Efficient Top K Temporal Spatial Keyword Search

Chengyuan Zhang†, Lei Zhu†, Weiren Yu‡, Jun Long†, Fang Huang†, Hongbo Zhao♮

† School of Information Science, Central South University, PR China
‡ School of Engineering and Applied Science, Aston University, United Kingdom
♮ School of Minerals Processing and Bioengineering, Central South University, PR China
{cyzhang,leizhu,jlong,hfang,zhbalexander}@csu.edu.cn,w.yu3@aston.ac.uk

Abstract. *Massive amount of data that are geo-tagged and associated with text information are being generated at an unprecedented scale in many emerging applications such as location based services and social networks. Due to their importance, a large body of work has focused on efficiently computing various spatial keyword queries. In this paper,we study the top-k temporal spatial keyword query which considers three important constraints during the search including time, spatial proximity and textual relevance. A novel index structure, namely SSG-tree, to efficiently insert/delete spatio-temporal web objects with high rates. Base on SSG-tree an efficient algorithm is developed to support top-k temporal spatial keyword query. We show via extensive experimentation with real spatial databases that our method has increased performance over alternate techniques .*

Keywords: Temporal spatial keyword, high rates, memory

1 Introduction

Due to the proliferation of user generated content and geo-equipped devices, massive amount of microblogs (e.g., tweets, Facebook comments, and Foursquare check-ins) that contain both text information [13] and geographical location information are being generated at an unprecedented scale on the Web. For instance, in the GPS navigation system, a POI (point of interest) is a geographically anchored pushpin that someone may find useful or interesting, which is usually annotated with textual information (e.g., descriptions and users reviews). In social media (e.g., Flickr, Facebook, FourSquare, and Twitter), a large number of posts and photos are usually associated with a geo-position as well as a short text. In the above applications, a large volume of spatio-textual objects may continuously arrive with high speed. For instance, it is reported that there are about 30 million people sending geo-tagged data out into the Twitterverse, and 2.2 percentage of the global tweets (about 4.4 million tweets a day) provide location data together with the text of their posts[1].

In this paper, we aim to take advantage of the combination of geo-tagged information within microblogs to support temporal spatial keyword search queries on microblogs, where users are interested in getting a set of recent microblogs each of which contains all keywords and closet to user's location. Due to the large numbers of microblogs that can satisfy the given constraints, we limit the query answer to k microblogs, deemed most relevant to the querying user based on a ranking function f_{st} that combines the time recency and the spatial proximity of each microblog to the querying user.

Example 1. In Fig. **??**, suppose there are a set of tweets, each of which is described by the send's interests, the sender's location and creation time of tweet. When a GPS-enabled smartphone user wants to find the most recent tweet who has the same interests as him and closes his location, he may send the local search server two keywords, *swimming* and *gym*. Based on the user's current location derived from the smartphone , the two query keywords, and the creation time, tweet o_1 is returned by the server. Note that although tweet o_3 is closer to Q than o_1, it doesn't satisfy the keyword constraint. Tweet o_4 satisfies the keyword constraint and spatial closer to Q than o_1, but the creation time of o_4 is too early.

Challenges. There are three key challenges in efficiently processing temporal spatial keyword queries over temporal spatial keyword microblogs streams. Firstly, a massive number of microblogs, typically in the order of millions, are posted in many applications, and hence even a small increase in efficiency results in significant savings. Secondly, the streaming temporal spatial keyword microblogs may continuously

[1] http://www.futurity.org/tweets-give-info-location

arrive in a rapid rate which also calls for high throughput performance for better user satisfaction. Thirdly, the above challenges call for relying on **only** in-memory data structures to index and query incoming microblogs, where memory is a scarce resource.

Based on the above challenges, we first discuss what kinds of techniques should be adopted from different angles. Then, we propose a novel index technique, namely the Segment Signature Grid trees (SSG-Trees for short), to effectively organize continuous temporal spatial keyword microblogs. In a nutshell, SSG-Trees is essentially a set of Signature Grid-Trees, each node of which is enriched with the reference to a frequency signature file for the objects contained in the sub-tree rooted at the node. Then, an efficient temporal spatial keyword search algorithm is designed to facilitate the online top-k temporal spatial keyword search. Extensive experiments show that our SSG-Trees based **TSK** algorithm achieves very substantial improvements over the nature extensions of existing techniques due to strong filtering power.

The rest of this paper is organized as follows. Section 2 formally defines the problem of top k temporal spatial keyword search. We introduce the techniques should be adopted in Section 3. Section 4 presents the framework of SSG-Trees and algorithm. Extensive experiments are reported in Section 5.

2 Preliminaires

In this section, we present problem definition and necessary preliminaries of top k temporal spatial keyword Search. Table 1 below summarizes the mathematical notations used throughout this section.

Notation	Definition
$o(q)$	s geo-textual object (query)
$o.\psi(q.\psi)$	a set of keywords (terms) used to describe o (query q)
$o.loc(q.loc)$	location of the object o (query q)
$o.t(q.t)$	timestamp of the object o (query q)
$\mathcal{V}$	vocabulary
w	a keyword (term) in $\mathcal{V}$
l	the number of query keywords in $q.\psi$
w	the number of results should be returned
m	the number of independent uniformly random hash functions
α	the preference parameter to balance the spatial proximity and temporal recency
b	size of a bit-block of GS
B	the sparse vector size
c	the leaf node capacity of SSG-Trees
$f_s(o.loc, q.loc)$	the spatial proximity between o.loc and q.loc
$f_t(o.t, q.t)$	the temporal recency between o.t and q.t
$f_{st}(o, q)$	the spatial-temporal ranking score between o and q

Table 1: Notations

In this section, $\mathcal{O}$ denotes a sequence of incoming stream geo-textual objects. A **geo-textual object** is a textual message with geo-location and timestamp, such as geo-tagged and check-in tweets. Formally, a geo-textual object o is modeled as $o = < \psi, loc, t >$, where $o.\psi$ denotes a set of distinct keywords (terms) from a vocabulary set $\mathcal{V}$, $o.loc$ represents a geo-location with latitude and longitude, and $o.t$ represented the timestamp of object.

Definition 1 (Top-k Temporal Spatial-Keyword (TSK) Query). *A top-k temporal spatial keyword query q is defined as $q = < \psi, loc, t, k >$, where $q.\psi$ is a set of distinct user-specified keywords (terms), q.loc is the query location, q.t is the user submitted timestamp, k is the number of the result user expected.*

Definition 2 (Spatial Proximity $f_s(o.loc, q.loc)$). *Let δ_{max} denote the maximal distance in the space, the spatial relevance between the object o and the query q, denoted by $f_s(o.loc, q.loc)$, is defined as $\frac{\delta(q.loc, o.loc)}{\delta_{max}}$.*

Definition 3 (Temporal Recency $f_t(o.t, q.t)$). *Let λ_{max} denote the maximal time difference in the timestamp, the temporal recency between the object o and the query q, denoted by $f_t(o.t, q.t)$, is defined as $\frac{\lambda(o.t,q.t)}{\lambda_{max}}$.*

Based on the spatial proximity and temporal recency between the query and the object, the **Spatial-temporal Ranking Score** of an object o regarding the query q can be defined as follows.

Definition 4 (Spatial-temporal Ranking Score $f_{st}(o, q)$). *Let α $(0 \leq \alpha \leq 1)$[2] be the preference parameter specifying the trade-off between the spatial proximity and temporal recency, we have*

$$f_{st}(o, q) = \alpha * f_s(o.loc, q.loc) + (1 - \alpha) f_t(o.t, q.t). \quad (1)$$

*Note that the objects with the **small score values** are preferred (i,e., ranked higher).*

Definition 5 (Temporal Spatial Keyword Search). *Given a set of geo-textual objects $\mathcal{O}$ and a temporal spatial keyword query q, we aim to find the top k geo-textual objects with **smallest** spatial-temporal score, and each of which contains **all** of the query keywords.*

In the section hereafter, we abbreviate the geo-textual object and the geo-textual query as *object* and *query* respectively, if there is no ambiguity. We assume there is a total order for keywords in $\mathcal{V}$, and the keywords in each query and object are sorted accordingly. For presentation simplicity, we assume $w_i < w_j$ if $i < j$.

3 Motivation

Due to massive amount of objects and queries are being generated at an unprecedented scale, it is imperative to devise efficient indexing technique such that high arrival rates of incoming objects can be inserted immediately, expired objects can be deleted from its contents with the approximative rate as insertion, a large number of unpromising objects can be filtered at a cheap cost, and the memory cost should linear to the object size increase. We show that a good indexing mechanism over continuous geo-textual objects should satisfy following three criterion.

3.1 Efficient Textual Retrieval

Existing textual retrieval indexes , which can effectively combine with other spatial or temporal indexes, are mainly falling into one of two categories: inverted index [12,15,21] and signature file [14,25,23,28]. Owing to only the indexes of related keywords have been extracted in inverted index, inverted index excels in query processing efficiency while compared with signature. However, signature has faster insertion speed and utilizes significantly less storage overhead. According to [2], inverted index requires much larger space overhead than signature file (≈10 times), and demands expensive updates of the index when insert a new document, due to many terms of inverted index needs to store more than once and frequently undergo re-organization triggers under intensive information insertion/updating procedures. Furthermore, the inverted index is also reported to perform poorly for multiple terms queries in [4]. Obviously, taking the properties of fast update online system into consideration, signature file seems a better choice.

3.2 Efficient Spatial Partition

To support high arrival rates of incoming objects, space-partitioning index (e.g., Quadtree [6,17,18,26,27], Pyramid [1,20,22], and Grid structure [10,12,16]) is more famous than object-partitioning index (e.g., R-tree). As stressed in [9], space-partitioning index is more suitable to high update system because of its disjoint space decomposition policy, while the shape of object-partitioning index is highly affected by the rate and order of incoming data, which may trigger a large number of node splitting and merging. Motivated by this, we should adopt space-partitioning index as our proposed spatial partition index.

[2] α=1 indicates that the user cares only about the spatial proximity of geo-textual objects, α=0 gives the k most recent geo-textual objects in dataset

3.3 Efficient Temporal Partition

Regarding the temporal partition techniques, which can combine with other textual or spatial index, are mainly divided into two categories: Log Structure [11] and Sliding Window [8]. Log Structure partitions the data into a sequence of indexes with exponentially increasing size, while Sliding Window partitions the data into a sequence of indexes with equal size or with equal time range. Obviously, the performance of Log Structure is better than Sliding Window if top-k results can be find in most recent data. However, a sharp drop will be met if the top-k results can be find in most recent data, due to its exponent increase size partition. Furthermore, the insertion cost of Log Structure will significant increase while combining with other index. Finally, the deletion cost of Log Structure is always higher than that of Sliding Window, due to the deletion operation can only occur at the oldest index. Motivated by the above reasons, our proposed temporal partition strategy should fall into Sliding Window.

4 SSG-TREE FRAMEWORK

Based on the above motivations, in this section, we present a segment signature grid trees (SSG-Trees for short) that supports update at high arrive rate and provides the following required functions for geo-textual object search and ranking: I)**textual filtering**: all the textually irrelevant trees, nodes and objects have to be discarded as early as possible to cut down the search cost; II)**spatial filtering**: all the spatially irrelevant nodes have to be filtered out as early as possible to shrink the search space; III)**temporal filtering**: all the spatially irrelevant trees, nodes and objects have to be accessed as late as possible to follow the chronological order; and IV)**relevance computation and ranking**: since only the top-k objects are returned and k is expected to be much smaller than the total number of match objects, it is desirable to have an incremental search process that integrates the computation of the joint relevance, and object ranking seamlessly so that the search process can stop as soon as the top-k objects are identified.

Below, we first introduce frequency signature to support keyword filtering in section 4.1. Section 4.2 presents the gird tree for spatial partition. Detail data structure and search algorithm are depicted in section **??** and section 4.3 respectively.

4.1 Frequency Signature

The traditional superimposed coding signature is widely used in many off-line indexes such as *spatio-textual* [3], tow-level superimposed coding [7,19,24], IR^2-tree [5] etc. It is well known that the frequency of keyword occurrence in large texts follows Zipf's law. However, the traditional superimposed coding signature does not differentiate the frequencies of different keywords in the dataset. Hence, in this subsection, we study frequency superimposed coding signature based on the keywords's frequency. In many applications, keyword frequencies are estimated or collected with historical data. Statistics of such information are maintained, especially for the high frequency keywords. Such data is useful in optimizing the optimal configuration of superimposed coding signature. The performance improvement is remarkable even with rough estimations of keyword frequencies.

Same as the traditional superimposed coding, the frequency superimposed coding signature also uses k independent uniformly random hash functions map an n-terms set $\mathcal{W}=\{w_1, \cdots, w_n\}$ into a B-bit array. But the major difference is that instead of hashing all terms in range [1,B], we divide the B-bit array into a set of different size frequency blocks, and hash the terms into different blocks based on their term frequency. More specifically, based on the distribution of term frequency, we partition the terms into different frequency block by a series of frequency thresholds. Assume the aggregate frequency of the entire historical data is ξ, the bit array is divided into u frequency blocks, and the aggregate frequency of the i-th frequency block is ξ_i. Hence, the size of the i-th block can be calculated by $\frac{\xi_i}{\xi} \times B$. Then, for each term in n-terms set, we can hash them into different frequency blocks. Due to the terms have similar frequency are partitioned into the same blocks and the blocks allocated to high frequency terms have been assign more bits and less terms, the frequency superimposed coding signature effectively avoid the interference from low frequency terms.

4.2 Grid-Tree

The existing space-partitioning techniques are mainly falling into two categories: Grid structure and Quadtree. However, both of them have their own limitations while combined with signature file. The Grid structure is insensitive to system update, but it is hard to decide its granularity. High granularity will cause massive amount of node signature, which will lead to tremendous memory cost. Low granularity will result in a great deal of fat leaf nodes that contain several thousand of objects, which will significantly reduce query performance. Different as Grid structure, Quadtree can dynamically adjust granularity according to object distribution to achieve balance allocation after sacrificing partial update efficiency. Because each internal node of Quadtree has exactly four children, for the leaf node, it is easy to satisfy the leaf node capacity constraint, trigger node split, and complicated object redistribution. Thus, instead of splitting into four children nodes in Quadtree, we partition node into a set of grids. More specifically, the Grid-Tree partitions the spatial node into n^2 equal non-overlapping girds, where $n \geq 2$, to delay the time of redistribution and avoid continual node split. Evidently, the spatial queries algorithms that can be applied on Quadtree can easily be applied on the Grid-Tree[3].

In order to support efficient geo-textual object search, the SSG-Trees clusters a set of geo-textual objects into a series of continual signature Grid-Tree, which cluster the objects into disjointed subsets of nodes and abstracts them in various granularities. By doing so, it capacitates the pruning of those (textually, spatially or temporally) irrelevant subsets or trees. The efficiency of SSG-Trees depends on its pruning power is highly related to the effectiveness of object clustering and the search algorithms. Our SSG-Trees clusters spatially and temporally close objects together and carries textual information in its node signatures.

SSG-Trees is essentially a set of Signature Grid-Trees, each node of which is enriched with the reference to a signature file for the objects contained in the sub-tree rooted at the node. In particular, each node of an SSG-Trees contains all spatial, temporal, and keyword information; the first is in the form of a rectangle, the second is in the form of timestamp, and the last is in the form of a signature.

More formally, the leaf node of SSG-Trees has the form ($nSig$, r, t, $oSig$). $oSig$ refers to a set of signatures created by the objects of current node, $nSig$ is the **OR-ing** of all signature in $gSig$, r is the area covered by current node, and t is the latest timestamp aggregated from the objects. An inner node has the form ($nSig$, r, t, cp). cp are the address of the children nodes, $nSig$ is the **OR-ing** of all the signatures of its children, r is the area covered by current node, and t is the latest timestamp aggregated from its children nodes. To simplify the following presentation we degrade the Grid-Tree to its special case Quadtree in the example.

4.3 Processing of TSK queries

We proceed to present an important metric, the minimum spatial-temporal distance $MIND_{st}$, which will be used in the query processing. Given a query q and a node N in the SSG-Trees, the metric $MIND_{st}$ offers a lower bound on the actual spatial-temporal distance between query q and the objects enclosed in the rectangle of node N. This bound can be used to order and efficiently prune the paths of the search space in the SSG-Trees.

Definition 6 ($MIND_{st}(q, N)$)**.** *The distance of a query point q from a node N in the SSG-Trees, denoted as $MIND_{st}(q, N)$, is defined as follows:*

$$MIND_{st}(q, N) = \alpha * \frac{MIND_s(q.loc, N.r)}{\delta_{max}} + (1 - \alpha) * \frac{MIND_t(q.t, N.t)}{\lambda_{max}} \tag{2}$$

where α, δ_{max}, and λ_{max} are the same as in Equation 1; $MIND_s(q.loc, N.r)$ is the minimum Euclidian distance between $q.loc$ and $N.r$, $MIND_t(q.t, N.t)$ is the minimum time difference between $q.t$ and $N.t$.

A salient feature of the proposed SSG-Trees structure is that it inherits the nice properties of the Quadtree for query processing.

[3] Obviously, if n equals two, the Grid-Tree degrades to Quadtree

Algorithm 1 TSK Search(q, k, $\mathcal{I}$)

Input: q : the spatial-keyword temporal query, k : the number of object return, $\mathcal{I}$: current SSG-Trees index
Output: $\mathcal{R}$: top-k query result results

```
1: R := ∅; H = ∅, λ_max = ∞
2: H ← new a min first heap
3: build frequency signature for query
4: H.Enqueue(I.root, MIND_st(q, I.root))
5: while H ≠ ∅ do
6:    e ← the node popped from H
7:    if e is a leaf node then
8:       for each object o in node e do
9:          if o passed the signature test AND f_st(q, o) ≤ λ_max then
10:            λ_max ← f_st(q, o)
11:            update R by (o, f_st(q, o))
12:         end if
13:      end for
14:   else
15:      for each child e' in node e do
16:         if e' passed the signature test AND MIND_st(q, e') ≤ λ_max then
17:            H.Enqueue(e', MIND_st(q, e'))
18:         end if
19:      end for
20:   end if
21:   process the root node of next SSG-Trees
22: end while
23: return R
```

Theorem 1. *Given a query point q, a node N, and a set of objects $\mathcal{O}$ in node N, for any $o \in \mathcal{O}$, we have $f_{st}(q, N) \leq DIST_{st}(q, o)$.*

Proof. Since object o is enclosed in the rectangle of node N, the minimum Euclidian distance between $q.loc$ and $N.r$ is no larger than the Euclidian distance between $q.loc$ and $o.loc$:

$$MISD_S(q.loc, N.r) \leq f_s(q.loc, o.loc)$$

For each timestamp t, $N.t$ is the maximum value $\mathcal{O}.t$ of all the object in node N. Hence:

$$MISD_t(q.loc, N.r) \leq f_t(q.loc, o.loc)$$

According to Equation 1 and Equation 2, we obtain:

$$MIND_{st}(q, N) \leq f_{st}(q, o)$$

thus completing the proof.

When searching the SSG-Trees for the k objects nearest to a query q, one must decide at each visited node of the SSG-Trees which entry to search first. Metric $MIND_{ST}$ offers an approximation of the spatial-temporal ranking score to every entry in the node and, therefore, can be used to direct the search. Note that only node satisfied the constraint of query keywords need to be loaded into memory and compute $MIND_{ST}$.

To process **TSK** queries with SSG-Trees framework, we exploit the best-first traversal algorithm for retrieving the top-k objects. With the best-first traversal algorithm, a priority queue is used to keep track of the nodes and objects that have yet to be visited. The values of f_{st} and $MIND_{st}$ are used as the keys of objects and nodes, respectively.

When deciding which node to visit next, the algorithm picks the node N with the smallest $MIND_{st}(q, N)$ value in the set of all nodes that have yet to be visited. The algorithm terminates when k nearest objects (ranked according to Equation 1) have been found.

Algorithm 1 illustrates the details of the SSG-Trees based **TSK** query. A minimum heap $\mathcal{H}$ is employed to keep the Grid-Tree's nodes where the key of a node is its minimal spatial-temporal ranking score. For

the input query, we calculate its frequency signature in Line 3. In Line 4, we find out the root node of current time interval, calculate the minimal spatial-temporal ranking score for the root node, and then pushed the root node into the $\mathcal{H}$. The the algorithm executes the while loop (Line 5-21)until the top-k results are ultimately reported in Line 23.

In each iteration, the top entry e with minimum spatial-temporal ranking score is popped from $\mathcal{H}$. When the popped node e is a leaf node(Line 7), for each signature in node e, we will iterator extract the objects that satisfy query constraint and check whether its spatial-temporal ranking score is less than λ_{max}. If its score is not larger than λ_{max}, we push o into result set and add update λ_{max}. When the popped node e is a non-leaf node(Line 15), a child node $e^{'}$ of e will be pushed to $\mathcal{H}$ if it can pass the query signature test and the minimal spatial-temporal ranking score between $e^{'}$ and q, denoted by $MIND_{st}(q, e^{'}$, is not larger than λ_{max} (Line 15- 17). We process the root node of next interval in Line 21. The algorithm terminates when $\mathcal{H}$ is empty and the results are kept in $\mathcal{R}$.

5 Experiments

In this section, we present the results of a comprehensive performance study to evaluate the effectiveness and efficiency of our techniques proposed in this section.

5.1 Baseline Algorithms

To the best of our knowledge, no existing work investigating top-k queries on spatial-keyword temporal data. Hence, for comprehensive performance evaluation, we discuss how to exploit existing techniques in [9] for processing **TSK** queries. And we develop two baselines by utilizing existing index structures, namely IFQ and SIFQ.

I) Inverted File plus Quadtree (**IFQ**). IFQ first employs Quadtree to partition objects into leaf cells according to their location information. Then, the objects inside each cell are stored in a reversed chronological order. Finally, for the objects in each leaf cell, we build inverted file for keyword filtering proposed and recursively construct the inverted file for its ancestral cells.

II) Segment Based Inverted File plus Quadtree (**SIFQ**). SIFQ is an enhanced version of IFQ. Similarly, it employs Quadtree to partition the objects into different cell, uses reversed chronological order to organize the object list in leaf cell, and build the inverted file for all the cells which contain objects. The major difference between them is that IFQ organizes all the objects in an single quadtrees, but SIFQ partitions all the incoming objects into a set of quadtrees or segments by time unit.

5.2 Experiment Setup

In this section, we implement and evaluate following algorithms.

- **IFQ.** IFQ based **TSK** algorithm proposed in baseline algorithm.
- **SIFQ.** Enhanced IFQ by partitioning the objects into a set of time unit in baseline algorithm.
- **SSG.** SSG-Trees based **TSK** algorithm proposed in Section 4.

Dataset. All experiments are based on a real-life dataset **TWEETS**. TWEETS is a real-life dataset collected from Twitter [5], containing 13 million geo-textual tweets from May 2012 to August 2012. The statistics of **TWEETS** are summarized in Table 2.

Property	# of objects	vocabulary	avg. # of term per obj.
TWEETS	$13.3M$	$6.89M$	10.05

Table 2: Dataset Details

Workload. The workload for the **TSK** query consists of 1000 queries, and the average query response time are employed to evaluate the performance of the algorithms. The query locations are randomly selected from the underlying dataset. On the other hand, the likelihood of a keyword t being chosen as query keyword is $\frac{freq(t)}{\sum_{t_i \in \mathcal{V}} freq(t_i)}$ where $freq(t)$ is the term frequency of t in the dataset. The number of

query keywords (l) varies from 1 to 5, the number of results (k) grows from 10 to 50, and the preference parameter α changes from 0.1 to 0.9. By default, l, k, and α are set to **3**, **10**, and **0.5** respectively. In addition, unless mentioned otherwise, the default value of cell capacity c is 100, and geo-textual object arrival rate τ is 4000. We random select 10% of the geo-textual objects from **TWEETS** as the historical object workload when SSG are constructed. We always keep the most recent 5M objects in.

All experiments are implemented in C++. The experiments are conducted on a PC with 2.9GHz Intel Xeon 2 cores CPU and 32GB memory running Red Hat Enterprise Linux. For a fair comparison, we tune the important parameters of the competitor algorithms for their best performance. Particularly, the leaf capacity of all algorithms is set to 100. The partition threshold of SIFQ is set to 400000. Our measures of performance include insertion time, deletion time, storage overhead, and response time. The rest of this section evaluates index maintenance (Section 5.3), and query processing (Section **??**).

5.3 Index Maintenance

In this subsection, we evaluate the insertion time, deletion time, storage overhead of all the algorithms[4]. Since all the objects are indexed in one single quadtree in IFQ, the insertion and deletion time are longer than the other algorithms. What's worse, the time to process 2000 objects for IFQ is large than one second. Thus, we ignore IFQ's insertion time and deletion time in comparison.

Evaluation on storage overhead. gives the performance when varying the number of objects from 5M to 13M. As the number of objects increase, the storage overhead is stable for all algorithms, owing to we only keep the most recent 5M objects in memory. The performance of signature based algorithms is always better than that of inverted index based algorithms. The storage overhead of signature based algorithms is at most one half of that of inverted index based algorithms. gives the same experiment with varying node capacity from 100 to 500. The storage overhead of algorithms meets a slight decrease, due to more information has been shared by the textual index.

6 Conclusions

To the best of our knowledge, this is the first work to study the problem of top-k continuous temporal spatial keyword queries over streaming temporal spatial keyword microblogs, which has a wide spectrum of application. To tackle with this problem, we propose a novel temporal spatial keyword partition indexing structure, namely SSG-Trees, efficiently organize a massive number of streaming temporal spatial keyword microblogs such that each incoming query submitted by users can rapidly find out the top-k results. Extensive experiments demonstrate that our technique achieves a high throughput performance over streaming temporal spatial keyword data.

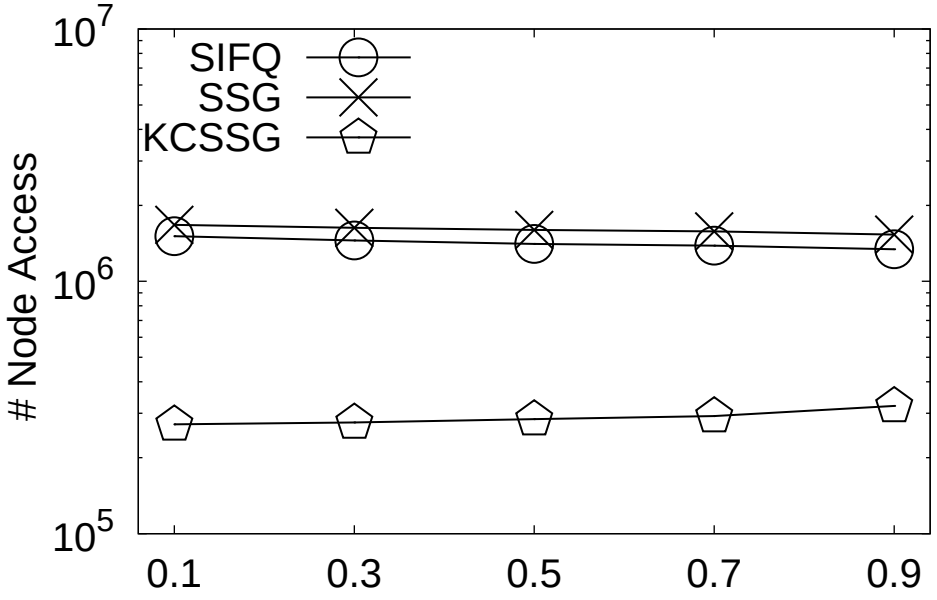
Acknowledgments: This work was supported in part by the National Natural Science Foundation of China (61272150, 61379110, 61472450, 61402165, 61702560, S1651002, M1450004), the Key Research Program of Hunan Province(2016JC2018), and project (2016JC2011, 2018JJ3691) of Science and Technology Plan of Hunan Province.

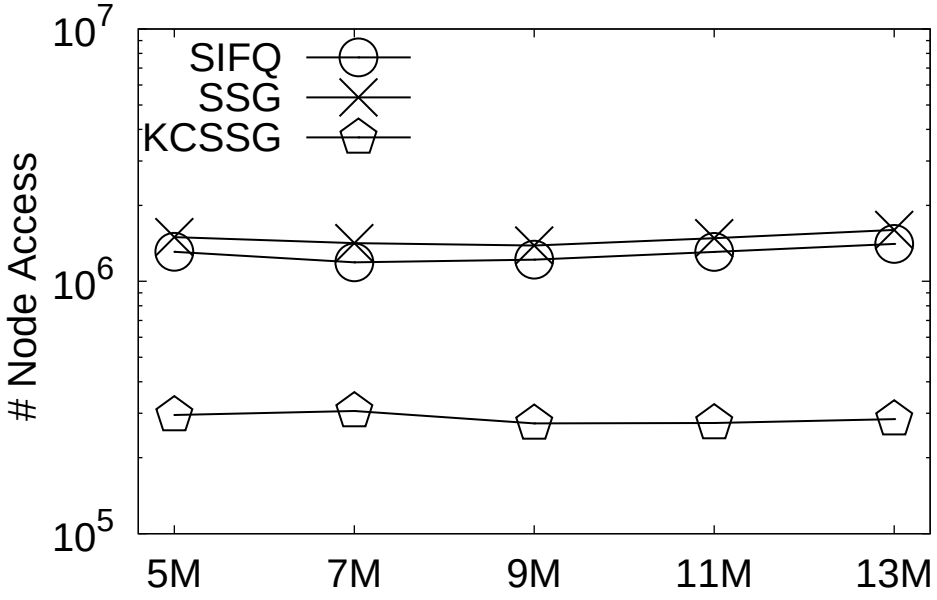
References

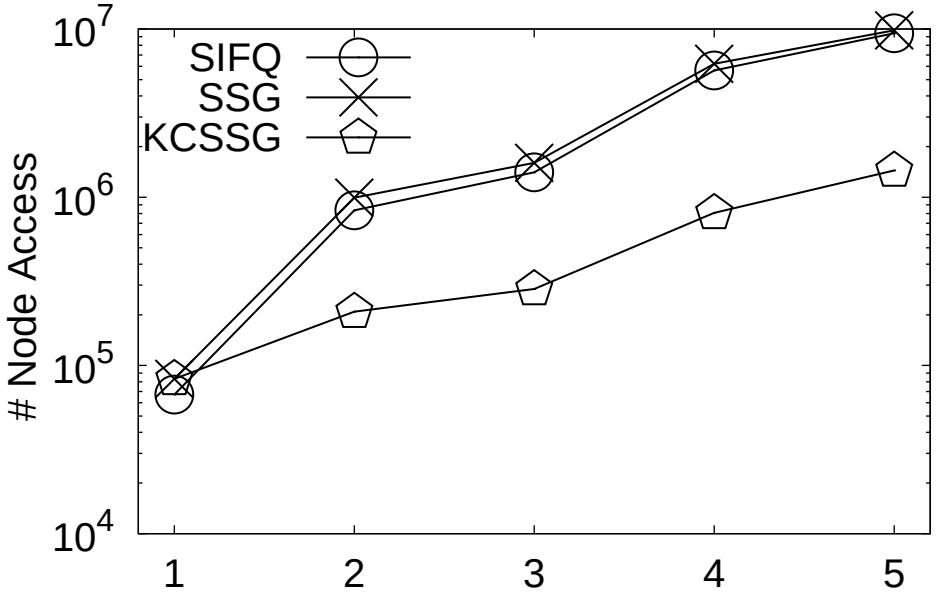
1. Aref, W.G., Samet, H.: Efficient processing of window queries in the pyramid data structure. In: Proceedings of the Ninth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, April 2-4, 1990, Nashville, Tennessee, USA, pp. 265–272 (1990)
2. Christodoulakis, S., Faloutsos, C.: Design considerations for a message file server. IEEE Trans. on Software Engineering **10**(2), 201–210 (1984)
3. Deppisch, U.: S-tree: A dynamic balanced signature index for office retrieval. In: SIGIR, pp. 77–87 (1986)
4. Faloutsos, C., Jagadish, H.V.: Hybrid index organizations for text databases. In: Advances in Database Technology - EDBT'92, 3rd International Conference on Extending Database Technology, Vienna, Austria, March 23-27, 1992, Proceedings, pp. 310–327 (1992)
5. Felipe, I.D., Hristidis, V., Rishe, N.: Keyword search on spatial databases. In: ICDE 2008
6. Gargantini, I.: An effective way to represent quadtrees. Commun. ACM **25**(12), 905–910 (1982)

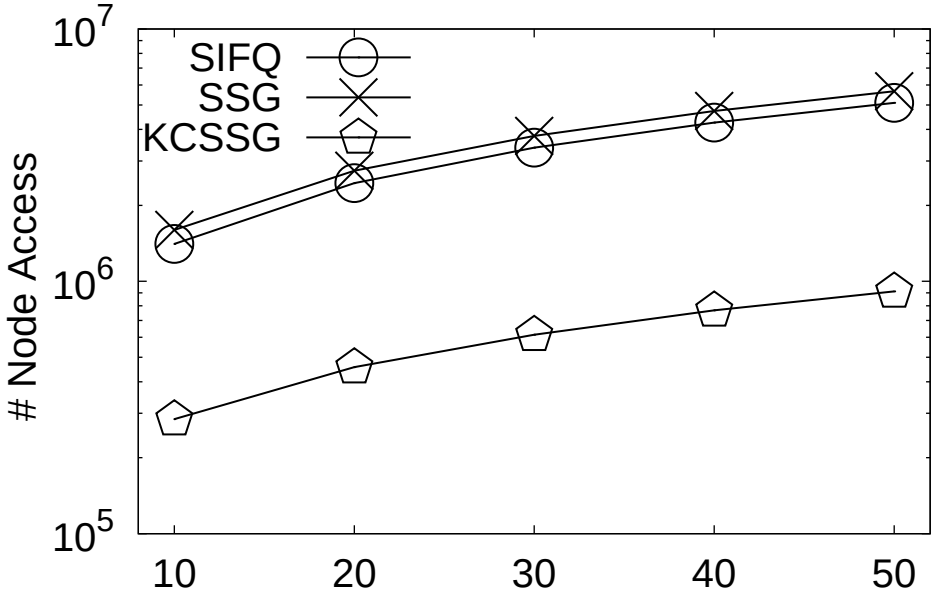
[4]We ignore IFQ's insertion time and deletion time in comparison, due to it cannot meet the current arrive rate.

7. Lee, D.L., Kim, Y.M., Patel, G.: Efficient signature file methods for text retrieval. IEEE Trans. Knowl. Data Eng. **7**(3), 423–435 (1995)
8. Magdy, A., Alarabi, L., Al-Harthi, S., Musleh, M., Ghanem, T.M., Ghani, S., Mokbe, M.F.: Taghreed: A system for querying, analyzing, and visualizing geotagged microblogs. In: SIGSPATIAL (2014)
9. Magdy, A., Mokbel, M.F., Elnikety, S., Nath, S., He, Y.: Mercury: A memory-constrained spatio-temporal real-time search on microblogs. In: IEEE 30th International Conference on Data Engineering, Chicago, ICDE 2014, IL, USA, March 31 - April 4, 2014, pp. 172–183 (2014)
10. Mouratidis, K., Hadjieleftheriou, M., Papadias, D.: Conceptual partitioning: An efficient method for continuous nearest neighbor monitoring. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, Baltimore, Maryland, USA, June 14-16, 2005, pp. 634–645 (2005)
11. O'Neil, P.E., Cheng, E., Gawlick, D., O'Neil, E.J.: The log-structured merge-tree (lsm-tree). Acta Inf. **33**(4), 351–385 (1996)
12. Wang, Y., Lin, X., Wu, L., Zhang, W.: Effective multi-query expansions: Collaborative deep networks for robust landmark retrieval. IEEE Trans. Image Processing **26**(3), 1393–1404 (2017)
13. Wang, Y., Lin, X., Wu, L., Zhang, W., Zhang, Q.: Exploiting correlation consensus: Towards subspace clustering for multi-modal data. In: Proceedings of the ACM International Conference on Multimedia, MM '14, Orlando, FL, USA, November 03 - 07, 2014, pp. 981–984 (2014)
14. Wang, Y., Lin, X., Wu, L., Zhang, W., Zhang, Q.: LBMCH: learning bridging mapping for cross-modal hashing. In: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, Santiago, Chile, August 9-13, 2015, pp. 999–1002 (2015)
15. Wang, Y., Lin, X., Wu, L., Zhang, W., Zhang, Q., Huang, X.: Robust subspace clustering for multi-view data by exploiting correlation consensus. IEEE Trans. Image Processing **24**(11), 3939–3949 (2015)
16. Wang, Y., Lin, X., Zhang, Q.: Towards metric fusion on multi-view data: a cross-view based graph random walk approach. In: 22nd ACM International Conference on Information and Knowledge Management, CIKM'13, San Francisco, CA, USA, October 27 - November 1, 2013, pp. 805–810 (2013)
17. Wang, Y., Lin, X., Zhang, Q., Wu, L.: Shifting hypergraphs by probabilistic voting. In: Advances in Knowledge Discovery and Data Mining - 18th Pacific-Asia Conference, PAKDD 2014, Tainan, Taiwan, May 13-16, 2014. Proceedings, Part II, pp. 234–246 (2014)
18. Wang, Y., Wu, L.: Beyond low-rank representations: Orthogonal clustering basis reconstruction with optimized graph structure for multi-view spectral clustering. Neural Networks (2018)
19. Wang, Y., Wu, L., Lin, X., Gao, J.: Multiview spectral clustering via structured low-rank matrix factorization. IEEE Trans. Neural Networks and Learning Systems (2018)
20. Wang, Y., Zhang, W., Wu, L., Lin, X., Fang, M., Pan, S.: Iterative views agreement: An iterative low-rank based structured optimization method to multi-view spectral clustering. In: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016, pp. 2153–2159 (2016)
21. Wang, Y., Zhang, W., Wu, L., Lin, X., Zhao, X.: Unsupervised metric fusion over multiview data by graph random walk-based cross-view diffusion. IEEE Trans. Neural Netw. Learning Syst. **28**(1), 57–70 (2017)
22. Wu, L., Wang, Y., Gao, J., Li, X.: Deep adaptive feature embedding with local sample distributions for person re-identification. Pattern Recognition **73**, 275–288 (2018)
23. Wu, L., Wang, Y., Ge, Z., Hu, Q., Li, X.: Structured deep hashing with convolutional neural networks for fast person re-identification. Computer Vision and Image Understanding **167**, 63–73 (2018)
24. Wu, L., Wang, Y., Li, X., Gao, J.: Deep attention-based spatially recursive networks for fine-grained visual recognition. IEEE Trans. Cybernetics (2018)
25. Wu, L., Wang, Y., Li, X., Gao, J.: What-and-where to match: Deep spatially multiplicative integration networks for person re-identification. Pattern Recognition **76**, 727–738 (2018)
26. Zhang, C., Zhang, Y., Zhang, W., Lin, X.: Inverted linear quadtree: Efficient top k spatial keyword search. In: 29th IEEE International Conference on Data Engineering, ICDE 2013, Brisbane, Australia, April 8-12, 2013, pp. 901–912 (2013)
27. Zhang, C., Zhang, Y., Zhang, W., Lin, X.: Inverted linear quadtree: Efficient top K spatial keyword search. IEEE Trans. Knowl. Data Eng. **28**(7), 1706–1721 (2016)
28. Zhang, C., Zhang, Y., Zhang, W., Lin, X., Cheema, M.A., Wang, X.: Diversified spatial keyword search on road networks. In: Proceedings of the 17th International Conference on Extending Database Technology, EDBT 2014, Athens, Greece, March 24-28, 2014., pp. 367–378 (2014)


SIFQ
SSG
KCSSG
Node Access
10^7
10^6
10^5
0.1
0.3
0.5
0.7
0.9


10^7
10^6
10^5
Node Access
SIFQ
SSG
KCSSG
5M
7M
9M
11M
13M


SIFQ
SSG
KCSSG
Node Access
10^7
10^6
10^5
10^4
1
2
3
4
5


10^7
10^6
10^5
Node Access
SIFQ
SSG
KCSSG
10
20
30
40
50