STQP: Spatio-Temporal Indexing and Query Processing

Nikhat Fatima
Department. of CSE,
Deccan College of Engineering and Technology,
Darussalam, Hyderabad, T.S

Ayesha Ameen
Associate Professor IT Dept.,
Deccan College of Engineering and Technology,
Darussalam, Hyderabad, T.S

Syed Raziuddin, PhD
Professor & Head of CSE Dept.,
Deccan College of Engineering and Technology,
Darussalam, Hyderabad, T.S

ABSTRACT

In this ongoing work, the location-aware ranking query (LRQ) are considered, an important category of location-aware query. Types of location-aware ranking query are the k-nearest neighbour (NN) query and location-aware keyword query(LKQ). NN LKQs and inquiries have vast applications in many domains. However, there are a great number of location-aware datasets that demand better and flexible location aware rank queries. They are a lot more complex than spatio-textual objects. These things are termed as location-aware things. For location-aware things, simple NN LKQs and queries may well not be expressive enough to find the objects of interests. In this particular proposed work the generic location-aware rank query is formulated, which retrieves the objects satisfying a query predicate, ranks and returns the full total results predicated on spatial proximity, textual relevance's and measures extracted from attribute values. We create a construction called location aware indexing and query processing(LINQ), for useful indexing and querying of GLRQs. LINQ evolves the synopses tree to work with synopses of non-spatial features, and combines the synopses tree with other indexes to query and index the GLRQ. The global buckets can be used to provide efficiency and faster computation time by using Bin sort algorithm this proposed method is recognized as STQP. The increased proposed system will provide better results with respect to faster and output for spatial query results.

Keywords

Location Aware, Query, Synopsys Tree

1. INTRODUCTION

With all the proliferation and popular adoption of mobile telephony, it is increasingly more convenient for users to fully capture and submit geo-locations. As a result, increasingly more location-aware datasets have been made and created on the Web. For instance, Flickr, one of the primary photo-sharing website, has an incredible number of geo-tagged items every full month. The popularity and large scale of the location-aware datasets make location-aware queries important.

In the ongoing work, an important class of location-awarequery techniques is considered and explored which is termed as Location-aware Ranking Query (LRQ), an important course of location-aware query. Types of location-aware list query are the k-nearest neighbour (NN) query and location-aware keyword query. NN questions and LKQs have extensive applications in many domains. However, there are always a complete great deal of location-aware datasets that demand better and adaptable location aware ranking concerns. For instance on Yelp, the restaurant, Yelp gives its location "1429 Mendel St, SAN FRANCISCO BAY AREA, CA 94124", categories "Soul Food, American Traditional, Music Venues", rating 4.5 celebrities as well as the true quantity of reviews 150. Another example is the photographs on Flickr, where as well as the geo-location and text, each image also has some numeric attributes, like the range of views, the real volume of favourites, the true variety of comments, etc. The restaurants on photographs and Yelp on Flickr are mixtures of location, text and other styles of information. They may be much more sophisticated than spatio-textual things. These items are termed as location-aware things. For location-aware items, simple NN LKQs and queries might not exactly be expressive enough to find the objects of interests. For instance, on the restaurant dataset, through LKQs, one will discover the most nearest and relevant restaurants. But you can desire to find the nearest and relevant restaurants gratifying certain conditions, such as no-smoking, healthy, or top-ranked. Within the Flickr dataset, users might want to fetch the relevant and nearest photographs that are highly-rated. On these datasets, people may decide to search by not only location and keywords, but conditions on other attributes also

In prior work Location-aware list query is recognized as an important school of query, and many subsets have been examined. Nearest neighbour query is the most well-known LRQ. It requires a spatial location as suggestions and outputs the closest items in the dataset. The prevailing algorithms use index buildings plus some pruning solutions to limit the search space. Several NN algorithms have been suggested, where in fact the best-first algorithm is one of the very most influential methods, which proves to attain the optimum I/O performance. Location-aware keyword query is another school of LRQ. The essential LKQ will not contain any predicates, and has a standing function incorporating spatial proximity and textual relevance. There are a few other variations of LKQs also. One variant specifies a spatial region, which restricts the locations of the full total results. Another variant interprets the keywords as Boolean conditions. That is, it retrieves the items including all the keywords, and ranks the results just in line with the spatial distances.

In this suggested work the universal location-aware list query is formulated, which retrieves the things gratifying a query predicate, rates and earnings the full total results predicated on spatial proximity, textual relevance's and steps obtained from feature values. A platform called Location aware Indexing and Query processing(LINQ), for useful indexing and querying of Generic Location Aware Rank Queries(GLRQs). LINQ evolves the synopses tree to work with synopses of non-spatial characteristics, and combines the synopses tree with other indexes to query and index the GLRQ. The experiments on real datasets and the results demonstrate the effectiveness and efficiency of the method. The enhancement of this proposed system is to add temporal data as yet another attribute by which the ranking function can be increased and can provide reliable scores over time frame. The global buckets can even be used to provide efficiency and faster computation time by using Bin variety algorithm and this suggested method is recognized as

STQP. The increased proposed system shall provide better results with respect to faster output for spatial query results.

2. PREVIOUS WORKS

Driven in part by the emergence of the mobile Internet, the conventional Internet is acquiring a geo-spatial dimension. On the one hand, many (geo-referenced) points of interest e.g., stores, tourist attractions, hotels, entertainment services, public transport, and public services are being associated with descriptive text documents. On the other hand, web documents are increasingly being geo-tagged. This fusion of geo-location and documents enables queries that take into account both location proximity and text relevancy. One study has found that about one fifth of web search queries are geographical and have local intent, as determined by the presence of geographical terms such as place names and postal codes. Indeed commercial search engines have started to provide location based services, such as map services, local search, and local advertisements. For example, Google Maps supports location-aware text retrieval queries. Additional examples of location-based services include online yellow pages.

The R-tree is arguably the dominant index for spatial queries, and the inverted file is the most efficient index for text information retrieval. These were developed separately and for different kinds of queries. We aim to develop an approach that is able to leverage both techniques for the efficient processing of L*k*T queries. To achieve this goal, a simple approach is to use the inverted file to generate a number of top candidate objects based on text relevancy and then compute the spatial distances (resp. text relevancy) of the candidate objects using the other index. However, this approach is not efficient since there is no sensible way to determine the number of candidate objects needed from the first step in order to ensure that *k* top-*k* objects are found in the end. Instead, we propose a hybrid indexing structure, the IRtree that utilizes both indexing structures in a combined fashion. The IR-tree is essentially an R-tree, each node of which is enriched with reference to an inverted file for the objects contained in the sub-tree rooted at the node. In the IR-tree, a leaf node *N* contains a number of entries of the form (*O; rectangle(O:di*), where *O* refers to an object in database *D*, *rectangle* is the bounding rectangle of object *O*, and *O:di* is the identifier of the document of object *O*. A leaf node also contains a pointer to an inverted file for the text documents of the objects being indexed. The inverted file is stored separately, for two reasons: First, it is more efficient to store each inverted file contiguously, rather than as a sequence of blocks or pages that are scattered across a disk. Second, the inverted file can be distributed across several machines while this is not easily possible for the R-tree. [2]

Let *D* be a database. Each spatial web object *O* in *D* is defined as a pair $(O.\lambda,O.\psi)$, where $O.\lambda$ is a location descriptor in multidimensional space and $O.\psi$ is a document (e.g., a dining menu) that describes the object (e.g., an Italian restaurant).A two-dimensional geographical space composed of latitude and longitude is assumed, but the paper's proposals generalize to other multidimensional spaces of low dimensionality. Intuitively, a *Location-aware top-k Text retrieval* (L*k*T) query retrieves *k* objects in database *D* for a given query *Q* such that their locations are the closest to the location specified in *Q* and their textual descriptions are the most relevant to the keywords in *Q*. Formally, given a query *Q*, defined as a pair $(Q.\lambda, Q.\psi)$, where $Q.\lambda$ is a location descriptor and $Q.\psi$ is a set of keywords, the objects returned are ranked according to a ranking function given by: $f\ (D\varepsilon(Q.\lambda,\ O.\lambda),\ P(Q.\psi|\ O.\psi))$, where $D\varepsilon(Q.\lambda,\ O.\lambda)$ is the Euclidian distance between *Q* and *O* and $P(Q.\psi|O.\psi)$ is the text relevancy of $O.\psi$ with regard to $Q.\psi$. The text relevancy can be computed as the probability of generating query $Q.\psi$ from the language models of the documents or other text models. We tackle the problem of efficiently answering L*k*T queries. Thus, given a query *Q*, we retrieve a ranked list of *k* objects according to their ranking scores as computed by the ranking function $f\ (\cdot,\ \cdot)$ introduced above. The paper's proposals are applicable to a wide range of ranking functions, namely all functions that are monotone with respect to distance proximity and text relevancy.

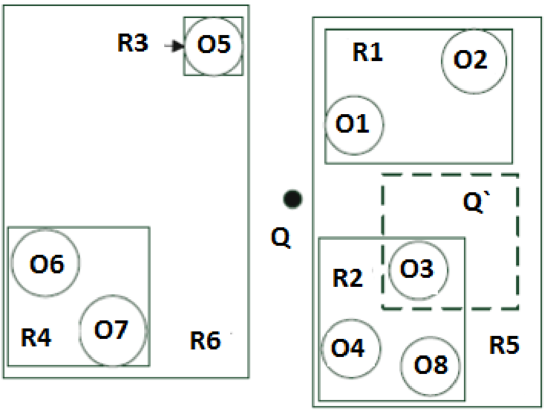

Figure 1: Objects and Bounding Rectangles

Table 1: Document by Term Matrix

	English	Hindi	Restaurant	Food
O1	0.5		0.5	
O2		0.5	0.5	
O3	0.7			0.1
O4			0.7	0.1
O5	0.4		0.4	
O6		0.4	0.3	
O7	0.1	0.1	0.4	0.1
O8		0.3	0.3	

The IR-tree is essentially an R-tree, each node of which is enriched with a reference to an inverted file for the objects contained in the sub tree rooted at the node. In the IR-tree, a leaf node contains a number of entries of the form *(O, O.r)*, where *O* refers to an object in database *D* and *O.r* is the bounding rectangle of object *O*. A leaf node also contains a pointer to an inverted file for the text documents of the objects being indexed. The inverted file is stored separately from the R-tree, for two reasons: First, it is more efficient to store each inverted file contiguously, rather than as a sequence of blocks or pages that are scattered across a disk. Second, the inverted file can be distributed across several machines, while this is not easily possible for the R-tree.

An inverted file consists of the following two main components.

- A vocabulary of all distinct terms in a collection of documents.
- A set of posting lists, each of which relates to a term *t*.

Each posting list is a sequence of pairs *O,w_*, where *O* refers to an object whose document *O.ψ* contains term *t*, and *w* is the weight of term *t* in document *O.ψ*. A non-leaf node *N* contains a number of entries of the form *(e, e.r)* where *e* points to a child node of *N* and *e.r* is the Minimum Bounding Rectangle (MBR) of all rectangles in entries of the child node. A pseudo document is constructed for each non-leaf entry in the IR-tree.

The pseudo document is an important concept in the IR-tree. It represents all documents in the entries of the child node, enabling us to estimate a bound on the text relevancy to a query of all documents contained in the sub tree rooted at *e*. The weight of a term *t* in the pseudo document referenced by *e* is the maximum weight of *t* in the documents contained in the sub tree rooted at node *e*. A non-leaf node *N* also contains a pointer to an inverted file for the pseudo documents of the entries stored in *N*. [3]

3. PROPOSED SYSTEM

3.1 Architecture

The user query Q is a query predicate of the form P1, P2, ….. Pn, where Pi is a simple selection predicate specified on a single attribute. The other component Q:f is a ranking function. In module tree of synopses is built. The histograms are constructed in holistic way. A bucket in the global histograms stands for a hyper-rectangle within the domain of data. The synopses tree summarizes the distribution of numeric attributes.

To answer the GLRQ, we need to combine synopses tree with other indexes on locations and texts. The state-of-the-art index structures supporting locations and texts are the IR-tree family. The basic structure of IR-tree index is an R-tree, where each entry is associated with an inverted file. The synopses tree can be easily combined with IR-tree index

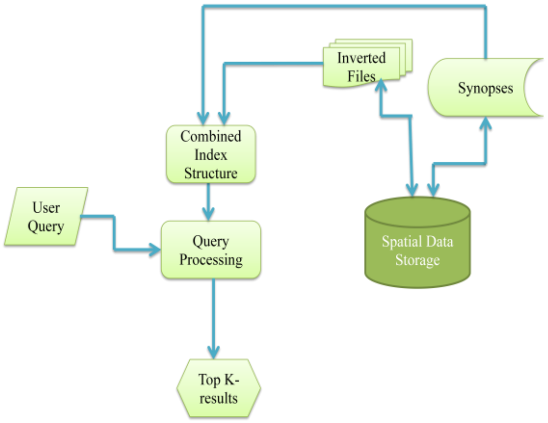

Figure 2: Architecture

3.2 Data Initialization

In this module initialization is done by denoting location-aware object O as a triple < λ, W, A>, where O: λ is a location descriptor, O: W is a set of keywords, and O: A {O:A1; O:A2; …… } is a set of attributes. We use O:Ai to denote the value of O on attribute Ai. The attributes in O:A are numeric attributes. The locations of the objects are implied by their positions on the plane, and the attributes and keywords are Retrieved. Formally, in a GLRQ Query Q:P is a query predicate of the form P1, P2, ….. Pn, where Pi is a simple selection predicate specified on a single attribute. The other component Q:f is a ranking function. Each attribute, including location and text, has a scoring function, measuring the "goodness" of an object in terms of the attribute. The scoring function on location returns spatial proximity, whereas the function on text measures the textual relevance.

3.3 Factorization and Buckets

After In this module histograms are used for synopses, as more than one attributes are concerned, we build multi-dimensional histograms to summarize the datasets. Multi-dimensional (nD) histogram has been widely used to summarize multi-dimensional data. Basically, an nD histogram is obtained by partitioning the multi-dimensional domain into a set of hyper-rectangular buckets, and then storing summary information for each bucket. As a synopsis, nD histogram is able to provide accurate approximation, but it is very expensive to construct and maintain an nD histogram. To reduce the complexity of nD histogram, nD factor data distribution into two-dimensional (2D) distributions is used. The modeling of a dataset generates a model called junction tree. The junction tree is a tree structure where each node is a pair of attributes. Then, according to the junction tree, the joint distribution of multiple attributes can be factorized into a set of 2D distributions.

In this module tree of synopses is built. If the nD-histograms in the synopses tree are constructed independently, the cost would be too high. It is proposed to construct the synopses with reduced cost. The datasets of different nodes may be similar or overlapping. The dataset of a node at a higher level is a superset of that of its descendants. Considering the characteristics, we construct the histograms in a holistic way. Specifically, we construct a set H { H1, H2, …….} of global histograms for the whole dataset D, where Hi is a global histogram, and Bij is a bucket the histogram Hi. The set of global histograms is used by all entries in the synopses tree. Let B be the set of buckets in all histograms. For any entry e, an array b elements is maintained, where each element in the array is the statistics about the D in a bucket. Thus the array records the summary statistics of e:D. By using global histogram buckets, the construction cost and storage overhead of histograms are decreased. This is because the global histograms are constructed and stored only once. For each entry in a non-leaf node, only some local statistics are kept.

3.4 Compact Information and Synopsys Tree

In A bucket in the global histograms stands for a hyper-rectangle within the domain of data. To approximate the data points falling in the hyper-rectangle, some statistics about the distribution of data in the rectangle are needed. Only one bit kept for a bucket, indicating whether the bucket is empty or not. To improve the accuracy of estimation, more bucket information is needed. In this module, we use a compact bit-based representation of the local information in each bucket. We split each bucket into M partitions, and then stores a M-bit string where each bit is 1 if the corresponding partition is not empty or 0 otherwise. We keep the splitting information, and the correspondence between the bits and the partitions. As all the buckets are split in the same way, this information can be stored only once as background information.

The synopses tree summarizes the distribution of numeric attributes. It is able to address part of the GLRQ processing problem. To answer the GLRQ, we need to combine synopses tree with other indexes on locations and texts. The state-of-the-art index structures supporting locations and texts are the IR-tree family. The basic structure of IR-tree index is an R-tree, where each entry is associated with an inverted file. Our synopses tree can be easily combined with IR-tree index. Each entry in the R-tree is associated with two additional components: the inverted file from the IR-tree, and the synopsis from the synopses tree. The inverted files and the synopses are stored separately from the R-tree. This structure is flexible in that the structure of the R-tree is not influenced by other parts, and the R-tree can be queried alone, with or without the inverted files and the synopses.

Algorithm STQP

Input: Dataset.

Output: Top-k results.

```
1.   Extract attributes

2.   Compute the ranking function
       d ← w1.prox(O.Q)+ w2(O.Q)+w3(O.rating)+w4(O.health) +w5(O.time)

3.   Create and factorize multidimensional histograms.

4.   Divide each histograms into buckets.
5.   Sort the buckets using Bin-sort algorithm.
6.   Create inverted indices.
7.   Create a synopses tree.
8.   Process the STQP query.
9.   Return the top-k result.
```

3.5 Query Processing and STQP

In this module Query processing algorithm is implemented which exploits the best-first strategy to search the combined index. In this priority queue is used to keep track of the nodes and objects to explore in decreasing order of their scores. A maximum matching score of entry e to query Q, which are used as the keys of entries in the queue. TopK keeps the current top-k results. The R-tree is traversed in a top-down manner. At each node N, if N is an internal node instead of a leaf, for each entry e in node N, the algorithm estimates maxf. If maxf > 0 , it implies that there may be objects enclosed by e satisfying the query predicate, so e with maxf is added to the queue. If N is a leaf, it computes the score of each entry, and pushes the entries with non-zero scores to the queue. If N is an object, N is directly reported as a top-k result. The algorithm terminates if the top-k results have been found. Given a query Q and a node N in the R-tree, we compute a metric maxf, which offers an upper bound on the actual scores of the objects enclosed by N with respect to Q. Conceptually, the synopsis associated with a node N as a multi-dimensional space (called data space of N) consisting of hyper-rectangles. Similarly, a query Q can be considered as a set of hyper-rectangles (called query space of Q) encompassing the points satisfying the query predicate

In this module Spatio Temporal Query processing method is implemented which exploits to include temporal data as an additional attribute through which the ranking function can be enhanced and will provide reliable scores over period of time. The global buckets can also be used to provide efficiency and faster computation time by using Bin sort algorithm this proposed method is known as STQP. Best-first strategy is used to search the combined index. This enhancement works the same way as previous module of query processing.

4. RESULTS

The concept of this paper is implemented using comments retrieved from Facebook API. Different results are shown below; The proposed paper is implemented in Java technology on Intel Core i3 Processor with minimum 20 GB hard-disk and 1GB RAM. Extensive experiments with UK location database are conducted.

The performance of the query methods is evaluated based on their efficiency in terms of computing time taken by each querying method. In this evaluation the LINQ and STQP graphs are mentioned.

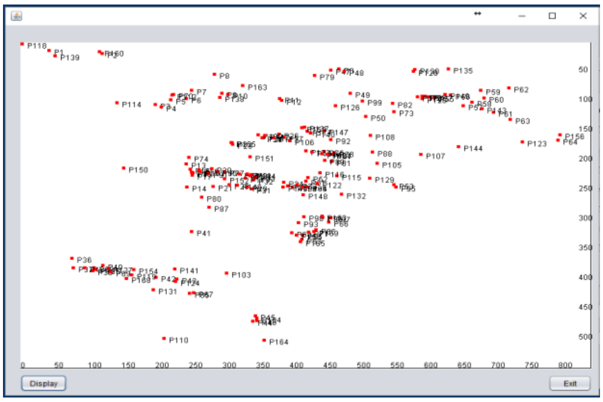

Figure 3: Location and Places from Dataset

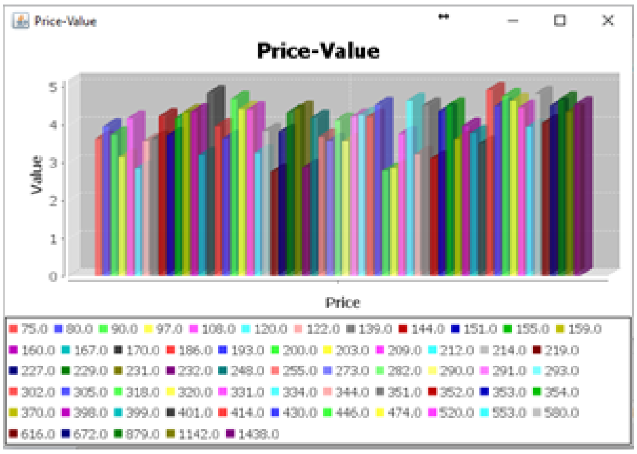

Figure 4: Graph for Factorization

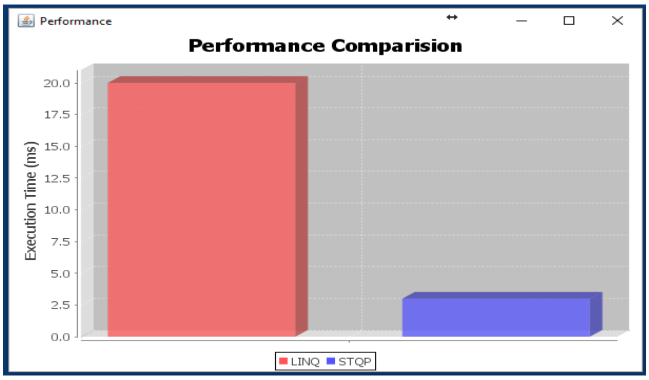

Figure 5: Graph for Execution Time

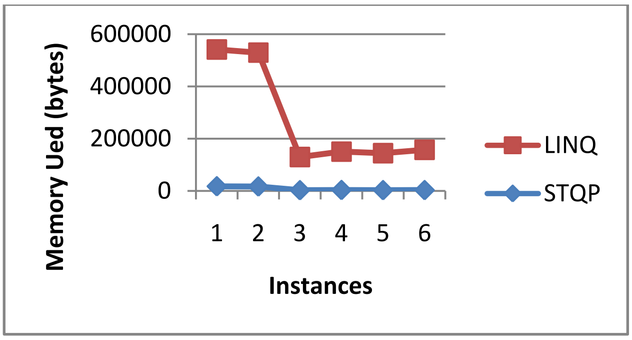

Figure 6: Graph for Memory Usage

The analysis is performed by using the above parameters. The comparison between LINQ and STQP is performed for execution time and Memory usage. It is observed that Memory and Execution time on the datasets clearly shows that STQP has better performance than LINQ.

5. CONCLUSION

In this paper, An important course of query, general location-aware get ranking query is formulated, and propose a construction called LINQ to process the query. In this particular platform, a novel index composition called synopses tree is made, which indexes synopses of things, and permits efficient pruning and estimation. This technique is increased by including temporal data as yet another attribute by which the ranking function can be increased and can provide reliable scores over time frame. The global buckets can even be used to provide efficiency and faster computation time by using Bin type algorithm this suggested method is recognized as STQP. The increased proposed system provided better results with respect to faster output for spatial query results. The synopses tree was created to reduce the price tag on construction while preserving accuracy. We show how to process GLRQs in the LINQ framework efficiently, leveraging synopses tree and other index set ups. Experimental results show that the proposed solution performs better than the existing solutions.

6. ACKNOWLEDGMENTS

This research was supported Mrs. AYESHA AMEEN, Associate Professor, D.C.E.T. We thank our colleagues from Deccan College Of Engineering and Technology,who provided insight and expertise that greatly assisted the research, although they may not agree with all of the interpretations/conclusions of this paper.

We thank Mr. Naqeeb Ahmed, Assistant professor for assistance and Dr. Syed Raziuddin, head, D.C.E.T for comments that greatly improved the manuscript.

We are also immensely grateful to (List names and positions) for their comments on an earlier version of the manuscript, although any errors are our own and should not tarnish the reputations of these esteemed persons

7. REFERENCES

[1] Xiping Liu, Lei Chen, and Changxuan Wan, “LINQ: A Framework for Location-Aware Indexing and Query Processing,” IEEE Transactions on Knowledge and Data Engineering, vol. 27, no 5, may 2015.

[2] G. Cong, C. S. Jensen, and D. Wu, “Efficient retrieval of the top-k most relevant spatial web objects,” Proc. VLDB Endowment, vol. 2, pp. 337–348, 2009.

[3] D. Wu, G. Cong, and C. Jensen, “A framework for efficient spatial web object retrieval,” VLDB J., vol. 21, pp. 797–822, 2012.

[4] Z. Li, K. Lee, B. Zheng, W.-C. Lee, D. L. Lee, andX. Wang, “IRTree: An efficient index for geographic document search,” IEEE Trans. Knowl. Data Eng., vol. 23, no. 4, pp. 585–599, Apr. 2011.

[5] J. A. B. Rocha-Junior, O. Gkorgkas, S. Jonassen, and K. Nørva g, “Efficient processing of top-k spatial keyword queries,” in Proc. Int. Conf Adv. Spatial Temporal Databases, 2011, pp. 205–222.

[6] C. Zhang, Y. Zhang, W. Zhang, and X. Lin, “Inverted linear quadtree: Efficient top k spatial keyword search,” in Proc. Int. Conf. Data Eng., 2013, pp. 901–912.

[7] G. Cormode, M. Garofalakis, P. J. Haas, and C. Jermaine, “Synopses for massive data: Samples, histograms, wavelets, sketches,” Found. Trends Databases, vol. 4, nos. 1–3, pp. 1–294, 2012.

[8] K. Tzoumas, A. Deshpande, and C. S. Jensen, “Lightweight graphical models for selectivity estimation without independence assumptions,” Proc. VLDB Endowment, vol. 4, no. 11, pp. 852–863, 2011.

[9] D. Lemire, O. Kaser, and K. Aouiche, “Sorting improves wordaligned bitmap indexes,” Data Knowl. Eng., vol. 69, no. 1, pp. 3–28, 2010.

[10] L. Chen, G. Cong, C. S. Jensen, and D. Wu, “Spatial keyword query processing: An experimental evaluation,” Proc. VLDB Endowment, vol. 6, no. 3, pp. 217–228, 2013.

[11] J. Fan, G. Li, L. Zhou, S. Chen, and J. Hu, “Seal: Spatio-textual similarity search,” Proc. VLDB Endowment, vol. 5, no. 9, pp. 824–835, 2012.

[12] X. Cao, L. Chen, G. Cong, C. S. Jensen, Q. Qu, A. Skovsgaard, D. Wu, and M. L. Yiu, “Spatial keyword querying,” in Proc. Int. Conf. Conceptual Model., 2012, pp. 16–29.

[13] J. Qi, R. Zhang, L. Kulik, D. Lin, and Y. Xue, “The min-dist location selection query,” in Proc. Int. Conf. Data Eng., 2012, pp. 366– 377.

[14] Y. Sun, J. Huang, Y. Chen, R. Zhang, and X. Du, “Location selection for utility maximization with capacity constraints,” in Proc. Int. Conf. Inf. Knowl. Manag., 2012, pp. 2154–2158.

[15] I. De Felipe, V. Hristidis, and N. Rishe, “Keyword search on spatial databases,” in Proc. Int. Conf. Data Eng., 2008, pp. 656–665.

.