AXE: Objects Search in Mobile Volunteered Service

Yao Wu[1,2], Wenjuan Liang[1,2], Yuncheng Wu[1,2], Hong Chen[1,2(✉)], and Cuiping Li[1,2]

[1] Key Laboratory of Data Engineering and Knowledge Engineering of Ministry of Education, Beijing, China
[2] School of Information, Renmin University of China, Beijing, China
{ideamaxwu,wenjuanliang,yunchengwu,licuiping,chong}@ruc.edu.cn

Abstract. Proliferation of ubiquitous smartphones makes location based services prevalent. People carry these devices around everyday and everywhere, which makes mobile volunteered services emerging. As far as we know, little work has been done on the search for mobile spatial textual objects, even though considerable researches have been done on moving objects query and spatial keyword query. In this paper, we study the problem of searching for mobile spatial textual objects in mobile volunteered services: given a set of mobile object and a user query, find the most relevant objects considering both spatial locations and textual descriptions. We model each mobile object as probabilistic instances with time recency. A new hybrid index is proposed for mobile spatial textual objects, called BIG-tree. And we propose an improved threshold algorithm to efficiently process the top-k query based on the index. We evaluate the performance of our approaches on real and synthetic datasets. Experimental results show our solutions outperform the baselines.

1 Introduction

Popularity of mobile devices makes location based services a compelling paradigm, which bring unique location-aware experiences to users. Location-based services hasten the advent of mobile crowdsourcing, and the need of searching for qualified mobile users becomes urgent [1]. Sometimes we submit certain queries, but cannot get satisfied results from traditional search engines, such as *get me a cab*, *fix my computer*, and *I need a house cleaner*. From these queries, what we expect is someone who can provide specific services with domain skills. In reality, people with skills like taxi driving, computer fixing and house cleaning are just around us. Xie et al. [2] propose a Volunteered Geographic Service system, which enables mobile users to efficiently monitor nearby services. In the proposed scenarios, mobile users are able to volunteer as providers of specific location-based services, and they are also able to subscribe specific services. With the diffusion of online mobile users, it is necessary to provide efficient means of enabling such queries for mobile volunteers.

I. Romdhani et al. (Eds.): CollaborateCom 2017, LNICST 252, pp. 73–87, 2018.
https://doi.org/10.1007/978-3-030-00916-8_8

Existing LBS systems employ a spatial keyword search approach to provide LBS services [3,4], in which, given a set of point of interests (POIs) or region of interests (ROIs) and a user query with a location and several keywords, all relevant POIs or ROIs are returned. Current researches mainly focus on query over static objects, such as restaurants, hotels or sights. Little research has been done on query for mobile objects with both spatial and textual information, for example, *"find the nearest people who can fix a computer"*. We name this evolved location based services as **mobile volunteered services** for its unique features.

In mobile volunteered services, both spatial and textual information should be concerned as well as the mobility of users. No existing techniques can be employed seamlessly to enable the query without appropriate modifications. To address these issues, we introduce mobile spAtial teXtual objects sEarch (AXE). Volunteered users are the objects with textual description and location information and they move stochastically. But the range they move around is generally in their routine hotspots (e.g., home, work place and malls). They change their locations without a high speed or lasting moving, such as taxi drivers. We focus more on the mobility rather than the motion perspective (e.g., direction and speed). Our contributions are four-folds:

- We study the problem of mobile spatial textual objects search, model the objects based on probabilistic instances and measure the spatial textual similarity with time recency.
- We propose BIG-tree, a hybrid index structure based on sorted lists of spatial linear quad-tree and textual inverted index. Two baselines are proposed based on modified classic indexes, i.e., IR-tree with bounding box (IRBB) and top-k aggregation with partial score (TAPS).
- We also propose an efficient improved threshold algorithm with lazy refinement and prior pause (NEWLP) based on the new index for top-k query.
- We have conducted extensive experiments on both real and synthetic datasets. Experimental results show that our methods outperform the baselines.

2 Related Work

Spatial keyword search and moving objects/query search have been well studied for years. **Spatial Keyword Search**: The R-tree based solution is arguably the dominant index for spatial keyword query. IR-tree [5] integrates each node of R-tree with a summary of the text content in the corresponding subtree. Several variants of IR-tree exist, which optimize the efficiency, including DIR-tree, CIR-tree, CDIR-tree [5], WIR-tree [6], KR*-tree [7], IR^2-tree [3] and SKI [8]. They integrate the textual information into R-tree with different data structures. Quad-tree is another choice in indexing the spatial textual objects. SFC-QUAD [9] is based on space filling curve. IQ-tree [10] is associated with an inverted file that organizes the keyword expression. I^3-tree [11] adopts the quad-tree structure to hierarchically partition the data space into cells. ILQ-tree [12] is proposed to keep the non-empty leaf node of the quad-tree in an auxiliary disk-based one dimensional structure. **Moving Objects/Query Search**: In moving

query researches [13–15], the query moves continuously while the data objects (e.g., hotels, restaurants) are stationary. *Safe region* (e.g., MW Voronoi) is used to model the moving query. In moving objects search [16], the objects move continuously while the query is stationary. Most existing solutions for moving objects indexes can be divided into object partitioning (e.g., TPR-tree [17]) and space partitioning (e.g., B^x-tree [18]).

Few work has been done on the mobile spatial textual objects search, which is different from existing works and essential in mobile volunteered services. There are two main challenges: (1) both spatial and textual information should be concerned. (2) the mobility of objects makes it difficult to model and index the objects. In this paper, we propose probabilistic instance model to capture the mobility of users and an improved threshold algorithm is presented to process the query based on our hybrid index structure BIG-tree.

3 Model AXE

3.1 Object Model

In mobile object environments, it is infeasible for the system to track the movement of objects and store the exact locations of objects at all times [16,19]. In this situation, it is impossible for queries to produce correct results based upon old data. However, if the degree of uncertainty is controlled, then the error of the answers to certain queries can be reduced. The routine hotspots of mobile users can be augmented with probabilistic estimates of the confidence of the occurrence. Generally speaking, there are two ways to model an object's current location. One is based on the history routine data and the other is based on the latest updated location. We introduce time recency to take both history data and latest data into consideration.

Definition 1 (Mobile Spatial Textual Object). *A mobile spatial textual object is represented with a triple* $o = \langle \psi, l, r \rangle$*, where* $o.\psi$ *is the set of keywords,* $o.l$ *is the last latitude and longitude of the objects with timestamp and* $o.r$ *is the set of probabilistic instances.* Correspondingly, a query is $q = \langle \psi, l \rangle$, where $q.\psi$ is the set of query keywords and $q.l$ is the query location with time. The probabilistic instance set $o.r$ for each object is represented by instances with probability, i.e., $c_j(p_i)$. For an instance c_j of an object o_i, the probability is p_i^j. A probabilistic instance represents a routine hotspot of an object, e.g., home. Given a set of moving objects $\mathcal{O} = \{o_1, o_2, ..., o_n\}$, the instance set $\mathcal{C}$ contains all instances with positive probability, i.e., $\mathcal{C} = \{c_j | \exists i (p_i(c_j) > 0)\}$.

In this paper, we give a naive statistic method to get the instance and corresponding probability of each object. Given the history locations $\mathcal{L} = \{l_1, l_2, ..., l_m\}$ of an object o, the probability p of instance c is, $p(c) = \frac{|l \in c|}{|\mathcal{L}|}$, i.e., of all the history locations, the ratio of the locations fall in a certain range. The inference method introduces distribution information of the objects' history locations, which indicates the probability of instances, i.e., routine hotspots.

In our implementation, each certain range is divided by quad-tree, which will be illustrated in the next section. More sophisticated methods (e.g., cluster techniques) can be exploited.

3.2 Similarity Measure

Definition 2 (Spatial Similarity). *Given an object o and a query q, the spatial similarity (spatial proximity) of o and q is defined as:*

$$Sim_S(o, q) = 1 - \frac{D(o.l, q.l)}{D_{max}} \tag{1}$$

where D_{max} is the maximum distance in the location space. For a mobile object, the update of the location is undetermined, and neither the history nor the updated last location can precisely predict the current location. We introduce time recency to infer the current location with probabilistic instances aggregation and last updated location. The spatial similarity can be expressed by,

$$D(o.l, q.l) = \theta d_1(o.l, q.l) + (1 - \theta) d_2(o.r, q.l) \tag{2}$$

where $d_1()$ is the Euclidean distance and $\theta = \lambda^{-(t_q - t_u)}$ is the time recency [20]. λ is the base number that determines the rate of recency decay, t_q is the query time and θ is monotonically decreasing with $t_q - t_u$. The similarity between the query location and probabilistic instance set is

$$d_2(o.r, q.l) = \sum_j p_j d_0(o.c_j, q.l) \tag{3}$$

where $p_j, o.c_j \in o.r$ and $d_0()$ is the Euclidean distance between $q.l$ and centroid of instance $o.c_j$. The spatial similarity is based on probabilistic instances set and last location with time recency. Time recency is a key factor in measuring the spatial similarity based on locations updated by mobile objects. It is introduced in [21] and is applied widely as the measurement of recency for stream data, in which, the exponential decay function has been shown to be effective [22]. In this paper, we use the time recency to measure the importance between probabilistic instances and latest location. The validity of a location decreases when it is not updated for a long time.

Definition 3 (Textual Similarity). *Given an object o and a query q, the textual similarity (textual relevancy) of o and q is defined as:*

$$Sim_T(o, q) = \frac{\sum_{t \in o.\psi \bigcap q.\psi} w(t)}{\sum_{t \in o.\psi \bigcup q.\psi} w(t)} \tag{4}$$

where $w(t)$ is the weight of keyword t in textual information. In this paper we use the inverted document frequency (denoted by idf) as keywords weight, i.e.,

$$w(t) = ln \frac{|\mathcal{O}|}{count(t, \mathcal{O})} \tag{5}$$

where $count(t, \mathcal{O})$ *is the number of objects containing token* t. Infrequent tokens have more weights for their rarity.

Definition 4 (Spatial Textual Similarity). *Given an object* o *and a query* q*, the spatial textual similarity between* o *and* q *is defined as:*

$$Sim(o, q) = \alpha Sim_S(o, q) + (1 - \alpha) Sim_T(o, q) \tag{6}$$

where $Sim_S(o, q)$ *is the spatial similarity and* $Sim_T(o, q)$ *is the textual similarity.* $\alpha \in [0, 1]$ is a preference parameter that balances the importance between distance proximity and text relevance.

Our mobile spatial textual objects search problem is substantially different from the existing problems. (1) We focus on the mobility rather than movement. (2) The spatial model is a set of discrete probabilistic instances rather than ROI or POI. (3) We take the temporal information to spatial textual similarity measurement.

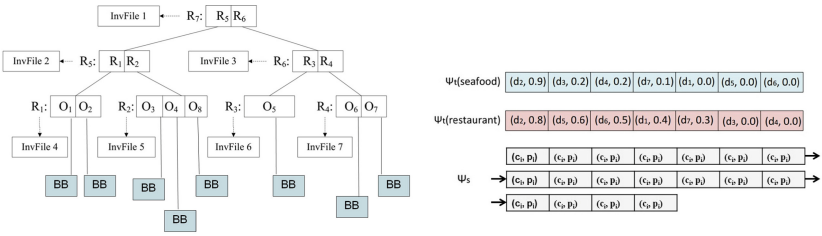
4 Handle AXE: Object Indexing

We first introduce two straightforward modified methods as baselines. Then, we present BIG-tree, a hybrid index based on sorted lists of spatial linear quad-tree and textual inverted index, which essentially consists of $\underline{B}^+$-tree and $\underline{I}$nverted file index based on $\underline{G}$rid partition.

4.1 Baseline Methods

Two types of classical indexes for spatial keyword objects are: the tree based integrated index and the sort lists based aggregation. The corresponding representatives are IR-tree [5] and RCA [23]. IR-tree integrates each node of R-tree with a summary of the text content of the objects in the corresponding subtree. Specifically, each node contains a pointer to an inverted file that describes the objects rooted at the node. RCA is a new approach based on modeling the spatial keyword query as a top-k aggregation problem [24]. Give a query with m keywords and a location, it constructs $m + 1$ sorted lists. RCA is a rank-aware combined algorithm by accessing the lists in a score-bounded manner, which is an improved TA. Experiments show RCA performs better than IR-tree when processing spatial keyword query [11,23].

We modify the IR-tree and RCA to adapt our situation. The two baselines are IR-tree with Bounding Box (IRBB) and Top-k Aggregation with Partial Score (TAPS). In the mobile spatial textual object model, each object is a set of probabilistic instances in terms of the spatial aspect. To adapt IR-tree, we draw a bounding box for the set of probabilistic instances. In IR-tree, The spatial attribute of each entry is a point, i.e., location coordinates. In IRBB, the spatial attribute is a rectangle with several probabilistic instances as shown in Fig. 1(a), i.e., the minimum quad-tree region that covers all the probabilistic instances.



(a) IR-tree with bounding box (b) Top-k aggregation with partial score

Fig. 1. Two baselines based on modified classical indexes.

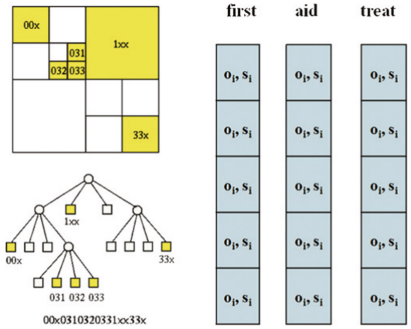


Fig. 2. Structure of BIG-tree. The left part is the linear encode based on extensive quad-tree and the right part is the sorted lists with score s_i based on keywords for object o_i.

To adapt RCA, each probabilistic instance in our model is an object in RCA model. And the similarity score of each mobile spatial textual object is the sum of all probabilistic instances. The differences of index structures between RCA and TAPS are shown as in Fig. 1(b). As we can see, the spatial list is far longer than the textual lists due to the several instances for each object. The query processing algorithms for IRBB and TAPS remain the same as in [5,23].

However, IRBB inherits the disadvantages from IR-tree. First, the centralized R-tree mechanism requires a high update cost. Second, the processing cost to examine whether a node is relevant to the query keywords is not negligible. Besides, TAPS is also not efficient when processing mobile spatial objects due to its large number of probabilistic instances. Each object in RCA becomes several instances in TAPS. Therefore, it is difficult for both baselines to scale.

4.2 Our Methods

The key idea of our optimization is to sort the spatial attribute list offline based on an approximate spatial order-preserving encoding such that the two-dimensional location attribute values are encoded into one-dimensional values with the desirable property. A pair of encoded location values that are close

together in the total order represents a pair of locations that are likely to be spatially close to each other. In this paper, we apply the well-known linear quad-tree [25] to obtain such a mapping. There are two useful properties of linear quad-tree encoding that we exploit in our algorithm. (1) A quad-tree is a space-partitioning tree data structure in which a d-dimensional space is recursively subdivided into 2^d regions, which shows simplicity and regularity in many applications. (2) As an efficient implementation of the disk-based quad-tree, the linear quad-tree is proposed to keep the non-empty leaf node of the quad-tree in an auxiliary disk-based one-dimensional structure (e.g., B^+ tree), where each node can be encoded by the space filling curve techniques.

In BIG-tree, the internal nodes are directory and the leaf nodes are the objects. The key in the tree is the one-dimensional space filling curve value, which mapped from the two-dimensional location coordinates. Each object at the leaf node is represented by a set of probabilistic instances with the last updated location. In the conventional quad-tree, the tree splits when the node is full. In our index structure, we map the bounding box into a **least quad cell**, i.e., the minimum quad-tree region that covers all the probabilistic instances. And the node splits not according to the number of items, but the size of bounding box. We call this **extensive quad-tree**. We make this modification because we only take quad-tree structure as the mapping method, and the data is stored in B^+-tree based on the one-dimensional value. The extensive quad-tree can hold a large number of objects under a quad cell, which facilitates the spatial search when we process the query.

In implementation, we encode the quad-tree nodes based on the Morton code (Z-order) because the Morton code of a node is encoded based on its split sequence, i.e., the path of the node in the quad-tree, and the code of a particular node (region) in the space is unique. This is essential because multiple quad-trees with different shapes are used in the paper. The way of how to derive the Morton code of a node based on its split sequence in two-dimensional space can be found in [25]. For the linear quad-tree, we only keep the solid leaf nodes on the disk by one-dimensional index structure (e.g., B^+-tree), which are ordered by their Morton codes. We apply a temporary buffer to solve the false positive problem as described in [23].

However, based on the properties of the linear quad-tree encoding, it is possible that some of the documents accessed in the searched spatial region (specified by some range of linear quad-tree values) could be false positives; i.e, the actual distance between the accessed document and query could be larger than the current search radius r_i at the i^{th} iteration. To avoid processing these false positives too early, we maintain buffers to temporarily store these false positive documents: a false positive document that should have been processed later in the j^{th} iteration (i.e., $j > i$) will be temporarily stored in the j^{th} buffer. Thus, the documents in the j^{th} buffer will be considered later during the j^{th} iteration.

Based on the encoded linear quad-tree, our algorithm progressively accesses the documents (objects) in the spatial attribute list in iterations. The textual lists can be easily obtained. We index the mobile spatial textual objects by BIG-tree, shown in Fig. 2, which is a new hybrid index structure. The BIG-tree

can be easily integrated into an existing relation database. We skip the BIG-tree manipulation (insertion and deletion) details due to the space limitation.

Existing geo-textual indexes are designed for processing spatial-keyword queries on static objects. They do not provide an efficient mechanism to maintain the index for the objects with mobile information. In **BIG-tree**, we encode the linear extensive quad-tree to construct the spatial sorted list and the textual lists with inverted files. To optimize the query processing, we apply quad-tree based spatial structure rather than R-tree for frequent updates. The reasons for using grid based structure are: (1) The grid based structure is update-friendly because the query region is indexed in mutually-exclusive cells. (2) We can use different indexing granularity for different queries by considering the spatial distribution and keyword distribution.

5 Wield AXE: Query Processing

5.1 Query Processing

In this paper, we focus on the top-k query. Given a query $q = \langle \psi, l, k \rangle$, retrieve k objects ranked according to the mobile spatial-textual similarity.

Definition 5 (Top-k Query). *Given a set of objects O and a query q, the answer set of a top-k mobile spatial textual objects search is a subset of O, A, such that*
(1) The size of A is k ,i.e., $|A| = k$, and (2) $\forall o^* \in A, \forall o \in O - A, Sim(q, o^*) \geq Sim(q, o)$
where Sim is the similarity function to evaluate the relevancy between a query and an object. According to the similarity measurement in Sect. 3, what differ us are the probabilistic instance set and time recency in measuring the similarity score between the query and each mobile spatial textual object.

Before the improved algorithm for TkQ, we present the threshold algorithm for top-k aggregation problem [24] in Algorithm 1, which consists of two main steps. First, perform a sorted access in parallel to each of sorted lists and compute the aggregated score of the object using the ranking function (Line 8). Then, update the threshold according to the aggregated score of the last seen object (Line 9–12). The algorithm terminates as soon as at least k objects have been seen whose score is at least equal to the threshold (Line 4–6).

Based on the threshold algorithm, we adapt our BIG-tree to the top-k query, specifically based on RCA [11], which is Rank-aware Combined Algorithm. We rank the spatial list based on quad-tree mapping rather than uniform grid in RCA. We name the modified RCA as mRCA. There are two main methods to optimize the threshold algorithm. One is to early terminate the iteration when accessing the sorted lists, as the RCA does. The other is to reduce the random access, which can reduce I/O cost. Based on these observations, to improve the efficiency, we propose optimized solution NEWLP for TkQ.

Algorithm 1. Threshold Algorithm

Input: $\mathcal{O}$: A set of objects
q: A query
$\mathcal{I}$: Inverted lists of spatial and textual attributes
Output: $\mathcal{R}$: Top-k results
1: $\mathcal{Q} \leftarrow$an empty prior queue
2: $\theta_{ta} \leftarrow 1, \theta_q \leftarrow 0$
3: **while** true **do**
4: **if** $\theta_{ta} < \theta_q$ **then**
5: $\mathcal{R} \leftarrow$Refinement($\mathcal{Q}$)
6: **return** $\mathcal{R}$
7: **end if**
8: candidate set $\{o_i\} \leftarrow$ SequentialAccess($q, \mathcal{I}$)
9: **if** $|\mathcal{Q}| < k$ or $Sim(o_i, q) > \theta_q$ **then**
10: add object o_i to queue $\mathcal{Q}$
11: update threshold θ_q
12: **end if**
13: $\mathcal{R} \leftarrow$Refinement($\mathcal{Q}$)
14: $\theta_{ta} \leftarrow Sim(o_i, q)$
15: **end while**

Algorithm 2. NEWLP Improvements

1: **SequentialAccess**($q, \mathcal{I}$)
2: **if** o_i.score$< \theta_{score}$ OR o_i.num$> \theta_{num}$ **then**
3: update $\theta_{score}, \theta_{num}$
4: **break** sequential access
5: **end if**
6:
7: **Refinement**($\mathcal{Q}$)
8: **if** o_i.score$> \theta_{ran}$ **then**
9: RandomAcess(o_i)
10: **end if**

5.2 Lazy Refinement

In conventional threshold algorithms, the refinement of top-k results is carried out in every iteration. The access of the index increases the disk I/O, which incurs time latency. In NEWLP, we proceed random access until the score of the top-k results reach the threshold. We set a threshold to stop the unnecessary random access. The refinement of the top-k results is processed once they exceed the threshold, otherwise, they will be refined later with the other candidates. The modification takes place in Line 13 in Algorithm 1.

5.3 Prior Pause

We terminate each iteration when the score-bounded or number-bounded conditions satisfy in the sequential access. There are two core factors affecting the efficiency of TA: termination conditions and the number of random access times. In contrast to the RCA algorithm, which accesses only a score-bounded documents in each iteration, the documents accessed by Prior Pause is determined by a score interval and a fixed number of documents corresponding to each iteration. This difference between RCA and Prior Pause is motivated by two reasons. First, the upper-bound score for the ranking function relies on the minimum distance of all the unseen objects to the query location. The upper bound score could decrease very slowly if a fixed number of documents is accessed per iteration. Second, The drawback of score bounded method is that it may lead too

Table 1. Dataset statistics.

Dataset	# of objs	# of locs	# of terms	Region	Size
Twitter	61413	9.6556	90.4253	(24.54, −134.54, 58.41, −67.45)	138M
Foursquare	4163	64.6002	138.8080	(−54.80, −175.19, 83.97, 178.59)	78M
Syndata	194466	6.5	9.5	(−54.80, −175.19, 83.97, 178.59)	100M

Table 2. Parameters and their settings.

Parameters	Settings
Different datasets	Twitter FS **Syndata**
# of syn objects	40K 60K 100K 150K **200K**
# of query kwds	2 3 **4** 5 6
# of tok-k results	1 5 **10** 20 50
Parameter α	0.1 0.3 **0.5** 0.7 0.9
Skewness ratio	5.18 3.58 2.88 1.28 0.58 **0**

much unnecessary access if there is a large amount of objects in a certain score interval. Prior Pause wield the power of these two tricks. The modification lies in Line 8 in Algorithm 1.

The overall improvement for TA is explained in Algorithm 2, which gives the outlines of SequentialAccess($q, \mathcal{I}$) and Refinement($\mathcal{Q}$). Details can refer to [11]. In the improvements, $\theta_{score}, \theta_{num}, \theta_{ran}$ indicate score-bounded threshold, fixed number threshold and random access threshold, respectively.

6 Performance Evaluation

We modify and extend two classical methods as baselines to handle our problem, i.e., IRBB based on IR-tree and TAPS based on top-k aggregation. We compare our proposed mRCA and improved version NEWLP with the two baselines. We conduct our experiments on two real data sets and one synthetic data set from different evaluation metrics. All the algorithms are implemented in Java and run on a Linux server machine with an Intel(R) Xeon(R) CPU E5-2670 0 @ 2.60 GHz and 256 GB memory. We run each experiment three times and report the average results.

6.1 Data Sets

To the best of our knowledge, no available dataset targeting the envisioned mobile volunteered service exists. We use two real datasets[1] and one synthetic dataset to evaluate the performance. The first one is a real Twitter dataset. It provides check-ins data with content information across USA. The second real dataset is Foursquare. This dataset contains check-in records with check-in venues. These two datasets all contains check-in locations, check-in time, text content and user ID. We aggregate the locations and text content of each user as one mobile spatial object. Each location acts as an instance with occurrence probability. As there are too many instances and keywords for one object, which

[1] https://sites.google.com/site/dbhongzhi/.

is far more to describe the object, we generate a synthetic dataset based on the two real datasets. We split one "fat" object from aggregated real datasets into several virtual objects in the synthetic dataset. This can preserve the spatial proximity and text relevance from each user and generate reasonable data with average 6.5 instances and 9.5 keywords for each user. It is enough to describe the service skills of a user by 10 keywords (instead of a long web-page document) and to represent the routine hotspots by 7 instances. Table 1 summarizes details of the three datasets.

6.2 Evaluation Metrics

In the following experiments, we evaluate the performance of different indexes and algorithms from several aspects. We study the metrics including: (1) index building time; (2) index disk storage; (3) query processing time. The different aspects are: (1) different datasets; (2) varying size of datasets; (3) varying number of query keywords; (4) varying number of return results; (5) varying α in similarity measure and (6) skewness of query keywords. Table 2 summarizes the settings, where values in bold represent the defaults.

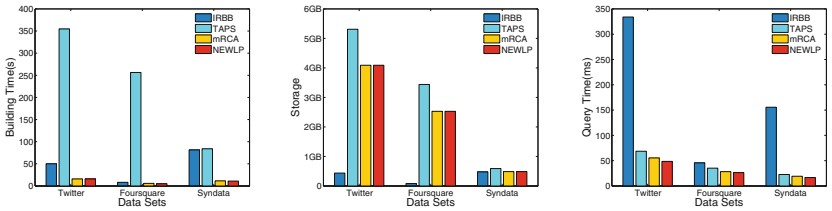
Index Construction. (1) Index Construction Time Index construction cost is used to measure the elapsed time when building the index. The time is affected by various factors, such as cache size and code optimization. It is difficult to provide a thoroughly fair comparison and we report the average time. **(2) Index Storage Cost** We use the size of index on disk to evaluate the space efficiency. The index storage includes the spatial and textual attribute.

Query Processing. The query parameters include number of keywords, query skewness, size of k and preference parameter α. To make the queries resemble what users would likely use, we generate the query sets randomly from the provided datasets. For each round of experiments, we generate query sets on each dataset, in which the number of query keywords is from 1 to 7 with an average 4. Each set consists of 30 queries. For evaluating the query performance free from cache impact, we generate 1,000 queries for experiments to warm up the experiment system.

6.3 Experiment Results

Different Datasets. We evaluate the index building time, index disk storage and query processing time on three different datasets as shown in Table 1. The two baseline methods IRBB, TAPS and our proposed algorithm mRCA and NEWLP are compared. The experiments results are shown in Fig. 3. TAPS shows high building time due to the possible massive probabilistic instances. In terms of index disk storage, the sorted lists based methods (i.e., TAPS, mRCA and NEWLP) on the real data sets (i.e., Twitter and Foursquare) incur higher costs. This is because for each object in the real datasets, the merged object has too

many instances and keywords, which leads to duplicates in the storage. When the instances and keywords decrease as shown in Fig. 3(b) - Syndata, the storage costs are similar for all indexes. In fact, the Syndata more resembles the real data format from mobile volunteered service if it exists. As for the query processing time, NEWLP performs best of all, which is about five times better than IRBB. To be noted, mRCA and NEWLP share the same index structure (BIG-tree), which makes the index building time and index storage quite the same.



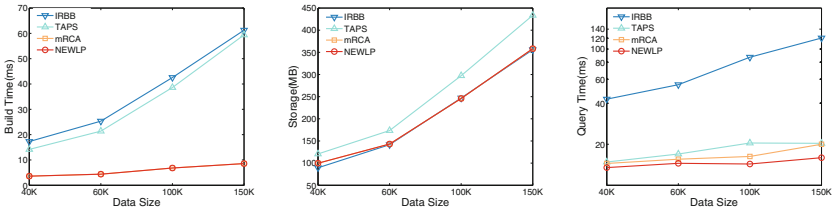
(a) Building Time (b) Disk Storage (c) Query Time

Fig. 3. Different datasets.

Varying Size of Datasets. In this experiment, we examine the scalability in terms of increasing size of datasets. We vary the size of synthetic data from 40K to 200K. As shown in Fig. 4, when the size of datasets increases, the efficiency of four indexes and processing algorithms decrease. However, the three aspects of building index time, index disk storage and query processing, our proposed BIG-tree and NEWLP perform best of all. As we have mentioned, mRCA and NEWLP share the same index as BIG-tree but NEWLP is an improved version in terms of query processing. BIG-tree maps the bounding box of set of probabilistic instances from quad-tree to one-dimensional value, which reduces the duplicates of storage. NEWLP optimize the threshold algorithm in two aspects, i.e., early iteration termination and reduced random access. Results show that these two optimizations work.

Varying Number of Query Keywords. We vary the number of query keywords from 2 to 6 to evaluate the query processing algorithms on the synthetic dataset. Since an object is considered relevant if it contains at least one query keyword, results in Fig. 5 show the query time increases as the number of query keywords increases especially for IRBB. However, the sorted lists based methods scale smoothly because the pruning techniques that take into account both spatial and textual relevance are more effective.

Varying Number of Query Results. The number of query results, i.e., the value of k is set from 1 to 50 to evaluate all the algorithms. The IRBB query time increases quickly while the sorted lists based methods work well without much fluctuation. The query processing latency for TAPS, mRCA and NEWLP



(a) Building Time (b) Disk Storage (c) Query Time

Fig. 4. Varying size of datasets.

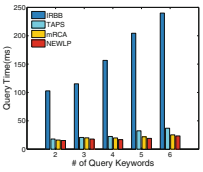


Fig. 5. Number of query keywords.

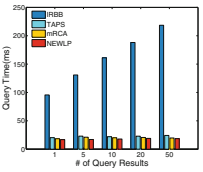


Fig. 6. Number of query results.

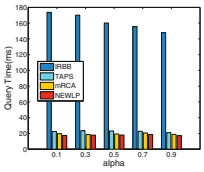


Fig. 7. Varying α in ranking function.

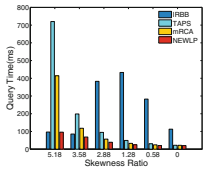


Fig. 8. Skewness of query keywords.

is mainly affected by the sorting of spatial lists, sequential access and random access of the sorted lists. As k increases, the overhead of sorting is fixed and the running time increases due to the sequential/random access. NEWLP performs better than mRCA due to its lazy refinement and prior pause that can reduce random access and terminate the iteration earlier (Fig. 6).

Varying α in Ranking Function Preference parameter α indicates the importance of spatial and textual attributes in the similarity measurement. As α increases, the spatial relevance plays a more important role in determining the final results. IRBB is more sensitive to α because it examines the spatial attribute first. When α is larger, the spatial relevance dominates the results and this reduces the running time. On the contrary, the other three algorithms are not sensitive to α because the spatial and textual sorted lists are accessed in parallel (Fig. 7).

Skewness of Query Keywords. We evaluate the impact of query term frequency skewness. The skewness ratio is defined as term frequency compared to average frequency of all keywords. For each keyword, the frequency is n and the average frequency is m. The skewness ratio for each keyword is $ln(n/m)$. Results in Fig. 8 show that, the frequent query keyword incurs high query time for sorted lists based methods. The reason is that when the query keyword frequency is high, the sorted lists for keywords become long, which incurs time latency to access the lists. But NEWLP still shows advantages over others. Notice that IRBB performs better for both high frequency and low frequency query keywords. One reasonable explanation is that, the spatial pruning power generates and dominates to terminate the iteration early.

7 Conclusions

We have studied the problem of mobile spatial textual objects search in mobile volunteered services. We model the objects based on probabilistic instances and measure the similarity with time recency. We propose a new hybrid index structure, BIG-tree, to manage the mobile objects, which encodes the extensive quadtree mapping into one-dimension value. We also propose improved threshold algorithm with lazy refinement and prior pause based on the index. Extensive experimental results show that our methods outperform the two proposed baselines and achieve better performance.

One apparent limitation of this study is that the data sets we use are slightly modified from mobile social networks rather than mobile volunteered services. Actually, we are under implementation of a system that supports the proposed mobile volunteered service, which will bring it to reality. A demonstration video can be found at Youtube[2]. We hope this study may open many new interesting and challenging problems that need further research.

Acknowledgment. This work is supported by National Science Foundation of China (No. 61532021), National Basic Research Program of China (973) (No. 2014CB340403), and National High Technology Research and Development Program of China (863) (No. 2014AA015204).

References

1. Cen Chen, S.-F.C., Hoong Chuin Lau, A.M.: Towards city-scale mobile crowdsourcing: task recommendations under trajectory uncertainties. In: IJCAI, pp. 1113–1119 (2015)
2. Xie, X., Jin, P., Yiu, M.L., Du, J., Yuan, M., Jensen, C.S.: Enabling scalable geographic service sharing with weighted imprecise voronoi cells. TKDE **28**(2), 439–453 (2016)
3. De Felipe, I., Hristidis, V., Rishe, N.: Keyword search on spatial databases. In: ICDE, pp. 656–665 (2008)
4. Cao, X., et al.: Spatial keyword querying. In: Atzeni, P., Cheung, D., Ram, S. (eds.) ER 2012. LNCS, vol. 7532, pp. 16–29. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34002-4_2
5. Cong, G., Jensen, C.S., Dingming, W.: Efficient retrieval of the top-k most relevant spatial web objects. PVLDB **2**(1), 337–348 (2009)
6. Wu, D., Yiu, M.L., Cong, G., Jensen, C.S.: Joint top-k spatial keyword query processing. TKDE **24**(10), 1889–1903 (2012)
7. Hariharan, R., Hore, B., Li, C., Mehrotra, S.: Processing spatial-keyword (SK) queries in geographic information retrieval (GIR) systems. In: SSDBM, p. 16 (2007)
8. Cary, A., Wolfson, O., Rishe, N.: Efficient and scalable method for processing top-k spatial boolean queries. In: Gertz, M., Ludäscher, B. (eds.) SSDBM 2010. LNCS, vol. 6187, pp. 87–95. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13818-8_8

[2] https://www.youtube.com/watch?v=8LtMZ3AqVTc.

9. Christoforaki, M., He, J., Dimopoulos, C., Markowetz, A., Suel, T.: Text vs. space: efficient geo-search query processing. In: CIKM, pp. 423–432 (2011)
10. Chen, L., Cong, G., Cao, X.: An efficient query indexing mechanism for filtering geo-textual data. In: SIGMOD, pp. 749–760 (2013)
11. Zhang, D., Tan, K.-L., Tung, A.K.H.: Scalable top-k spatial keyword search. In: EDBT/ICDT, pp. 359–370 (2013)
12. Zhang, C., Zhang, Y., Zhang, W., Lin, X.: Inverted linear quadtree: efficient top k spatial keyword search. In: ICDE, pp. 901–912 (2013)
13. Wu, D., Yiu, M.L., Jensen, C.S., Cong, G.: Efficient continuously moving top-k spatial keyword query processing. In: ICDE, pp. 541–552 (2011)
14. Huang, W., Li, G., Tan, K.-L., Feng, J.: Efficient safe-region construction for moving top-k spatial keyword queries. In: CIKM, pp. 932–941 (2012)
15. Wu, D., Yiu, M.L., Jensen, C.S.: Moving spatial keyword queries: formulation, methods, and analysis. ACM Trans. Database Syst. **38**(1), 7 (2013)
16. Zhang, M., Chen, S., Jensen, C.S., Ooi, B.C., Zhang, Z.: Effectively indexing uncertain moving objects for predictive queries. PVLDB **2**(1), 1198–1209 (2009)
17. Saltenis, S., Jensen, C.S., Leutenegger, S.T., López, M.A.: Indexing the positions of continuously moving objects. In: SIGMOD, pp. 331–342 (2000)
18. Jensen, C.S., Lin, D., Ooi, B.C.: Query and update efficient B+-tree based indexing of moving objects. In: VLDB, pp. 768–779 (2004)
19. Cheng, R., Prabhakar, S., Kalashnikov, D.V.: Querying imprecise data in moving object environments. In: ICDE, pp. 723–725 (2003)
20. Chen, L., Cong, G., Cao, X., Tan, K.-L.: Temporal spatial-keyword top-k publish/subscribe. In: ICDE, pp. 255–266 (2015)
21. Li, X., Croft, W.B.: Time-based language models. In: CIKM, pp. 469–475 (2003)
22. Efron, M., Golovchinsky, G.: Estimation methods for ranking recent information. In: SIGIR, pp. 495–504 (2011)
23. Zhang, D., Chan, C.-Y., Tan, K.-L.: Processing spatial keyword query as a top-k aggregation query. In: SIGIR, pp. 355–364 (2014)
24. Fagin, R., Lotem, A., Naor, M.: Optimal aggregation algorithms for middleware. J. Comput. Syst. Sci. **66**(4), 614–656 (2003)
25. Gargantini, I.: An effective way to represent quadtrees. Commun. ACM **25**(12), 905–910 (1982)