Each view is computed by a linear combination of the hidden states:

$$H = (h_1, \cdots, h_n);$$
$$V = softmax(W_2 tanh(W_1 H^T))H$$

where $W_1 \in \mathbb{R}^{d' \times 2d}$, $W_2 \in \mathbb{R}^{\mu \times d'}$, d is the hidden size for each unidirectional LSTM, d' is a hyperparameter, and μ is the number of extracted views. The softmax function is executed on the second dimension of the input data. In addition, the final representation is the concatenation of all views. In our experiment, we find that the mean-pooling gives the highest accuracy of similarity approximation.

Next we introduce the ITA module. As Fig. 1 shows, at each time step t, the goal of ITA module is to extract relevant information from the similar neighbors of current trajectory to help the encoding procedure. Inspired by the spatial distance estimation method [Yao et al. 2019], we propose to use two memory tensors to store the encoded information of all trajectories over two directions. One naive idea is to store their encoded information of all time steps. However, it would lead to a huge amount of memory consumption. To address this issue, we propose a trajectory clustering and segmentation based method. Specifically, we first partition the spatial space into grid cells. Then, each trajectory can be segmented by the boundary of grid cells. That is, the segment number of a trajectory is exactly the number of grid cells it has passed through. For each grid cell, we use a greedy clustering algorithm [Zhang et al. 2017] to group its trajectory segments based on the similarities between the associated trajectories. We store two d-dimensional vector representations for each group in the two memory tensors. Each vector holds summary information of the group of trajectories in one direction. In addition, to moderate the size of memory tensors, we also define a parameter γ to restrict the maximum number of groups in each grid cell.

More specifically, take the positive time direction as an example, the state update of a LSTM unit [Hochreiter and Schmidhuber 1997] are adapted as follows.

$$(f_t, i_t, o_t) = \sigma(W_g \cdot p_t + U_g \cdot h_{t-1} + A_g \cdot a_t + b_g);$$
$$c_t = f_t \cdot c_{t-1} + i_t \cdot tanh(W_c \cdot p_t + U_c \cdot h_{t-1} + A_c \cdot a_t + b_c);$$
$$h_t = o_t \cdot tanh(c_t)$$

where $W_g \in \mathbb{R}^{3d \times (d'+2)}$, $U_g, A_g \in \mathbb{R}^{3d \times d}$, $W_c \in \mathbb{R}^{d \times (d'+2)}$, $U_c, A_c \in \mathbb{R}^{d \times d}$. d is the hidden state size and d' is the word embedding size. It is worth noting that ITA module takes advantage of the attention vector $a_t \in \mathbb{R}^d$ to help encode the trajectory. The attention mechanism adaptively selects relevant information from other similar trajectories to derive a_t. Specifically, at each time step t, we retrieve K grid cells that are close to l_t (recall that l_t is the location of p_t) and get a set of vector representations $\mathcal{V}_t$. Each $v_i \in \mathcal{V}_t$ corresponds to the vector representation of a trajectory group that resides in the retrieved grid cells. Then, the attention vector a_t is computed as a weighted sum of v_i.

$$u_t^i = m^T tanh(W_a \cdot p_t + U_a \cdot h_{t-1} + Q_a \cdot v_i + b_a);$$
$$\beta_t^i = \frac{exp(u_t^i)}{\sum_{j=1}^{|\mathcal{V}_t|} exp(u_t^j)}; \quad a_t = \sum_{i=1}^{|\mathcal{V}_t|} \beta_t^i v_i$$

where $m^T, b_a \in \mathbb{R}^d$, $W_a \in \mathbb{R}^{d \times (d'+2)}$ and $U_a, Q_a \in \mathbb{R}^{d \times d}$ are network parameters. The attention weight is jointly determined by the previous hidden state h_{t-1}, the input point p_t and the retrieved vector representation v_i. This weight represents the influence of each v_i.

During the training process, we encode the information of newly processed trajectory into the two representation vectors after processing each point p_t. Suppose p_t is contained in the group g_i of trajectory segments and v_i is the associated vector representation of g_i, we update v_i as follows:

$$z_t = \sigma(W_z \cdot p_t + U_z \cdot h_{t-1} + A_z \cdot a_t + b_z);$$
$$v_i^{new} = (1 - z_t) v_i^{old} + z_t h_t$$

where $W_z \in \mathbb{R}^{d \times (d'+2)}$, $A_z, U_z \in \mathbb{R}^{d \times d}$ and $b_z \in \mathbb{R}^d$.

Complexity Analysis. The time complexity of directly computing the trajectory similarity is $\mathcal{O}((d'+2)\bar{l}^2)$, where $\bar{l}$ is the average trajectory length. Our embedding based method reduces the time complexity to $\mathcal{O}(d)$.

Training Methods

For any trajectories τ_1 and τ_2, our model independently projects them into two vectors τ_1^E and τ_2^E. We learn network parameters so that the distance between the generated embeddings ($D(\tau_1, \tau_2) = e^{-||\tau_1^E - \tau_2^E||_2}$) approximates the similarity. However, it is computationally prohibitive to fit the similarities between all trajectory pairs. Inspired by the negative sampling [Mikolov et al. 2013], for each trajectory τ, we sample η similar and dissimilar trajectories (denoted as $\mathcal{T}_s$ and $\mathcal{T}_d$) from the training data to form positive and

negative pairs, respectively. Then, we set the loss function as the weighted sum of mean squared errors (MSE) based on the real similarity between sampled pairs.

$$\mathcal{L} = \sum_{\tau_s \in \mathcal{T}_s} Sim(\tau, \tau_s)(Sim(\tau, \tau_s) - D(\tau, \tau_s))^2 + \sum_{\tau_d \in \mathcal{T}_d} Sim(\tau, \tau_d)(Sim(\tau, \tau_d) - D(\tau, \tau_d))^2$$

Diversity-Aware Activity Trajectory Selection

We first prove that the DaATS problem is NP-hard. Then, we propose a greedy algorithm with a suite of optimizations. Finally, we give the approximation ratio analysis.

NP-hardness of the DaATS Problem

Theorem 1. *The DaATS problem is NP-hard.*

Proof. We reduce the DaATS problem from the minimum independent dominating set (MIDS) problem [Garey and Johnson 1979], which finds a minimum subset D of V for a graph $G = (V, E)$ such that: (1) $\forall v_i, v_j \in D$, $v_i \notin Neighbors(v_j)$; and (2) $\forall v_i \in V$, $v_i \in D$ or $\exists v_j \in D\ s.t.\ v_i \in Neighbors(v_j)$. We build a DaATS problem to solve the decision problem of MIDS, which answers whether there exists a result set with the size no more than k. Given a graph G and a set S of trajectories, we map each $\tau_i \in S$ to a vertex $v_i \in V$. If there exists an edge between v_i and v_j, we set $||\tau_i^E - \tau_j^E||_2 < -ln\theta$ (i.e., $e^{-||\tau_i^E - \tau_j^E||_2} > \theta$). Otherwise, we set $||\tau_i^E - \tau_j^E||_2 \geq -ln\theta$.

Let $R = \{\tau_1, \cdots, \tau_k\}$ be the result of a given DaATS instance, and $D = \{v_1, \cdots, v_k\}$ be the mapped vertices in G. Suppose that we have $\mathcal{C}(R, S) = |S|$, from preliminaries section, we can infer that for any $\tau \in S$, we have either $\tau \in R$, or τ is covered by R (i.e., $\exists \tau' \in R\ s.t.\ ||\tau^E - \tau'^E||_2 < -ln\theta$). On the other hand, considering the mapped graph G, it's obvious that for any $v_i, v_j \in D$, we have $(v_i, v_j) \notin E$. In addition, for any $v_i \in V$, we have: (1) $v_i \in D$; or (2) $\exists v_j \in D,\ (v_i, v_j) \in E$. Thus, D is the result of the decision problem of MIDS. Since MIDS is NP-hard, the theorem is proved. □

Greedy Based Algorithm

Since the DaATS problem is NP-hard, we resort to a greedy heuristic. We consider two trajectories have a neighborhood relationship if they are similar. Given a set S of trajectories, we iteratively finds the "best candidate" that has the maximum number of uncovered neighbors. To meet the requirements of diversity, after adding a new candidate τ_{new} to the result set R, we remove the uncovered neighbors that are similar to τ_{new} from S. Once we have found k results, we terminate the algorithm. The main technical challenge here is how to find the "best candidate" in each iteration. A straight forward solution is to calculate the number of uncovered neighbors for each $\tau \in S \backslash R$. However, the computation cost is prohibitively expensive. To address this issue, we propose a suite of optimization techniques based on the following lemma.

Lemma 1. *Let M and N be two sets of trajectories where $M \subseteq N$, and τ_{new} is a new trajectory. Let M' denote M inserted with τ_{new} and N' denote N inserted with τ_{new}. Then we have $\mathcal{C}(M', S) - \mathcal{C}(M, S) \geq \mathcal{C}(N', S) - \mathcal{C}(N, S)$.*

Proof. Suppose that the coverage increases by l after inserting τ_{new} into M, and $L = \{\tau'_1, \cdots, \tau'_l\}$ is the set of newly covered trajectories. For any $\tau \in N \backslash M$, if it does not cover any $\tau' \in L$, then we have $\mathcal{C}(N', S) - \mathcal{C}(N, S) = \mathcal{C}(M', S) - \mathcal{C}(M, S) = l$. Otherwise, some trajectories in L have been covered by N, thus we have $\mathcal{C}(N', S) - \mathcal{C}(N, S) < \mathcal{C}(M', S) - \mathcal{C}(M, S) = l$. Together, the lemma is proved. □

From Lemma 1, we can see that, given any trajectory, the number of its uncovered neighbors will decrease when we proceed the algorithm. This motivates us to design an upper bound based pruning technique, which recalculates the number of uncovered neighbors for a few promising candidates after adding a new result.

Upper Bound Based Pruning. We create a triple entry $\langle \tau, \delta(\tau), \mathcal{I}(\tau) \rangle$ for each trajectory τ, where $\delta(\tau)$ is the number of its uncovered neighbors, and $\mathcal{I}(\tau)$ is the result size when $\delta(\tau)$ is computed. Note, according to Lemma 1, $\delta(\tau)$ can be regarded as the upper bound on the number of τ's uncovered neighbors for $|R| > \mathcal{I}(\tau)$. To find a candidate with the maximum number of uncovered neighbors, we visit the triple entries in descending order of $\delta(\tau)$. A max-heap $\mathcal{H}$ is used to manage all the entries and a "lazy" update strategy [Zhou et al. 2013] is applied. When selecting a new candidate, we pop the heap to get a trajectory τ^{top} with the globally largest $\delta(\tau^{top})$. If $\mathcal{I}(\tau^{top}) < |R|$, we recalculate $\delta(\tau^{top})$ by checking its neighbors. This is because $\delta(\tau^{top})$ is evaluated when the result size is $\mathcal{I}(\tau^{top})$. Next, we set $\mathcal{I}(\tau^{top})$ as $|R|$ and reinsert it to $\mathcal{H}$. We repeat this until $\delta(\tau^{top})$ is evaluated regarding current result set. Then, we add τ^{top} to the result set since it has a larger number of uncovered neighbors than any of the remaining ones in $\mathcal{H}$.

Data Partition. The heap based method needs to maintain a triple entry for each $\tau \in S$. Thus, the heap size is $|S|$, which is relatively large and leads to high cost of heap adjustments (the time cost is $log|S|$ for each heap adjustment). To address this issue, we propose a partition based method. Recall that the number of uncovered neighbors for all trajectories will decrease as we proceed the algorithm, and we are interested in a few candidates with the largest number of uncovered neighbors in each iteration. Thus, we propose to partition the candidates by the range of their recorded number of uncovered neighbors, and then process each range in descending order. Let $\mathcal{I}_{max}$ and $\mathcal{I}_{min}$ be the maximum and minimum number of uncovered neighbors for all trajectories in iteration 0. We partition the range $[\mathcal{I}_{min}, \mathcal{I}_{max}]$ into equal intervals. Specifically, we set the interval width as 1 and build an inverted list for each partition. That is, all entries in an inverted list have the same recorded number of uncovered neighbors. With this method, we can perform each triple entry adjustment in $\mathcal{O}(1)$ time. Thus, the heap adjusting cost is eliminated.

Algorithm 1 presents the implementation of our method. We first build a graph G by mapping each $\tau_i \in S$ to a vertex

$v_i \in V$ and adding edges based on their similarities (Line 1). Clearly, the coverage of a trajectory is exactly the degree of the mapped vertex. For ease of presentation, we call a vertex *black* if it is in the result set R or covered by R. Otherwise, we call the vertex *white*. Initially, R is empty and all vertices are white (Line 2). Then we partition the vertices (Line 3), build inverted lists for each partition (Line 4∼6), and select the subset iteratively (Line 7∼23). For each candidate selection, we get an entry t out of the inverted list that has the largest recorded number of white neighbors (Line 9). If $t.v$ is black, we directly drop it because it is similar to some trajectories in R. If $t.v$ is white, we check whether $t.\delta(v)$ is computed regarding current result set (Line 12). If not, we update it and reinsert t into the corresponding inverted list (Line 13∼14). Otherwise, we add $t.v$ to the result set and color black all vertices covered by it (Line 16∼17). The iteration is repeated until we have selected k diverse results or all vertices have been colored black.

Complexity Analysis. The time complexity of building the mapped graph is $\mathcal{O}(n^2)$, where $n = |S|$. The time complexity of the subset selection is $\mathcal{O}(n_s \cdot n_i)$, where n_s is the average neighborhood size and n_i is the total number of trajectories whose uncovered neighbors are computed. Note, n_s and n_i are usually much smaller than n. In practice, suppose the minimum similarity threshold a system can support is θ' (e,g., 0.7), we can pre-compute the neighbors for each trajectory regarding θ'. In this way, for any user specified threshold θ ($\theta \geq \theta'$), we do not have to check the similarity of all trajectory pairs to build the graph.

Algorithm 1: GreedyTrajectorySelection(S, k, θ)

```
1  Build a graph G = (V, E) using the method described in
   Theorem 1;
2  Set R as empty and color all vertices in V white;
3  Partition vertices based on their degrees;
4  foreach vertex v ∈ V do
5  |  Put ⟨v, v.degree, 0⟩ into the inverted list L_{v.degree};
6  end
7  L ← pick a non-empty inverted list with the largest recorded
   number of white neighbors;
8  while L ≠ ∅ and |R| < k do
9  |  t ← get a triple entry out of L;
10 |  if t.v is white then
11 |  |  N^W(t.v) ← the set of the white neighbors of t.v;
12 |  |  if t.I(v) < |R| then
13 |  |  |  t.δ(v) ← |N^W(t.v)|;
14 |  |  |  Put ⟨t.v, t.δ(v), |R|⟩ into the inverted list
   |  |  |  L_{t.δ(v)};
15 |  |  else
16 |  |  |  R ← R ∪ {t.v};
17 |  |  |  color t.v and each v' ∈ N^W(t.v) black;
18 |  |  end
19 |  end
20 |  if L = ∅ and there exists a non-empty inverted list then
21 |  |  L ← pick a non-empty inverted list with the largest
   |  |  recorded number of white neighbors;
22 |  end
23 end
24 return R;
```

Approximation Ratio Analysis

We first introduce the following lemma, which will help with the approximation ratio analysis.

Lemma 2. *Given two d-dimensional vertices v_i and v_j, we call v_i conflicts with v_j if v_i is covered by v_j (i.e., $||v_i - v_j||_2 < -ln\theta$). Let R be a diverse vertex set in which no two vertices conflict with each other. Given a vertex $v \notin R$, at most C_d vertices in R conflict with v, where C_d is a function of the dimensionality d and independent of $|R|$.*

Proof. If $v_1, v_2 \in R$ conflict with v, then v_1 and v_2 must be inside the sphere $\mathbb{S}_v$ which is centered at v with radius $-ln\theta$. Because v_1 and v_2 do not conflict with each other, we have $||v_1 - v_2||_2 \geq -ln\theta$. Obviously, the angle between vv_1 and vv_2 is at least $60°$. Next, we prove that there are at most C_d rays sharing the same initial vertex in d-dimensional space, where the angle between any two rays is at least $60°$.

For any $\vartheta \in (0, \pi/3]$, previous study [Yao 1982] has shown that the space E^d can be covered with a certain number of convex cones $\mathbb{V} = \{V_1, \cdots, V_{C_d}\}$ such that: (1) all convex cones share the same initial vertex v; and (2) each convex cone's angular diameter is smaller than ϑ. Let $\mathbb{R}_v = \{vu_1, \cdots, vu_{C_d+1}\}$ be any $C_d + 1$ rays sharing the same initial vertex v. Based on the pigeonhole principle, at least two rays are contained in the same convex cone. Suppose that vu_1 and vu_2 lie in the same convex V_1, we have $\angle u_1 v u_2 < 60°$ because the angular diameter of V_1 is smaller than $\pi/3$. For any two rays vu_i and vu_j, if the inequality $\angle u_i v u_j \geq 60°$ always holds, then the number of rays must be smaller than $C_d + 1$, where C_d is a function of the dimensionality d. Note, we can further prove that $C_d = \mathcal{O}(d^2)$. This is because there are at most $\mathcal{O}(d^2)$ rays sharing the same initial vertex in d-dimensional space, where the angle between any two rays is at least $45°$.

Given the above, the lemma is proved. □

From Lemma 2, we have the approximation ratio of our greedy algorithm as follows. Due to space constraints, we leave the full proof to our technical report.

Theorem 2. *The approximation ratio of our greedy algorithm is at least $\frac{1}{C_d+1}$.*

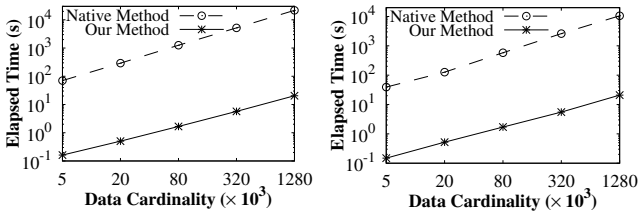
Experiment

Experimental Setup

Dataset. We experimented on two real-world datasets: T-Drive [Yuan et al. 2011] and NYCTL [Donovan and Work 2015]. The dataset T-Drive consists of taxi trajectories in Beijing. The original trajectories are very long (the average length is $1,450$) and usually last days. To create trips with realistic duration, we divided them into 0.52 million hour-long sub-trajectories. The dataset NYCTL covers four years of taxi trips in NYC. We selected 0.5 million trips from the dataset and used the New York Road Network to infer the routes of each trip. In addition, we used real-world POI data [Yang et al. 2015; Center 2017] to map each POI

Methods	T-Drive		NYCTL	
	HR@50	R10@50	HR@50	R10@50
LSTM	0.4965	0.7386	0.5861	0.8179
LSTM + ITA	0.5197	0.7584	0.6150	0.8299
BiLSTM + Last	0.4991	0.7398	0.5899	0.8188
BiLSTM + Self-Attention	0.4874	0.7299	0.5784	0.8104
BiLSTM + Max-Pooling	0.4695	0.7192	0.5627	0.7996
BiLSTM + Mean-Pooling	0.5028	0.7425	0.5933	0.8226
Our Model + Last	0.5259	0.7598	0.6184	0.8316
Our Model + Self-Attention	0.5131	0.746	0.5952	0.8197
Our Model + Max-Pooling	0.4953	0.7396	0.5838	0.8101
Our Model + Mean-Pooling	**0.5301**	**0.7626**	**0.6216**	**0.8365**

Table 1: Comparing different deep metric learning methods.



(a) T-Drive (b) NYCTL

Figure 2: Comparing the similarity computation efficiency.

to the nearest trajectory point, which can describe the activity of associated trajectory. We chosen the trajectories in central city and removed trajectories whose points are less than 10. Finally, we got 0.29/0.17 million trajectories in T-Drive/NYCTL.

Baselines. We compared our deep metric learning method with the following baselines:

(1) *LSTM*. The last d-dimensional hidden state was selected as the trajectory embedding. Generally, the accuracy of similarity approximation first improves with the increase of d and then levels off when d is large enough. In contrast, the efficiency decreases with the increase of d since the time complexity is $\mathcal{O}(d)$. We tuned d using grid search and set $d = 512$ because it has stable performance.

(2) *BiLSTM based methods*. The BiLSTM computes a set of 512-d hidden states in each direction. We generated 4 baselines with different ways of combing the hidden states. For self-attention, we searched hyperparameters in a wide range and found that $d' = 600$ and $u = 5$ worked best.

(3) *Variants of our method.* We also experimented with 4 ways of combing the hidden states in our method, and used the same parameters as the BiLSTM based methods. In addition, we included the variant of *LSTM + ITA* as well. We tuned K with grid search and set $K = 9$ as it worked best.

For the subset selection problem, we compared our greedy method with the following baselines:

(1) *MaxMin*, *MaxSum*. The two methods [Drosou and Pitoura 2014] are commonly used to find a set of most diverse trajectories. Note, they may return a result set without fulfilling the diversity constraint stated in our problem definition.

(2) *Random*. We repeatedly picked a trajectory τ if adding τ to the result set does not break the diversity constraint.

Settings. To moderate the size of memory tensors, we divided the space into $200m \times 200m$ grid cells and set the maximum number of groups γ in each grid cell as 6. Due to the numerous number of trajectories, it's impractical to compute the similarity of all trajectory pairs. Thus we randomly chosen $15k$ trajectories to compute the ground truth. In addition, 30%/10%/60% of the ground truth data was used for training/parameter tuning/testing. The sampling size η was set as 10. For the DaATS problem, we used the whole dataset and set the similarity threshold θ as 0.8. We randomly sampled 100 square-shape regions as the explored regions. By default, we set the region size as 0.01 of the city size and selected a subset of size 100.

Evaluation Metrics. We studied the top-k similarity search problem, and used two metrics to evaluate the effectiveness of metric learning methods: (1) $HR@50$. It measures the overlap of the ground truth and the top-50 results; (2) $R10@50$. It measures how many results in top-10 ground truth can be found from the top-50 results. For the methods of subset selection, we used CPU time and coverage ratio (defined as $\mathcal{C}(R, S)/|S|$) to evaluate their efficiency and effectiveness.

Environment. We conducted experiments on a workstation powered by Intel Xeon Gold-6148 CPU on Linux (Ubuntu 16.04), having a Nvidia Titan Xp GPU.

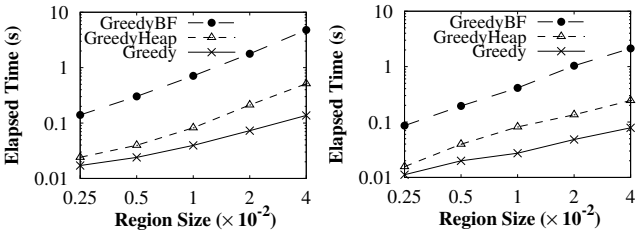
Evaluating the Deep Metric Learning Methods

Overall effectiveness. Table 1 shows the effectiveness performance comparison. We observed that our method significantly outperformed the baselines on both datasets. This is because our method considers both intra- and inter-trajectory correlations, which is very helpful to generate high-quality embeddings. Our method improved other methods by up to 6.8%. For example, on T-Drive dataset, the $HR@50$ of LSTM and our method were 0.4965 and 0.5301, respectively. Note that the self-attention and max-pooling based methods had unsatisfactory performance. It might be because they over-specialize to the training data without capturing the general information of input trajectories.

Overall efficiency. We compared our method with the native method that directly computes the exact similarity. As all metric learning methods have the same time complexity $\mathcal{O}(d)$, we didn't include the result of other baselines. Figure 2 shows their run time for computing the similarity of varying number of trajectory pairs. We observed that our method always achieved the best performance and improved the native method by up to 1060×. Moreover, we can see that our method had steady performance. This is because the fixed-size embedding has the advantage of stable time cost for similarity computation.

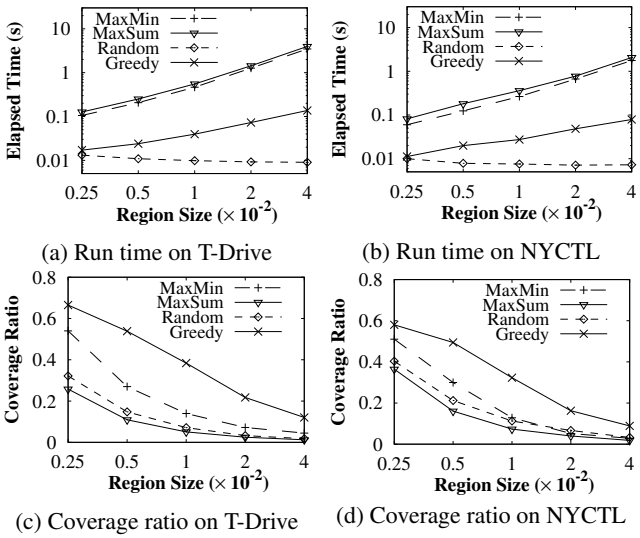
Evaluating the Greedy Based Method

Evaluating Proposed Techniques. We implemented the following methods to evaluate the efficiency of our proposed techniques: (1) GreedyBF (brute force) recalculates the number of uncovered neighbors for all trajectories in each iteration; (2) GreedyHeap improves GreedyBF with the



(a) T-Drive (b) NYCTL

Figure 3: Evaluation of proposed techniques.



(a) Run time on T-Drive (b) Run time on NYCTL

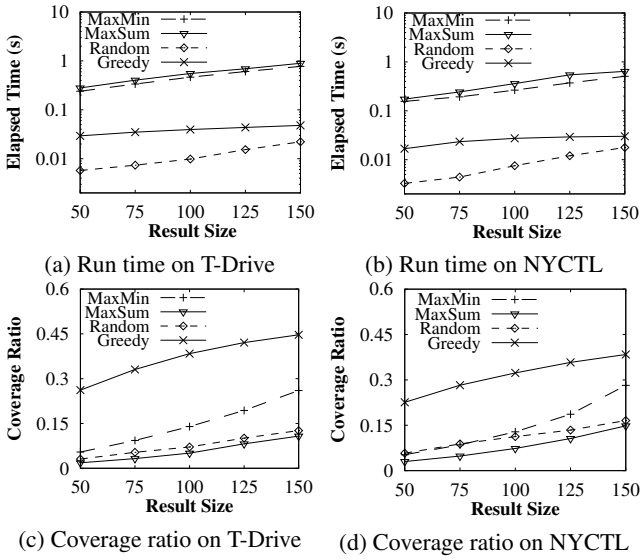
(c) Coverage ratio on T-Drive (d) Coverage ratio on NYCTL

Figure 4: Comparing with baselines under different region size.

upper bound based pruning; (3) Greedy (our method) integrates the data partition method into GreedyHeap. Figure 3 shows the result on different region size. We can see that GreedyHeap outperformed GreedyBF by up to 8.7× when the region size was large. This is because the upper bound based pruning technique can reduce the computation cost by skipping unpromising candidates. We also observed that the partition technique could help improve the performance. This is because it eliminates the cost of heap adjustments.

Effect of the region size. Figure 4 depicts the effect of scaling the explored regions. We observed that our method showed good scalability. It always achieved the highest coverage ratio and had comparable run time with the Random method. This is because our method always seeks to retrieve the trajectory with maximum increment of coverage. Moreover, the proposed optimization techniques can significantly improve the efficiency of subset selection. Note that the coverage ratio of MaxSum was lower than Random. The main reason is that it tends to focus on the boundary or sparse area where the trajectories are farthest to all of the others.

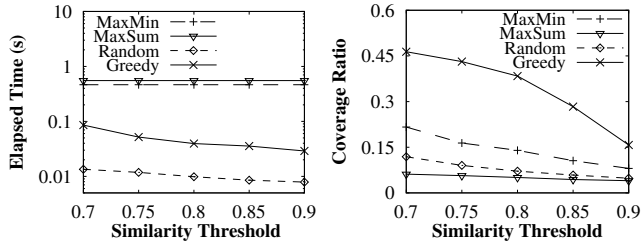
Effect of the result size. Figure 5 displays the effect of the result size. We observed that our method scaled very well when the result size was increased. We also observed that our method was able to keep a high coverage ratio under



(a) Run time on T-Drive (b) Run time on NYCTL

(c) Coverage ratio on T-Drive (d) Coverage ratio on NYCTL

Figure 5: Comparing with baselines under different result size.



(a) Run time on T-Drive (b) Coverage ratio on T-Drive

Figure 6: Comparing with baselines under different value of θ.

different result size. This is due to the fact that our method has performance guarantees on the approximation ratio.

Effect of the similarity threshold. Lastly, we investigated the effect of similarity threshold θ. We only included the result on T-Drive dataset due to space constraints. Figure 6 shows the result. We can see than our method performed well under different value of θ. Our method outperformed MaxMin and MaxSum by up to 16× and 19× in terms of efficiency, and outperformed all the baselines by up to 7.6× in terms of coverage ratio.

Conclusion

In this paper, we studied the DaATS problem and proposed a novel approximation solution. To speed up the similarity computation, we developed a deep metric learning method to project the trajectories into fixed-size embeddings. For efficient and effective subset selection, we proposed a greedy algorithm with a suite of heuristic optimization techniques. The experimental results demonstrate the superiority of our proposal.

Acknowledgments

Shuo Shang is supported by NSFC U2001212, 62032001 and 61932004.

References

Angel, A.; and Koudas, N. 2011. Efficient diversity-aware search. In *SIGMOD*, 781–792.

Ashkan, A.; Kveton, B.; Berkovsky, S.; and Wen, Z. 2015. Optimal greedy diversity for recommendation. In *IJCAI*, 1742–1748.

Center, S. I. 2017. Map POI (Point of Interest) data. doi: 10.18170/DVN/WSXCNM. URL https://doi.org/10.18170/DVN/WSXCNM.

Chen, L.; and Cong, G. 2015. Diversity-aware top-k publish/subscribe for text stream. In *SIGMOD*, 347–362.

Chen, L.; Shang, S.; Jensen, C. S.; Yao, B.; and Kalnis, P. 2020. Parallel semantic trajectory similarity join. In *ICDE*, 997–1008.

Coskun, H.; Tan, D. J.; Conjeti, S.; Navab, N.; and Tombari, F. 2018. Human motion analysis with deep metric learning. In *ECCV*, 693–710.

Donovan, B.; and Work, D. B. 2015. Using coarse GPS data to quantify city-scale transportation system resilience to extreme events. *CoRR* abs/1507.06011.

Drosou, M.; and Pitoura, E. 2012. DisC diversity: result diversification based on dissimilarity and coverage. *PVLDB* 6(1): 13–24.

Drosou, M.; and Pitoura, E. 2014. Diverse set selection over dynamic data. *IEEE Trans. Knowl. Data Eng.* 26(5): 1102–1116.

Garey, M. R.; and Johnson, D. S. 1979. *Computers and intractability: a guide to the theory of NP-Completeness*. W. H. Freeman.

Guo, T.; Feng, K.; Cong, G.; and Bao, Z. 2018. Efficient selection of geospatial data on maps for interactive and visualized exploration. In *SIGMOD*, 567–582.

He, J.; Tong, H.; Mei, Q.; and Szymanski, B. K. 2012. GenDeR: a generic diversified ranking algorithm. In *NIPS*, 1151–1159.

Hochreiter, S.; and Schmidhuber, J. 1997. Long short-term memory. *Neural Computation* 9(8): 1735–1780.

Li, X.; Zhao, K.; Cong, G.; Jensen, C. S.; and Wei, W. 2018. Deep representation learning for trajectory similarity computation. In *ICDE*, 617–628.

Liu, Y.; Zhao, K.; and Cong, G. 2018. Efficient similar region search with deep metric learning. In *SIGKDD*, 1850–1859.

Lu, Y.; Lu, J.; Cong, G.; Wu, W.; and Shahabi, C. 2014. Efficient algorithms and cost models for reverse spatial-keyword *k*-nearest neighbor search. *ACM Trans. Database Syst.* 39(2): 13:1–13:46.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*, 3111–3119.

Mueller, J.; and Thyagarajan, A. 2016. Siamese recurrent architectures for learning sentence similarity. In *AAAI*, 2786–2792.

Parambath, S. P.; Usunier, N.; and Grandvalet, Y. 2016. A coverage-based approach to recommendation diversity on similarity Graph. In *RecSys*, 15–22.

Pei, W.; Tax, D. M. J.; and Maaten, L. V. D. 2016. Modeling time series similarity with siamese recurrent networks. *CoRR* abs/1603.04713.

Shang, S.; Chen, L.; Wei, Z.; Jensen, C. S.; Zheng, K.; and Kalnis, P. 2017. Trajectory similarity join in spatial networks. *PVLDB* 10(11): 1178–1189.

Tolosana, R.; Vera-Rodríguez, R.; Fiérrez, J.; and Ortega-Garcia, J. 2018. Exploring recurrent neural networks for on-line handwritten signature biometrics. *IEEE Access* 6: 5128–5138.

Vee, E.; Srivastava, U.; Shanmugasundaram, J.; Bhat, P.; and Amer-Yahia, S. 2008. Efficient computation of diverse query results. In *ICDE*, 228–236.

Wang, Z.; Long, C.; Cong, G.; and Ju, C. 2019. Effective and efficient sports play retrieval with deep representation learning. In *SIGKDD*, 499–509.

Ward, M. O.; Grinstein, G. G.; and Keim, D. A. 2010. *Interactive data visualization - foundations, techniques, and applications*. A K Peters.

Yang, D.; Zhang, D.; Zheng, V. W.; and Yu, Z. 2015. Modeling user activity preference by leveraging user spatial temporal characteristics in LBSNs. *IEEE Trans. Syst. Man Cybern. Syst.* 45(1): 129–142.

Yao, A. C. 1982. On constructing minimum spanning trees in k-dimensional spaces and related problems. *SIAM J. Comput.* 11(4): 721–736.

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 *ICDE*, 1358–1369.

Yuan, J.; Zheng, Y.; Xie, X.; and Sun, G. 2011. Driving with knowledge from the physical world. In *SIGKDD*, 316–324.

Zhang, Y.; Li, X.; Wang, J.; Zhang, Y.; Xing, C.; and Yuan, X. 2017. An efficient framework for exact set similarity search using tree structure indexes. In *ICDE*, 759–770.

Zhang, Y.; Liu, A.; Liu, G.; Li, Z.; and Li, Q. 2019. Deep representation learning of activity trajectory similarity computation. In *ICWS*, 312–319.

Zheng, Y. 2015. Trajectory data mining: an overview. *ACM TIST* 6(3): 29:1–29:41.

Zhou, C.; Zhang, P.; Guo, J.; Zhu, X.; and Guo, L. 2013. UBLF: an upper bound based approach to discover influential nodes in social networks. In *ICDM*, 907–916.