International Journal of Engineering Science Invention
ISSN (Online): 2319 – 6734, ISSN (Print): 2319 – 6726
www.ijesi.org ||Volume 4 Issue 9|| September 2015 || PP.06-09

Spatial Approximate String Keyword content Query processing

[1]S.Geetha , [2]A.Saranya
[1]*Assistant Professor , Department Of Information Technology And Computer Application Shrimati Indira Gandhi College,Trichy-2.*
[2]*Research Scholar Department of computer science Shrimati Indira Gandhi College, Trichy-2.*

ABSTRACT : *Web users and content are increasingly being geo-positioned, and increased focus is being given to serving local content in response to web queries. This development calls for spatial keyword queries that take into account both the locations and textual descriptions of content. We study the efficient, joint processing of multiple top-k spatial keyword queries. Such joint processing is attractive during high query loads and also occurs when multiple queries are used to obfuscate a user's true query. To propose a novel algorithm and index structure for the joint processing of top-k spatial keyword queries. Empirical studies show that the proposed solution is efficient on real datasets. They also offer analytical studies on synthetic datasets to demonstrate the efficiency of the proposed solution.*

Key words: *H.2.4 k Spatial database, H.2.4 n Textual database.*

I. INTRODUCTION

A range of technologies combine to afford the web and its users a geographical dimension. Geo-positioning technologies such as GPS and Wi-Fi and cellular geo-location services, e.g., as offered by Skyhook, Google, and Spotigo, are being used increasingly; and different geo-coding technologies enable the tagging of web content with positions. Studies suggest that some 20% of all web queries from desktop users exhibit local intent, i.e., query for local content. The percentage is likely to be higher for mobile users. This renders so-called *spatial keyword queries* important. Such queries take a location and a set of keywords as arguments and return the content that best matches these arguments. Spatial keyword queries are important in local search services such as those offered by Google Maps and a variety of yellow pages, where they enable search for, e.g., nearby restaurants or cafes that serve a particular type of food. Travel sites such as Trip Advisor and Travellers Point may use spatial keyword queries to find nearby hotels with particular facilities. As an example, in Figure 1, a service provider's database D stores the spatial locations and textual descriptions (sets of keywords) of restaurants $p1$, $p2$, $p3$, and $p4$. For instance, restaurant $p1$ is described by the keywords: 'pizza' and 'grill.'
User $q1$ (shown as a shaded dot) issues the following query: Find the nearest restaurant that serves 'curry' and 'sushi.' The service returns restaurant $p2$, which is the closest one that contains all the keywords in $q1$.

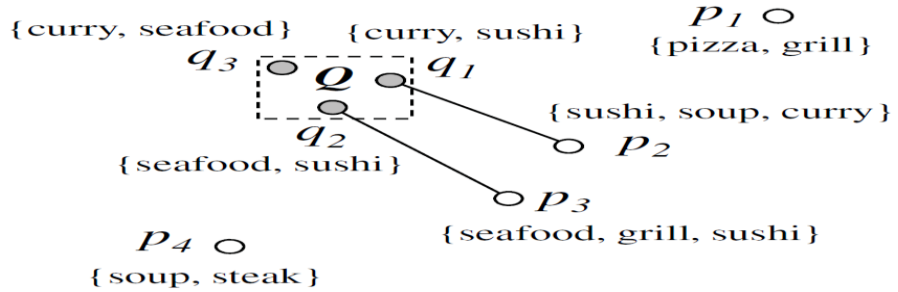
ALGORITHM 1: ITERATE (Joint query *Q*, Tree root *root*, Integer *k*)

For each subquery *qi* do
$V_i \leftarrow$ new max-priority queue; ►maintain the top *k* objects
Initialize V_i with *k* null objects with distance ∞;
$U \leftarrow$ new min-priority queue;
U.Enqueue(*root* , 0);
while *U* is not empty do
$e \leftarrow U$.Dequeue();
if *e* is an object then
update V_i by (e, $dist(q_i.\lambda, e.\lambda)$);
if V_i has *k* non-null objects then
break the while-loop;
else ►*e* points to a child node
read the node *CN* of *e*;
read the posting lists of *CN* for keywords in $q_i.\psi$;
for each entry *e'* in the node *CN* do
if $q_i.\psi \leq e'.\psi$ then
U.Enqueue(*e'*, $mindist(q_i.\lambda, e'.\lambda)$);
return $\{V_i\}$; ►top-*k* results of each subquery

ALGORITHM 2 GROUP (Joint query *Q*, Tree root *root*, Integer *k*)

$U \leftarrow$ new min-priority queue;
U.Enqueue(*root* , 0);
for each subquery $q_i \in Q$ do
$V_i \leftarrow$ new max-priority queue; ►maintain the top k objects
initialize V_i with k null objects with distance ∞;
let $q_i.\tau$ be the maximum distance in V_i;
while U is not empty do
$e \leftarrow U$.Dequeue(); ►phase I: checking dequeued entry
if $e.key \geq Q.\tau$ or $|Q.\psi \cap e.\psi| < Q.m$ then ►Rules 1, 2
continue the while-loop;
$Q^* \leftarrow \{q_i \in Q \mid mindist(q_i.\lambda, e.\lambda) \leq q_i.\tau \wedge q_i.\psi \leq e.\psi\}$;
if Q^* is empty then ►Rule 3
continue the while-loop;
$e.key \leftarrow \min_{qi \in Q^*} mindist(q_i.\lambda, e.\lambda)$;
if $e.key$ has increased (in line 14) then ►update key
U.Enqueue(e, $e.key$);
continue the while-loop;
read the child node CN of e; ►phase II: processing child node
read the posting lists of CN for keywords in $Q^*.\psi$;
if CN is a non-leaf node then
for each entry e' in the node CN do
if $|Q^*.\psi \cap e\_.\psi| \geq Q^*.m$ and $mindist(Q^*.\lambda, e\_.\lambda) <$
$Q^*.\tau$ then ►Rules 1,2
$Q\_ \leftarrow \{qi \in Q^* \mid mindist(q_i.\lambda, e\_.\lambda) \leq q_i.\tau \wedge$
$qi.\psi \leq e'.\psi\}$;
if Q' is not empty then ►Rule 3
U.Enqueue(e', $\min_{qi \in Q'} mindist(q_i.\lambda, e'.\lambda)$);
else ►CN is a leaf node
for each object p in the leaf node CN do
if $|Q^*.\psi \cap p.\psi| \geq Q^*.m$ and $mindist(Q^*.\lambda, p.\lambda) <$
$Q^*.\tau$ then ►Rules 1, 2
for each subquery $q_i \in Q^*$ such that
$dist(q_i.\lambda, p.\lambda) < q_i.\tau$ and $q_i.\psi \leq p.\psi$
do
update V_i by (p, $dist(q_i.\lambda, p.\lambda)$);
return $\{V_i\}$; ►top-k results of each subquery

Top-K Spatial Keyword Queries Diagram



A top-k spatial keyword query retrieves k objects that are closest to the query location and that contain all the argument keywords. To study the joint processing of sets of such queries. Consider again Figure 1 where the dashed rectangle represents a theater. After a concert, users $q1$, $q2$, and $q3$ each wish to find a nearest restaurant (k = 1) that matches their preferences. User $q1$ prefers 'curry' and 'sushi,' user $q2$ prefers 'seafood' and 'sushi,' and user $q3$ prefers 'curry' and 'seafood.' The result returned to $q1$ is $p2$, as this is the nearest object that contains all keywords of this user's query. Similarly, the result returned to $q2$ is $p3$, and the result returned to $q3$ is empty, as no object contains 'curry' and 'seafood. These three queries can be processed jointly as a single joint top-k spatial keyword query Q that consists of three sub queries. The joint processing is motivated by two main applications. Application I: Multiple Query Optimization.

Existing work can be roughly divided into three categories. *Grouping / partitioning a large set of queries.* Papadopoulos et al. use a space filling curve to group queries so as to improve the overall performance of processing all queries. Zhang et al. Study the processing of multiple nearest neighbor queries; they propose R-tree-based solutions and heuristics for the grouping of queries. However, the joint query we consider is actually one group of queries. We thus aim to compute one group of queries efficiently. Any grouping/partitioning approach can be applied before our algorithm.

Cache the result objects as well as the index that supports these objects as the results so as to optimize query response time. Consider a scenario where objects are stationary while queries are mobile. They develop a semantic caching scheme that records a cached item (object) as well as its valid range. However, their approach cannot be directly applied to our problem because their valid range concept ignores the query keywords in our problem. In our experiments, traditional caching, i.e., LRU buffering, is considered as a competitor to our proposals.

II. PROPOSED ALGORITHMS

To present the existing IR-tree and then proceed to develop a basic and an advanced algorithm, in Sections 3.2 and 3.3, respectively, for processing joint top*k* spatial keyword queries. The algorithms are generic and are not tied to a particular index.

Preliminaries: the IR-Tree

The IR-tree, which to use as a baseline, is essentially an R-tree extended with inverted files. The IR-tree's leaf nodes contain entries of the form (*p, p.λ, p.di*), where *p* refers to an object in dataset *D*, *p.λ* is the bounding rectangle of *p*, and *p.di* is the identifier of the description of *p*. Each leaf bnode also contains a pointer to an inverted file with the text descriptions of the objects stored in the node. An inverted file index has two main components. • A vocabulary of all distinct words appearing in the description of an object. • A posting list for each word *t* that is a sequence of identifiers of the objects whose descriptions contain *t*. Each non-leaf node *R* in the IR-tree contains a number of entries of the form (*cp, rect, cp.di*) where *cp* is the address of a child node of *R*, *rect* is the MBR of all rectangles in entries of the child node, and *cp. di* is the identifier of a pseudo text description that is the union of all text descriptions in the entries of the child node. As an example, contains eight spatial objects *p*1, *p*2, . . . , *p*8, and Figure 2b shows the words appearing in the description of each object. Figure 3a illustrates the corresponding IR-tree, and Figure 3b shows the contents of the inverted files associated with the nodes.

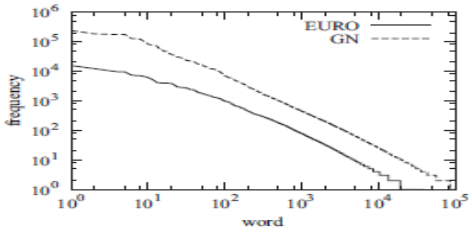
III. EXPERIMENTAL STUDY

To use two real datasets for studying the robustness and performance of the different approaches. Dataset "EURO" contains points of interest (e.g., pubs, banks, cinemas) in Europe (www.pocketgpsworld.com). Dataset "GN" is obtained from the U.S. Board on Geographic Names (geonames.usgs.gov). Here, each object has a geographic location and a short description.

(a) offers details on EURO and GN. The spatial domain of each dataset is normalized to the unit square
(b) plots the word frequency distributions in the two datasets.

Dataset	EURO	GN
# of objects	162,033	1,868,821
# of distinct words	35,315	222,407
Average # of words per object	18	4

(a) Dataset Details



(b) Word Frequencies

Effect of Different Types of Keywords

This experiment studies the performance of the solutions for different types of query keywords:

• Random keywords (R-key), picking a random data object and then random words of the object as the query keyword set, as mentioned earlier.

• Partitioning keywords (P-key): picking words that are used for the partitioning of objects in the W-IBR-tree.

• Non-partitioning keywords (NP-key): picking words that are not used for the partitioning of objects in the W-IBRtree. The total I/O costs of the methods for these three types of keywords. The W-IBR-tree outperforms the CDIBR-tree using R-key and P-key, while they have comparable performance for NP-key. The W-IBR-tree beats the Inverted- R-trees and the IR2-tree for all three types of keywords. R key uses randomly generated keys and does not favor any particular index. In this setting, the W-IBR-tree achieves better performance than the other indexes. Effect of Overlap Among Query Keywords. This experiment evaluates the effect of overlap among keywords in sub queries. A higher overlap means that the sub queries share more common keywords, while a low overlap means sub queries tend to have different keywords.

IV. CONCLUSIONS AND FUTURE WORK

This paper introduces the joint top-*k* spatial keyword query and presents efficient means of computing the query. Our solution consists of the W-IBR-tree that exploits keyword partitioning and inverted bitmaps for indexing spatial keyword data, and (ii) the GROUP algorithm that processes multiple queries jointly. In addition, to describe how to adapt the solution to existing index structures for spatial keyword data. Empirical studies with combinations of two algorithms and a range of indexes demonstrate that the GROUP algorithm on the W-IBR-tree is the most efficient combination for processing joint top-*k* spatial keyword queries.

It is straightforward to extend our solution to process top-*k* spatial keyword queries in spatial networks. To take advantage of Euclidean distance being a lower bound on network distance. While reporting the top-*k* objects incrementally, if the current object is farther away from the query in terms of Euclidean distance than is the *kth* candidate object in terms of network distance, the algorithm stops and the top-*k* result objects in the spatial network are found. The network distance from each object to a query can be easily computed using an existing, efficient approach. An interesting research direction is to study the processing of joint moving queries, which is useful in environments with continuously moving users.

References

[1] A. Guttman. R-trees: a dynamic index structure for spatial searching. In *SIGMOD*, pp. 47–57, 1984.

[2] R. Hariharan, B. Hore, C. Li, and S. Mehrotra. Processing spatial-keyword (SK) queries in geographic information retrieval (GIR) systems. In *SSDBM*, p. 16, 2007.

[3] G. R. Hjaltason and H. Samet. Distance browsing in spatial databases. *ACM TODS*, 24(2):265–318, 1999.

[4] M. Hong, M. Riedewald, C. Koch, J. Gehrke, and A. J. Demers. Rule-based multi-query optimization. In *EDBT*, pp. 120–131, 2009.

[5] H. Hu, J. Xu, W. S. Wong, B. Zheng, D. L. Lee, and W. C Lee. Proactive caching for spatial queries in mobile environments. In *ICDE*, pp. 403–414, 2005.

[6] P. Kalnis and D. Papadias. Multi-query optimization for online analytical processing. *Inf. Syst.*, 28(5):457–473, 2003.

[7] L. Kaufman and P. J. Rousseeuw. *Finding groups in data: an introduction to cluster analysis*. New York: Wiley, 1990.

[8] H. Kido, Y. Yanagisawa, and T. Satoh. An anonymous communication technique using dummies for location-based services. In *IEEE Conf. on Pervasive Services*, pp. 88–97, 2005.

[9] H. Lu, C. S. Jensen, and M. L. Yiu. PAD: privacy-area aware, dummy-based location privacy in mobile Services. In *MobiDE*, pp. 16–23, 2008.

[9] B. Martins, M. J. Silva, and L. Andrade. Indexing and ranking in geo-IR systems. In *GIR*, pp. 31–34, 2005.

[10] M. Murugesan and C. Clifton. Providing privacy through plausibly deniable search. In *SDM*, pp. 768–779, 2009.

[11] S. Naranan and V. K. Balasubrahmanyan. Models for power law relations in linguistics and information science. *Journal of Quantitative Linguistics*, 5(1-2):35–61, 1998.

[12] A. Papadopoulos and Y. Manolopoulos. Multiple range query optimization in spatial databases. In *ADBIS*, pp. 71–82, 1998.

[13] P. Roy, S. Seshadri, S. Sudarshan, and S. Bhobe. Efficient and extensible algorithms for multi query optimization. In *SIGMOD*, pp. 249–260, 2000.

[14] F. Saint-Jean, A. Johnson, D. Boneh, and J. Feigenbaum. Private web search. In *WPES*, pp. 84–90, 2007.

[15] H. Samet, J. Sankaranarayanan, and H. Alborzi. Scalable network distance browsing in spatial databases. In *SIGMOD*, pp. 43–54, 2008.

[16] M. Sanderson and J. Kohler. Analyzing geographic queries. In [22] T. K. Sellis. Multiple-query optimization. *ACM TODS*, 13(1):23–52, 1988.

[17] Y. Tao, J. Zhang, D. Papadias, and N. Mamoulis. An efficient cost model for optimization of nearest neighbor search in low and medium dimensional spaces. *IEEE TKDE*, 16(10):1169–1184, 2004.

[18] Y. Theodoridis and T. K. Sellis. A model for the prediction of R-tree performance. In *PODS*, pp. 161–171, 1996.