K-NEAREST NEIGHBOURS APPROXIMATE KEYWORD SEARCH FOR SPATIAL DATABASE

[1]SU NANDAR AUNG, [2]MYINT MYINT SEIN

[1]University of Computer Studies (Yangon, Myanmar),
[2]Research and Development Department (University of Computer Studies, Yangon, Myanmar)
E-mail: sunandaraung@ucsy.edu.mm, myint@ucsy.edu.mm

Abstract- Keyword search over a large amount of data is an important operation in a wide range of domains. Nearest neighbor objects with required for location-based search from spatial database has been well studied for years due to its importance to commercial search engines. Specially, the top-k spatial keyword query takes a user location and user-supplied keywords as arguments and returns objects that is nearest k objects from user current location and textually relevant to the user required keyword. In these systems, inconsistencies and errors can exist in the user's typed queries. This paper proposes new index structure that combines K-d tree and inverted file to answer such query efficiently and we also discuss how to answer the queries that contain the user typing error in keyword.

Keywords- KNN Approximate Keyword Search, Experimental Results.

I. INTRODUCTION

During the past several years traditional databases have been enhanced to include spatially referenced data. This type of data is an essential component of existing applications such as Geographic Information Systems (GIS), Computer-Aided Design (CAD), multimedia information systems, data warehousing, and NASA's Earth Observing System (EOS). Spatial databases have been defined as database systems with a model and query language that support spatial data types and provide spatial indexing and efficient algorithms for spatial query processing. In spatial databases, data are associated with spatial coordinates and extents, and are retrieved based on spatial proximity. 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. In reality, for many scenarios, keyword search for retrieving approximate string matches is required. The spatial keyword query consists of a spatial location, a set of keywords 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.

Keyword search over a large amount of data is an important operation in a wide range of domains. Approximate string search could be necessary when users have a fuzzy search condition or simply a spelling error when submitting the query. In the context of spatial databases approximate string search could be combined with any type of spatial queries, including range and nearest neighbor queries. For example, "find k objects that are nearest neighbors of the user current location and have a description that is similar to Super Market". Similar example could be constructed for range queries. A key issue in understanding the semantics of these queries is to define the similarity between two strings. The edit distance metric is often adopted for such approximate string queries.

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. 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. 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 with inconsistencies of user input keyword. 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 nearest neighbours keyword queries within minimum time.
2) K-Nearest Neighbors keyword search algorithm is developed using the proposed index structure to

efficiently answer kNN queries and to explore useful information that contains user required keywords using approximate string matching method.

II. 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 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. Among them, Felipe et al. Reference 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) are two geo-textual indices that loosely combine the R*-tree and inverted file. Hariharan et al. Reference proposed the KR*-tree.

This paper proposed a framework for GIR systems and focus on indexing strategies.Reference uses R*-tree for spatial index and inverted file for text index. Cary et al, proposed SKI that combines and R-tree with an inverted index by the inclusion of spatial references in posting lists. In the posting list of term contains all its term bitmaps rather than documents. The IR tree 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. Reference proposed S2I index structure based on R-tree and inverted file. The objects in are stored differently according to the document frequency and infrequency of the term. 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. Reference 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. Reference proposed the system that studies how to efficiently combine these two types of indexes, and how to search the resulting index to find answers. This paper use the IR tree index and try to bridge the gap between queries and data, it is important to support approximate keyword search on spatial database. Reference proposed the MHR-tree that supports a wide range of query including range and NN queries. This paper used the R-tree, the min-wise signature and the linear hashing technique for efficiently answering approximate string match queries in large spatial databases.

III. PROBLEM STATEMENT

The propose system intends to anwer the kNN queries with the user's required keyword that can have typing error called kNN approximate keyword query. 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 kNN query is defined by $q=(\lambda, \psi, k)$ that takes three arguments, where λ is a point location, ψ is a set of keyword , and k is the number of objects 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. A query q return k objects o_k from D that are nearest neighbor of λ with the highest scores $\alpha(o_l,\lambda), \alpha(o_{1l},\lambda) \geq \alpha(o_{2l},\lambda) \geq \alpha(o_{3l},\lambda) \geq \cdots \geq \alpha(o_{nl},\lambda)$, according to the Euclidean Distance or return k objects ok from D that are nearest neighbors of λ with the less edit distance between query keyword and text document of objects in D, $ed(o_d,\psi), ed(o_{1d},\psi) \leq ed(o_{2d},\psi) \leq ed(o_{3d},\psi) \leq \cdots \leq \alpha(o_{nd},\psi)$ using Levenshtein Distance for approximate string matching.

Given a query string 'q' and a set of strings 's' from DB, the edit distance between 'q' and 's' , denoted as ed(q,s), is defined as the minimum number of edit operations required to transform one string into the other. The edit operation refers to an insertion, deletion, or substitution of a single character. This can estimate the size of the answer set {s | ed(q,s) <= t and s Є DB}, where t is the edit distance threshold. Mathematically, the Levenshtein distance between two strings a,b is given by $lev_{a,b}(a|b)$ where

$$lev_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0, \\ \min \begin{cases} lev_{a,b}(i-1,j)+1 \\ lev_{a,b}(i,j-1)+1 \\ lev_{a,b}(i-1,j-1)+1_{(a_i \neq b_i)} \end{cases} & \text{otherwise} \end{cases}$$

where $1_{(ai\neq bj)}$ is the indicator function equal to 0 when $a_i=b_j$ and equal to 1 otherwise. The distance is the number of deletions, insertions, or substitutions required to transform s into t. The greater the Levenshtein distance, the more different the strings are.

IV. 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). This paper proposes kNN keyword search algorithm using K-d tree and inverted file. And also the propose system use Levenshtein Distance for approximate string search.

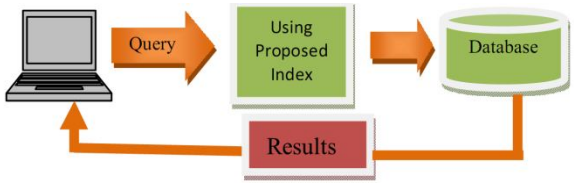


Fig1: Framework for Proposed System

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 calculates the edit distance between query keyword and document of that point by using Levenshtein Distance(point.keyword, Q.keyword) ≤ t (thresholds t=length of Q.keyword /2). In the algorithm, the procedure ComputeBounding Boxes(…) 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. $d(q,p) = \sqrt{(q_{lat} - p_{lat})^2 + (q_{lon} - p_{lon})^2}$. LEVENSHTEIN (...) procedure calculates the edit distance between input keyword and each point keyword.

Algorithm1. K-NN Approximate 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);
     edit_distance<- LEVENSHTEIN (
     point.keyword,Q.keyword)

if distance<minDistance&&edit_distance ≤ t
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;
```

V. EXPERIMENTAL RESULTS

Fig2 shows the index construction time (second) depending on the size of datasets. Fig3 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). Fig4 shows the searching time depending on the varying number of objects k.

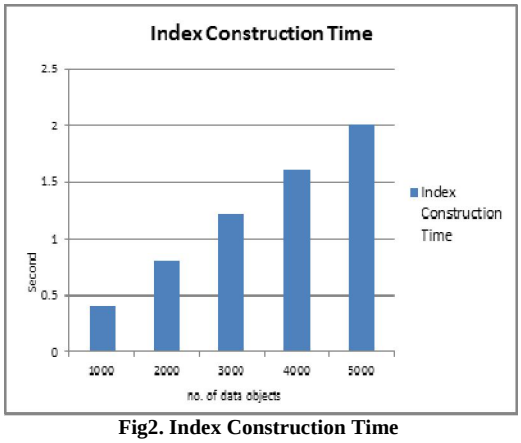


Fig2. Index Construction Time

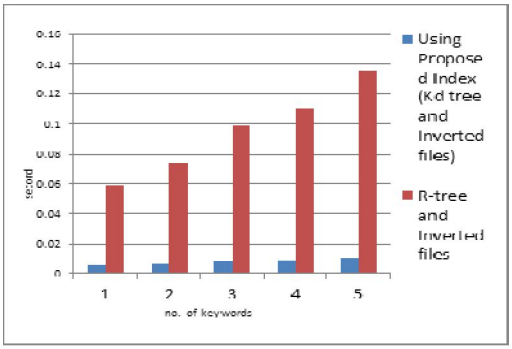


Fig3. Searching Time for varying number of keywords

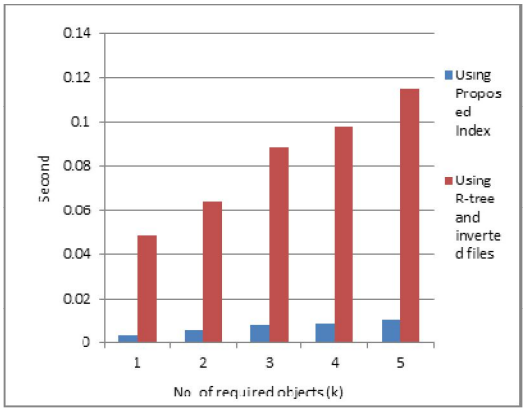


Fig4. Searching Time for varying number of required objects

CONCLUSION

The proposed 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. It can efficiently process kNN keyword query searching with minimum IO costs and CPU costs and can also correct the inconsistencies of the user's input keyword.

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", inER, 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", inSSDBM, 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.

[17] S. Alsubaiee, A.Behm, C.Li, " SupportingLoction-Based Approximate-Keyword Queries", ACM GIS' 10, November 2-5, 2010. San Jose, CA, USA.

[18] B.Yao, F.Li, M. Hadjieleftheriou, K. Hou, " Approximate String Search in Spatial Database".

[19] http://en.wikipedia.org/wiki/Levenshtein_distance

[20] http://en.wikibooks.org/wiki/Algorithm_Implementation/Strings/Levenshtein_distance