Future Generation Computer Systems ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

HIB-tree: An efficient index method for the big data analytics of large-scale human activity trajectories

Xu Chen [a], Jie Zhang [a], Zheng Xu [b,*], Jin Liu [a]

[a] School of Computer Science, Wuhan University, China
[b] The Third Research Institute of the Ministry of Public Security, Shanghai, 201204, China

HIGHLIGHTS

- A relevant keyword extraction method based on word embedding are proposed.
- Point popularity and keyword influence factor are used for similarity function.
- An efficient HIB-tree is designed for activity trajectories.

ARTICLE INFO

Article history:
Received 1 July 2017
Received in revised form 17 November 2017
Accepted 2 January 2018
Available online xxxx

Keywords:
Word embedding
Spatial keyword query
Activity trajectory
Trajectory index

ABSTRACT

Research on traditional trajectory data has been widely carried out, and the popularization of mobile social network and location based service have enabled traditional trajectory data correlate with more human activity information, named as activity trajectory. How to generate large-scale, high-quality activity trajectories and provide efficient indexing structure to support big data management and analysis, has become a new challenge. This paper proposes a new method to generate activity trajectory, which produces relevant keywords for activity trajectory based on word embedding, then an efficient index HIB-tree and an enhanced similarity algorithm based on point popularity and keyword influence factor are proposed to recommend activity trajectory. Experiments show that the performance of HIB-tree is better than the existing index methods of activity trajectory.

1. Introduction

The popularity of mobile devices and location-based technology make it easier to obtain human trajectory and location. More and more individual users use the mobile application to publish the latest information with the location and the time. The trajectory of human has evolved from the traditional spatial–temporal information to multidimensional semantic information with various activities, which has become the activity trajectory [1]. With the increasing of trajectory size and complexity, the traditional trajectory indexing methods are not suitable for big data management and analytics of large-scale human activity trajectory.

There is no existing activity trajectory but traditional spatial–temporal trajectories. However, activity trajectories can be formed by traditional spatial–temporal trajectories and activity information which can be extracted from text of social media website. Problem of keyword extraction from texts is classic, the famous baseline is the term frequency-inverse document frequency (TFIDF) [2]. TFIDF is a universal method suited for many occasions. TextRank [3] is another widely used method originated from PageRank algorithm. However, the effect of TextRank is not as good as TFIDF in most cases. In this paper, an extraction method based on word embedding is proposed. Word2vec [4,5] is a famous implementation tools for word embedding, which converts words into multidimensional vectors.

Zheng proposed a popularity-aware spatial keyword search on activity trajectories, which utilizes ITB-tree [1], the novel similarity function enables user to search for places considering by not only distance factor but also keyword popularity, and the optimization of trajectory refinement and early stop condition improve the efficiency. However, ITB-tree is complex and less efficient for activity trajectories, and the definition of keyword popularity is not accurate enough. Besides, their work does not discuss how keywords are extracted from semantic places and explain the keyword relevance with the semantic places. As a result, this paper proposes a more efficient index named Hierarchical Inverted B-tree (HIB). To sum up, our paper has three main contributions:

* Corresponding author.
E-mail address: xuzheng@shu.edu.cn (Z. Xu).

https://doi.org/10.1016/j.future.2018.01.004
0167-739X/

(1) **Relevant keyword extraction method.** In related works, keyword extraction is not discussed clearly and traditional keywords extraction method is unsatisfactory for activity trajectory. However, it is quite important since the keyword relevance determine the recommendation quality. This paper proposes a new extraction method based on word embedding and produces relevant keywords from online check-in service.
(2) **Enhanced similarity function.** Similarity function plays an important role in spatial keyword queries. This paper proposes an enhanced similarity function based on the distance factor and the point popularity with keyword influence factor. Compared with the former similarity function, enhanced similarity function supports the semantics of query better.
(3) **Indexing method.** ITB-tree is designed for popularity-aware spatial keyword search. However, ITB-tree is not suitable for activity trajectory management since the popularity of each keyword is precomputed offline. When we repeat the experiment of ITB-tree, we find that building ITB-tree on activity trajectory is not efficient enough because the operation needs to traverse tree to update the keyword popularity if the number of trajectories keeps increasing. For the problems of ITB-Tree, this paper designs a new hierarchical inverted B-tree (HIB-tree) for better building index and improving the spatial keyword query efficiency as well.

The rest of paper is organized as follows. Section 2 discusses the related work, Section 3 describes some definitions used in the paper. Section 4 discusses how to extract relevant keywords from check-in services and generate activity trajectory. Section 5 presents the HIB-tree structure, building and querying method of HIB-tree for large-scale activity trajectories. Experimental results are presented and discussed in Section 6. Finally, we conclude the paper in Section 7.

2. Related work

Index of trajectory data. The main approaches of spatial index are tree-based methods. B-tree [6] is commonly used in databases and file systems, there are many variants of B-tree, such as BSP-tree [7], K-D-B-tree [8], R-tree [9] and so on. R-tree is used widely in spatial data, which has many variants. Two extended dimensional R*-tree contends well with now-relative data, permits the indexing of data regions that grow continuously over time, lets the internal bounding regions grow [10]. A four-dimensional R-tree index structure based on R*-tree supports time dimensions together with spatial dimensions [11]. Besides tree-based method, a grid structure index [12] adapts gracefully to its contents under insertion and deletion, achieving an upper bound of two disk accesses for single record retrieval and handling range queries and partially specified queries efficiently. Space-filling curves [13] that collect points that are close to each other in space on one curve are also proposed. These index methods are baselines for us when designing our own index structure.

Index of trajectory with textual information. Spatial object with textual information can be fetched from many sources such as DIANPING, Twitter and Sina Weibo. Researchers are fond of using tree structure enhanced with inverted file for trajectory with textual information. For example, Christoforaki et al. utilize tree-based methods and space-filling curves to support top-k spatial keyword queries [14]. Hariharan et al. propose a geographic information retrieval systems over efficient index mechanisms which uses R*Tree and an inverted file index for each keyword, which stores a sorted list of object ids in which the keyword appears, its score, and frequency [15]. Zhou et al. discuss the performance of different combining scheming of R*-tree and inverted file towards textual and location aware queries [16]. IR2-tree is proposed to combines an R-tree with superimposed text signatures [17], the paper presents an efficient incremental algorithm that constructs and maintains an IR2-tree, and use it to answer top-k spatial keyword queries. Cong G et al. propose a location-aware top-k text retrieval (LkT) query [18], which utilizes means of language model and a probabilistic ranking function to compute the text relevancy to a query. After that, variants of LkT query are proposed. Wu D et al. study the efficient processing of continuously moving top-k spatial keyword (MkSK) [19] queries over spatial keyword data and Lu J et al. define Reverse Spatial–Textual k Nearest Neighbor (RSTkNN) query [20] for finding objects that take the query object as one of their k most spatial–textual similar objects. Cao et al. [21] propose a location-aware top-k prestige-based text retrieval (LkPT) query, to retrieve the top-k spatial web objects ranked according to both prestige-based text relevance (PR) and location proximity, in which the PR score of an objects takes the presence of its nearby objects into account, that are also relevant to a query. ITB-tree is designed for popularity-aware spatial keywords search, which is based on TB-tree with each node enhanced with an inverted file [1]. These index methods perform well in their situation, however they are not suitable considering both reading and writing performance. Our work is derived from ITB-Tree, which considers the distance between query location and trajectories, and takes the influence factor of each keyword into consideration. Unlike ITB-Tree, the keyword influence factor is calculated in real time. Although both ITB-Tree and our problem take keyword into consideration for similarity calculation, they can be quite different.

Mining trajectory with textual information. There are plenty of mining types of trajectory data enriched with textual information, such as the discovery of potential areas of interest, tourism path planning, trajectory cluster, trajectory prediction, trajectory similarity measurement and so on. Alvares et al. propose a data preprocessing model to add semantic information to trajectories, and utilize spatial joins between trajectories and ROIs (Region of Interests) to compute the frequent moves between stops [22]. Shang et al. propose and investigate a novel path nearby cluster query, which identifies clusters of spatial objects with respect to a user-specified travel route and aims to find potential regions of interest along the travel route and recommend them to travelers [23]. Shang and his partners also investigate the collective travel planning query, that finds the lowest-cost route connecting multiple query sources and a destination via at most k meeting points [24]. Tanzmeister et al. present a trajectory clustering approach that group trajectories without the need of manually-tuned distance thresholds [25], the approach uses a homotopy-like binary clustering predicate to find trajectories belonging to the same cluster. Hashem et al. introduce a novel group trip planning query which returns for each type of data points those locations that minimize the total travel distance for the entire group [26]. Giannotti et al. introduce a novel form of spatio-temporal pattern, which formalizes the aggregate movement behavior of user [27]. The new pattern, called a trajectory pattern, represents a set of individual trajectories that share the property of visiting the same sequence of places with similar travel times. Yan presents a conceptual and computational approach for semantically trajectory analyzing, which redefines trajectory as the trace of a moving object that has not only geometric spatio-temporal features but also semantic features in terms of integrating geographic and application domain knowledge [28].

Keywords extraction. Keyword extraction has two well-known statistical approach baselines, TFIDF and TextRank. Most natural language processing packages in Python or Java have implemented

Please cite this article in press as: X. Chen, et al., HIB-tree: An efficient index method for the big data analytics of large-scale human activity trajectories, Future Generation Computer Systems (2018), https://doi.org/10.1016/j.future.2018.01.004.

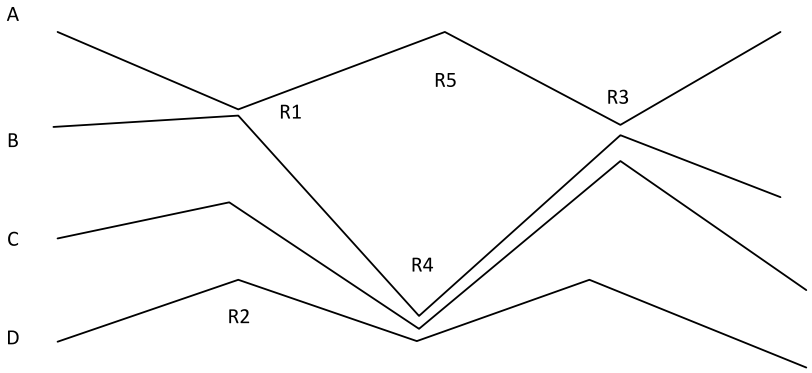


Fig. 1. Activity trajectory example.

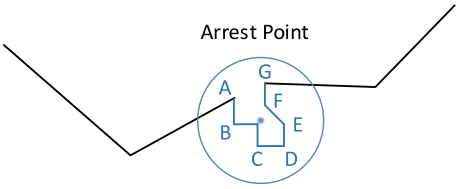


Fig. 2. Example of arrest point.

the two methods and its variants. Many researches in keywords extraction are based on the baselines [2,29–31]. Bruce and Burkey use heuristics to extract keywords from documents [32]. The heuristics are based on syntactic clues, such as the use of italics, the presence of phrases in section headers, and the use of acronyms. Their algorithm tends to produce a relatively large list of phrases, with low precision. Witten and Paynter et al. use Kea algorithm for automatically extracting keyphrases from text [33]. Kea algorithm identifies candidate keyphrases using lexical methods, calculates feature values for each candidate, and uses a machine learning algorithm to predict which candidates are good keyphrases. Kea algorithm is based on TFIDF algorithm and Naive Bayes model. Turney et al. present enhancements to the Kea keyphrase extraction algorithm that are designed to increase the coherence of the extracted keyphrases [34], it uses the degree of statistical association among candidate keyphrases as evidence that they may be semantically related. The statistical association is measured using web mining.

3. Definition presentation

To describe the concept of activity trajectory [1] conveniently, definitions of this paper follow the related concepts in [1] and our paper defines other related concepts in the following. Table 1 shows the relevant symbolic representations and definitions involved in the subsequent formal description.

Definition 1 (*Point Popularity*)**.** Point Popularity is renamed from Keyword Popularity in [1] because it is more accurate in our definition set. The popularity of the point bound with sp is defined as the number of trajectories that go through the point, which is also bound with *sp*, i.e.

$$\rho(P) = |T \in D | \exists P \in T, P.sp = sp| \tag{1}$$

As shown in Fig. 1, which contains four trajectories A, B, C, D and five semantic place R_1, R_2, R_3, R_4 and R_5, then $\rho(R_1) = 2$, $\rho(R_2) = 1$, $\rho(R_3) = 3$, $\rho(R_4) = 3$, $\rho(R_5) = 1$.

Table 1
Summarize of notations.

Notations	Definitions
D	Trajectory database
T	An activity trajectory
Q	A spatial keyword query
P	A point in T
ω	A keyword
A	The keyword set of a trajectory point
$\rho(P)$	The point popularity at point P
ρ	The maximum point popularity
$S(T, Q)$	The similarity between T and Q
$S(P, Q)$	The similarity between P and Q
T_p	The time interval at point P
AP	Arrest point
sp	Semantic place
α	The weighting parameter
d	Distance threshold
T_{sp}	The topic collection of sp

Definition 2 (*Keyword Influence Factor*)**.** Keywords are hidden in semantic place and need to be extracted. However, the contribution to the semantic place of each keyword is not the same as others. The keyword influence factor is defined as the similarity of each word correspond to the topic of the semantic place, i.e,

$$\partial(\omega) = \frac{1}{|T_{sp}|} \sum_{w \in T_{sp}} sim(\omega, w) \tag{2}$$

For example, in semantic place (116.31295, 39.977344), it represents a Chinese Sichuan restaurant, So "川菜" would be the topic of this place and for each keywords generated from comments will have its similarity value correspond to the topic. The top 10 keywords generated via word2vec is "(川菜, 1.0), (火锅, 0.722956897178), (小吃, 0.662984934093), (快餐, 0.656568687436), (川菜馆, 0.646319085702), (火锅店, 0.641461488406), (川味, 0.6413045426), (麻辣烫, 0.619468359814), (菜, 0.605398866501), (担担面, 0.58440103714)", the second digital value of each pair represents the similarity between the word and the topic.

Now we are in position to propose our enhanced similarity function between a trajectory and a spatial keyword query $Q = (x, y, tw, qw, \alpha)$, in which (x, y) is the query location, tw is a time window, qw is a set of query keywords, and $\alpha \in (0, 1)$ is a weight parameter.

Definition 3 (*Enhanced Similarity*)**.** Given a spatial keyword query $Q = (x, y, tw, qw, \alpha)$, the similarity between Q and trajectory T and is defined as

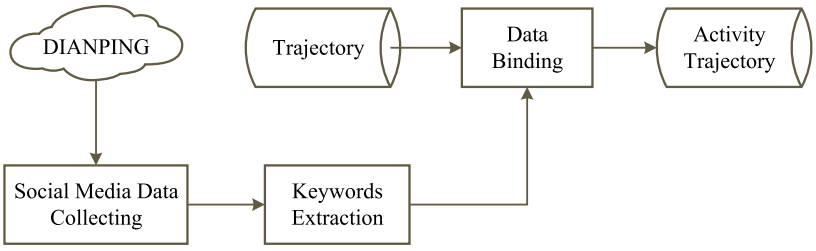


Fig. 3. Generation of activity trajectory.

$$S(T, Q) = \frac{1}{|q\omega|} \sum_{\omega=q\omega} S_\omega(T, Q) \tag{3}$$

$$S_\omega(T, Q) = \max_{P\in T, P.t\in t\omega} \begin{cases} \alpha \times (1 - \frac{d(P, Q)}{D_{\max}}) + (1-\alpha) \times \frac{\rho(P)}{\rho} \times \partial(\omega), & \text{if } \omega \in P.A \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

where $D_{\max}$ is the maximum distance between any two trajectory points in the dataset, ρ is the maximum point popularity. The ratio $\rho(P)/\rho$ is called the relative point popularity and $S_\omega(T, Q)$ is called the similarity between T and Q with respect to the keyword ω.

In enhanced similarity function, we take account of the point popularity and keyword influence factor, which are regarded as the semantic property of a point. The parameter α is to let users specify their preference between the distance of locations and the semantic property of the place.

Definition 4 (*Arrest Point*)**.** The arrest point represents the center of a geographic area where a user's arrest time exceeds a given time threshold in the area and the speed of every point is lower than the speed threshold. The arrest point is determined by three factors, time factor, distance factor and speed factor which are produced by time and distance. Generally speaking, for trajectory $T(P_1, P_2, \ldots, P_{max})$, if $\forall(m \le i \le j \le n)$ where $1 \le m \le n \le max$ we have

$$\begin{aligned} &Distance\left(P_i, P_j\right) < DistanceThreshold \\ &\sum_{i=m}^{n} P_i.timeInterval > TimeThreshold \\ &speed_{p_i} < SpeedThreshold \end{aligned} \tag{5}$$

then the arrest point of this area can be defined as $AP(x, y, TimeInterval)$, where

$$\begin{aligned} AP.x &= 1/|P| * \sum_{i=m}^{n} P_i.x \\ AP.y &= 1/|P| * \sum_{i=m}^{n} P_i.y \\ AP.TimeInterval &= \sum_{i=m}^{n} P_i.timeInterval \end{aligned} \tag{6}$$

In Fig. 2, the distance between arbitrary points is less than the given distance threshold. The sum of the timeIntervals of all points in the region is greater than the given time threshold. The speed of every point in the area is lower than speed threshold. If the conditions above are satisfied, then an arrest point is established. In [35], the stay point has been defined, however in some real situations, a person may run from place to place in a small range with fast speed, perhaps finding the place they are going or just in a hurry doing something here and there. To eliminate the above situation for a more accurate stay point, we add speed factor and call it as arrest point.

4. Generation of activity trajectory

Activity trajectory includes two parts, trajectory information and activity information. Fig. 3 shows the generation of activity trajectory.

Traditional spatio-temporal trajectory information can be easily fetched on mobile terminals or GPS. Activity information can be accessed from website. There are three steps to generate useful activity information: First, crawling useful comments from social media website, e.g. DIANPING or Foursquare. Second, extracting keywords from comments. Third, binding keywords with trajectory to generate activity trajectories.

4.1. Keyword extraction

The details of keywords extraction is not described in [1]. Existing algorithms of keyword extraction are not suitable for activity keywords extraction, such as TFIDF, TextRank and so on. According to the investigation report in [36], research on this topic is rare in recent years. We try to use TFIDF and TextRank methods on keywords extraction from comments of users in DIANPING, however the result is not expected, therefore we decided to use word2vec for extracting keywords and the effects is good.

Word2vec is a group of related models that implement word embedding. These models are shallow, two-layer neural networks that are trained to reconstruct linguistic contexts of words. Word2vec takes a large corpus of text as inputs and constructs a vector space, typically of several hundred dimensions, with each unique word in the corpus being assigned a corresponding vector in the space. Words that share common contexts in the corpus are located in close proximity to one another in the space [4]. Word2vec utilize two model architectures to produce a distributed representation of words: either continuous bag-of-words (CBOW) or continuous skip-gram. In the continuous bag-of-words architecture, the model predicts the current word from a window of surrounding context words. The order of context words does not influence prediction (bag-of-words assumption). In the continuous skip-gram architecture, the model uses the current word to predict the surrounding window of context words. The skip-gram architecture weighs nearby context words more heavily than the others. According to the authors' note [4], CBOW is faster while skip-gram is slower but does a better job for infrequent words.

To utilize word2vec, word vectors are trained by a large number of Chinese corpus. The training data we used is website corpus from Sogou Labs [37], a total size of 3 Terabytes. A lot of work need to be done for training word2vec model. As shown in Fig. 4, firstly, segmentation words are generated from original corpus using HanLP tools,[1] which produce nearly 100 Gigabytes word files, and each word is separated by a space or a carriage return character. Secondly, loading all segmentation words into the gensim training

[1] http://hanlp.linrunsoft.com/.

system, which utilizes a skip-gram model architecture, a more suitable architecture for infrequent words supporting for incremental training. Thirdly, for each word in files, loading file and training words with the context information. Finally, persistent word2vec model is prepared to use.

Each semantic place will have a certain number of tags, which are considered as the most related topics for the semantic place. Then the keyword extraction method goes as follows. For each word separated from comment, the similarities among these words and tags are calculated using word2vec model. After fetching all the similarity information, all words are sorted according to the average similarity with tagged words. And the top 100 words are selected as the keywords of this semantic place. The extraction algorithm is shown in Algorithm 1.

Algorithm 1 Keyword Extraction Algorithm

```
Input: sp
Output: keyword set
1: kw←emptylist;
2: initialize word2vec model;
3: cK←keywords from comment text of semantic place
4: tags←valid tags of semantic place
5: for w in cK do
6:     if w not in word2vec model then
7:         score[w]←0;
8:         continue;
9:     end if
10:    for t in tags do
11:        tS←similarity(t, w)
12:        score[w]=average(score[w], tS);
13:    end for
14: end for
15: kw←top n keywords sorted by score;
16: return kw;
```

4.2. Data binding

Generally, trajectory points are not exactly equal to the semantic place. In our experiment, we use e as the distance threshold for binding trajectory point with semantic place. When the distance of semantic place and the trajectory point is no greater than e, we bind the trajectory point to the semantic place. As we can see, e is an adjustable value, in our experiment the value of e is set as 100 m. For a trajectory, not every point can bind a semantic place, those points will be ignored in HIB-tree. Data binding algorithm is shown in Algorithm 2.

Algorithm 2 Keywords Binding Algorithm

```
Input: T, sp, e
Output: n: num of binded points
1: n←0;
2: for p in T do kSet←empty set;
3:     for t in sp.Points do
4:         if distance(t, p) ≤ e then
5:             kSet.add(t.getKeywords());
6:             n++;
7:         end if
8:     end for
9:     t.setKeywords(kSet);
10: end for
11: kw←top n keywords sorted by score;
12: return kw;
```

5. Storage and index

5.1. Problem of ITB-tree

ITB-tree is an activity trajectory index, which combines the TB-tree and the reverse index file [1]. ITB-tree considers keyword popularity, and it is necessary to traverse the B-tree to update the keyword popularity information each time when the trajectory point information is update. Therefore, constructing the ITB-tree is very slow. ITB-tree index structure is complex and the query process is complicate. TB-tree is the basic structure of ITB-tree. TB-tree requires that a leaf node can only store one trajectory information, if a leaf node is full, the trajectory is cut into segments and stored by separate leaf nodes. ITB-tree uses a double linked list to connect the leaf nodes that store different pieces of the same trajectory. Considering the index structure, querying process is complex and time-consuming.

5.2. The proposed HIB-tree

Considering the above problems of the existence of ITB-tree, this paper proposes a hierarchical inverted B-tree index (HIB-tree). Using HIB-tree to store the active trajectory can improve the efficiency of index construction and optimize the query time.

The HIB-tree index is divided into two parts, as shown in Fig. 5. The nodes of the B-tree are divided into leaf nodes and intermediate nodes. A leaf node N of the HIB-tree contains a number of entries in the form of (ID, MBR, Time Interval, TID). ID is the identification of node while TID is the identification of the trajectory. Since one leaf node stores only one trajectory information, TID can uniquely identifies the trajectory. MBR is the minimum bounding rectangle of all the trajectory points contained in each node of the B-tree. Time Interval represents the maximum time interval of all the child nodes. The entries in a non-leaf node N is in the form of (ID, MBR, Time Interval, TIDS), which remains the same as leaf node except that non-leaf node stores a list of TID to indicate how many leaf-nodes will be accessed through this non-leaf node. The second part is a hierarchical index file. We divide the hierarchical index file into three blocks, namely trajectory point information, point activity information, and node keyword information. The trajectory point information records the spatial information of the trajectory points in each leaf node, and TID is the key index for point information, and these points are clustered by keywords. The point activity information uses the trajectory point as the key index for various types of activity information, for example keyword influence factor in the form of {<key1, factor>, <key2, factor>, …}, point popularity in the form of {tid1, tid2, …} which can be calculated through the length of list and a constant time interval. The node keyword information uses ID to index for the activity information contained in each node, which is used to query filter according to the keyword information.

5.3. Building and querying of HIB-tree

HIB-tree building is divided into two steps, the building of B-tree and the recording of hierarchical file data. When inserting a trajectory point, HIB-tree searches a suitable leaf node to record point information, in the meantime, a non-leaf node updates MBR, Time Interval and TIDS information, a leaf node records TID, MBR

Please cite this article in press as: X. Chen, et al., HIB-tree: An efficient index method for the big data analytics of large-scale human activity trajectories, Future Generation Computer Systems (2018), https://doi.org/10.1016/j.future.2018.01.004.

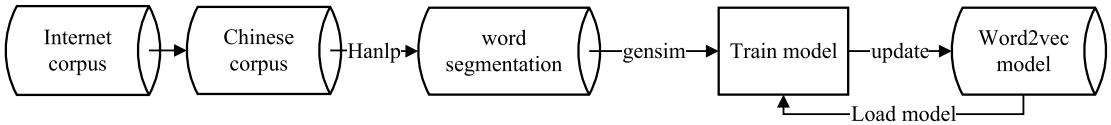


Fig. 4. Process of training word2vec.

and Time Interval in the hierarchical inverted information. The building algorithm is shown in Algorithm 3.

Algorithm 3 HIB-Tree Building Algorithm

```
Input: P(tid, x, y, timeInterval, keywords)
Output: root
1: function Insert(P, root)
2:     node←searchFromRootForSpace(root);
3:     if node instanceof InteriorNode then
4:         if node is full then
5:             node ← new InteriorNode;
6:         end if
7:         node.add(P);
8:         while node is not Leaf do
9:             node←createSonNode();
10:            node.add(P);
11:        end while
12:    else
13:        if node is full then
14:            newNode←new LeafNode;
15:            newNode.add(P);
16:        else
17:            node.add(P);
18:        end if
19:    end if
20:    return root;
21: end function
```

Query operation of HIB-tree is also divided into two steps, B-tree pruning operation and trajectory refinement. Given a query $Q(x, y, t, A)$, in which x is the latitude of query point while y is the longitude of the query point and t represents the time interval of a query and A is the keyword set of query location, select the tid that satisfies the condition based on the query keywords, time interval and MBR. This process uses the breadth-first traversal and it is very fast. After obtaining the tids set, calculate the similarity between Q and the trajectory through the trajectory point information and the point activity information of each point. Finally, recommend Top-k trajectories based on the similarity function. The querying algorithm is shown in Algorithm 4.

Table 2
Parameter settings.

Parameter	Default value
Number of results k	10
Number of query keywords $\|qw\|$	3
Query time window $\|tw\|$	1 h
Weighting parameter α	0.5
Number of trajectories $\|D\|$	5K

Algorithm 4 HIB-Tree Query Process

```
Input: Q(x, y, t, A)
Output: satisfied tid list
1: tids←initEmptyList();
2: function Query(Q, root)
3:     P←new Point(Q.x, Q.y);
4:     queue←initPriorityQueue();
5:     queue.add(root);
6:     while queue is not empty do
7:         node ← queue.removeFirst();
8:         if P ∈ node.MBR & node.t ∈ t & node.A ∈ A then
9:             if node instanceof LeafNode then
10:                refine(node.tid, Q);
11:            else
12:                for each son ∈ node.sons do
13:                    queue.add(son)
14:                end for
15:            end if
16:        end if
17:    end while
18:    return tids;
19: end function
20:
21: function Refine(tid, Q)
22:     points, keywords ← readFile(tid);
23:     similarity←calculate S(T,Q);
24:     minSim←tidsimilarityHashMap.getMinSim();
25:     if similarity ⩾ minSim then
26:         tid.removeMin();
27:         tids.add(tid);
28:     end if
29: end function
```

6. Experiment

6.1. Setup

The trajectory data is derived from the GeoLife Project,[2] which records the outdoor trajectory information from 182 users from Microsoft Asia Research Institute in Beijing for more than two years (from April 2007 to August 2009) Such as shopping, dinner, exercise and so on. These trajectory data include 18176 tracks and 20442467 track points. Activity information is always available on the Internet. DIANPING is the popular third-party consumer comment website in China, it has more than 20,000 active users per month, a wide range of comments, and cooperative shops are throughout the China. Website crawler is used to crawl semantic information from the DIANPING site. Crawled Information includes shop tags, customer comments and tags representing the topic of

2 https://www.microsoft.com/en-us/download/details.aspx?id=52367.

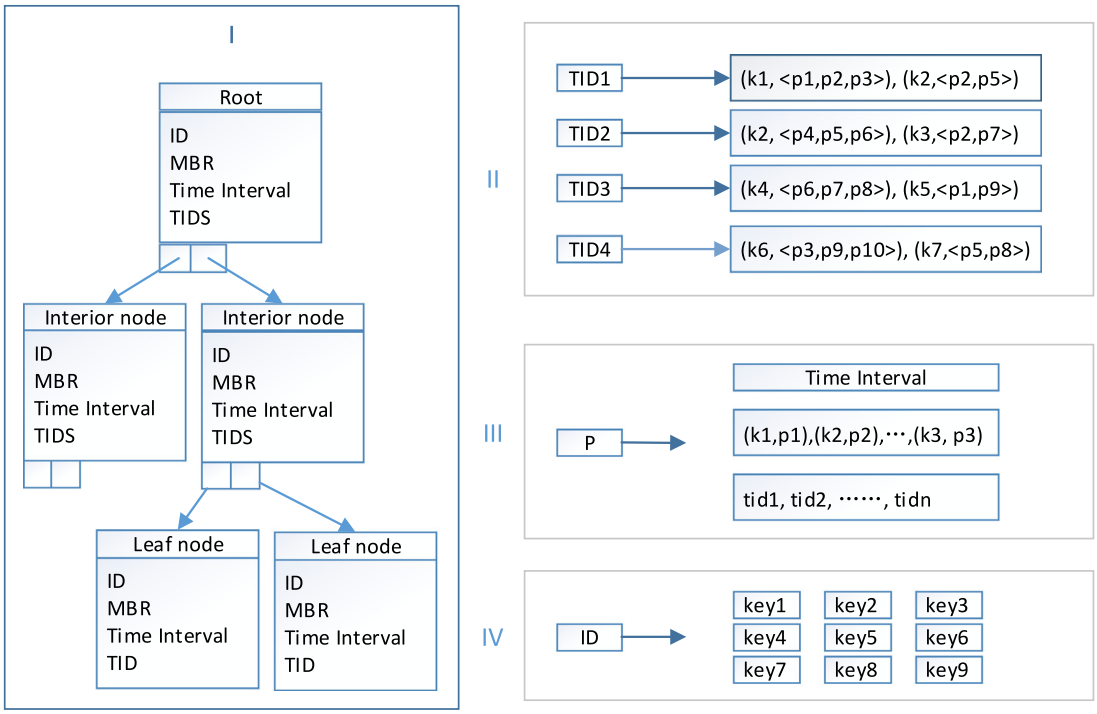


Fig. 5. Hierarchical inverted B-tree index for activity trajectories: (I) B-tree structure. (II) Trajectory point information. (III) Point activity information. (IV) Node keywords.

semantic place. For each semantic place, 100 keywords are extracted from the corresponding comments as activity information and the top 10 keywords are chosen to represent the activity information of semantic place. To extract useful keywords, we use work2vec model. And to train word2vec model, we use a high performance computer with 64G memory and 3 Terabytes disk storage with Linux version 2.6.32-504.23.4.el6.x86_64 to deal with 3 Terabytes corpus data from Sogou Labs and the code is implemented in python.

We study and compare the performance between ITB-tree and HIB-tree. The query location and keyword set are both set randomly. The performance of the algorithms are measured in terms of runtime by taking the average records of 100 query processing. The default values of all the parameters used in our experiments are summarized in Table 2. All tests were implemented in Java, and executed on a Server with Linux version 2.6.32-504.23.4.el6.x86_64, 2.60 GHz CPU and 4 GB RAM.

6.2. Performance study

Index building. Building ITB-tree is very time consuming in practice, therefore, we choose a relatively small amount of data to show how ITB-tree and HIB-tree react in building index. We randomly choose 100, 500, 1000 activity trajectory to test building operation and the result is shown in Fig. 6. As expected, HIB-tree is better than ITB-tree in data writing, as the index sizes increase, the writing rate of ITB-tree dropped rapidly since the calculation of keyword popularity require traversing tree for each trajectory point enhanced with activity information, while the HIB-tree writing rate maintains a faster level. Therefore, HIB-tree is more suitable for indexing large-scale activity trajectories. In the following experiments, keyword popularity is precomputed offline for faster building of ITB-Tree.

Index querying. Compared with ITB-tree, HIB-tree gets rid of doubly linked list, simplifies the tree structure and changes the role of non-leaf node. In the following part, we present our experiment results base on several aspects that affect the performance of index

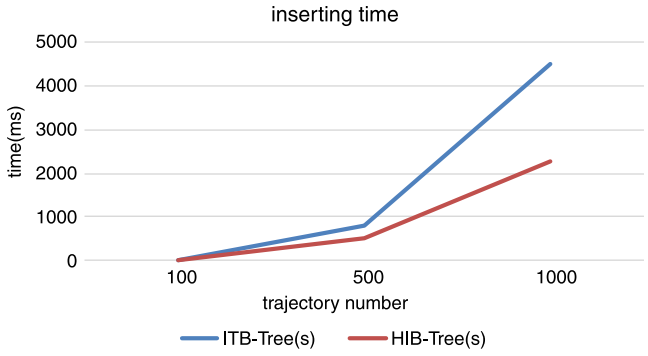


Fig. 6. Experiment on index building.

structure. Since the baseline index structure is ITB-tree, the test cases refer to ITB-tree. In ITB-tree, all other parameters are set either default values or the most optimized parameters.

Effects of k. In the first experiment we study the effects of the requested number of results. As shown in Fig. 7, our implemented index structure significantly outperforms the ITB-tree for all values of k in terms of runtime. The decoupling of storage modules in HIB-Tree make it more accurate to fetch relative data for analysis and ignore other useless data which decrease the number of disk IOs during querying phase. Besides, separating indexes into different part make it easier for LRU (Least Recently Used) cache which accelerate query phase.

Effects of the number of keywords. We also investigate how the number of query keywords affects the performance of both index. In Fig. 8, the performances of both index structures declines with the increasing number of keywords. The increase of the participated nodes leads to more similarity calculation. However, HIB-tree significantly outperforms the ITB-tree structure for all values of the number of keywords in terms of runtime.

Please cite this article in press as: X. Chen, et al., HIB-tree: An efficient index method for the big data analytics of large-scale human activity trajectories, Future Generation Computer Systems (2018), https://doi.org/10.1016/j.future.2018.01.004.

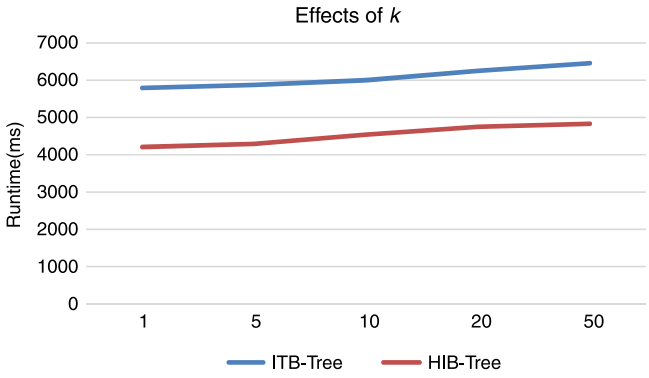


Fig. 7. Experiment of K.

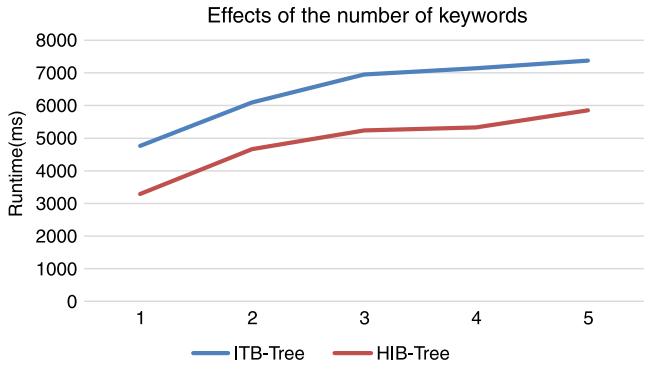


Fig. 8. Experiment of the number of keywords.

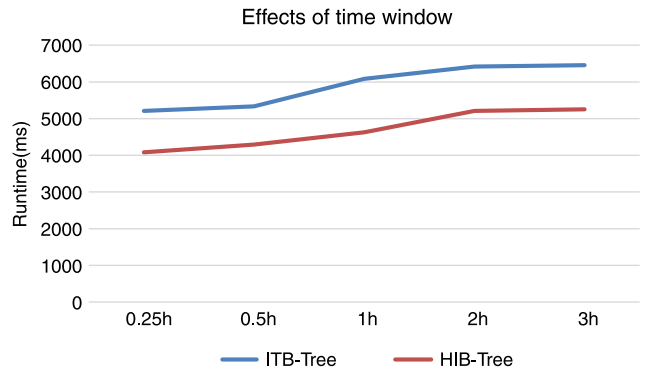


Fig. 9. Experiment of time window.

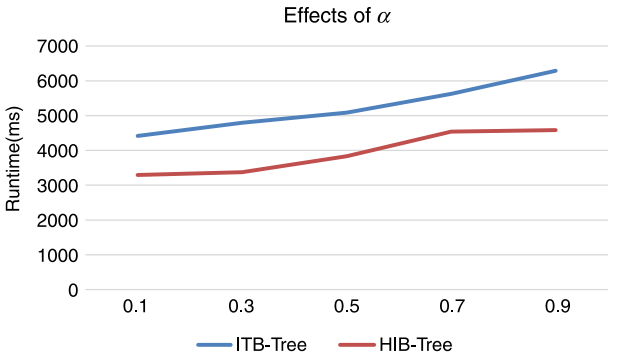


Fig. 10. Experiment of α.

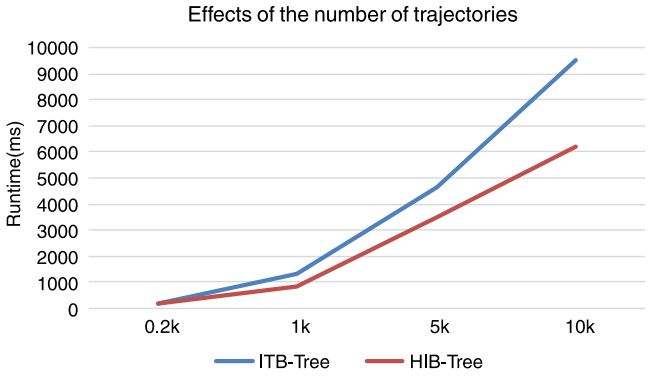


Fig. 11. Runtime effects of the number of trajectories.

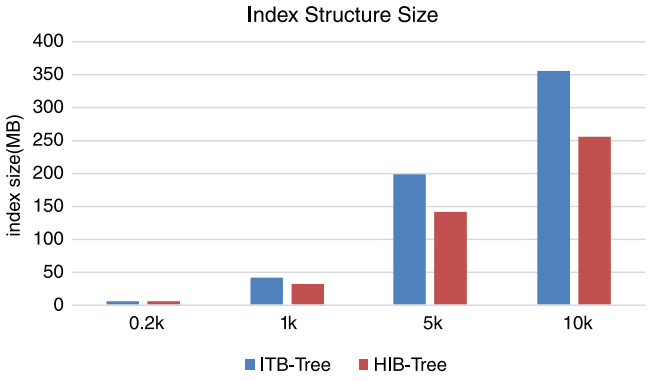


Fig. 12. Index structure size.

Effects of time window. We also study the effects of the length of the query time window by changing it from 0.25 h to 3 h. As illustrated in Fig. 9, the runtime of both indexes tend to increase with the length of the time window. And the increase of time window results in the increase of number of nodes fetched for further refinement, which leads to more disk IO. We can also find that HIB-Tree has better performance than ITB-Tree.

Effects of α. Weighting parameter α in the similarity function is to let users specify their preference between spatial proximity and semantic property. A larger α means that the distance is more important while a smaller α means the popularity is more important, and α will influence the value of similarity and affect pruning process. The experiments test the effects of α, which plays equal role in both index. Fig. 10 shows the results of HIB-tree outperforms ITB-tree.

Effects of the number of trajectories. To study the scalability of the ITB-tree and HIB-tree, we generate 4 datasets containing 200, 1000, 5000, 10000 trajectories by random selection from the original activity trajectory set. Then we run experiments on these four datasets, the results are shown in Fig. 11. Besides, we show the space cost of HIB-tree and ITB-tree when the size of the dataset varies in Fig. 12. The additional storage cost in the ITB-tree is the overhead introduced by the inverted files and other auxiliary information in each node. As we can see, HIB-tree has smaller index size than ITB-tree, and HIB-tree outperforms ITB-tree for all the values of the number of trajectories.

7. Conclusion

Our paper proposes an efficient index method for the big data analytics of large-scale human activity trajectories. For the generation of activity trajectory, we present a keyword extraction

Please cite this article in press as: X. Chen, et al., HIB-tree: An efficient index method for the big data analytics of large-scale human activity trajectories, Future Generation Computer Systems (2018), https://doi.org/10.1016/j.future.2018.01.004.

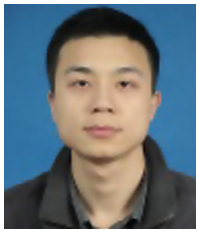
method based on word embedding innovatively. For the index of the activity trajectory, this paper proposes the HIB-tree index structure, which has the advantage of performance over ITB-tree. The HIB-tree index structure improves the insert and query efficiency, and the index structure of the inverted file makes the HIB-tree have better scalability. In future work, firstly, we would be glad to further our study in index method on large scale human trajectory data with distributed storage system. In this case, the availability and query speed will be the focus of the index system. Besides, the current work focus on the static activity trajectory, it is worth studying how to apply our work to the index of dynamic human activity trajectory.

Acknowledgments

This work was supported by the National Key Research and Development Program of China (Grant No. 2016YFB0502601, No. 2017YFC0803700) and Natural Science Foundation of Shanghai. This work was sponsored by CCF-Venustech Open Research Fund (Grant No. CCF-VenustechRP2017006) and Shanghai Rising-Star Program.

References

[1] K. Zheng, B. Zheng, J. Xu, et al., Popularity-aware spatial keyword search on activity trajectories, World Wide Web 20 (4) (2017) 749–773.
[2] J. Ramos, Using tf-idf to determine word relevance in document queries, in: Proceedings of the First Instructional Conference on Machine Learning, vol. 242, 2003, pp. 133–142.
[3] R. Mihalcea, P. Tarau, TextRank: Bringing order into text, in: EMNLP, vol. 4, 2004, pp. 404–411.
[4] T. Mikolov, K. Chen, G. Corrado, et al., Efficient estimation of word representations in vector space, Comput. Sci. (2013).
[5] T. Mikolov, I. Sutskever, K. Chen, et al., Distributed representations of words and phrases and their compositionality, in: Advances in Neural Information Processing System, 2013, pp. 3111–3119.
[6] R. Bayer, E. McCreight, Organization and maintenance of large ordered indexes, in: Software Pioneers, Springer Berlin Heidelberg, 2002, pp. 245–262.
[7] H.M.S. Radha, Efficient image representation using binary space partitioning trees, Signal Process. 35 (2) (1994) 174–181.
[8] J.T. Robinson, The KDB-tree: A search structure for large multidimensional dynamic indexes, in: Proceedings of the 1981 ACM SIGMOD International Conference on Management of Data, ACM, 1981, pp. 10–18.
[9] A. Guttman, R-Trees: A Dynamic Index Structure for Spatial Searching, ACM, 1984.
[10] R. Bliujute, C.S. Jensen, S. Saltenis, et al., R-tree based indexing of now-relative bitemporal data, in: VLDB, 1998, pp. 345–356.
[11] S. Šaltenis, C.S. Jensen, R-tree based indexing of general spatio-temporal data, History (1999).
[12] J. Nievergelt, H. Hinterberger, K.C. Sevcik, The grid file: An adaptable symmetric multikey file structure, ACM Trans. Database Syst. 9 (1) (1984) 38–71.
[13] V. Gaede, O. Günther, Multidimensional access methods, ACM Comput. Surv. 30 (2) (1998) 170–231.
[14] M. Christoforaki, J. He, C. Dimopoulos, et al., Text vs. space: Efficient geo-search query processing, in: Proceedings of the 20th ACM International Conference on Information and Knowledge Management, ACM, 2011, pp. 423–432.
[15] R. Hariharan, B. Hore, C. Li, et al., Processing spatial-keyword (SK) queries in geographic information retrieval (GIR) systems, in: Scientific and Statistical Database Management, IEEE, 2007, p. 16.
[16] Y. Zhou, X. Xie, C. Wang, et al., Hybrid index structures for location-based web search, in: Proceedings of the 14th ACM International Conference on Information and Knowledge Management, ACM, 2005, pp. 155–162.
[17] I. De Felipe, V. Hristidis, N. Rishe, Keyword search on spatial databases, in: Data Engineering, 2008, pp. 656–665.
[18] G. Cong, C.S. Jensen, D. Wu, Efficient retrieval of the top-k most relevant spatial web objects, Proc. VLDB Endow. 2 (1) (2009) 337–348.
[19] D. Wu, M.L. Yiu, C.S. Jensen, et al., Efficient continuously moving top-k spatial keyword query processing, in: Data Engineering, ICDE, 2011 IEEE 27th International Conference on, IEEE, 2011, pp. 541–552.
[20] J. Lu, Y. Lu, G. Cong, Reverse spatial and textual k nearest neighbor search, in: Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data, ACM, 2011, pp. 349–360.
[21] X. Cao, G. Cong, C.S. Jensen, Retrieving top-k prestige-based relevant spatial web objects, Proc. VLDB Endow. 3 (1–2) (2010) 373–384.
[22] L.O. Alvares, V. Bogorny, B. Kuijpers, et al., A model for enriching trajectories with semantic geographical information, in: Proceedings of the 15th Annual ACM International Symposium on Advances in Geographic Information Systems, ACM, 2007, p. 22.
[23] S. Shang, K. Zheng, C.S. Jensen, et al., Discovery of path nearby clusters in spatial networks, IEEE Trans. Knowl. Data Eng. 27 (6) (2015) 1505–1518.
[24] S. Shang, L. Chen, Z. Wei, et al., Collective travel planning in spatial networks, IEEE Trans. Knowl. Data Eng. 28 (5) (2016) 1132–1146.
[25] G. Tanzmeister, D. Wollherr, M. Buss, Environment-based trajectory clustering to extract principal directions for autonomous vehicles, in: Intelligent Robots and Systems, IROS 2014, 2014, pp. 667–673.
[26] T. Hashem, T. Hashem, M.E. Ali, et al., Group trip planning queries in spatial databases, in: International Symposium on Spatial and Temporal Databases, Springer, 2013, pp. 259–276.
[27] F. Giannotti, M. Nanni, F. Pinelli, et al., Trajectory pattern mining, in: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2007, pp. 330–339.
[28] Z. Yan, S. Spaccapietra, Towards semantic trajectory data analysis: A conceptual and computational approach, in: VLDB Ph.D. Workshop, 2009.
[29] A. Aizawa, An information-theoretic perspective of tf–idf measures, Inf. Process. Manage. 39 (1) (2003) 45–65.
[30] H.C. Wu, R.W.P. Luk, K.F. Wong, et al., Interpreting tf-idf term weights as making relevance decisions, ACM Trans. Inf. Syst. 26 (3) (2008) 13.
[31] K.S. Hasan, V. Ng, Automatic keyphrase extraction: A survey of the state of the art, ACL (1) (2014) 1262–1273.
[32] Krulwich Bruce, C. Burkey, Learning user information interests through extraction of semantically significant phrases, Adv. Artif. Intell. 25 (1996) 110–112.
[33] I.H. Witten, G.W. Paynter, E. Frank, et al., KEA: Practical automatic keyphrase extraction, in: Proceedings of the Fourth ACM Conference on Digital Libraries, ACM, 1999, pp. 254–255.
[34] P.D. Turney, Coherent keyphrase extraction via web mining, in: Proceedings of Ijcai, cs.lg/0308033, 2003, 434–439.
[35] Y. Zheng, L. Zhang, X. Xie, et al., Mining interesting locations and travel sequences from GPS trajectories, in: Proceedings of the 18th International Conference on World Wide Web, ACM, 2009, pp. 791–800.
[36] S.K. Bharti, K.S. Babu, Automatic keyword extraction for text summarization: A survey, 2017.
[37] Y. Liu, F. Chen, W. Kong, et al., Identifying web spam with the wisdom of the crowds, ACM Trans. Web 6 (1) (2012) 1–30.



Xu Chen is an associate professor at Wuhan University, China. His works focuses on Geographic Information Retrieval and Natural Language Processing. He holds a Ph.D. degree in Geographic Information Science from Wuhan University (2010 China) and was a visiting scholar at Carnegie Mellon University (2013 USA).

Jie Zhang is a master candidate at the State Key Laboratory of Software Engineering, Computer School, Wuhan University, China. He received his bachelor degree in Computer Science from Wuhan University in 2016. His work focuses on Geographic Information Retrieval.

Zheng Xu was born in Shanghai, China. He received the Diploma and Ph.D. degrees from the School of Computing Engineering and Science, Shanghai University, Shanghai, in 2007 and 2012, respectively. He worked as the postdoctoral in Tsinghua University from 2014 to 2016. He is currently working in the third research institute of ministry of public security, China. His current research interests include big data, public security and mobile crowd sensing.

Jin Liu is a professor in the State Key Laboratory of Software Engineering, Computer School, Wuhan University. His research interests include Software Engineering and interactive collaboration on the Web. His work has been published in several international journals including CCPE, J SUPERCOMPUT and IEEE TSE.

Please cite this article in press as: X. Chen, et al., HIB-tree: An efficient index method for the big data analytics of large-scale human activity trajectories, Future Generation Computer Systems (2018), https://doi.org/10.1016/j.future.2018.01.004.