Efficient Combined Index Structure for K-Nearest Neighbours Keyword Search on Spatial Database

Su Nandar Aung, Myint Myint Sein
University of Computer Studies, Yangon
sunandaraung@ucsy.edu.mm, myint@ucsy.edu.mm

Abstract

Spatial keyword search on spatial database has been well studied for years due to its importance to commercial search engines.Specially, a spatial keyword query takes a user location and user-supplied keywords as arguments and returns object that is nearest k objects from user current location and textually relevant to the user required keyword. Geo-textual index play an important role in spatial keyword querying. This paper proposes the efficient combined index structure for K-Nearest Neighbours Keyword Search on Spatial Database. That combine K-d tree and inverted file for nearest neighbor keyword query which is based on the most spatial and textual relevance to query point and required keyword. It can search required k results with minimum IO costs and CPU costs. The k-results are ranked according to the distance or keyword. The own dataset is created for Yangon (Myanmar) region which contains latitude, longitude, name, description and category type of each object.

Keywords: *Combination Scheme, Spatial Keyword Queries, Problem Statement, Proposed Index, K-NN Keyword Search Algorithm.*

1. Introduction

Spatial database systems manage large collections of spatial data, which apart from spatial attributes contain non spatial information. Spatial data are data that have a location (spatial) and mainly required for Geographic Information Systems (GIS) whose information is related to geographic locations. GIS model supports spatial data types, such as point, line and polygon. A geospatial collections increase in size, the demand of efficient processing of spatial queries with text constraints becomes more prevalent.

An increasing number of applications require the efficient execution of nearest neighbor (NN) queries constrained by the properties of the spatial objects. Due to the popularity of keyword search, particularly on the Internet, many of these applications allow the user to provide a list of keywords that the spatial objects should contain, in their name or description or categories. Spatial keyword search is an important tool in exploring useful information from a spatial database and has been studied for years. The query consists of a spatial location, a set of keywords and a parameter k and the answer is a list of objects ranked according to a combination of their distance to the query point and the relevance of their text description to the query keyword. *The spatial relevance* is measured by the distance between the location associated with the candidate document to the query location, and *the textual relevance* is said to be textually relevant to a query if object contains queried keywords. [1]

During the design of a spatial index, issues that need to be minimized are:

(a) The area of covering rectangles maintained in internal nodes,
(b) The overlaps between covering rectangles for indexes developed based on the overlapping native space indexing approach,
(c) The number of objects being duplicated for indexes developed based on the non-overlapping native space indexing approach
(d) The directory size and its height.

Many index structures that have been proposed in recent years mainly use R-tree and then combine with inverted file, namely the families of IR-tree [4, 5, 6, 7, 8, 9, 10]. All use R-tree for spatial (latitude/longitude) index and inverted file for textual index. They all created hybrid index structure according to the combination schemes: (1) *Text first loose combination scheme*, employs the inverted as the top-level index and then arrange the postings in each inverted list in a spatial structure. (2) *Spatial-first loose combination scheme* employs the spatial index as the top-level index and its leaf nodes contain inverted files or bitmaps for the text information of objects contained in the nodes. (3) *Tight combination index*combines a spatial and a text index tightly such that both types of information can be used to prune the search space simultaneously during query processing.

The construction of an efficient index structure should take into account overlaps between nodes and coverage of a node. Minimization of a node coverage leads to more precise searching within the tree and minimization of the overlap between nodes reduces the number of paths tested in the tree during a search that

can reduce search time. As the data objects in the R-tree can be overlapping and covering each other, the search process in the R-tree might suffer from unnecessary node visits and higher IO cost [16]. Moreover, the IR-trees suffer from high update cost. Each node has to maintain an inverted index for all the keywords of documents associated with this node's MBR. When a node is full and split into two new nodes, all the textual information in the node has to be re-organized [1]. As the R-tree need to reorganized, it suffers from higher CUP costs.

This paper intends to reduce IO costs, CUP costs and searching time for kNN keyword search.

This paper includes the following contributions:

(1) The main contribution is to create index structure that combine K-d tree and inverted file for efficiently process spatial keyword queries within minimum time.
(2) Nearest neighbor keyword search algorithm is developed using the proposed index structure to efficiently answer Boolean kNN queries and to explore useful and exact information that user required.

2. Related Works

There has been lot of interest in building geographic information retrieval system. Spatial Keyword search has been well studied for years due to its importance to commercial search engines. Various types of spatial keyword queries have been proposed. For spatial keyword search, the index structure is created for both spatial and textual relevance. Most index structures [10, 5, 6, 8, 9] use R-tree and its variants as spatial index and inverted file for text index. They all combine both indices depending on the combination schemes [15]. Among them [8] integrates signature file instead of inverted file into each node of the R-tree. Inverted file-R*tree (IF-R*) and R*-tree-inverted file (R*-IF) [10] are two geo-textual indices that loosely combine the R*-tree and inverted file. Hariharan et al. R. G¨obel, A. Henrich, R. Niemann, and D. Blank [8] proposed the KR*-tree. This paper proposed a framework for GIR systems and focus on indexing strategies. I. D. Felipe, V. Hristidis, and N. Rishe [9] uses R*-tree for spatial index and inverted file for text index. Cary et al, [5] proposed SKI that combines and R-tree with an inverted index by the inclusion of spatial references in posting lists. In [5] the posting list of term contains all its term bitmaps rather than documents. The IR tree [6] creates each nodes of the R-tree with a summary of the text content of the objects in the corresponding subtree. Li et al. proposed an index structure, which is also called IR tree that stores one integrated inverted file for all the nodes. X. Cao, L. Chen, G. Cong, C. S. Jensen, Q. Qu, A. Skovsgaard, D. Wu, and M. L. Yiu [3] proposed S2I index structure based on R-tree and inverted file. The objects in [3] are stored differently according to the document frequency and infrequency of the term.

D. Zhang, K.L. Tan, Anthony K.H. Tung [1] proposed I^3 (Integrated Inverted Index), which adopts the Quad tree structures to hierarchically partition the data space into cells. The basis unit of I^3 is the keyword cell, which captures the spatial locality of a keyword. X.Cao, G.Cong, Christian S. Jensen, Jun.J. Ng, BengC.Ooi, N.T. Phan, D. Wu [15] proposes a Web Object Retrieval System (SWORS) that is capable of efficiently retrieving spatial web objects that satisfy spatial keyword queries. This system use IR tree and inverted file for index. It supports two types of queries that are location aware top-k text retrieval (Lkt) query and spatial keyword group (SKG) query.

3. Problem Statement

Let D is a spatial database that contains D= $\{o_1,o_2,o_3\ldots..,o_n\}$ such that every object o in D has many attributes $<o_{id}, o_l, o_d>$ where o_{id} is an identifier of an object, o_l is a spatial location that contain latitude and longitude and o_d is an text document of each object for keyword querying.

A *keyword query* q_k is a set of keywords k_1, k_2, k_3, …., k_m. The result is a set of objects ordered by the relevance of their textual description to the query keywords.

Boolean kNN Keyword Queries: Let kNN query $q=<q_k, q_l, k>$ be a Boolean kNN query where q_k is user required keywords $w_1\ldots..w_m$, q_l is a user current location (latitude, longitude) and k is the number of result objects. A query q return k objects o_k from D that are nearest neighbor of q_l with the highest scores according to the Euclidean Distance and Boolean Model in which corresponding point contain required keywords $q_k=\{w_1, w_2,\ldots., w_m\}$.

$$Ans(q) = \begin{cases} o_k \ first\ ordered\ by\ distance\ (o_l, q_l) \\ \vdots \forall_w \in q_k.\ w \in o_d \end{cases}$$

4. Proposed System

The proposed system creates hybrid geo-textual index structure that integrates spatial index and text index to process spatial keyword queries efficiently. In this proposed system K-d tree loosely combined with inverted file. K-d tree is used for spatial queries and inverted file is used for keywords information that is the most efficient index for text information retrieval.For each node of K-d tree,an inverted file is created for indexing the text components of objects contained in the node. As K-d trees represent a disjoint partition, the proposed system

can't cause more IO costs and also K-d trees don't need to rebalance the textual information so the proposed can reduce update cost (CPU Costs).

Table1. Example Dataset

id	Latitude	Longitude	Keywords
Obj1	16.779568	96.152687	Mobile, Shopping, Mall, Telecommunication, Electronics, Tools
Obj2	16.779533	96.15269	May, Shopping, Center, Super Market
Obj3	16.813517	96.08475	Cat, Walk, Foot, Wear
Obj4	16.779565	96.135581	NorthPoint, Shopping, Center, Super, Market, Food, Drink
Obj5	16.881351	96.152549	Gamonpwint, Shopping, Center, Super, Market, Food, Drink
Obj6	16.779581	96.169647	Gamonpwint, Shopping, Center, Super, Market, Food, Drink
Obj7	16.779568	96.152719	Asia, Shopping, Center, Super, Market
Obj8	16.830324	96.186432	Moon, Bakery, Food, Drink

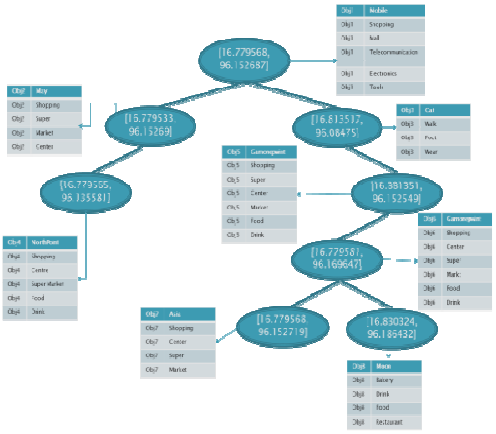

Figure1. Proposed Index Structure for Dataset of Table 1

Most geo-textual indices use the inverted file for text indexing. An inverted file has a vocabulary of terms, and each term is associated with an inverted file. The frequency information is not included in the inverted file that is developed to handle Boolean queries.

Inverted file can be used to check the query keywords contain or not. K-d tree structure is known as point indexing structures as it is designed to index data objects which are points in a multi-dimensional space. It can be used efficiently for nearest neighbor query and range query. This paper proposes nearest neighbor keyword search algorithm using K-d tree and inverted file.

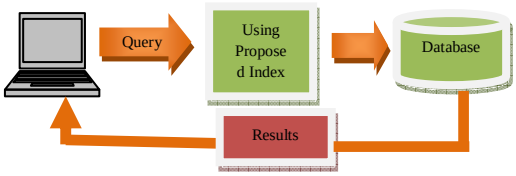

Figure2. Framework for Proposed System

5. Three Types of Spatial Keyword Queries

Standard spatial keyword queries involve different conditions on the spatial and textual aspects of places. In spatial databases, the arguably most fundamental queries are range queries and *k* nearest neighbor queries. In text retrieval, queries may be Boolean, requiring results to contain the query keywords, or ranking-based, returning the *k* places that rank the highest according to a text similarity function. [3]

Three types of spatial keyword queries are receiving particular attention. The *Boolean range query* $q = (\rho, \psi)$ where ρ is a spatial region and ψ is a set of keywords, returns all places that are located in region ρ and that contain all the keywords in ψ. Variations of this query may rank the qualifying places. The *Boolean kNN query* $q=(\lambda, \psi, k)$ takes three arguments, where λ is a point location, ψ is as above, and k is the number of places to return. The result consists of up to k places, each of which contains all the keywords in ψ, ranked in increasing spatial distance from λ. Next, the *top-k range query* $q = (\rho, \psi, k)$ where ρ, ψ, and k are as above, returns up to k places that are located in the query region ρ, now ranked according to their text relevance to ψ. Finally, the *top-k kNN query* takes the same arguments as the Boolean *k*NN query. It retrieves k objects ranked according to a score that takes into consideration spatial proximity and text relevance.

Among these queries, the latter two ones that perform textual ranking are the most similar to standard web querying, and the last one is the one that is most interesting and novel, as it integrates the spatial and textual aspects in the ranking.

6. K-NN Keyword Search Algorithm

Algorithm1. K-NN Keyword Search in Hybrid Index Structure

```
NNKeywordSearch (T,Q)
T: kd tree;
Q: Query that contains current location Q.l, required keyword
    Q.key, number of required nearest neighbours objects Q.k;
L: Arraylist;
pqResult: Priority Queue;
count ← 0;
pqResult ← NNSearch (T, Q);
while count <Q.k do
    L.add (pqResult.remove());
return L;
          ----------------------------------------------------
NNSearch (T, Q)
pq: Proirity Queue;
pqResult: Priority Queue;
Search: tuple kd tree, bounding_box, potential_distance and
            tuple;
nnPoint: undefine;
minDistance: infinity ( ∞)
pq.add( Search ( T, B_Box, 0));
whilepq.size>0 and pq.TOP(().potential_distance<minDistance
        do
        T ← pq.TOP().kdtree;
B_Box ← pq.TOP ().bounding_box ;
pq.remove();
    if T≠ leaf then
      point ← T.key;
      i ← T.discr;
      distance ← DISTANCE ( point.l , Q.l);
if distance<minDistance&&Q.keyword Є point.keywords
    pqResult.ADD(point);
    minDistance ← distance;
BOUNDINGBOX (left_BB,right_BB,BB,point[i])
potential_distance ←MINDISTANCE(left_BB,Q.l);If
    potential_distance< distance then
    pq.ADD(Search(T.left, left_BB, potential_distance));
potential_distance← MINDISTANCE (right_BB,Q.l);
If potential_distance< distance   then
        pq.ADD(Search(T.right, right_BB,potential_distance));
returnpqResult;
```

The algorithm-1 returns the closest points to a given user's current location according to a certain distance function.When the algorithm explores some points of the kd-tree, it starts computing the distance between this points and query point and then check the required keywords contain or not.

In the algorithm, the procedure ComputeBoundingBoxes(…) returns the bounding boxes lBB and rBB for the left and the right subtrees, respectively. The function MinimumDistance(BB; c) returns the potential distance between any point located inside the bounding box BB and query point. The DISTANCE (…) procedure calculates the distance between two points using Euclidean distance.

$$d(q,p) = \sqrt{(q_{lat} - p_{lat})^2 + (q_{lon} - p_{lon})^2}$$

7. Architecture of Proposed System

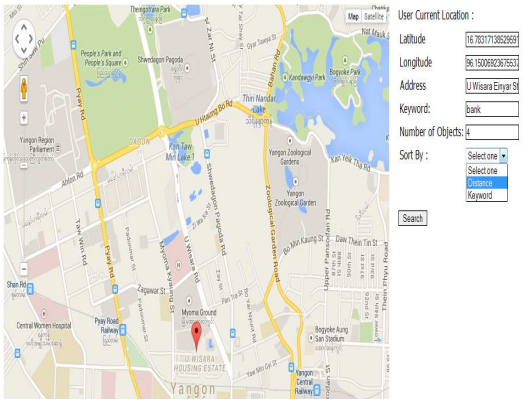

Figure3. User Interface for the Proposed System

The propose system adopts the browser-server model for desktop and laptop computer. Figure3 shows the user interface for the proposed system. Users can input their queries through the web browser and the queries are sent to the server for processing. After the queries are processed, the results are sent back and displayed using Google Maps in the users' browser. Queries are sent from the browser to the server by the HTTP post operation.

The browser side use Google Map API to provide interfaces to users for generating queries and viewing the returned spatial web objects. Users can specify the current's location by clicking a location in Google Map to get the latitude and longitude of that location and can type the required keywords. And then the required number of objects k and the sort by type. The query is sent to the server and then relevant k-objects are retrieved by the server that are nearest neighbours and contain the specified keyword. The results are sorted by the distance or keyword and then are displayed on Google Maps in the browser.

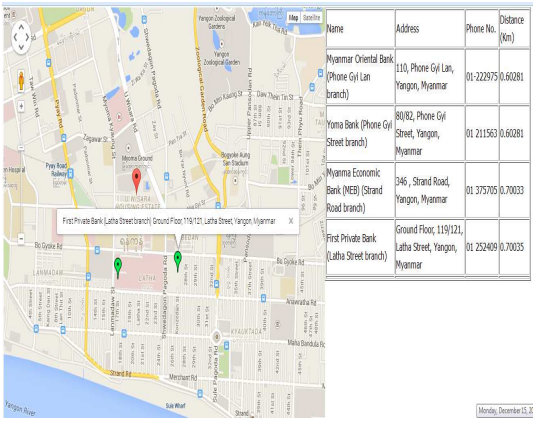

Figure4. User Interface for the Result

8. Experimental Results

Figure5 shows the index construction time (second) depending on the size of datasets. Figure6 compare the searching time (second) depending on the number of required keywords between using proposed index structure and other index that combine R-tree and inverted file. Searching time using proposed index structure is faster than other index (R-tree and inverted files) about 100-times in second. Figure7 shows the searching time depending on the varying number of objects k.

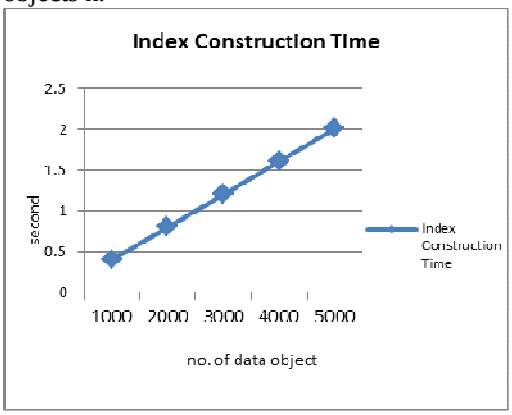

Figure 5. Index Construction Time

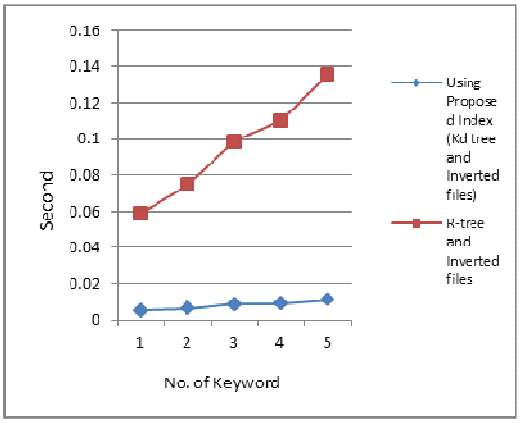

Figure 6. Searching Time for varying number of keywords

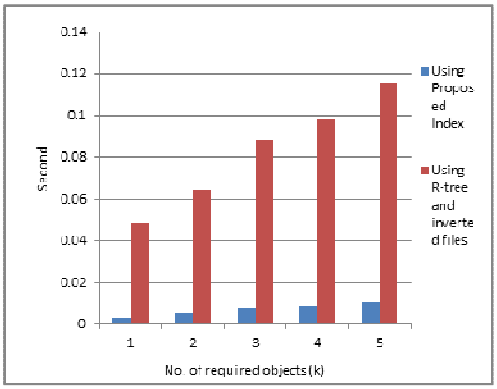

Figure 7. Searching Time for varying number of required objects

9. Conclusion and Further Extension

This paper presented hybrid index structure for range keyword query searching with minimum IO costs and CPU costs. This index structure can avoid searching in overlapping area. So it can reduce searching time in overlap area. Moreover, it can't cause node overflow, so it doesn't need to re-organize the textual data and spatial data. Many Further extensions can be considered for efficient hybrid index structure for spatial database. As a further extension, we'll add an efficient spatial approximate keyword search and Boolean keyword search within given range and nearest neighbour and approximate keyword search in this proposed index structure.

REFERENCES

[1] D. Zhang, K.L. Tan, Anthony K.H. Tung, "Scalable Top-K Spatial Keyword Search", EDBT/ICDT'13 March 18-22, 2013

[2] L. Chen, G. Cong, C. S. Jensen, and D. Wu, "Spatial Keyword Query Processing: An Experimental Evaluation", in proceddings of the VLDB Endowment, Vol.6, No.3, 2013.

[3] X. Cao, L. Chen, G. Cong, C. S. Jensen, Q. Qu, A. Skovsgaard, D. Wu, and M. L. Yiu, "Spatial keyword querying", in*ER*, pages 16–29, 2012.

[4] J. B. Rocha-Junior, O. Gkorgkas, S. Jonassen, and K. Nørv°ag, "Efficient processing of top-k spatial keyword queries", in *SSTD*, pages 205–222, 2011.

[5] Z. Li, K. C. K. Lee, B. Zheng, W.-C. Lee, D. L. Lee, and X. Wang, "Ir-tree: An efficient index for geographic document search", *IEEE TKDE*, 23(4):585–599, 2011.

[6] A. Cary, O. Wolfson, and N. Rishe, "Efficient and scalable method for processing top-k spatial Boolean queries", in*SSDBM*, pages 87–95, 2010.

[7] G. Cong, C. S. Jensen, and D. Wu, "Efficient retrieval of the top-k most relevant spatial web objects", *PVLDB*, 2(1):337–348, 2009.

[8] R. G¨obel, A. Henrich, R. Niemann, and D. Blank, (2009), "A hybrid index structure for geo-textual searches", in *CIKM*, pages 1625–1628, 2009.

[9] I. D. Felipe, V. Hristidis, and N. Rishe, "Keyword search on spatial databases", in *ICDE*, pages 656–665, 2008.

[10] R. Hariharan, B. Hore, C. Li, and S. Mehrotra, (2007), "Processing spatial-keyword (sk) queries in geographic information retrieval (gir) systems", in *SSDBM*, page 16, 2007.

[11] Y. Zhou, X. Xie, C. Wang, Y. Gong, and W.-Y. Ma, "Hybrid index structures for location-based web search", in *CIKM*, pages 155–162, 2005.

[12] H.M. Kakde, "Range Searching using Kd Tree", 2005.

[13] A. Guttman, "R-trees: A dynamic index structure for spatial searching", in *SIGMOD*, pages 47–57, 1984.

[14] B.C. Ooi, R. Sacks-Davis, J.Han, "Indexing in Spatial Databases".

[15] X.Cao, G.Cong, Christian S. Jensen, Jun.J. Ng, BengC.Ooi, N.T. Phan, D. Wu, "SWROS: A System for the Efficient Retrieval of Relevant Spatial Web Objects".

[16] Y. Theodoridis, T. Sellis, "Optimization Issues in R-tree Construction", Technical Report KDBSLAB-TR-93-08.