2011 12th IEEE International Conference on Mobile Data Management

# Efficient Algorithms for Top-k Keyword Queries on Spatial Databases

Ting Wang [#1], Guoliang Li [#2], Jianhua Feng [#3]

[#] *Department of Computer Science, Tsinghua University, Beijing 100084, China*

[1]wangting421@gmail.com; [2]liguoliang@tsinghua.edu.cn; [3]fengjh@tsinghua.edu.cn

***Abstract*— With the ever-increasing number of spatio-textual objects on the Internet, many applications require to find objects in a given range that have the best scores to a keyword query. In this paper, we study the problem of top-k keyword search on spatial databases, which given a range, a keyword query, and a ranking function, finds $k$ objects in the range that has the maximal scores to the keyword query. We study research challenges to address this problem. We extend the well-known R-tree to store both the textual and spatial information and propose a new index structure to index the objects. We devise an efficient threshold-based algorithm and develop effective pruning techniques to efficiently find the best answers. The experiments show that our algorithm achieves high performance and outperforms state-of-the-art methods.**

## I. Introduction

Nowadays, a lot of services for the spatial data are available on the Internet, such as Google maps[5] and Yahoo! maps[6]. Users can search on the spatial data and give ratings to objects on the spatial data. For instance, a user can search for restaurants and give ratings for the restaurants. The rating collected from users' feedback is very important since it can reflect the quality of an object. Users can use these ratings when they search for objects. For example, a user might be willing to go to the restaurant with the best rating of all candidate restaurants in a given region. In this paper, we focus on keyword-based top-$k$ region queries on spatial data: $\langle \mathcal{R}, \mathcal{K}, k \rangle$, where $R$ is a search region, $\mathcal{K}$ is a set of query keywords, and $k$ is the number of returned answers. In our work, we use a rectangle to denote the region. The answer of a query is $k$ objects in the given region that contain all the keywords with the highest score, ranked by relevance.

Figure 1 shows an example. The query is "find the best `primary school` (with the highest rating) in the given region". According to the figure, there are two schools in the given region: $O_5$ and $O_6$. The best result will be $O_5$ because it has the highest rating of these two schools.

To achieve our goal, we propose a new method to address this problem. We propose a new index structure, spatial keyword R-tree, called SKR-Tree, which is extended from the R-Tree with an R-tree node storing both spatial and keyword information.

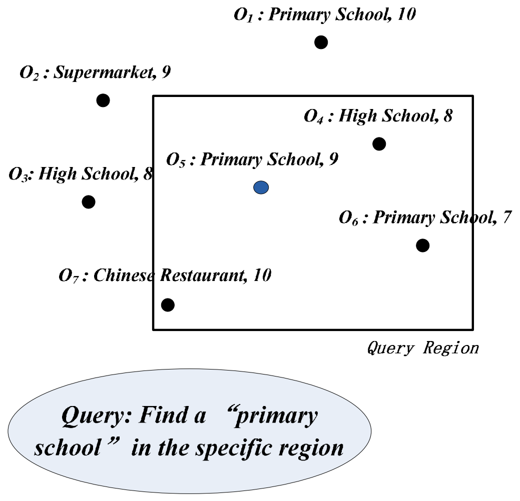

$O_1$ : Primary School, 10
$O_2$ : Supermarket, 9
$O_4$ : High School, 8
$O_5$ : Primary School, 9
$O_3$: High School, 8
$O_6$ : Primary School, 7
$O_7$ : Chinese Restaurant, 10
Query Region
Query: Find a "primary school" in the specific region

Fig. 1. Example: finding a primary school in a specific region

## II. Algorithm

As R-tree only keeps spatial information, to capture both textual and spatial information, we build a *spatial keyword-based R-tree* to address our problem, called SKR-Tree.

### A. SKR-Tree's structure

The structure of SKR-Tree is similar to R-Tree. Each node of SKR-Tree has MBR. MBR is the minimum bounding rectangle of all the objects which are recorded in the subtree of this node. In the leaf node of SKR-Tree, there are some pointers to the objects which are stored in the disk.

The additional information of SKR-Tree includes keyword's max possible value, keyword MBR and max-value set.

Each node stores the max possible value for every keyword in this node. It is calculated from all the objects of the subtree of this node. Keyword MBR means the minimum bounding rectangle of a specific keyword. We choose the top-$C$ objects that has the highest ranking from the nodes of subtree as the max value set.

## B. *Search Algorithm*

Our search starts at the root of the tree. We use a priority queue to store the nodes from the SKR-Tree. The initialized queue has only one element: root of the SKR-Tree. Every time, we process the first element from the priority queue. There are three situations as follows.

1. The first element of the queue is a node, and it is not a leaf node. There are two steps.

First, the algorithm scans all the objects of the max-value set from current node. We check the object whether it contains all the keywords which are given by the query. If it satisfies both spatial and keyword constraints, it can be added into the priority queue.

Next we process all the children of this node. We calculate the intersection of the keyword MBRs from the child node and given region from the query. The algorithm could make a pruning obviously if the intersection of MBRs is empty. After that, the child node will be added into the priority queue with an upper bound value.

2. The first element of the queue is a node, and it is a leaf node. We just load all the objects from this leaf node. The objects can be added into the priority queue if they satisfy the constraints.

3. The first element of the queue is an object. We should add this object to the result list. If the size of the result set is equal to $k$, it means we have got all the results which we need. The algorithm should be stopped and return the result list.

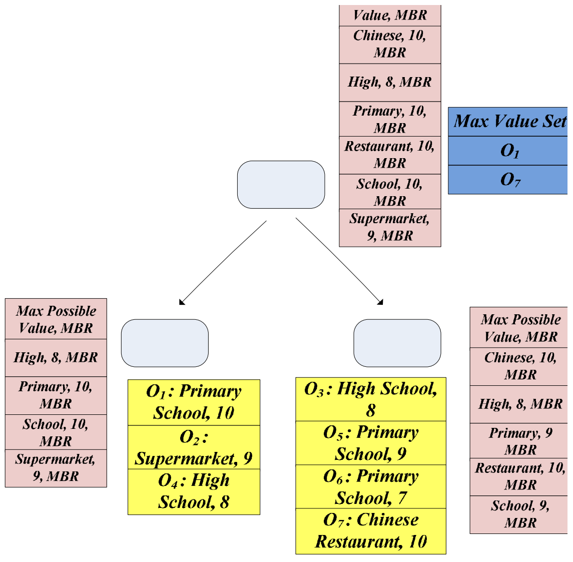


Fig. 2. Example: a SKR-Tree

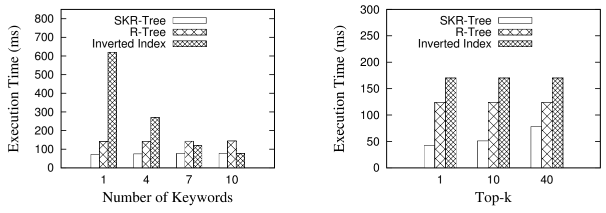
## C. *Overview*

Figure 2 shows an example. It describes a SKR-Tree to store the data in figure 1. Here we set $C = 2$. In the root, it stores keyword's max possible value, keyword MBR and max-value set. The leaf nodes store keyword's max possible value and keyword MBR and some pointers to the objects stored in the disk.

## III. Experiments

We have conducted extensive experiments to show our algorithm has a good performance. We implement the baseline algorithms to compare with our method. We make several experiments to show our algorithm is better than any other baseline algorithm whatever $k$ and the number of keywords are. We used a computer with an Intel(R) Core(TM)2 Duo P8600 @ 2.40GHz CPU and 3GB RAM to finish all the experiments.

In our experiment, We choose the California (CA) data set from the CloudMade project [7]. We use tf-idf as its value of each keyword.



(a) Varying Number of Keywords (b) Varying k

Fig. 3. Execution Time

## A. *Varying Number of Keywords*

Figure 3($a$) shows that the number of keywords varies from 1 to 10 in this experiment. Our algorithm has a better performance than the baseline algorithms.

## B. *Varying k*

Figure 3($b$) shows that $k$ varies from 1 to 40 in this experiment. Our algorithm has a better performance than the baseline algorithms.

## IV. Acknowledgements

This work is partly supported by the National Natural Science Foundation of China under Grant No. 61003004, the National Grand Fundamental Research 973 Program of China under Grant No. 2011CB302206, and National S&T Major Project of China under grant no. 2011ZX01042-001-002.

## References

[1] I. D. Felipe, V. Hristidis, and N. Rishe, "Keyword search on spatial databases," in *ICDE*. IEEE, 2008, pp. 656–665.
[2] D. Zhang, Y. M. Chee, A. Mondal, A. K. H. Tung, and M. Kitsuregawa, "Keyword search in spatial databases: Towards searching by document," in *ICDE*. IEEE, 2009, pp. 688–699.
[3] D. Zhang, "Improving min/max aggregation over spatial objects," in *Proc. of GIS*, 2001, pp. 88–93.
[4] A. Guttman, "R-trees: A dynamic index structure for spatial searching," in *SIGMOD Conference*, B. Yormark, Ed. ACM Press, 1984, pp. 47–57.
[5] "Google maps," http://maps.google.com/.
[6] "Yahoo! maps," http://maps.yahoo.com/.
[7] "Cloudmade," http://cloudmade.com/.