IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 7, Issue 4, April 2018

A Novel Approach to Search Top K Spatial Objects Using Inverted Linear Quadtree Method

Sumedha Holla[1], Sumithra R. M[2] , Lavanya G[3] , Megha Shastri[4], Prof. Pushpaveni H.P[5]

Scholars, Dr. Ambedkar Institute of Technology, Bengaluru[1,2,3,4]

Assistant Professor, Dept. of CSE, Dr. Ambedkar Institute of Technology, Bengaluru[5]

Abstract: The advances in geo-positioning technologies and geo-location services, has paved way for the rapid growth of spatio-textual objects. These are theobjects that are collected in many applications such as location based services and social networks. The spatio-textual objects are the objects in which an object is described by its spatial location and a set of keywords (terms). The approach to handle the spatio textual objects can be done we study two fundamental ways: top k spatial keyword search (TOPK-SK), and batch top k spatial keyword search (BTOPK-SK). Given a set of spatio-textual objects, a query location and a set of query keywords, the TOPK-SK retrieves the closest k objects each of which contains all keywords in the query. BTOPK-SK is the batch processing of sets of TOPK-SK queries. We also take into consideration a data structure which is based on the inverted index and the linear quadtree, a novel index structure, called inverted linear quadtree (ILQuadtree). This data structure is carefully designed to exploit both spatial and keyword based pruning techniques to effectively reduce the search space. To deal with BTOPK-SK, a new computing paradigm is to be implemented which partitions the queries into groups based on both spatial proximity and the textual relevance between queries. To efficiently support BTOPK-SK, we implement IL-Quadtree technique in this paper.

Keywords: Spatial, Spatio-textual objects, Keyword, Batch processing.

1 INTRODUCTION

*Spatio-textual objects*are available in many applications. For example, the local search service provides the location information as well as short descriptions of the businesses. In the GPS navigation system,ageographically anchored pushpin called Point Of Interest(POI) is used. It is usually annotated with texture information.These uploaded images are usually associated with multiple text labels. As a result, in recent years various spatial keyword query

models and techniques have emerged such that users can effectively exploit both spatial and textual information of these spatiotextual objects.In the paper, we investigate the problem of conducting *top k spatial keyword search* (TOPK-SK) [1], [2]; that is, given a set of spatio-textual objects, a query location q and a set of keywords, we aim to retrieve the k closest objects each of which contains all keywords in the query. The top k spatial keyword search is important in spatial keyword queries and has a wide range of applications. Below is the example

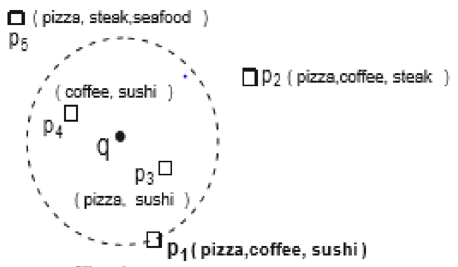


Fig. 1: Online Yellow Pages Example

In Fig. 1, suppose there are a set of businesses whose locations (represented by squares) and service lists (a set of keywords). When a GPS-enabled smartphone user wants to find a nearby restaurant to have a piece of pizza and a cup of coffee, she may send the local search server two keywords, coffee and pizza. Based on the user's current location (e.g., the point q in Fig. 1) derived from the smartphone and the two query keywords, business p1 is returned by the server. Note that although businesses p3 and p4 are closer to q than p1, they do not satisfy the keyword constraint.

In many real applications, the query workload may vary from time to time, and the system may encounter a burst of queries. In this scenario, the system throughout is poor if a large number of queries are processed one by one. The batch query processing techniqueshandles aa large number of queries in the queue with a reasonable delay. A large number of fake spatial keyword searches may be issued in order to protect the user's privacy. This may lead to dramatic degrade of the system throughout if queries are processed individually. To alleviate this issue, we also investigate the problem of batch spatial keyword query (BTOPK-SK) which aims to efficiently support a large number of spatial keyword queries at the same time.

Motivation. The performance of the spatial keyword query is poor if objects are separately organized by textual index and spatial index. Therefore, the main challenge is how to effectively combine the spatial index and textual information. In the paper, we propose the inverted linear quadtree (ILQuadtree) indexing technique which naturally combines the spatial and textual features of the objects. Specifically, for each keyword we build a linear quadtree for the related objects so that the objects which do not contain any query keyword can be immediately excluded from computation. Besides facilitating spatial search, the quadtree for each keyword can also serve as the signature of the objects in the sense that we can effectively prune a group of objects based on the AND semantics. Moreover, as the space filling curve technique is employed by the linear quadtree, the objects are clustered on the disk based on their spatial proximity, which enhances the I/O efficiency of our search algorithm. We further enhance the pruning capability of the signature technique by partitioning objects into groups. The key challenge of the BTOPK-SK problem is the computation sharing of the spatial keyword queries. In this paper, we develop effective partition approach to group queries such that the queries in the same groups can share computation costs because they are likely to have common search paths. Based on the inverted linear quadtree (IL-Quadtree) and the groups of the queries, we devise efficient batch processing algorithm to support BTOPK-SK queries.

Contributions. The principle contributions of this paper are as follows.
• To facilitate the top k spatial keyword search (TOPK-SK), we propose a novel indexing structure called IL-Quadtree to effectively organize spatio-textual objects.
• Based on IL-Quadtree, we develop an efficient top k spatial keyword search algorithm
• Efficient batch spatial keyword query processing techniques are developed to improve the throughout of the system when there are a large amount of spatial keyword queries.

2 PRELIMINARY

2.1 Problem Definition

Let us denote thespatio-textual object as o and denote the location by o.loc and the keyword of terms byo.Tfrom the vocabulary V. The top k spatial keyword search (TOPK-SK) yields the following result. Consider a set O of spatio-textual objects, a query object q where q.loc is the query location and q.T is a set of query keywords, we obtain the closest k objects each of which contains all of the query keywords. In this paper, a batch top k spatial keyword query, denoted by Q, consists of a set of top k spatial keyword queries. Theminimum bounding rectangle of the locations of the queries in Qis denoted by Q.mbr; Q.T represents the union of the keyword sets of the queries in Q; and Q.µ is the smallest keyword set size of the queries in Q. BTOPK-SK. Given a set O of spatio-textual objects, and a batch spatial keyword query (BTOPK-SK) Q, we try to get the best top k results for each top k search.

3 IL-QUADTREE

In Section 3.1 we show the disadvantages of the existing indexing approaches. Section 3.2 is written to overcome the disadvantages shown in the existing system.

3.1 Motivation The two great challenges faced by current works that uses TOPSK search may be listed as follows: i) how to effectively reduce the number of objects with the search region. ii) how to effectively decrease the average surviving probability of these objects.These disadvantages has to be tackled with a new index structure has following properties. Firstly,It should involve inverted index i.e., related objects are grouped by a spatial index for each keyword. In this way, the objects which do not contain any query keyword is immediately eliminated. Secondly, the new index structure should be adaptive to the distribution of the objects for each keyword. Thirdly, we need to exploit the AND semantic for the computation of the results.

3.2 IL-Quadtree Structure

In this paper, we have introduced a datastructure called as quadtree. A quadtree is a used for space partitioning in which a d-dimensional space is recursively subdivided into 2d regions. This structure is very simple. As shown in Fig. 2(a), assuming that quadtrees resulting from a split are numbered in the order
SW, SE, NW and NE, which are represented by 00, 01, 10 and 11 respectively 1. As shown in Fig. 2(c), in the paper, we use two conventions i.e. a circle and a square to denote the non-leaf node and leaf node respectively. Moreover, a leaf node is set black if it is not empty, i.e., it contains at least one point. Otherwise, it is a white leaf node. In Fig. 2(b)

and (c), the codes of the leaf nodes are labelled by the corresponding integer number of their split sequences (i.e, Morton code).

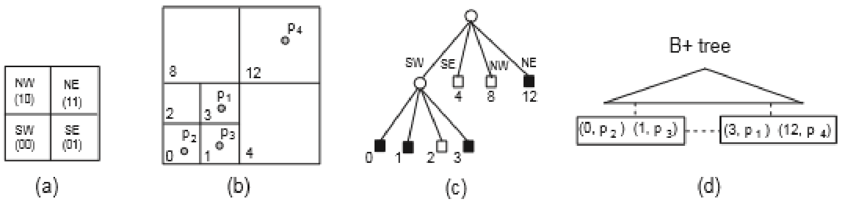


Fig. 2: Linear Quadtree Example

Note that in the paper the quadtree structure is kept as the space partition based signature of the objects. Fig. 2(d) shows an ordering results of these black leaf nodes as well as the objects resident on them, where the node codes are represented as integer numbers.

3.3 IL-QUADTREE BASED TOPK-SK QUERY

Here, we introduce an efficient TOPK-SK query algorithm assuming that objects are organized by an IL- Quadtrees. Same as other inverted index based approaches, we also conduct incremental nearest neighbor search on the IL-Quadtrees. The main difference is that we can make use of the space partition based *signatures* (i.e., quadtree structures) to eliminate non-promising objects.

Algorithm 1: TOPK-SK (Q, q, k)

```
Input :LQ : the IL-Quadtree, q : the spatio-textual query
k : number of objects returned,
Output :R : TOPK-SK query result.
$R = \varnothing ; H = \varnothing ; \lambda = \infty;$
For each LQi  wherei belongs to I(q.T) do
Push root node of LQi into H;

While H not equal to $\varnothing$ do
e←the quadtree node popped from H;
if e is a black leaf node then
suppose e is from LQi;
for each LQj where j not equa to i and j belongs to I(q.T) do
CheckSignature(e,LQj);
If e passed the signature test then
      Load objects contained by e;
For each object o contained by e do
Ohit=Ohit+1;
          If Ohit=l then
$Compute \delta(o, q);$
$if \delta(o, q) > \lambda then$
Break;
   Else

   Update $\lambda$ and R;
Else if e is a non-leaf node then
     For each child node e' of e do

        If e is not a white leaf node and $\delta_{min}$(e',q) less than or equal to $\lambda$ then
Push e' into H;
Return R
```

4 BATCH TOP k SPATIAL KEYWORD SEARCH

In this Section, we investigate how to effectively process the batch top-k spatial keyword queries.

4.1 Motivation

The system efficiency may reduce and it is time consuming if we separately process each individual query in the BTOPK-SK, especially when there are more number of queries. Therefore, it is advisable to partition the queries into groups. By grouping the queries, it may share computation and hence significantly reduce the overall costs. Moreover, novel query processing techniques are mandatory to cope with groups of queries based on the IL-Quadtree proposed in this paper.

4.2 Effective query partition.
Query partition plays an important role in batch query processing because it can notably reduce the processing time by grouping similar queries so that the CPU and I/O costs can be shared between queries in the same group. This strategy has been widely adopted in many research papers such as [4]. However, this method only considers spatial information, and does not considers textual composition. Thus, we can take both spatial feature and textual information into consideration to further improve the effectiveness of query partition.

```
Algorithm 2: Partition(O, Q, D)
Input :O : the spatial textual objects,
Q : the query log
D : the depth of recursion
$hfSet = \Phi; hpSet = \Phi; fpSet = \varphi; fpList = \varphi;$
hfSet←extracts the terms whose local frequency            larger than $\gamma$;
fpSet← enumerates all unique term pairs from    hfSet;
hpSet←from objects in O enumerate the unique term pairs;
fplist←fpSet\hpSet;
Q←extracts the query whose terms covered by O from Q;
$cList_1 = \varphi; cList_2 = \varphi; \min = inf; pset = \varphi;$
for each pair  pi belongs to fpList do
assumetj is the jth term from pi;$tList_1 = \varphi; tList_2 = \varphi;$
tList1←the objects in O contains t1;
tList2←the objects in O contains t2;
the other objects assigned to the list that has higher similarities;
$if\ \mathcal{E}(Q.CtList_1) + \mathcal{E}(Q.CtList_2) \leq \min\ then$
Update min cList1,cList2 and pset;
fpList←fpList\pset;D+=1;
$if \eta > 2^D then$
$c \leftarrow c \cup Partition(cList_1, fpList, Q, \eta/2);$
$c \leftarrow c \cup Partition(cList_2, fpList, Q, \eta/2);$
Else
Update C by cList1 and cList2;
```

4.3 Efficient batch query processing
After dividing the batch query into a set of groups, it is essential to devise new query processing techniques based on IL-Quadtree structure proposed in this paper. The key is the computation sharing strategies. We carefully maintain the objects loaded during the query processing so that the location information of these objects can be used to facilitate the computation. More specifically, assume there are a set of objects loaded from cell c of LQtwhere c is a black leaf node of LQt. If there are some candidate queries on c, we can further push down these objects for LQt, i.e., generate refined signatures for child cells of c on the fly, to facilitate the signature tests of these queries.

```
Algorithm 3:Grouping(Q,k)
Input:Q:a set consisting of spatio textual queries
        k:the number of assigned group
Output:G:a set consisting of queries assigned to k groups.
$foreachqueryqi \in Qdo$
qi.val←calculate z-order value for q;
G←sample k points which is uniformly based on z-order values queries;
isMove←true;
whileisMovedo
```

```
$foreach query qi \in Q do$
$score < -\infty, set < -0;$
$foreach cluster gj \in G then$
if f(gj,gi)<gscore then
score←f(gj,gi);set←j;
Gset←GsetUqi;
recalculate the k centroids;
if the k centroids can no longer move then
isMove←false;
return G
```

Algorithm 5:Node Processor(Q*,e',R)
Input:Q*:the term constraint of satisfied spatio-textual queries, e':current node,R:top-k results of each subquery for Q
Output:R:top-k results of each subquery for Q

```
$foreach qi \in Q * do$
hit←0;
$foreach t \in q.T do$
        if e' is a leaf node belonging to LQt then
hit:=hit+1;
*if e' belongs to the object list of LQt is empty then
load the objects from LQt and then update e';
if hit=|q.T| then
objs←intersects the objects of related terms in e';
$foreach o \in objs do$
 $if \delta\min(q,o) < q.\lambda then$
$updateRi by(o, \delta\min(q,o))$
      $update \lambda\max;$

else
update e' by qi;
return R
```

Algorithm 4:BTOPK-SK(Q,k)
Input:k:number of objects returned,Q:the query group,
Output:R:top-k results of each query in Q

```
$R := \Phi; H := \varphi \lambda\max = \infty \; Q.T = \varphi$
e←new a root node;
for each subquery qi belongs to Q do
Ri←new max heap;
Initialise Ri with k null object with distance infinity;
let qi.$\lambda$ be the maximum distance in Ri;qi.$\lambda$←$\infty$
e.Q←e.QUqi;
for each LQt where t belongs to I(qi.T) do
if t doesnot belong to e.T then
e.T←e.Tut;
update e by the root node of LQt;
push e into H,$\lambda$max ←$\infty$
while H not equal to $\varnothing$ do
e←the node popped from H;
if emin greater than or equal to $\lambda$max then
```

```
break;
Q'←{qi $\in$ e.Q|$\delta$min(qi,e) $\leq$ qi.$\lambda$^qi.T $\in$ e.T};
If Q'is $\varnothing$ then
Continue the while-loop;
For each t $\in$ e.T do
If the object list of e'of t is not empty then
Split the objects into child entries e';
For eah child entries e'of e do
If |Q'.T $\cap$ e'.T| $\geq$ Q.$\mu$ and $\delta$min(Q',e')< Q'.$\lambda$ then
Q*←{qi $\in$ Q'|$\delta$min(qi,e') $\leq$ qi.$\lambda$^qi.T $\in$ e'.T};
If Q* is not $\varnothing$ then
If e'.level>=w'then
Node-processor(Q*,e',R);

Else
Update e' by Q*;
Push e' into H;
Return R
```

5 PERFORMANCE EVALUATION

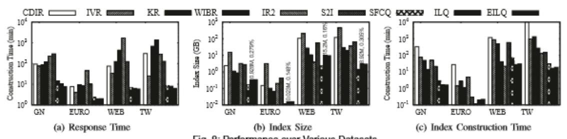
The results of the performance study to evaluate the efficiency and scalability of the proposed techniques in the paper is analysed here. The following algorithms are evaluated for the TOPK-SK query.

- **ILQ**. IL-Quadtree based TOPK-SK algorithm.
- **IVR**. The inverted R-tree based TOPK-SK algorithm.
- **IR2**. The enhanced IR2-tree technique based TOPKSK algorithm.
- **KR∗** TOPK-SK algorithm developed based on the KR∗tree [8].
- **WIBR**. The WIBR-tree technique is proposed in [4].
- **CDIR**. The CDIR-tree technique is proposed in [9].
- **S2I**. The S2I3 technique is proposed in [7].
- **SFCQ**. The SFCQ4 technique is proposed in [6].
- **EILQ**. Enhanced IL-Quadtree by object partition technique.

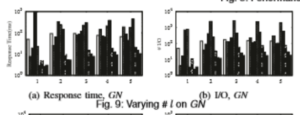
To evaluate the efficiency of batch algorithm, the following algorithms are evaluated for the BTOPK-SK query.

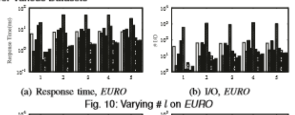
- **I-ILQ**. The IL-Quadtree based TOPK-SK algorithm.

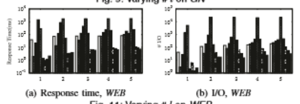
5.1 Evaluating TOPK-SK query

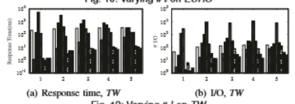


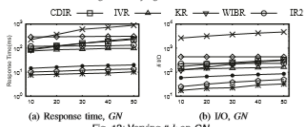
(a) Response Time (b) Index Size (c) Index Construction Time
Fig. 8: Performance over Various Datasets


(a) Response time, GN (b) I/O, GN
Fig. 9: Varying # l on GN

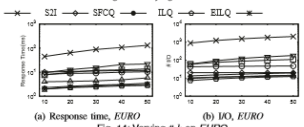

(a) Response time, EURO (b) I/O, EURO
Fig. 10: Varying # l on EURO


(a) Response time, WEB (b) I/O, WEB
Fig. 11: Varying # l on WEB


(a) Response time, TW (b) I/O, TW
Fig. 12: Varying # l on TW



(a) Response time, GN (b) I/O, GN
Fig. 13: Varying # k on GN


(a) Response time, EURO (b) I/O, EURO
Fig. 14: Varying # k on EURO

5.2Evaluating BTOPK-SK Query

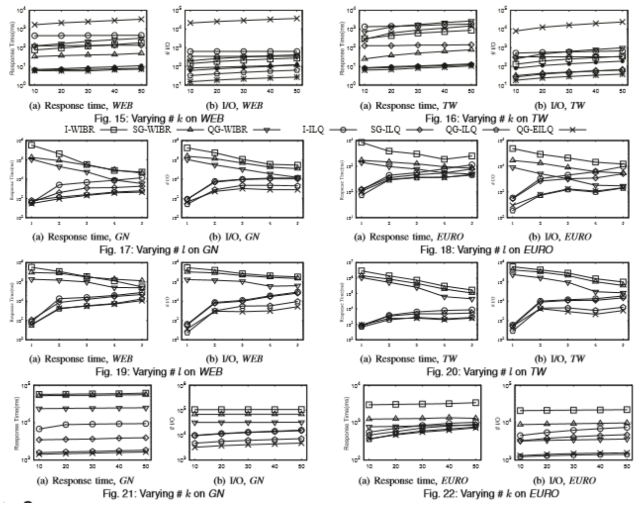


Fig. 15: Varying # *k* on *WEB*
Fig. 16: Varying # *k* on *TW*
Fig. 17: Varying # *l* on *GN*
Fig. 18: Varying # *l* on *EURO*
Fig. 19: Varying # *l* on *WEB*
Fig. 20: Varying # *l* on *TW*
Fig. 21: Varying # *k* on *GN*
Fig. 22: Varying # *k* on *EURO*

6 CONCLUSIONS

The *spatio-textual* objects are used in a wide range of applications. Hence the problem of top k spatial search is important which uses this technique. In the paper, we propose a novel index structure, namely IL-Quadtree, to organize the *spatio-textual* objects. An efficient algorithm is developed to support the top k spatial keyword search by taking advantage of the IL-Quadtree. We further propose a partition based method to enhance the effectiveness of the signature of linear quadtree. To facilitate a large amount of spatial keyword queries, we propose a BTOPK-SK algorithm as well as a query group algorithm to enhance the performance of the system. The main of our project convincingly demonstrate the efficiency of our techniques.

REFERENCES

[1] Y. Zhou, X. Xie, C. Wang, Y. Gong, and W.-Y. Ma, "Hybrid index structures for location-based web search," in *CIKM*, 2005.
[2] I. D. Felipe, V. Hristidis, and N. Rishe, "Keyword search on spatial databases," in *ICDE*, 2008.
[3] S. Ding, J. Attenberg, R. A. Baeza-Yates, and T. Suel, "Batch query processing for web search engines," in *Proceedings of the Forth International Conference on Web Search and Web Data Mining, WSDM 2011, Hong Kong, China, February 9-12, 2011*, 2011, pp. 137–146. [Online]. Available: http://doi.acm.org/10.1145/1935826.1935858
[4] D. Wu, M. L. Yiu, G. Cong, and C. S. Jensen, "Joint top k spatial keyword query processing," *TKDE*, 2011.
[5] G. R. Hjaltason and H. Samet, "Distance browsing in spatial databases." *TODS*, vol. 24, no. 2, pp. 265–318, 1999.
[6] M. Christoforaki, J. He, C. Dimopoulos, A. Markowetz, and T. Suel, "Text vs. space: efficient geo-search query processing," in *CIKM*, 2011.
[7] J. B. Rocha-Junior, O. Gkorgkas, S. Jonassen, and K. Nørv°ag, "Efficient processing of top-k spatial keyword queries," in *SSTD*, 2011.
[8] R. Hariharan, B. Hore, C. Li, and S. Mehrotra, "Processing spatial-keyword (sk) queries in geographic information retrieval (gir) systems," in*SSDBM*, 2007.
[9] G. Cong, C. S. Jensen, and D. Wu, "Efficient retrieval of the top-k most relevant spatial web objects," *PVLDB*, vol. 2, no. 1, 2009.
[10] D. Wu, G. Cong, and C. S. Jensen, "A framework for efficient spatial web object retrieval," *VLDB J.*, 2012.
[11] A. Cary, O. Wolfson, and N. Rishe, "Efficient and scalable method for processing top-k spatial boolean queries," in *SSDBM*, 2010, pp. 87–95.
[12] 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 Trans.Knowl. Data Eng.*, vol. 23, no. 4, pp. 585–599, 2011.
[13] D. Zhang, K.-L. Tan, and A. K. H. Tung, "Scalable top-k spatial keyword search," in *EDBT*, 2013, pp. 359–370.
[14] G. Li, J. Feng, and J. Xu, "Desks: Direction-aware spatial keyword search," in *ICDE*, 2012.
[15] S. B. Roy and K. Chakrabarti, "Location-aware type ahead search on spatial databases: semantics and efficiency," in *SIGMOD Conference*, 2011.
[16] J. B. Rocha-Junior and K. Nørv°ag, "Top-k spatial keyword queries on road networks," in *EDBT*, 2012.
[17] C. Zhang, Y. Zhang, W. Zhang, X. Lin, M. A. Cheema, and X. Wang, "Diversified spatial keyword search on road networks," pp. 367–378, 2014.
[18] I. Gargantini, "An effective way to represent quadtrees," *Commun. ACM*, vol. 25, no. 12, pp. 905–910, 1982.
[19] G. M. Morton, "A computer oriented geodetic data base and a new technique in file sequencing," *Technical Report, Ottawa, Canada: IBM Ltd*, 1966.
[20] C. Faloutsos, "Multiattribute hashing using gray codes," in *SIGMOD Conference*, 1986.