Efficient Group Top-k Spatial Keyword Query Processing

Kai Yao[1], Jianjun Li[1(✉)], Guohui Li[1], and Changyin Luo[2]

[1] School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China
{kaiyao,jianjunli,guohuili}@hust.edu.cn
[2] School of Computer, Central China Normal University, Wuhan, China
luochangyin@hust.edu.cn

Abstract. With the proliferation of geo-positioning and geo-tagging, spatial web objects that possess both a geographical location and textual description are gaining in prevalence. Given a spatial location and a set of keywords, a top-k spatial keyword query returns the k best spatio-textual objects ranked according to their proximity to the query location and relevance to the query keywords. To our knowledge, existing study on spatial keyword query processing only focuses on single query point scenario. In this paper, we take the first step to study the problem of multiple query points (or group queries) top-k spatial keyword query processing. We first propose a threshold-based algorithm, which first performs incremental top-k spatial keyword query for each query point and then combines their results. Next, we propose another more efficient algorithm by treating the whole query set as a query unit, which can significantly reduce the objects to be examined, and thus achieve higher performance. Extensive experiments using real datasets demonstrate that our approaches are efficient in terms of runtime and I/O cost, as compared to the baseline algorithm.

Keywords: Spatial keyword query · Top-k query · Spatial databases

1 Introduction

In modern applications such as Google Maps, Points of Interest (PoI) are typically augmented with textual descriptions. This development gives prominence to spatial keyword queries [1–4]. A typical such query takes a location and a set of keywords as arguments and returns relevant content that matches the arguments. To our knowledge, most of the existing proposals for answering spatial keyword queries are only focused on single query point scenario. However, in some cases,

The work is partially supported by the State Key Program of National Natural Science of China under Grant No. 61332001, National Natural Science Foundation of China under Grants Nos. 61572215, 61300045, China Postdoctoral Science Special Foundation under Grant No. 2015T80802, and the Fundamental Research Funds for the Central Universities, HUST-2016YXMS076.

F. Li et al. (Eds.): APWeb 2016, Part I, LNCS 9931, pp. 153–165, 2016.
DOI: 10.1007/978-3-319-45814-4_13

there may be multiple queries. For example, two friends at different locations want to find a restaurant to have dinner together, and they may have different preferences (e.g., one might like restaurant that offers "pizza" and "dessert" in a "cozy" environment, the other might prefer restaurant that offers "pizza" and "coffee" and has "friendly" service). As another example, consider a group of tourists who visit a city and stay in different hotels. These people may want to visit an attraction of the city, which is not far from their locations and at the same time is relevant to their different preferences (e.g., some tourists would like to visit "museum" and "library", while others want to visit "art gallery" and "theater").

In this paper, we take the first step to study the group location-aware top-k text retrieval (GLkT) query, which takes into account multiple query points. This query enables multiple mobile users (or drivers) to get the k spatio-textual objects that best match their queries with respect to location and text relevancy.

Note that each query point may have different locations and keywords. It can be widely utilized in various decision support systems and multiple domains like service recommendation, investment planning, etc. The most relevant study on this problem is IR-tree [1] which embeds an inverted index in each node of R-tree [5]. The inverted index at each node refers to a pseudo-document that represents all the objects under the node. Therefore, in order to verify if a node is relevant for a set of query keywords, the current approaches access the inverted index at each node to evaluate the similarity between the query keywords and the pseudo-document associated with the node. However, they only consider single query point and their methods are not suitable for GLkT query. Most traditional spatial queries on spatial database such as group nearest neighbor queries [6] and aggregate nearest neighbor queries [7], which find the k objects with the minimum total distance to a group of query points. However, both group NN and aggregate NN queries do not concern non-spatial information (e.g., name, description, and type etc.). To address this limitations and improve the performance, we first propose a threshold-based algorithm, which first performs incremental top-k spatial keyword query for each query point and then combines their results. To achieve higher performance, we propose another more efficient algorithm by treating the whole query set as a query unit, which can significantly reduce the objects to be examined.

The main contributions of this paper can be summarized as follows:

- We propose a threshold algorithm, which first performs incremental top-k spatial keyword query for each query point and then combines their results, to address the GLkT query processing problem.
- By treating the whole query set as a query unit, we propose a more efficient algorithm namely UA. We also propose a heuristic to prune the search space and reduce the number of node accesses significantly.
- Extensive experiments using real datasets demonstrate that our approaches are efficient in terms of both runtime and I/O cost, as compared to the baseline algorithm.

The remainder of the paper is organized as follows. Section 2 gives a formal problem definition. Section 3 presents a threshold algorithm and a Unit-based

algorithm for GLkT query. Experimental results are provided in Sect. 4. We review related work in Sect. 5 and make a conclusion in Sect. 6.

2 Preliminaries

Problem Statement. Let $D = \{p_1, p_2, ..., p_n\}$ be a dataset of spatio-textual objects. Each object $p \in D$ includes a spatial location $p.l$ and textual description $p.d$, denoted by $p = \{p.l, p.d\}$. Let $Q = \{q_1, q_2, ..., q_m\}$ be a collection of query points. Each $q \in Q$ is represented by $q = \{q.l, q.d, k, \alpha\}$, where $q.l$ is the query location, $q.d$ is the set of query keywords and parameter $\alpha \in (0, 1)$ is used to balance between the spatial and textual components. Given a query set Q, our goal is to return k spatio-textual objects $\{r_1, r_2, ..., r_k\}$ from D with the highest relevance score $T(r_1, Q) \geq T(r_2, Q) \geq ... \geq T(r_k, Q)$. We use a linear interpolation function to compute the spatio-textual relevance score. This paper's proposals are applicable to a wide range of ranking functions, namely all functions that are monotone with respect to spatial proximity and text relevancy.

Definition 1 ***(Spatial Proximity).*** *Given an object p and a query q, the spatial proximity is defined in the following equation:*

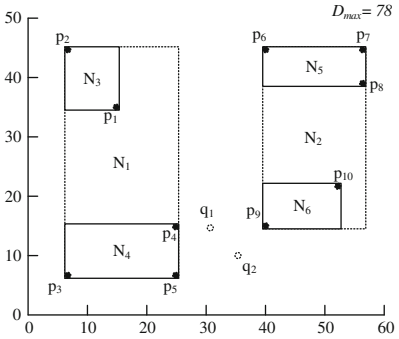
$$\delta(p, q) = 1 - \frac{D_\varepsilon(p.l, q.l)}{D_{\max}} \tag{1}$$

where $D_\varepsilon(p.l, q.l)$ is the Euclidean distance between $p.l$ and $q.l$, and $D_{\max}$ is the maximum distance in the location space. The maximum distance may be obtained by getting the largest diagonal of the Euclidean space of the application.

Definition 2 ***(Textual Relevance).*** *Given an object p and a query q, the textual relevance id defined in the following equation:*

$$\theta(p, q) = \frac{\sum_{t \in q.d} \omega_{t,p.d} \cdot \omega_{t,q.d}}{\sqrt{\sum_{t \in p.d} (\omega_{t,p.d})^2 \cdot \sum_{t \in q.d} (\omega_{t,q.d})^2}} \tag{2}$$

There are several similarity measures that can be used to evaluate the textual relevance between the query keywords $q.d$ and the text description $p.d$ [8]. In order to compute the cosine, we adopt the approach employed by Zobel and Moffat [9]. Therefore, the weight $\omega_{t,p.d}$ is computed as $\omega_{t,p.d} = 1 + \ln(f_{t,p.d})$, where $f_{t,p.d}$ is the number of occurrences (frequency) of t in $p.d$; and the weight $\omega_{t,q.d}$ is obtained from the following formula $\omega_{t,q.d} = \ln(1 + \frac{|P|}{df_t})$, where $|P|$ is the total number of documents in the collection. The document frequency df_t of a term t gives the number of documents in P that contains t. In this paper, we adopt the well-known cosine similarity between the vectors composed by the weights of the terms in $q.d$ and $p.d$. The textual relevance is a value within the range $[0, 1]$ (property of cosine).



(a) Location

P_1:{(t_2,2) , (t_4,3)}
P_2:{(t_3,2)}
P_3:{(t_2,1) , (t_4,1)}
P_4:{(t_1,2) , (t_2,4)}
P_5:{(t_2,1) , (t_6,2)}
P_6:{(t_3,1)}
P_7:{(t_1,1) , (t_5,2)}
P_8:{(t_2,2)}
P_9:{(t_1,2) , (t_2,2) , (t_6,3)}
P_{10}:{(t_3,1)}

(b) Description

t_1: pizza
t_2: dessert
t_3: coffee
t_4: delicious
t_5: cozy
t_6: friendly

(c) Tokens

Fig. 1. Group Top-k spatial keyword query

Definition 3 ***(Spatio-textual Relevance Score).*** *Given an object p and a query q, the spatio-textual relevance score between p and q is defined as:*

$$\tau(p,q) = \alpha \cdot \delta(p,q) + (1-\alpha) \cdot \theta(p,q) \tag{3}$$

where $\alpha \in (0,1)$ is a parameter used to balance between the spatial and textual components. For example, $\alpha = 0.5$ means that spatial proximity and textual relevance are equally important.

Definition 4 ***(Group Spatio-textual Relevance Score).*** *Given an object p and a query set $Q = \{q_1, q_2, ..., q_m\}$, the group spatio-textual relevance score between p and Q is defined as:*

$$T(p,Q) = \sum\nolimits_{q \in Q} \tau(p,q) \tag{4}$$

Example 1. Suppose in a geographic search engine there is a set $D = \{p_1, ..., p_{10}\}$, each of which is associated with one spatial location and a set of keywords. Figure 1(a) and (b) plot the spatial location of the objects and the frequencies of keyword, respectively.

Assume that two friends q_1 and q_2 want to find a restaurant to have dinner together, and q_1 prefer restaurant that offers "pizza" and "dessert" in a "cozy" environment, and q_2 prefer restaurant that offers "pizza" and "coffee" and has "friendly" service. Therefore, the query set Q is consist of $q_1 = \{[30,15], t_1, t_2, t_5, 1, 0.5\}$ and $q_2 = \{[35,10], t_1, t_3, t_6, 1, 0.5\}$. For object p_4 where $p_4.l = (25,15)$, its spatial proximity between q_1 and q_2 is $\delta(p_4,q_1) = 0.94$ and $\delta(p_4,q_2) = 0.86$ respectively, its textual relevance between q_1 and q_2 is $\theta(p_4,q_1) = 0.55$ and $\theta(p_4,q_2) = 0.31$ respectively. Thus, the spatio-textual relevance score $\tau(p_4,q_1) = 0.5 * 0.94 + (1-0.5) * 0.55 = 0.75$ and $\tau(p_4,q_2) = 0.5 * 0.86 + (1-0.5) * 0.31 = 0.58$. According to Definition 4, the group spatio-textual relevance score $T(p_4,Q) = 0.75 + 0.58 = 1.33$. Similarly, for object p_9,

$T(p_9, Q) = 0.71 + 0.81 = 1.52$. Thus, p_9 is better than p_4 since $1.52 > 1.33$. By calculating the relevance score of all the objects, we take the one object with highest score $\{p_9/1.52\}$ as the result.

3 GLkT Query Algorithm

In this section, we present efficient algorithms for processing GLkT query with the assumption that spatio-textual objects are organized by IR-Tree. Section 3.1 gives a brief description of the baseline method. Section 3.2 gives the threshold algorithm and the Unit-based algorithm will be introduced in Sect. 3.3.

3.1 Baseline Algorithm

The idea of the baseline algorithm is as follows. For each query $q_i \in Q$, we compute the ranked list of all objects according to the ranking function τ respectively. Then, we combine them to gain the comprehensive relevance score, thus we can use classical selecting algorithm to get the k most relevant spatio-textual objects from dataset D.

Though this method can solve our problem, it is rather inefficient and may generate large amounts of candidates. For each $q_i \in Q$, it requires retrieving all the spatio-textual objects and computing relevance score (high I/O cost), without any pruning.

3.2 Threshold Algorithm

In this section, we propose a threshold algorithm (TA) to efficiently find top-k most relevant objects from dataset D. The basic idea is that, it first performs incremental top-k spatial keyword query for each $q_i \in Q$ and then combines their results. As shown in Fig. 1, the query set Q is consist of $q_1 = \{[30, 20], t_1, t_2, t_5, 1, 0.5\}$ and $q_2 = \{[35, 10], t_1, t_3, t_6, 1, 0.5\}$, TA retrieves the most relevant object of q_1 (object p_4 with $\tau(p_4, q_1) = 0.75$) and computes the spatio-textual relevance about q_2 and p_4 ($\tau(p_4, q_2) = 0.58$). Similarly, TA finds the most relevant object of q_2 (object p_9 with $\tau(p_9, q_2) = 0.81$) and computes the spatio-textual relevance about q_1 and p_9 ($\tau(p_9, q_1) = 0.71$). Thus, object p_9 with the larger group spatio-textual relevance score than p_4 ($1.52 > 1.33$) becomes the temporary result.

For each query q_i, TA stores a threshold λ_i, which is the score of the currently most relevant object, i.e., $\lambda_1 = 0.75$ and $\lambda_2 = 0.81$. The total threshold Λ is defined as the sum of each λ_i, i.e., $\Lambda = \lambda_1 + \lambda_2 = 1.56$. Continuing our algorithm, since $\Lambda > \tau(p_9, Q)$, it is possible that there exists an object in D whose relevance score is larger than p_9. So we continue to retrieve the second most relevant object of q_1 and q_2. As shown in Fig. 2, the second most relevant object of q_1 and q_2 is p_9 and p_5, respectively. Then we update $\lambda_1 = 0.71$, $\lambda_2 = 0.72$ and $\Lambda = 1.43$. Continuing TA, since $\Lambda < \tau(p_9, Q)$, it indicates that there exists no object in D

Object	$\tau(p, q_1)$
p_4	0.75
p_9	0.71
p_5	0.51
...	...

(a) Rank list of q_1

Object	$\tau(p, q_2)$
p_9	0.81
p_5	0.72
p_4	0.58
...	...

(b) Rank list of q_2

Object	$T(p, Q)$
p_9	1.52
p_4	1.33
p_5	1.23
...	...

(c) Rank list of Q

Fig. 2. Example of threshold algorithm

whose relevance score is larger than p_9, so we terminate algorithm and return p_9 as the ultimate result.

Algorithm 1 shows the pseudo code for TA, where C is the candidate list of the k most relevant objects found so far, and $best\_k$ is the relevance score of the k-th most relevant object in C. Note that if C contains fewer than k members, $best\_k$ is set to zero. When an object is inserted into C, an existing member is replaced if the list already contains k members.

Algorithm 1. TA(Query Q, Index $IR\text{-}tree$)

```
1  begin
2      Λ = ∞; best_k = 0; C = ∅;
3      while Λ > best_k do
4          foreach q_i in Q do
5              get the next relevant object p_j of q_i;
6              λ_i = τ(p_j, q_i);
7              update Λ;
8              compute the relevance score between p_j and Q : T(p_j, Q);
9              if T(p_j, Q) > best_k then
10                 Add p_j to C;
11                 if |C| ≥ k then
12                     update best_k;
13     return C;
```

3.3 Unit-Based Alogorithm

TA may cause multiple accesses to the same node of the index structure(i.e., IR-Tree) due to multiple query points. To avoid this problem, we propose Unit-based algorithm (UA) which could process GLkT query by a single traversal. The main idea is to treat the whole query set as a query unit, then we could use it to prune the search space. Specifically, starting from the root of the IR-tree for dataset D, UA visits only nodes that may contain candidate objects. In the following, we will use symbol $\mathcal{U}$ to represent the query unit. Firstly, we give the

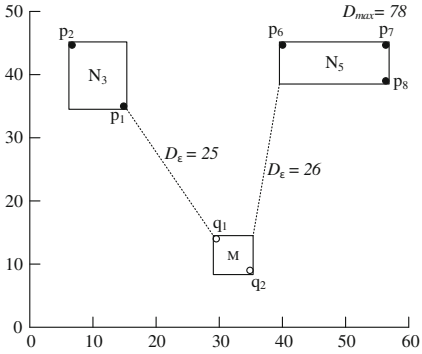


Fig. 3. Example of Definition 5

definitions of spatial proximity and textual relevance between object p and $\mathcal{U}$ (i.e., $\delta_u(p,\mathcal{U})$ and $\theta_u(p,\mathcal{U})$).

Definition 5 ($\delta_u(p,\mathcal{U})$)**.** *Given an object p and a query set Q. The spatial proximity between p and $\mathcal{U}$ is defined in the following equation:*

$$\delta_u(p,\mathcal{U}) = 1 - \frac{D_\varepsilon(p.l, MBR.l)}{D_{\max}} \tag{5}$$

where MBR is the minimum bounding rectangle of Q, and D_{max} is the maximum distance in the location space.

According to the characters of minimum bounding rectangle, we have $D_\varepsilon(p.l, MBR.l) \leq D_\varepsilon(p.l, q.l), q \in Q$. Therefore, we have $\delta_u(p,\mathcal{U}) \geq \delta(p,q), q \in Q$. Note that p could be an intermediary-node N of IR-tree. As shown in Fig. 3, Q is consist of q_1 and q_2, N_3 and N_5 are intermediary-nodes of IR-tree, we have $D_\varepsilon(N_3.l, MBR.l) = 25$ and $D_\varepsilon(N_5.l, MBR.l) = 26$.

Definition 6 ($\theta_u(p,\mathcal{U})$)**.** *Given an object p and a query set Q. The textual relevance between p and $\mathcal{U}$ is defined in the following equation:*

$$\theta_u(p,\mathcal{U}) = MAX\{\theta(p,q)\}, q \in Q \tag{6}$$

That is, $\theta_u(p,\mathcal{U})$ is the largest textual relevance score of p with q that belongs to Q. Therefore, it is easy to see that $\theta_u(p,\mathcal{U}) \geq \theta(p,q), q \in Q$.

Definition 7 ($\tau_u(p,\mathcal{U})$)**.** *Given an object p and a query set Q. The spatio-textual relevance score between p and $\mathcal{U}$ is defined as:*

$$\tau_u(p,\mathcal{U}) = \alpha \cdot \delta_u(p,\mathcal{U}) + (1-\alpha) \cdot \theta_u(p,\mathcal{U}) \tag{7}$$

Theorem 1. *Given an object p and a query set Q, the following is true:*

$$\tau_u(p,\mathcal{U}) \geq \tau(p,q), \forall q \in Q$$

Proof. Since the query q is enclosed in the MBR of Q, the minimum Euclidean distance between MBR and p is no larger than the Euclidean distance between q and p: $\delta_u(p,\mathcal{U}) \geq \delta(p,q), q \in Q$*; Then, the textual relevance between p and* $\mathcal{U}$ *is larger than the textual relevance between p and q:* $\theta_u(p,\mathcal{U}) \geq \theta(p,q), q \in Q$*; According to Definitions 3 and 7, we have:* $\tau_u(p,\mathcal{U}) \geq \tau(p,q), \forall q \in Q$.

Based on the Theorem 1 we could use the query unit $\mathcal{U}$ to prune the search space.

Heuristic 1: Let Q be the query set and *best_k* be the relevance score of the k-th most relevant object found so far. A node N will be pruned if:

$$\tau_u(N,\mathcal{U}) \leq \frac{best\_k}{|Q|}$$

where $\tau_u(N,\mathcal{U})$ is the spatio-textual relevance score between N and $\mathcal{U}$, and $|Q|$ is the size of Q. The concept of heuristic 1 also applies to the leaf entries of IR-tree. When an object p is encountered, we first compute relevance score between p and $\mathcal{U}$. If $\tau_u(p,\mathcal{U}) \leq \frac{best\_k}{|Q|}$, p is discarded since its relevance score cannot be larger than *best_k*. In this way we avoid performing the computations between p and the members of Q.

Based on example 1, assuming we try to find top-1 spatio-textual object and $best\_k = 1.52$. When N_3 is encountered, as shown in Fig. 3, according to Definitions 5 and 6 we could have $\delta_u(N_3,\mathcal{U}) = 1 - \frac{D_\varepsilon(N_3,\mathcal{U})}{D_{\max}} = 1 - \frac{25}{78} = 0.68$ and $\theta_u(N_3,\mathcal{U}) = MAX\{\theta(N_3,q_1),\theta(N_3,q_2)\} = MAX\{0.25,0.28\} = 0.28$. Since $\tau_u(N_3,\mathcal{U}) = 0.5 \cdot 0.68 + 0.5 \cdot 0.28 = 0.48 < \frac{best\_k}{|Q|} = 0.76$, N_3 can be pruned without being visited.

Algorithm 2 shows the pseudo code for UA, where *TopkList* denotes the candidate list of the k most relevant objects found so far, and *best_k* is the relevance score of the k-th most relevant object in *TopkList*. Note that if *TopkList* contains fewer than k members, *best_k* is set to zero. When an object is inserted into *TopkList*, an existing member is replaced if the list already contains k members.

In UA, if *Node* is a non-leaf node, its entries are sorted in a *List* according to $\tau_u(entry,\mathcal{U})$ (lines 2–4). Next, we iterate through the *List* (in the sorted order) and recursively invoke UA on the entries (lines 5–9). Once the first entry with $\tau_u(entry,\mathcal{U}) \leq \frac{best\_k}{|Q|}$ has been found, we ignore this entry and the rest entries in *List* (line 7). We do this because no object in the subtree of this entry will be inserted into *TopkList*. If *Node* is a leaf node, its entries are sorted in a *List* according to $\tau_u(object,\mathcal{U})$ and are visited in this order (lines 10–12). When an object passes heuristic 1 and $T(object,Q) > best\_k$, we insert it into *TopkList* and update *best_k* (lines 13–19). Note that if *TopkList* contains fewer than k members, *best_k* is kept at zero.

Algorithm 2. UA($Node$: IR-tree node, Q: collection of query)

```
1  begin
2      if Node is an intermediate node then
3          List ← entries in Node;
4          sort entries in List according to τ_u(entry,U);
5          foreach entry in List do
6              if τ_u(entry,U) ≤ best_k/|Q| then
7                  break; /* pruned by heuristic 1 */
8              else
9                  UA(entry,Q);
10     else
11         List ← objects in Node;
12         sort objects in List according to τ_u(object,U);
13         foreach object in List do
14             if τ_u(object,U) ≤ best_k/|Q| then
15                 break; /* pruned by heuristic 1 */
16             else
17                 if T(object,Q) > best_k then
18                     Add object to TopkList;
19                     update best_k;
20     return TopkList;
```

4 Experiments

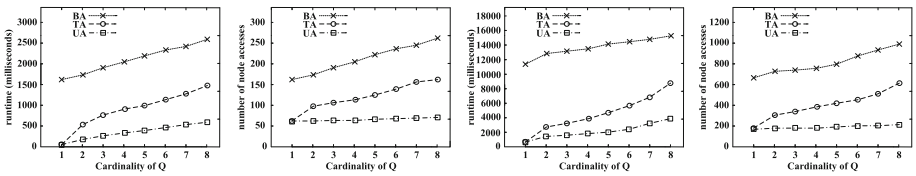
In this section we evaluate the efficiency of our algorithms, using two real datasets: EURO[1] and TWITTER[2]. Table 1 summarizes these two datasets. EURO is a real dataset that contains points of interest (e.g., restaurant, hotel, park) in Europe. Each point of interest, which can be regarded as a spatial web object, contains a geographical location and a short description (name, features, etc.). TWITTER is a dataset generated from 3 million real tweets with location and textual information. All algorithms were implemented in Java, and an Intel(R) Core(TM) i5-4210M CPU @2.60 GHz with 4 GB RAM was used for the experiments. The index structure is disk resident, and the page size is 4 KB and the maximum number entries in internal nodes is set to 100.

In order to make the query Q resemble what users would like use, each $q_i \in Q$ is distributed uniformly in the MBR of Q and has 3 tokens which are randomly generated from datasets. Unless stated explicitly, parameters are set as follows by default: $\alpha = 0.5$, $k = 10$. In all experiments, we use workloads of 100 queries and report average costs of the queries.

[1] http://www.pocketgpsworld.com.
[2] http://twitter.com.

Table 1. Dataset properties

Property	EURO	TWITTER
Total number of objects	193,263	3,000,000
Total number of keywords	105,877	957,716
Average number of keywords per object	11.52	8.74
Data size	43.8 M	264 M



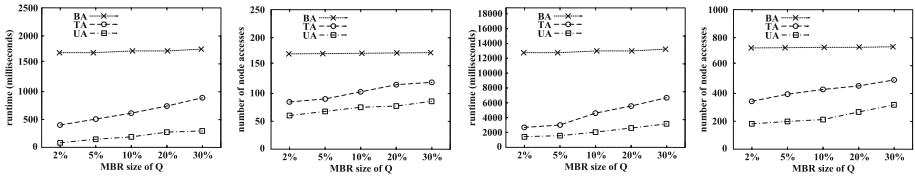
(a) Runtime(EURO) **(b)** I/O(EURO) **(c)** Runtime(TWITTER) **(d)** I/O(TWITTER)

Fig. 4. Results for varying cardinality of Q

The first experiment shows the effect of the cardinality of Q. We fix MBR of Q to 5 % of the workspace of dataset and vary $|Q|$ from 1 to 8. The results are shown in Fig. 4. The CPU cost increases in all algorithms when the cardinality of Q increases from 1 to 8. This is because the distance and textual relevance computations for qualifying objects increase with the number of query points. When $|Q|$ equals 1, the problem is converted to a LkT query. In this case TA has similar performance as UA. However, the performance of UA better than TA when $|Q|$ is larger than 1, due to the high pruning power of heuristic 1. On the other hand, the cardinality of Q has little effect on the node accesses of UA because it does not influence the pruning power of heuristic 1.

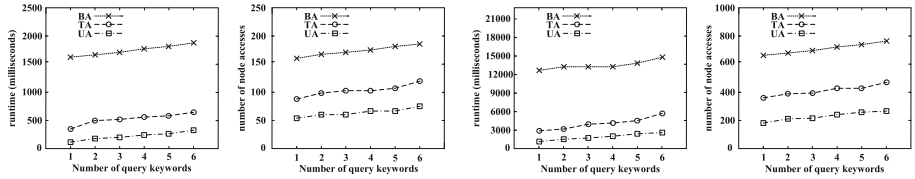
To evaluate the effect of the MBR size of Q, we set $|Q|$ to 2 and vary MBR from 2 % to 30 % of the datasets. The results are shown in Fig. 5. We can see that the MBR size of Q does not influence the CPU cost and node accesses of BA significantly, because it does not play an important role in the process of BA. However, the cost of the other two algorithms increase with the MBR size. For TA, the termination condition is that the total threshold Λ should be smaller than $best\_k$, which increases with the MBR size. Therefore, TA retrieves more spatio-textual objects for each query $q_i \in Q$. For UA, the reason is the degradation of pruning power of heuristic 1 with the MBR size of Q, but it still has remarkable performance than BA and TA.

To evaluate the performance under different number of query keywords, we vary the number of keywords of each $q_i \in Q$ from 1 to 6 by setting $|Q|$ to 2 and MBR to 5 %. The results are shown in Fig. 6. We can see that UA outperforms BA and TA in terms of both runtime and I/O cost. When the number of query keywords increase, all three algorithms need more time and node accesses to finish the search query.



(a) Runtime(EURO) (b) I/O(EURO) (c) Runtime(TWITTER) (d) I/O(TWITTER)

Fig. 5. Results for varying MBR size of Q



(a) Runtime(EURO) (b) I/O(EURO) (c) Runtime(TWITTER) (d) I/O(TWITTER)

Fig. 6. Results for varying number of keywords

5 Related Work

Spatial keyword queries [3] extends classic keyword search to retrieve spatio-textual objects considering relevance to a set of input keywords as well as proximity to the location of the query. A comprehensive comparison of indexing techniques for spatial keyword search appears in [4]. Several proposals use loose combinations of text indexing (e.g., inverted lists) and spatial indexing (e.g., the R*-tree or the grid index) [10,11]. They usually employ the two structures in separate stages. Later, Ian de Felipe et al. [2] proposed a data structure that integrates signature files and R-trees. The main idea was indexing the spatial objects in an R-tree employing a signature on the nodes to indicate the presence of a given keyword in the sub-tree of the node. Cong et al. [1] and Li et al. [12] propose augmenting the nodes of an R-tree with textual indexes such as inverted files. The inverted files are used to prune nodes that cannot contribute with relevant objects. The mCK query [13] takes a set of m keywords as an argument, and retrieves groups of spatial keyword objects. It returns m objects of minimum diameter that match the m keywords. Rocha-Junior et al. [14] propose an indexing structure that associates each term to a different data structure (block or aggregated R-tree) and can process top-k spatial keyword queries more efficiently. Wu et al. [15] proposed an index structure called WIR-tree, which partitions objects into multiple groups such that each group shares as few keywords as possible. Their work aims to find the object closet to the query location that covers all the keywords specified in the query. However, our problem definition is differ significantly from joint spatial keyword query processing.

Current approaches for processing spatial keyword queries only focuses on single query point scenario, and the techniques proposed cannot be applied to the problem of multiple query points (or group queries) top-k spatial keyword query processing. Our GLkT query takes as input a query set Q and returns k spatio-textual objects which have highest relevance score based on their spatial proximity and textual relevance to the query set Q.

6 Conclusion

In this paper, we study the problem of multiple query points (or group queries) top-k spatial keyword query processing (GLkT). We propose two algorithms for processing GLkT query using IR-tree as index structure. First, we propose a threshold-based algorithm (TA). Next, based on the minimum bounding rectangle of query set Q, we propose another more efficient algorithm (UA), which utilizes a pruning heuristic to prune the spatio-textual objects and reduce the cost significantly. Finally, we demonstrate the efficiency of our approach through an extensive experimental evaluation.

References

1. Cong, G., Jensen, C.S., Wu, D.: Efficient retrieval of the top-k most relevant spatial web objects. Proc. VLDB Endow. **2**(1), 337–348 (2009)
2. De Felipe, I., Hristidis, V., Rishe, N.: Keyword search on spatial databases. In: IEEE 24th International Conference on Data Engineering, ICDE 2008, pp. 656–665. IEEE (2008)
3. Cao, X., Chen, L., Cong, G., Jensen, C.S., Qu, Q., Skovsgaard, A., Wu, D., Yiu, M.L.: Spatial keyword querying. In: Atzeni, P., Cheung, D., Ram, S. (eds.) ER 2012 Main Conference 2012. LNCS, vol. 7532, pp. 16–29. Springer, Heidelberg (2012)
4. Chen, L., Cong, G., Jensen, C.S., Dingming, W.: Spatial keyword query processing: an experimental evaluation. In: Proceedings of the VLDB Endowment, vol. 6, pp. 217–228. VLDB Endowment (2013)
5. Guttman, A.: R-trees: a dynamic index structure for spatial searching, vol. 14. ACM (1984)
6. Papadias, D., Shen, Q., Tao, Y., Mouratidis, K.: Group nearest neighbor queries. In: Proceedings of 20th International Conference on Data Engineering, pp. 301–312. IEEE (2004)
7. Yiu, M.L., Mamoulis, N., Papadias, D.: Aggregate nearest neighbor queries in road networks. IEEE Trans. Knowl. Data Eng. **17**(6), 820–833 (2005)
8. Manning, C.D., Raghavan, P., Schütze, H., et al.: Introduction to Information Retrieval, vol. 1. Cambridge University Press, Cambridge (2008)
9. Zobel, J., Moffat, A.: Inverted files for text search engines. ACM Comput. Surv. (CSUR) **38**(2), 6 (2006)
10. Zhou, Y., Xie, X., Wang, C., Gong, Y., Ma, W.-Y.: Hybrid index structures for location-based web search. In: Proceedings of the 14th ACM International Conference on Information and Knowledge Management, pp. 155–162. ACM (2005)
11. Chen, Y.-Y., Suel, T., Markowetz, A.: Efficient query processing in geographic web search engines. In: Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data, pp. 277–288. ACM (2006)

12. Li, Z., Lee, K.C., Zheng, B., Lee, W.-C., Lee, D., Wang, X.: Ir-tree: an efficient index for geographic document search. IEEE Trans. Knowl. Data Eng. **23**(4), 585–599 (2011)
13. Zhang, D., Chee, Y.M., Mondal, A., Tung, A.K., Kitsuregawa, M.: Keyword search in spatial databases: towards searching by document. In: IEEE 25th International Conference on Data Engineering, ICDE 2009, pp. 688–699. IEEE (2009)
14. Rocha-Junior, J.B., Gkorgkas, O., Jonassen, S., Nørvåg, K.: Efficient processing of Top-k spatial keyword queries. In: Pfoser, D., Tao, Y., Mouratidis, K., Nascimento, M.A., Mokbel, M., Shekhar, S., Huang, Y. (eds.) SSTD 2011. LNCS, vol. 6849, pp. 205–222. Springer, Heidelberg (2011)
15. Dingming, W., Yiu, M.L., Cong, G., Jensen, C.S.: Joint top-k spatial keyword query processing. IEEE Trans. Knowl. Data Eng. **24**(10), 1889–1903 (2012)