Semantic-aware Query Processing for Activity Trajectories

Huiwen Liu
[1] School of Computer Science and Technology, Soochow University
[2] Collaborative Innovation Center of Novel Software Technology and Industrialization
hwliu@stu.suda.edu.cn

Jiajie Xu
[1] School of Computer Science and Technology, Soochow University
[2] Collaborative Innovation Center of Novel Software Technology and Industrialization
xujj@suda.edu.cn

Kai Zheng
[1] School of Computer Science and Technology, Soochow University
[2] Collaborative Innovation Center of Novel Software Technology and Industrialization
zhengkai@suda.edu.cn

Chengfei Liu
Faculty of SET, Swinbourne Univesity of Technology
cliu@swin.edu.au

Lan Du
Monash University
dulan520@gmail.com

Xian Wu
School of Computer Science and Technology, Soochow University
wuxian@suda.edu.cn

ABSTRACT

Nowadays, users of social networks like tweets and weibo have generated massive geo-tagged records, and these records reveal their activities in the physical world together with spatio-temporal dynamics. Existing trajectory data management studies mainly focus on analyzing the spatio-temporal properties of trajectories, while leaving the understanding of their activities largely untouched. In this paper, we incorporate the semantic analysis of the activity information embedded in trajectories into query modelling and processing, with the aim of providing end users more accurate and meaningful trip recommendations. To this end, we propose a novel trajectory query that not only considers the spatio-temporal closeness but also, more importantly, leverages probabilistic topic modelling to capture the semantic relevance of the activities between data and query. To support efficient query processing, we design a novel hybrid index structure, namely ST-tree, to organize the trajectory points hierarchically, which enables us to prune the search space in spatial and topic dimensions simultaneously. The experimental results on real datasets demonstrate the efficiency and scalability of the proposed index structure and search algorithms.

CCS Concepts

•**Information systems** → **Information retrieval query processing; Evaluation of retrieval results;** *Similarity measures;*

Keywords

activity trajectories query; spatial keywords; semantic relevance

1. INTRODUCTION

WSDM '17, February 6–10, 2017, Cambridge, United Kingdom.

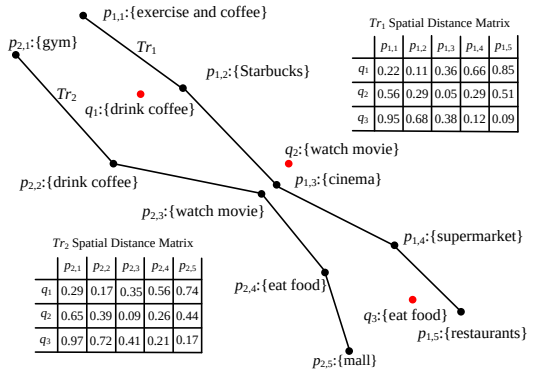
ISBN 978-1-4503-4675-7/17/02...$15.00

DOI: http://dx.doi.org/XXXX.XXXX

In recent years, the effectiveness of trajectory data management in the development of location based systems (LBS), such as trip planning and place recommendation systems, has been demonstrated in [5][16][10][35][29]. With the widespread use of smart phones nowadays, users from the social networks have generated massive geo-tagged records such as tweets and check-ins, which tell not only the spatio-temporal dynamics of users, but also their activities in the physical world. To model this enriched trajectory information, the concept of *activity trajectory* is first proposed in [33], where a set of keywords (describing user activities) is associated with each time-stamped location in the trajectory, and then a new activity trajectory query has been formulated [33], which requires the returned trajectories to match the keywords in query and be geographically close to the query positions simultaneously.

Existing activity trajectory querying approaches [33] require accurate keyword match, as a result, it tends to lead no or too few qualified results to be found. To address this problem, an effective alternative is to apply the approximate keyword search based on certain textual similarity measures [32]. This method can find trajectories that have similar spellings of activity terms with respect to the query, which increases the robustness of system for handling misspelling or conventional spelling difference (e.g., 'center' vs. 'centre'). However, it still cannot retrieve the activities which are synonyms but morphologically different, such as 'gym' and 'exercise'. This gap caused by the limited understanding of the activities in trajectory data motivates us to investigate other approaches aiming to capture the semantic relevance between activities.

Example 1: Considering the example in Figure 1, a tourist plans for a series of activities close to the locations q_1, q_2 and q_3 respectively and would like some recommendations based on the activities of other people in the surrounding area, such as Tr_1 and Tr_2, as reference. Conventional keyword based trajectory search methods [33][32] tend to return Tr_2 as result because the query keywords can be textually matched, though it is farther from $Q = \{q_1, q_2, q_3\}$ than Tr_1. However, if checking these two trajectories more carefully, we will find Tr_1 is a more relevant candidate to recommend since 'drink coffee', 'watch movie' and 'eat food' are very similar to 'Starbucks' (in $p_{1,2}$), 'cinema' (in $p_{1,3}$) and 'restaurants' (in $p_{1,5}$) in terms of their semantic relevance, though their spellings have nothing in common. In order to achieve a more rational trajectory recommendation, the key problem is to interpret



Tr_1	$p_{1,1}$	$p_{1,2}$	$p_{1,3}$	$p_{1,4}$	$p_{1,5}$
q_1	0.22	0.11	0.36	0.66	0.85
q_2	0.56	0.29	0.05	0.29	0.51
q_3	0.95	0.68	0.38	0.12	0.09

Tr_2	$p_{2,1}$	$p_{2,2}$	$p_{2,3}$	$p_{2,4}$	$p_{2,5}$
q_1	0.29	0.17	0.35	0.56	0.74
q_2	0.65	0.39	0.09	0.26	0.44
q_3	0.97	0.72	0.41	0.21	0.17

Figure 1: Example of user-oriented trajectory similarity query

and model user activities based on the keyword descriptions, and then take the semantic relevance of the activity into consideration when searching the trajectory database.

In this paper, we use probabilistic topic model (e.g., Latent Dirichlet Allocation [3][30], LDA in short), one of the proven successful techniques in machine learning area, to transform the activity descriptions in trajectories into their semantic representations, which can then be used to quantify the semantic correlation between activity descriptions. For instance in Figure 1, by applying LDA model [3], we can obtain the topic distribution of $p_{1,1}$ over five latent topics $(0.31, 0.03, 0.60, 0.03, 0.03)$ (in Table 3), in which each component indicates the relevance between activity description in $p_{1,1}$ and a topic. Analogously, we can get the topic distributions of all other trajectory points and all query points (in Table 3). Since the textual records of all trajectory points are now represented by high dimensional vectors in topic space, it is easy to measure their similarity based on the vectors. This approach has turned out to be an effective measure to evaluate the thematic relevance between web documents [26]. By incorporating the topic distance into the trajectory query processing, we can expect the results to have high spatial proximity and semantic correlation w.r.t. the given query.

Although the topic model based query is easy to understand, answering the query efficiently is challenging. The reasons can be summarized in three aspects. Firstly, compared to conventional keywords based trajectory search [33][32], the topic distribution based index method has higher dimensions of each trajectory point, which severely deteriorates pruning efficiency (also known as the 'curse of dimensionality' [19]) of most multi-dimensional search algorithms. Secondly, compared to conditional trajectory data which contain spatial information only, the vast majority of activity trajectories usually consist of several places spanning a large area, and such spatial dynamics tend to result in huge computational and I/O cost in retrieval. Last but not least, the query no longer tends to be a single point, but a few user specified locations of interest, which increases the search space of query processing greatly.

To address above difficulties for efficient query processing, we define a new type of trajectory similarity measure that considers both geometric distance and semantic relevance. Since Locality-Sensitive Hashing (LSH) [17] is a powerful tool for high dimensional search, which coincides to our topic embedded trajectory similarity measurement, we propose Quadtree and LSH combined hybrid index structure called ST-tree to integrate spatial and semantic information seamlessly. It keeps the advantage of hierarchial structure while avoiding the flaws of large dead space when indexing scalable trajectory points in high dimensional space. Particularly, we maintain a trajectory oriented multi-probe index scheme to reduce high memory cost caused by massive hash tables in LSH. On top of the index, a novel ST-tree based search algorithm is designed to retrieve huge volume of trajectories efficiently based on some theocratical bounds. To sum up, the major contributions of this paper can be briefly summarized as follows:

- We introduce and formalize a new type of probabilistic topic model based similarity measure between a query and a candidate trajectory, by which it is possible to find trajectories that can best satisfy the user intention.
- We propose a novel hierarchial index structure, namely IDQ-tree, to integrate the spatial and semantic information seamlessly. Moreover, an LSH based index called ST-tree is further designed for superior effectiveness on high dimensional similarity search.
- On top of the index structures, we develop efficient user oriented trajectory search algorithms, which greatly reduce the search space during trajectory search, and hence, significantly reduces the computational and I/O cost.
- We conduct an extensive experimental analysis based on real activity trajectory datasets, which includes the performance comparisons of the proposed algorithms. The experimental results demonstrate the efficiency of our proposed method.

The rest of paper is organized as follows. We define necessary concepts and formulate similarity query in Section 2. Section 3 presents baseline methods. Proposed index and search algorithm are discussed in Section 4 and Section 5. Section 6 reports the experimental observations. This paper is concluded in Section 8 after a brief review of related work in Section 7.

2. PROBLEM DEFINITION

In this section, we provide the necessary definitions and give the formal problem statement. Table 1 summarizes the notations used throughout the paper.

2.1 Activity Trajectory

Definition 1. (Activity Trajectory): An activity trajectory Tr is defined as a sequence of geo-spatial points related to textual records about user activities, i.e., $Tr = (p_1, p_2, ..., p_n)$. Each $p_i (1 \leq i \leq n)$ is a trajectory point in the form of $(l, tmsp, W)$, where l is the geographical location and $tmsp$ is the time-stamp; W is the textual record formed by a set of keywords in vocabulary V to describe user activities at location l. Essentially, an activity trajectory is a series of historical records describing what users did and where they did it. In the rest of the paper, we simply use *trajectory* to represent *activity trajectory* when no ambiguity.

2.2 Probabilistic Topic Model based Semantic Interpretation

Probabilistic topic model is a powerful tool for semantic interpretation of a large archive of documents over latent topics. By regarding the textual record in each trajectory point as a document and the thematic meanings as topics, we can apply a particular probabilistic topic model, e.g. Latent Dirichlet Allocation (LDA) [3] model, to derive textual record of each trajectory point as a distribution of topics. Here a topic can be understood as the semantic meaning of user activities as the following definition.

Definition 2. (Latent Topic): A latent topic (or topic in short) z represents a type of activity that a user can take at some sites of interest, such as 'exercise', 'food' and 'shopping'. We use the finite $\mathbb{Z}$ to denote a pre-processed topic set, which is the union of all the topics associated with user activity descriptions.

How to derive the set $\mathbb{Z}$ of latent topics is an interesting research problem but outside the scope of this paper, and we assume that a given number of latent topics have already been extracted by some topic modeling algorithms (e.g., LDA model [3]). Based on statistical concurrence, textual record of each trajectory point has its probabilistic topic distribution over the latent topics in $\mathbb{Z}$, indicating its correlation to topics at semantic level.

Table 1: Summary of Notations

Notation	Description
$Tr; p$	Activity trajectory; A point in the trajectory
$p.l$	Location of point p
$p.W$	Textual record attached to p
$\beta_w[z]$	Topic proportion of keyword w on topic z
$TD_W[z]$	Topic proportion of keywords W on topic z
$\mathcal{D}_T(W, W')$	Topic distance between W and W'
Q	A set of query points
q	A query point with form (l, W) in Q
$Tr.MRP(q)$	Most relevant point from Tr to q
$Tr.MRPS(Q)$	Most relevant point set from Tr to Q
$\mathcal{D}_S(q.l, p.l)$	Spatial distance from p to q
$d(q, p)$	Distance from p to q
$\mathcal{D}_{mrp}(q, Tr)$	Most relevant point distance from q to Tr
$\mathcal{D}_Q(Tr)$	Distance from Tr to Q

Definition 3. (Topic Distribution of Textual Record): Given the topic set $\mathbb{Z}$ and the collection V of all keywords that may appear in textual records of trajectories, the matrix $\beta = \mathbb{Z} \times V_z$ denotes the word distributions of each topic $z \subseteq \mathbb{Z}$, where V_z ($V_z \subseteq V$) is the collection of keywords belonging to the topic z. Each β_z represents a distribution of a single topic z over all words which belong to this topic z (i.e., V_z), and $\beta_z[w]$ denotes the occurrence frequency of word w in V_z, calculated as $\beta_z[w] = T_w / \sum_{v \in V_z} T_v$ where T_w represents the occurrence number of each word w in all trajectories. Table 2 is a matrix based on Figure 1. Let $z =$ '$drink$', then $V_{\text{'}drink\text{'}} = \{\text{'}coffee\text{'}, \text{'}drink\text{'}, \text{'}Starbucks\text{'}\}$ is the collection of all keywords related to z. The third tuple in Table 2 is the word distribution $\beta_{\text{'}drink\text{'}}$, and $\beta_{\text{'}drink\text{'}}[\text{'}coffee\text{'}] = 0.5$ represents the occurrence frequency of 'coffee' in $V_{\text{'}drink\text{'}}$. Such a matrix β can be initialized by applying probabilistic topic models like LDA [3].

Given a textual record W formed by a sequence of keywords, the *topic distribution* of W over the topic set $\mathbb{Z}$, denoted as TD_W, is the statistical proportion for each keyword in W (excluding stop words that contain little topical content), where the topic proportion $TD_W[z]$ from W to topic z is calculated as

$$TD_W[z] = \frac{N_{w \in (W \bigcap V_z)} + \alpha}{|W| + |\mathbb{Z}| * \alpha} \tag{1}$$

where $N_{w \in (W \bigcap V_z)}$ is the number of keywords belonging to V_z in W; α is the symmetric Dirichlet prior and generally $\alpha = 0.1$; $|W|$ and $|\mathbb{Z}|$ are the number of keywords in W and topics in $\mathbb{Z}$ respectively. So we can guarantee that the sum of topic proportions of any textual record W satisfies $\sum_{z \in \mathbb{Z}} TD_W[z] = 1$.

Definition 4. (Topic Distance Measure): Given two textual records W and W', their topic distance $D_T(W, W')$ indicates their semantic similarity which can be measured by several standardized similarity estimation techniques (e.g., cosine distance and l_p distance) based on their topic distributions. Here, we use the well-known Euclidean distance to measure their topic distance in high dimensional topic space, and adapt the sigmoid function to normalize their distance to range $[0, 1]$ as follows:

$$\mathcal{D}_T(W, W') = \frac{2}{1 + e^{-\sqrt{\sum_{z \in \mathbb{Z}} (TD_W[z] - TD_{W'}[z])^2}}} - 1 \tag{2}$$

The greater topic distance is, the less semantic relevance between the two given textual records.

Table 2: Topic Distributions over Words

topics	words distribution of each topic		
exercise	exercise (0.5)	gym (0.5)	
movie	movie (0.4)	watch (0.4)	cinema (0.2)
drink	coffee (0.5)	drink (0.3)	Starbucks (0.2)
food	food (0.4)	eat (0.4)	restaurant (0.2)
shop	supermarket (0.5)	mall (0.5)	

Table 3: Topic Distributions of Textual Records

textual records \ topics	exercise	movie	drink	food	shop
exercise and coffee	0.44	0.04	0.44	0.04	0.04
gym	0.72	0.07	0.07	0.07	0.07
Starbucks	0.07	0.07	0.72	0.07	0.07
drink coffee	0.04	0.04	0.84	0.04	0.04
watch movie	0.04	0.84	0.04	0.04	0.04
cinema	0.07	0.72	0.07	0.07	0.07
supermarket	0.07	0.07	0.07	0.07	0.72
eat food	0.04	0.04	0.04	0.84	0.04
mall	0.07	0.07	0.07	0.07	0.72
restaurants	0.07	0.07	0.07	0.72	0.07

Example 2: Table 2 and Table 3 show the LDA interpretation on the trajectories Tr_1 and Tr_2 in Figure 1. When running LDA, we first derive the words distribution of each topic (e.g. 'exercise') over its relevant keywords (e.g. 'exercise', 'gym') in all trajectories based on statistical concurrence as shown in Table 2. Then we derive the topic distribution of each textual record in Table 3 based on Equation 1, where each component is a topic proportion (e.g. $\beta_{\text{'}gym\text{'}}[exercise] = 0.72$) that indicates semantic relevance of the text record w.r.t a topic. The topic distance between the textual record of a query point q_1 and that of a trajectory point $p_{1,2}$ can be quantified as $\mathcal{D}_T(q_1.W, p_{1,2}.W) = 0.009$ based on Equation 2, meaning that have high similarity in semantics.

2.3 Problem Definition

Definition 5. (Distance to Query Point): Given a query point q with a textual record $q.W$ and geographical location $q.l$, the distance from a trajectory point p to q is measured according their spatial closeness and topic relevance as follows:

$$d(q, p) = \lambda \cdot \mathcal{D}_S(q, p) + (1 - \lambda) \cdot \mathcal{D}_T(q, p) \tag{3}$$

where $\lambda \in [0, 1]$ is a user specified parameter to adjust the weight between spatial proximity and topic similarity; $\mathcal{D}_S(q, p)$ is the spatial distance where we use the sigmoid function to normalize the Euclidean distance of q and p to the range $[0, 1]$; $\mathcal{D}_T(q, p)$, representing $\mathcal{D}_T(q.W, p.W)$ for simplicity, is the topic distance of their textual records and defined in Equation 2.

Definition 6. (Most Relevant Point): Given a query point q and a trajectory Tr, a trajectory point $p \in Tr$ is the most relevant point (MRP) to q, denoted as $Tr.MRP(q)$, if for any other trajectory point p' we have $d(q, p) \leq d(q, p')$. The distance from the most relevant point $Tr.MRP(q)$ to query point q is called the distance from Tr to q, denoted as $\mathcal{D}_{mrp}(q, Tr) = min_{p \in Tr} d(q, p)$.

Example 3: Consider the example in Figure 1 where λ = 0.5, we can derive $d(q_1, p_{1,2})$=0.08 according to the distance measure, and $p_{1,2}$ has the minimum distance to query point q_1 among all trajectory points in Tr_1, so it is the most relevant point with respect to q_1 such that $Tr_1.MRP(q_1) = p_{1,2}$ and $\mathcal{D}_{mrp}(q_1, Tr_1)$ = 0.08.

Definition 7. (Distance to Query): Given a query with a set of query points $Q = q_1, q_2, ..., q_m$ and a trajectory Tr, we define the distance from Tr to Q, denoted as $\mathcal{D}_Q(Tr)$, as the sum of the distances from Tr to each query point in Q:

$$\mathcal{D}_Q(Tr) = \sum\nolimits_{q_i \in Q} \mathcal{D}_{mrp}(q_i, Tr) \tag{4}$$

The set of MRPs for each query point forms the most relevant point

set, denoted as $Tr.MRPS(Q)$. Finding MRP set for a query Q can be decomposed into finding MRP of each query point in Q.

Example 4: Continuing the above example in Figure 1 where $Q = \{q_1, q_2, q_3\}$, we can easily derive $Tr_1.MRPS(Q) = \{p_{1,2}, p_{1,3}, p_{1,5}\}$ and $Tr_2.MRPS(Q) = \{p_{2,2}, p_{2,3}, p_{2,4}\}$. In comparison, Tr_1 is considered to be more relevant to the query than Tr_2 because $\mathcal{D}_Q(Tr_1) < \mathcal{D}_Q(Tr_2)$.

Problem Statement. Given an trajectory set τ, a finite topic set $\mathbb{Z}$, a query Q with a set of query points q , a user-specified integer k ($k < |\tau|$), an User-oriented Trajectory Similarity Query (UTSQ) returns k distinct trajectories from τ that have the top-k smallest distances with respect to Q.

3. BASELINE ALGORITHMS

In this section, we propose two baseline algorithms which explore the possibility of using existing techniques to solve UTSQ.

3.1 Quadtree based Algorithm

The first baseline method, called Quadtree [15], which only utilizes the spatial coordinates of trajectory points to prune search space. In this method, the points of all trajectories are treated as a point set and Quadtree is used to index these points in two-dimensional space directly. Given a query Q, this baseline method traverses the index structure to find the spatial nearest trajectory incrementally in terms of the *spatial best match distance* which is computed by $\mathcal{D}_{sbm}(Q, Tr) = \lambda \cdot \sum_{q \in Q} minD_S(q, Tr)$, where $minD_S(q, Tr) = \min_{p \in Tr} \mathcal{D}_S(q, p)$. It is easy to see that spatial best match distance is always the lower bound of the distance between Tr and Q, formally presented by the following lemma:

Lemma 1: $\mathcal{D}_{sbm}(Q, Tr) \leq \mathcal{D}_Q(Tr)$

Similar to the k-BCT query in [10], we keep finding the next nearest trajectory Tr (based on spatial best match distance) and computing its distance $\mathcal{D}_Q(Tr)$ from Tr to Q. During the process, we keep track of the k-th minimum distance as the upper bound of final results $\mathcal{D}_{ub}$. If the spatial best match distance of next obtained trajectory exceeds $\mathcal{D}_{ub}$ already, the search algorithm terminates since all 'unscanned' trajectories have no chance to be better than the current top-k results according to Lemma 1.

3.2 iDistance based Algorithm

The second baseline algorithm uses iDistance [31][20] as index structure to prune the search space in pure topic dimension. In the first place, the points of all trajectories are treated as a series of high-dimensional points. In iDistance, a data clustering algorithm (e.g., k-means, k-medoids, etc.) is deployed firstly to group all trajectory points into m clusters based on their distributions in topic space. Each cluster $C_i (i \in [1, m])$ covers trajectory points referring to similar topics by a reference point O_i and contains nearest and farthest radii which define the topic space range. Afterwards, all high-dimensional trajectory points are transformed into a single dimensional space by an index key which is computed as follows:

$$key = i * c + D_T(p, O_i) \quad (5)$$

where c is a constant to partition the single dimension space into regions, i.e., all points in cluster C_i will be mapped to the range $[i*c, (i+1)*c)$. Finally, a B$^+$-tree is adopted to index the transformed points to speed up retrieval. In detail, each leaf node of the B$^+$-tree is linked to both the left and right siblings to facilitate searching the neighboring nodes when the search region is gradually enlarged.

In processing a query Q, we access the iDistance index structure to browse the topic space by expanding the hyper-sphere centered at each query point in Q, and R is the radius of the corresponding search sphere. At each time, R is increased by Δr ($R = R + \Delta r$). If a trajectory point p is inside the searching sphere of query point q, their topic distance $\mathcal{D}_T(q, p)$ can be retrieved. Similar to the Quadtree-based method, we keep finding the nearest trajectory with the minimum *topic best match distance* and computing its distance to query, and the *topic best match distance* to Q is computed by

$$\mathcal{D}_{tbm}(Q, Tr) = (1 - \lambda) \cdot \sum_{q \in Q} minD_T(q, Tr) \quad (6)$$

where $minD_T(q, Tr) = \min_{p \in Tr} D_T(q.W, p.W)$.

The topic best match distance is always the lower bound of distance from Tr to Q obviously, formally presented by the Lemma 2. Trajectory search stops when the topic best match of next obtained trajectory exceeds the dynamically maintained k-th minimum distance of the trajectories we have scanned.

Lemma 2: $\mathcal{D}_{tbm}(Q, Tr) \leq \mathcal{D}_Q(Tr)$

4. HYBRID INDEX BASED ALGORITHM

In this section, we design a hybrid trajectory index structure called IDQ-tree to prune the search space efficiently in both spatial and topic domains. The basic idea of the aigorithm is to model points of all trajectories as a point set in high dimensional space with respect to its spatial and topic features simultaneously.

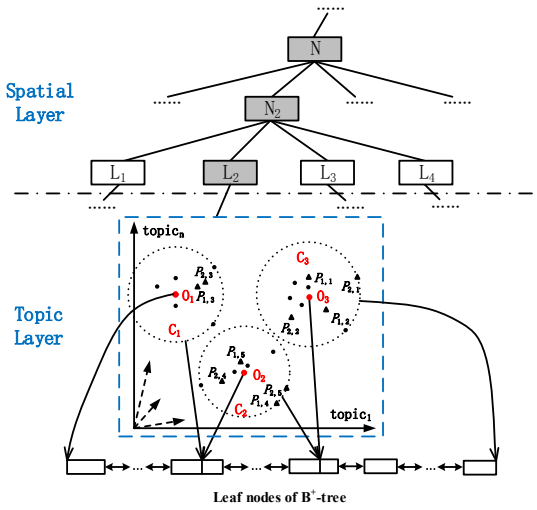


Figure 2: IDQ-tree

Index Structure. The IDQ-tree which is shown in Figure 2 has two layers: the spatial layer at the top, and the topic layer at the bottom. We design IDQ-tree as a spatial first index structure because of the better pruning effect of spatial domain due to its two-dimensional nature (v.s. high dimensional topic domain). In IDQ-tree, all nodes in spatial layer are organized by Quadtree, in the form of $N = (sp, srect, so, sr)$, where $srect$ is the minimum bounding rectangle (MBR) of all trajectory points contained by N; so and r are the center point and radius of a hyper-sphere that covers the topic distributions of all trajectory points in N respectively; sp is the pointer(s) to its child node(s) if N is a non-leaf node, or a pointer to a B$^+$-tree for topic layer if N is a leaf node. For topic layer, all nodes of IDQ-tree are organized by iDistance, with the form as $N = (key, tp, trect, to, tr)$, where key is the minimum key value of all trajectory points of this node, tp refers to pointer(s) to child node(s), $trect$ is the MBR, to and tr are the center point and radius of a hyper-sphere. For leaf nodes, two additional pointers are used to refer to its left and right siblings respectively.

Example 5: A two-layered IDQ-tree is shown in Figure 2, where we assume all trajectory points of the two trajectories Tr_1 and Tr_2 in Figure 1 are close each other and placed in the same spatial leaf

node L_2 (after Quadtree based space partition). In topic domain, the trajectory points are partitioned into three clusters, where C_1 contains $\{p_{1,3}, p_{2,3}\}$, C_2 contains $\{p_{1,4}, p_{1,5}, p_{2,4}, p_{2,5}\}$, C_3 contains $\{p_{1,1}, p_{1,2}, p_{2,1}, p_{2,2}\}$. Trajectory points in the same cluster have high semantic similarity, and the clusters are further organized by B^+-tree in hierarchy in iDistance.

In construction of the IDQ-tree, we first build up a Quadtree for all trajectory points in spatial space. Then for all trajectory points in each leaf node of the Quadtree, we further construct a B^+-tree on top of the iDistance clusters in topic space like the second algorithm. Finally, we compute the $N.so$ and $N.sr$ for all spatial layer nodes of the IDQ-tree in a bottom up fashion.

Algorithm 1: IDQ-Tree based Search Algorithm

input : Trajectory set τ, query Q, parameters k and λ
output: top-k result set $\mathcal{R}$

```
1  D_ub = +∞; D_lb = 0;
2  Visited Trajectory Set ν = null;
3  while true do
4      Retrieve a new nearest Quadtree leaf node N;
5      if |Q| * mdist(q, N) > D_ub then
           break;
6      while true do
7          Retrieve a new best match trajectory Tr ∈ τ − ν;
8          Compute D_Q(Tr);
9          Insert Tr into R by asc order of D_Q(Tr);
10         Update D_ub and D_lb;
11         if D_lb > D_ub then
               break;
12         Insert Tr into ν;
13 keep the top-k results in R;
14 return R;
```

Trajectory Search. The basic structure of the search algorithm is illustrated by Algorithm 1. As the index structure has two layers, the trajectory search is proceeded in the spatial and topic domain alternately. Given a query Q, we start from the spatial layer to visit spatial layer nodes based on a priority queue in the ascending order of $mdist(q, N)$ defined as follows:

$$mdist(q, N) = \lambda \cdot \mathcal{D}_S(q, N) + (1 - \lambda) \cdot \mathcal{D}_T(q, N) \quad (7)$$

where $\mathcal{D}_S(q, N)$ is the minimum spatial distance from q to the node N according to its $N.rect$, and $D_T(q, N)$ is minimum topic distance from q to any point belonging to node N. Let $||TD_q, N.o||$ be the Euclidean distance between textual record of q and reference point $N.o$ in topic space, we compute $\mathcal{D}_T(q, N)$ as follows:

$$\mathcal{D}_T(q, N) = \begin{cases} 0 & ||TD_q, N.o|| \leq N.r \\ \frac{1-e^{-||TD_q,N.o||}}{1+e^{-||TD_q,N.o||}} - N.r & otherwise \end{cases} \quad (8)$$

Lemma 3: $|Q| * mdist(q, N)$ is the lower bound $\mathcal{D}_Q(Tr)$ for all 'unscanned' trajectories in the database.

Proof: For each query point $q \in Q$, we know that $mdist(q, N)$ is the lower bound distance to any point in N. Since all 'unscanned' trajectory points belong to a spatial layer leaf node N' (possibly $N' = N$) such that $mdist(q, N') \leq mdist(q, N)$, we can easily get $d(p, q) \leq mdist(q, N)$. For any unscanned trajectory Tr, we have $|Q| * mdist(q, N) \leq \sum_{q \in Q} \mathcal{D}_q(Tr) \leq \mathcal{D}_Q(Tr)$. Therefore, $|Q| * mdist(q, N)$ must be the strict lower bound $\mathcal{D}_Q(Tr)$ of all 'unscanned' trajectories. □

In the search process, we dynamically maintain the k-th minimum distance for all scanned trajectories $\mathcal{D}_{ub}$. If condition $|Q| * mdist(q, N) > D_{ub}$ hold for a spatial layer leaf node N, it is unnecessary to explore its child nodes anymore because all trajectories covered by N can be pruned according to Lemma 3. Otherwise, we go to the topic layer and find the nearest trajectory in the ascending order of the *best match distance* which is computed in the following formula:

$$\mathcal{D}_{bm}(Q, Tr) = \sum_{q \in Q} (\lambda \cdot minD_S(q, N) + (1 - \lambda) \cdot \mathcal{D}_T(q, Tr)) \quad (9)$$

where $minD_S(q, N)$ is the minimum spatial distance from query q to node N and $\mathcal{D}_T(q, Tr) = \min_{p \in (Tr \bigcap N)} \mathcal{D}_T(q.W, p.W)$. Whenever the next nearest trajectory is retrieved, we compute its distance to Q and update $\mathcal{D}_{ub}$. $\mathcal{D}_{bm}(Q, Tr)$ is the lower bound of distance to Q, so that all unvisited trajectories in the leaf node can be pruned directly once the best match distance of next obtained trajectory is greater than $\mathcal{D}_{ub}$ of current best top-k trajectories.

Example 6: Given a query $Q = \{q_1, q_2, q_3\}$, we prune the search space in spatial dimension first. In Figure 2, we can directly filter the spatial nodes such as L_1, L_3 and L_4 according to Lemma 3. For the nodes spatially close to Q (e.g. L_2), we process the topic layer nodes to validate trajectories in the ascending order of their best match distance to Q, such that T_1 and T_2 are checked prior to trajectories in the same region with lower semantic relevance.

5. LSH BASED ALGORITHM

Since the IDQ-tree relies on high-dimensional clustering in topic space, the issue of 'dimensionality curse' may severely deteriorate the efficiency of query processing. To this end, we further propose a novel index structure, namely ST-tree, which combines Quadtree (for spatial domain) and LSH (for topic domain) seamlessly. Following a dimensionality reduction method in topic space, the ST-tree hashes the trajectory points with similar topic distributions (i.e., they have highly semantic relevance) into a same hash bucket. In addition to the spatial based pruning in query processing, a rational mechanism can be used to measure which buckets are relevant to the query points in semantics, so that trajectory points not belonging to those buckets can be pruned directly. In this way, it is able to find qualified activity results in a more efficient manner.

5.1 LSH based Index Structure

As shown in Figure 3, the novel index structure is similar to IDQ-tree, and we skillfully use LSH to replace the iDistance in the topic layer to solve the problem of 'dimensionality curse' caused by high-dimensional clustering in the iDistance method. The LSH is a well-known index scheme for high-dimensional similarity search, with the basic idea to use a family of locality-sensitive hash functions to hash similar objects into the same bucket, which can effectively reduce the dimensionality and realize fast retrieval. To build LSH structure, we need to establish the LSH hash family first.

LSH Family for Euclidean distance. LSH hash families have the property that objects close to each other have a higher colliding probability than objects far apart, which are determined by different distance measure functions [13] [18]. Based on p-stable distributions [19][37], Datar et al.[13] proposed LSH family $\mathcal{H}$ for l_p norms, and each hash function in hash family $\mathcal{H}$ is defined as:

$$h_{a,b}(p) = \lfloor \frac{a \cdot p + b}{d} \rfloor \quad (10)$$

where a is a random vector in $|\mathbb{Z}|$ dimensionality with entries chosen independently from a p-stable distribution; b is a real number chosen uniformly from $[0, d]$; d is a real constant representing the width of the LSH function, and it is determined by the value range of topic distance between trajectory points. Each hash function $h_{a,b}(p)$ maps a trajectory point p in $|\mathbb{Z}|$ dimensionality onto the set

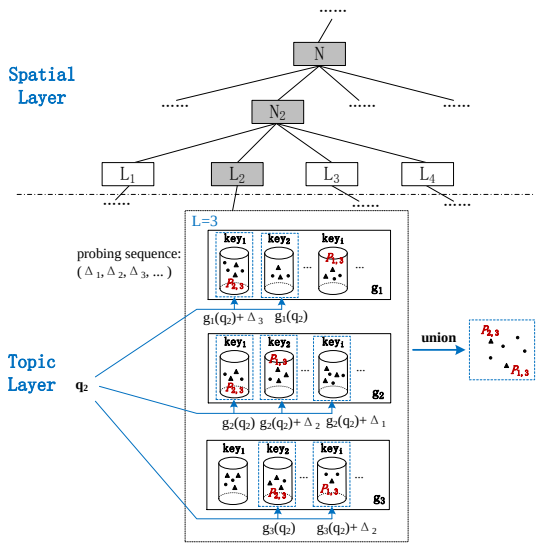


Figure 3: ST-tree

of integers. The p-stable for our work is the Gaussian distribution, which is 2-stable and works for the Euclidean distance.

LSH Index Structure. Using the hash family $\mathcal{H}$ of Euclidean distance defined above, we can construct an LSH structure for trajectory point set P of each Quadtree leaf node in topic layer. The process of the construction can be summarized as follows [17][24]:

- For a selected integer M, define a function family $G = \{g : S \rightarrow U^M\}$ to construct hash tables, and for $g \in G$, $g(p) = (h_1(p), ..., h_M(p))$ to construct each hash table, where $h_j \in \mathcal{H}$ $(1 \leq j \leq M)$ with randomly chosen a and b, i.e., g is the concatenation of M LSH functions in $\mathcal{H}$.
- For a selected integer L, choose $g_1, ..., g_L$ from G independently and uniformly at random to construct L hash tables.
- For each trajectory point p in the Quadtree leaf node, compute the hash values $g_i(p)$ for the i-th hash table and place p into the hash bucket to which $g_i(p)$ points, for $i = 1, ..., L$.

By concatenating multiple LSH functions (i.e., L), the collision probability of far away points becomes very small, but accordingly, the collision probability of nearby points is also reduced. To overcome this drawback, multiple hash tables (i.e., M) are used to increase collision accuracy of nearby points, but this improvement needs more space cost. In practice, parameter settings of M and L are subject to memory capacity and requirements on semantic relevance, and we evaluate their impacts on efficiency and accuracy over real datasets in section 6.

5.2 Multi-probe based Search Algorithm

Based on ST-tree, we conduct efficient trajectory search as illustrated by Algorithm 2. Specifically, we first retrieve a new set of candidate trajectories, which contains some places near any of the query locations and has a high textual similarity with the query point. To avoid the flaws of basic LSH which probes only one bucket which the query object q is hashed into in a hash table, we incorporate the multi-probe search technique into the candidates search methodology to achieve higher retrieval efficiency with lower space cost, by means of probing multiple buckets that are likely to contain candidates in a hash table. Then the distance to query Q is computed for each trajectory in the candidate set. Finally, we insert the trajectory into the result set in ascending order of distance to query. During this process, we keep track of k-th smallest distance to query found so far as an upper bound (D_{ub}) and a lower bound distance (D_{lb}) for all unvisited trajectories. As long as $D_{ub} < D_{lb}$, the search algorithm can terminate safely because all unvisited trajectories are impossible to become the top-k results. Otherwise, we will fetch more candidates and repeat the process again.

Algorithm 2: ST-tree based Trajectory Search

input : Trajectory set τ, query Q, parameters k and λ
output: top-k result set $\mathcal{R}$

$D_{ub} = +\infty$; $D_{lb} = 0$; Visited Trajectory Set ν = null
while $D_{lb} \leq D_{ub}$ **do**
  $D_{lb} \leftarrow |Q| * mdist(q, N)$; $CS \leftarrow \emptyset$;
  for *next nearest Quadtree leaf node* N *of* q_i **do**
    for $k \in [1, L]$ **do**
      /*multi-probe*/
      compute the $g_k(q_i) = (h_1(q_i), ..., h_M(q_i))$;
      apply the perturbation vector Δ to $g_k(q_i)$;
      probe bucket $g_k(q_i) + \Delta$ to get a point set p_k;
    $p \leftarrow$ the union of p_k $(k \in [1, L])$;
    $CS \leftarrow$ a new candidate set containing points in p;
    for *each* $Tr \in CS$ *and* $Tr \in \tau - \nu$ **do**
      Compute $\mathcal{D}_Q(Tr)$;
      Insert Tr into $\mathcal{R}$ by asc order of $\mathcal{D}_Q(Tr)$;
      Update D_{ub} and insert Tr into ν;
keep the top-k results in $\mathcal{R}$;
return $\mathcal{R}$;

Retrieving Candidate Trajectories. A candidate is a trajectory that is possible to become part of the query result. Since the query trajectory can consist of several places spanning a large area, we will obtain a set of candidates which are close to at least one of the query locations and have a high textual similarity at that location.

Similar to IDQ-tree, the candidate retrieval is proceeded in the spatial and topic dimension alternately, and we adapt the best-first search paradigm to obtain the nearest Quadtree leaf node in the spatial layer with a priority queue PQ in the ascending order of $mdist(q, N)$ defined in Equation 6. In order to reach high search accuracy while using much less hash tables compared with basic LSH, we utilize the multi-probe search scheme [24] to generate a candidate trajectory point set for query point q_i in the latest dequeued entry of PQ in topic layer. In addition to the hash bucket $g_k(q_i) = (h_{k1}(q_i), ..., h_{kM}(q_i))$ $(1 \leq k \leq L)$ that the query point q_i is hashed into by basic LSH, we use a carefully derived probing sequence, denoted as $(\Delta_1, \Delta_2, \Delta_3, ...)$, to additionally check those buckets $g_k(q_i) + \Delta$ that are "close" enough (i.e., the hash values of the two buckets only differ slightly) to $g_k(q_i)$ of each hash table, where $\Delta = (\delta_1, ..., \delta_M)$ which is randomly chosen from probing sequence denotes a *hash perturbation vector* defined in [24]. Recall that the LSH functions for Euclidean distance are of the form $h_{a,b}(p) = \lfloor \frac{a \cdot p + b}{d} \rfloor$, if we pick d to be reasonably large, similar trajectory points can be hashed to the same or adjacent values (i.e., differ by at most 1). Hence we restrict our attention to perturbation vectors Δ with $\delta_i \in \{-1, 0, 1\}$.

In order to construct an optimized probing sequence of perturbation vectors $(\Delta_1, \Delta_2, \Delta_3, ...)$, which each vector Δ in this sequence maps to a unique set of hash values to guarantee one hash bucket of a hash table is probed no more than once, we adapt the

query-directed probing approach in [24] which is based on

$$score(\Delta) = \sum_{i=1}^{M} x_i(\delta_i)^2 \tag{11}$$

The $score(\Delta)$ for each possible perpetuation vector Δ determines the probing sequence, and is computed as follows: firstly, each hash function $h_{a,b}(q) = \lfloor \frac{a \cdot p + b}{d} \rfloor$ maps q to a line and the line is divided into slots (intervals) of length d which is numbered from left to right and the hash value is the number of the slot that q falls to. Then assigning $f_i(q) = a_i \cdot q + b_i$ as the projection of query q onto the line for the i-th hash function and $h_i(q) = \lfloor \frac{a_i \cdot q + b_i}{d} \rfloor$ as the slot to which q is hashed. For $\delta_i \in \{-1, +1\}$, letting $x_i(\delta)$ be the distance from $f_i(q)$ to the boundary of the slot $h_i(q) + \delta$, then $x_i(-1) = f_i(q) - h_i(q) \times d$, $x_i(1) = d - x_i(-1)$, and $x_i(0) = 0$ for convenience. For any fixed trajectory point p, $f_i(p) - f_i(q)$ is a Gaussian random variable with mean 0. The variance of this random variable is proportional to $||p-q||_2^2$. Assuming that d is chosen to be large enough, so that all trajectory points in the Quadtree leaf node will fall in one of the three slots numbered $h_i(q)$, $h_i(q) - 1$ or $h_i(q) + 1$ with high probability. Thus the probability that one trajectory point p falls into slot $h_i(q) + \delta$ can be estimated by $Pr[h_i(p) = h_i(q) + \delta] \approx e^{-C x_i(\delta)^2}$, where C is a constant depending on $||p - q||_2$. Hence, the success probability of finding a trajectory point p close to a query point q for a perturbation vector Δ can be estimated as follows:

$$\begin{aligned} Pr[g_i(p) = g_i(q) + \Delta] &= \prod_{i=1}^{M} Pr[h_i(p) = h_i(q) + \delta_i] \\ &\approx \prod_{i=1}^{M} e^{-C x_i(\delta_i)^2} = e^{-C \sum_i^M x_i(\delta_i)^2} \end{aligned} \tag{12}$$

Therefore, the perturbation vector Δ which can find a trajectory point p close to q is related to Equation 10 and this is also the basis for *query-directed* probing method, which orders perturbation vectors in increasing order of their scores. Obviously, perturbation vectors with smaller scores have higher probability of yielding points near to q.The specific implementation process of the query-directed probing algorithm can be referred in [24], and we will implement it in the experimental section. Finally, we put all the trajectories which contain any point in the trajectory point set but are not in $\mathcal{R}$ into the candidate trajectory set CS.

Example 7: Figure 3 shows a ST-tree index, in which all trajectories points in Figure 1 are covered by the leaf node L_2 in spatial layer and organized by LSH buckets in topic layer. If the basic LSH method is used for a query point q_2, then only a number of L buckets with hash values $\{g_1(q_2), g_2(q_2), g_3(q_2)\}$ are probed, and obviously, the trajectory point $p_{1,3}$ that is highly relevant to q_2 in semantic is missed. To cover $p_{1,3}$, the basic LSH method needs more hash tables. In contrast, if the query-directed probing approach is used, no relevant trajectory points will be missed because $p_{1,3}$ is included in the $\{g_2(q_2) + \Delta_2, g_3(q_2) + \Delta_2\}$ buckets.

Computing Lower Bound Distance. During the candidate retrieval process, another important task is to maintain and update a lower bound distance D_{lb} for all unvisited trajectories in the spatial layer. Here, we directly use the $mdist(q, N)$ in the top entry of priority queue PQ as the D_{lb}. Obviously, $|Q| * mdist(q, N)$ is a lower bound of $\mathcal{D}_Q(Tr)$ for all unvisited trajectories which has been proven in Lemma 3. In the search process, if condition $|Q| * mdist(q, N) > D_{ub}$ (the k-th minimum distance for all scanned trajectories) is satisfied, the search algorithm can terminate according to Lemma 3.

6. EXPERIMENTAL STUDY

We conduct extensive experiments on real datasets to evaluate the performance of our proposed index and search algorithms.

6.1 Experimental Settings

We use the real activity trajectory datasets by crawling the check-in records of Foursquare within the areas of Los Angeles. Each check-in record contains the user ID, venue with geo-location, time of check-in and the tips written in plain English. We put the records belonging to the same user in the chronological order to form the trajectory of this user, and each check-in record becomes a trajectory point. Then we use the LDA tool to interpret the tips associated with the check-in records to topic distributions. The generated dataset includes 315567 activity trajectories, and the number of trajectory points is 6011342 in total.

Table 4: Default Parameter Values

Parameter	Default Value	Description
λ	0.5	weight factor
k	10	top-k results
$\|Q\|$	3	No. of query points in Q
$\|\mathbb{Z}\|$	50	No. of topics
$\|\mathcal{D}\|$	100K	No. of trajectories

We first investigate the impact of important parameters d, M and L over the performance of ST-tree. Afterwards, we compare query time cost and number of visited trajectories (denoting I/O) of all proposed algorithms. The default values for parameters are summarized in Table 4. In the experiments, we vary these values to investigate the effect of each parameter. For each set of experiment, we generate 100 queries randomly and report average query time and number of visited trajectories. All algorithms are implemented in Java and run on a PC with 2-core CPUs at 3.2GHz and 8GB memory. For the IDQ-tree and ST-tree, we set each leaf node in spatial layer to corresponds to a 4k disk page by default.

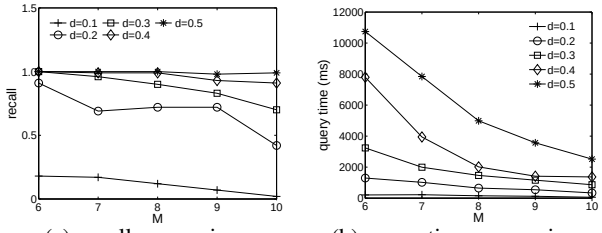
6.2 Performance Evaluation

1) Evaluation on LSH Parameters.

In ST-tree, there are several LSH related parameters including d, M and L that may affect the query performance. We will tune these parameters to evaluate the performance of ST-tree in sequence. Since the LSH is a similarity index structure, we could get inaccurate results for some query Q. So we analyze the performance of ST-tree in three aspects: search quality, search speed and space cost. Ideally, the ST-tree search system should be able to achieve high-quality search results with higher speed and lower space cost.

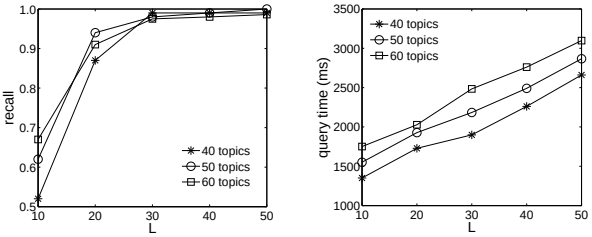
Similar to LSH, we use $recall$ to measure the ST-tree search quality, which can be computed as $recall = \frac{|A(Q) \bigcap I(Q)|}{I(Q)}$, where $I(Q)$ is the set of ideal answers (i.e., the top-k nearest trajectories of query Q) and $A(Q)$ is the set of actual answers. In ideal case, the $recall$ score is 1.0, which means all the k nearest neighbors are returned. In order to get more robust $recall$, we set the number of returned neighbor trajectories k as 100. Search speed is measured by query time, which is the time spent to answer a query. Space requirement is measured by total number of hash tables needed.

Effect of d and M. d and M are two fundamental parameters for constructing a hash table. As the value of topic distance between trajectory points is in the range [0, 1], we vary d from 0.1 to 0.5 and M from 6 to 10 with L fixed as 30. A group of experiments are conducted to evaluate performance of ST-tree under different d and M. The space requirement just vary slightly for different settings with fixed L, so we are only concerned with the precision of returned trajectories measured by $recall$ and the running time for disposing a query. The results are presented in Figure 4. Generally,



(a) recall comparison (b) query time comparison

Figure 4: Effect of d and M



(a) recall comparison (b) query time comparison

Figure 5: Effect of L

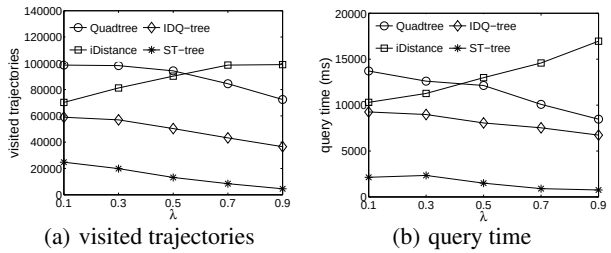
the choices of d and M do co-affect the accuracy and the query time according to Figure 4. When d is small, fewer LSH functions with smaller d can achieve an utterly perfect distinguishing capacity and the algorithm can retrieve results faster. However the recall of the results is small at the same time. On the other hand, if d is too large, more LSH functions need to be employed to acquire a good distinguishing capacity, which increases the time cost for the query. According to Figure 4, we set (d, M) as $(0.4, 8)$ for the datasets in the following experiments, where ST-tree exhibits the best performance w.r.t. both recall and query time.

Effect of L. L is the number of hash tables which exhibits impact on the precision of returned trajectories and space consumption. Intuitively, a larger L indicates more information provided by the ST-tree, which facilitates the accuracy of returned trajectories. To achieve higher recall with lower time and space cost, we vary L from 10 to 50 with d and M fixed as mentioned above and the results are shown in Figure 5. As expected, $recall$ increases when L ranges from 10 to 50 as shown in Figure 5(a). Moreover, increasing L contributes little to the accuracy when L is large enough. Especially, when L exceeds 30, $recall$ remains relatively unchanged. As to query time, increasing L leads to more time cost for searching in ST-tree. Figure 5(b) shows that query time increases significantly as L goes up. As the performances of ST-tree w.r.t. $recall$ and query time are satisfactory and do not vary much when L is greater than 30, we set L as 30 in the following experiment.

2) Comparison on Proposed Algorithms.

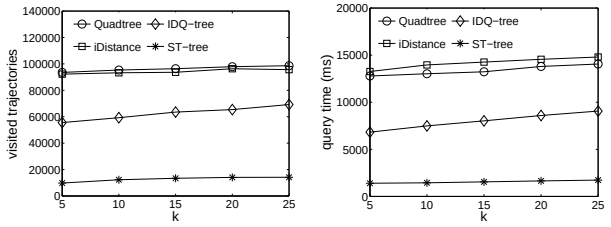
In this part, we vary parameters in Table 4 to compare ST-tree with IDQ-tree and two baseline algorithms.

Effect of λ. In the first set of experiments, we study the effect of different weight factors λ. As shown in Figure 6, the visited trajectories and the query time of all algorithms except iDistance have the similar trend, i.e., the visited trajectories and the query time reduce when λ goes up, notably a sharp decrease happens when λ is less than 0.7. This phenomenon can be explained by the spatial first nature of proposed algorithms, which leads to better pruning effects on spatial. However, the iDistance algorithm has the opposite trend because of the pruning of the pure topic domain. In comparison, ST-tree based algorithm outperforms three baseline algorithms in terms of CPU time and I/O, and it has a relatively stable performance in varying λ. Moreover, Figure 6 shows that the hybrid index is generally superior to the Quadtree and iDistance.



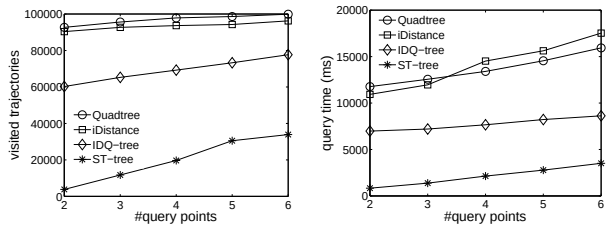
(a) visited trajectories (b) query time

Figure 6: Effect of λ



(a) visited trajectories (b) query time

Figure 7: Effect of k



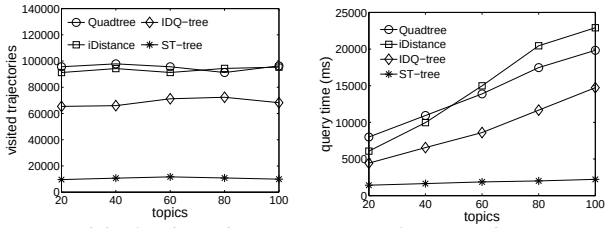
(a) visited trajectories (b) query time

Figure 8: Effect of $|Q|$

Effect of k. Next we study the effect of the intended number of results k by plotting the average query time cost and visited trajectories during search on the dataset. As shown in Figure 7, ST-tree significantly outperforms all other three baseline indexing methods on the dataset. Particularly, ST-tree is at least one order of magnitude faster than Quadtree and iDistance and 4-5 times faster than hybrid index with respect to query time. All algorithms incur higher cost in both number of visited trajectories and query time as k increases. This is expected that more candidates need to be retrieved in query processing when the value of k goes up.

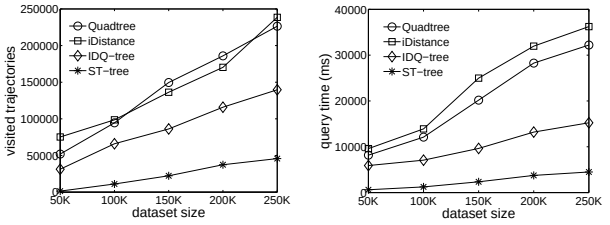
Effect of $|Q|$. Then we investigate the query performance when the number of query locations, $|Q|$, is varying. As shown in Figure 8, ST-tree has superior performance than all baseline methods as well. With the increase of query points, all algorithms incur more time cost and more trajectories are visited since they all utilize high dimensional index to retrieve candidates around each query point (regarding to the spatial and topic distributions of the query). Hence more query points generally result in more candidates to be retrieved, as shown in Figure 8(a).

Effect of $|\mathbb{Z}|$. We now proceed to examine the effect of the number of topics for topic models. The results are shown in Figure 9. We can observe that the number of visited trajectories for all approaches remains constant for all values of topics, while the query time for all methods except ST-tree increases sharply. When the value of topics exceeds 60, the query time for iDistance exceeds that of Quadtree because of the computation for high dimensionality. This phenomenon can explain the worse pruning efficiency of iDistance for higher dimensionality. As shown in Figure 9, ST-tree



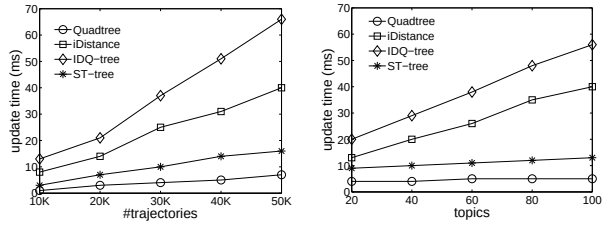
(a) visited trajectories (b) query time

Figure 9: Effect of $|\mathbb{Z}|$



(a) visited trajectories (b) query time

Figure 10: Effect of $|\mathcal{D}|$



(a) update time comparison (b) update time comparison

Figure 11: Comparisons on Updating Cost

has superior performance than three baseline algorithms. In particular, when the number of topics exceeds 60, ST-tree is at least one order of magnitude faster than other indices.

Effect of $|\mathcal{D}|$. To evaluate the scalability of all the approaches, we sample the LA dataset to generate datasets with different number of trajectories varying from 50K to 250K, and report the average query time and visited trajectories in Figure 10. With no surprise, the query time and the number of visited trajectories of all four methods increase linearly w.r.t. the size of dataset. It is worth to note that ST-tree scales much better than the others.

Comparisons on updating cost. We also evaluate the performance of all algorithms inserting an activity trajectory in varying trajectory number and topic number, the results of which are reported in Figure 11. In comparison, the hybrid index and iDistance have the worst updating performance, which can be explained by the fact that iDistance tends to require higher updating cost. From Figure 11, we can easily observe that Quadtree and ST-tree have the best overall updating performance among all proposed methods.

To sum up, compared with other indexing methods, ST-tree has better query performance in all settings because of the less query time and lower number of visited trajectories to read from disk.

7. RELATED WORK

Probabilistic Topic Model. The probabilistic topic models are statistical methods to analyze the words in documents and to discover the themes that run through them, how those themes are connected to each other, with no prior annotations or labeling of documents been required [1]. Based on topic models, it is possible to measure the relevance of a text with regard to a theme, as well as the relevance between different texts (at semantic level). The most classical topic models include LDA [3], Dynamic Topic Model [2], Dynamic HDP [21], etc. Some efforts have been made to apply topic models in location based service recommendation systems [22] and in shortest text interpretation [30]. The aforementioned techniques have been widely used in applications like document classification, user behavior understanding, functional region discovery, etc. In this paper, we tend to integrate topic model and activity trajectory for user-oriented trajectory similarity search.

Spatial Keyword Search. Searching spatial objects associated with textual information has gained significant attentions due to the prevalence of spatial web objects on the Internet. Well-known methods like IR-tree [11], IR^2-tree [14] and inverted Grid index [9] are proposed to study the spatial keyword search problem for high dimensionality. More recently, lots of efforts are made to handle spatial keyword search on road network [23], collective querying [6], confidentiality support [7], interactively querying [34] and semantic aware querying [25]. Generally, those methods are mature enough to handle the misspelling or spelling convention difference in spatial object search, and even under road network constraint. However, they are not capable of retrieving spatial objects which are with highly semantic relevance but morphologically different. Therefore in this paper, we investigate the topic model based activity trajectory search to better satisfy user intentions.

Trajectory Similarity Search. The problem of trajectory similarity search has been extensively studied in the last two decades [5] [16][36][10][33][27][12][28]. Initially, trajectory search is mostly constrained in spatial and temporal domains only [10]. In order to achieve better utilization of trajectory data, research efforts have been made to transform a raw GPS trajectory into a semantic trajectory [4], which consists of a sequence of stops at places of interest (PoIs) labelled with semantic tags. More recently, some academic efforts are made to deal with the trajectory search in high dimension. The work [27] proposes effective index and search algorithm for queries which have specified the query location and user-preference. In [33], a novel index structure is proposed to speed up the search of trajectory which can cover the activities in query. But this approach can only find the ones containing all desired activities in exact textual match, which does not fulfill the requirement of recommendation. The work [8] finds trajectories which can satisfy all activities of users and maximize users' experience, and the work [32] applies some approximate keyword search techniques to increase the robustness of system for handling misspelling or conventional spelling difference. But they still cannot retrieve the activities which are synonyms but morphologically different.

To the best of our knowledge, this is the only work to consider the fusion of topic model and activity trajectory search, so that activity trajectories can be recommended more rationally by the interpretation of activity descriptions and user intentions.

8. CONCLUSION AND FUTURE WORK

This paper studies the problem of searching activity trajectories more effectively by replacing keyword matching with semantic interpretation. The probabilistic topic model is utilized to interpret the textual records attached to trajectories and user queries into topic distributions. To support the efficient top-k activity trajectories search with respect to multiple query points in both spatial and topic dimensions, we develop a novel hybrid index structure called ST-tree, and propose effective searching algorithms to prune the search space. Extensive experimental results on real datasets demonstrate the efficiency of our proposed method.

In the future, it will be interesting to investigate how to incorporate the frequent human mobility and activity patterns into the system, to recommend users more informative activity trajectories.

9. ACKNOWLEDGEMENT

This work was partially supported by Chinese NSFC project under grant numbers 61402312, 61572335, 61232006, and Collaborative Innovation Center of Novel Software Technology and Industrialization.

10. REFERENCES

[1] D. M. Blei. Probabilistic topic models. *Communications of the ACM*, 55:77–84, 2012.

[2] D. M. Blei and J. D. Lafferty. Dynamic topic models. In *Proceedings of the 23rd international conference on Machine learning*, pages 113–120, 2006.

[3] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[4] V. Bogorny, B. Kuijpers, and L. O. Alvares. St-dmql: a semantic trajectory data mining query language. *IJGIS*, 22:1245–1276, 2009.

[5] X. Cao, G. Cong, and C. S. Jensen. Mining significant semantic locations from gps data. *Proceedings of the VLDB Endowment*, 3(1-2):569–571, 1999.

[6] X. Cao, G. Cong, C. S. Jensen, and B. C. Ooi. Collective spatial keyword querying. In *SIGMOD*, pages 373–384, 2011.

[7] Q. Chen, H. Hu, and J. Xu. Authenticating top-k queries in location-based services with confidentiality. *VLDB*, 7:49–60, 2014.

[8] W. Chen, L. Zhao, J.-J. Xu, G.-F. Liu, K. Zheng, and X. Zhou. Trip oriented search on activity trajectory. *Journal of Computer Science and Technology*, 30:745–761, 2015.

[9] Y.-Y. Chen, T. Suel, and A. Markowetz. Efficient query processing in geographic web search engines. In *SIGMOD*, pages 277–288, 2006.

[10] Z. Chen, H. T. Shen, X. Zhou, Y. Zheng, and X. Xie. Searching trajectories by locations: an efficiency study. In *SIGMOD*, pages 255–266, 2010.

[11] G. Cong, C. S. Jensen, and D. Wu. Efficient retrieval of the top-k most relevant spatial web objects. *VLDB*, 2:337–348, 2009.

[12] J. Dai, C. Liu, J. Xu, and Z. Ding. On personalized and sequenced route planning. *World Wide Web Journal*, 19:679–705, 2016.

[13] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*, pages 253–262, 2004.

[14] I. De Felipe, V. Hristidis, and N. Rishe. Keyword search on spatial databases. In *ICDE*, pages 656–665, 2008.

[15] R. A. Finkel and J. L. Bentley. Quad trees a data structure for retrieval on composite keys. *Acta informatic*, 4:1–9, 1974.

[16] E. Frentzos, K. Gratsias, and Y. Theodoridis. Index-based most similar trajectory search. In *ICDE*, pages 816–825, 2007.

[17] A. Gionis, P. Indyk, R. Motwani, et al. Similarity search in high dimensions via hashing. In *VLDB*, pages 518–529, 1999.

[18] P. Indyk. Stable distributions, pseudorandom generators, embeddings and data stream computation. In *Foundations of Computer Science*, pages 189–197, 2000.

[19] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *ACM symposium on Theory of computing*, pages 604–613, 1998.

[20] H. V. Jagadish, B. C. Ooi, K.-L. Tan, C. Yu, and R. Zhang. idistance: An adaptive b+-tree based indexing method for nearest neighbor search. *TODS*, 30:364–397, 2005.

[21] S. Kim and P. Smyth. Hierarchical dirichlet processes with random effects. In *Advances in Neural Information Processing Systems*, pages 697–704, 2006.

[22] Q. Liu, Y. Ge, Z. Li, E. Chen, and H. Xiong. Personalized travel package recommendation. In *ICDM*, pages 407–416, 2011.

[23] S. Luo, Y. Luo, S. Zhou, G. Cong, J. Guan, and Z. Yong. Distributed spatial keyword querying on road networks. In *EDBT*, pages 235–246, 2014.

[24] Q. Lv, W. Josephson, Z. Wang, M. Charikar, and K. Li. Multi-probe lsh: efficient indexing for high-dimensional similarity search. In *VLDB*, pages 950–961, 2007.

[25] Z. Qian, J. Xu, K. Zheng, W. Sun, Z. Li, and H. Guo. On efficient spatial keyword querying with semantics. In *International Conference on Database Systems for Advanced Applications*, pages 149–164, 2016.

[26] D. Ramage, D. Hall, R. Nallapati, and C. D. Manning. Labeled lda: A supervised topic model for credit attribution in multi-labeled corpora. In *EMNLP*, pages 248–256, 2009.

[27] S. Shang, R. Ding, B. Yuan, K. Xie, K. Zheng, and P. Kalnis. User oriented trajectory search for trip recommendation. In *EDBT*, pages 156–167, 2012.

[28] H. Su, K. Zheng, K. Zeng, J. Huang, S. Sadiq, N. J. Yuan, and X. Zhou. Making sense of trajectory data: A partition-and-summarization approach. In *2015 IEEE 31st International Conference on Data Engineering*, pages 963–974, 2015.

[29] H. Wang, K. Zheng, J. Xu, B. Zheng, X. Zhou, and S. Sadiq. Sharkdb: An in-memory column-oriented trajectory storage. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 1409–1418, 2014.

[30] X. Yan, J. Guo, Y. Lan, and X. Cheng. A biterm topic model for short texts. In *Proceedings of the 22nd international conference on World Wide Web*, pages 1445–1456, 2013.

[31] C. Yu, B. C. Ooi, K.-L. Tan, and H. Jagadish. Indexing the distance: An efficient method to knn processing. In *VLDB*, pages 421–430, 2001.

[32] B. Zheng, N. J. Yuan, K. Zheng, X. Xie, S. W. Sadiq, and X. Zhou. Approximate keyword search in semantic trajectory database. In *ICDE*, pages 975–986, 2015.

[33] K. Zheng, S. Shang, N. J. Yuan, and Y. Yang. Towards efficient search for activity trajectories. In *ICDE*, pages 230–241, 2013.

[34] K. Zheng, H. Su, B. Zheng, S. Shang, J. Xu, J. Liu, and X. Zhou. Interactive top-k spatial keyword queries. In *2015 IEEE 31st International Conference on Data Engineering*, pages 423–434, 2015.

[35] K. Zheng, Y. Zheng, N. J. Yuan, S. Shang, and X. Zhou. Online discovery of gathering patterns over trajectories. *IEEE Transactions on Knowledge and Data Engineering*, 26:1974–1988, 2014.

[36] Y. Zheng and X. Xie. Learning location correlation from gps trajectories. In *MDM*, pages 27–32, 2010.

[37] V. Zolotarev. *One-dimensional stable distributions*. American Mathematical Soc., 1986.