Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 3, Issue. 9, September 2014, pg.340 – 350

RESEARCH ARTICLE

FAST NEAREST NEIGHBOR SEARCH WITH KEYWORDS

K. Shivakrishna[1], M. Jayapal[2]
shivakrishna8808@gmail.com, jayapalmedida@gmail.com
[1]M. Tech IV Semester, Dept. of CS &E, Malla Reddy College of Engineering and Technology, Hyderabad
[2]Associate Professor, Dept. of CS&E, Malla Reddy College of Engineering and Technology, Hyderabad

Abstract- Conventional spatial queries such as vary search and nearest neighbor retrieval, involve solely conditions on objects' geometric properties. Today, several trendy applications call for novel varieties of queries that aim to seek out objects satisfying both a spatial predicate, and a predicate on their associated texts. As an example, rather than considering all the restaurants, a nearest neighbor query would instead invite the restaurant that is the nearest among those whose menus contain "steak, spaghetti, brandy" all at identical time. Presently the simplest solution to such queries relies on the IR2-tree, which, as shown in this paper, includes a few deficiencies that seriously impact its efficiency. Impelled by this, we have a tendency to develop a brand new access methodology called the spatial inverted index that extends the standard inverted index to address multidimensional information, and comes with algorithms which will answer nearest neighbor queries with keywords in real time. As verified by experiments, the projected techniques outperform the IR2- tree in query reaction time significantly, typically by an element of orders of magnitude. Spatial queries, such as range search and nearest neighbor retrieval, involve only conditions on objects geometric properties. A spatial database manages multidimensional objects (such as points, rectangles, etc.), and provides fast access to those objects based on different selection criteria. Now-a-days many applications call a new form of queries to find the objects that satisfying both a spatial predicate, and a predicate on their associated texts. For example, instead of considering all the restaurants, a nearest neighbor query would instead ask for the restaurant that is the closest among those whose menus contain the specified keywords all at the sametime.IR2-tree is used in the existing system for providing best solution for finding nearest neighbor. This method has few deficiencies. So we implement the new method called spatial inverted index to improve the space and query efficiency. And enhanced search is used to search the required objects based on the user priority level. Thus the proposed algorithm is scalable to find the required objects.

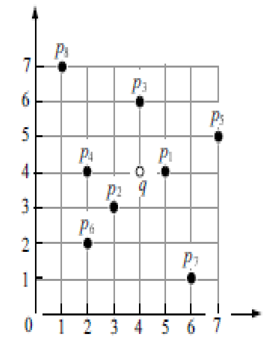
Keywords- Nearest Neighbor Search, Information Retrieval Tree, Spatial Inverted Index

I. INTRODUCTION

An increasing number of applications require an 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 users to provide keywords that the spatial objects should contain description of spatial keyword query. A spatial database manages multidimensional objects (such as points, rectangles, etc.) and provides fast access to those objects based on different selection criteria. The importance of spatial databases is, the real entities are represented in a geometric manner. For example, locations of restaurants, hotels, hospitals and so on are represented as points in a map, while larger areas such as parks, lakes and landscapes as a combination of rectangles. Queries focus on objects' geometric properties only, such as whether a point is in a rectangle or how close two points are from each other. Some modern applications that call for the ability to select objects based on both of their geometric coordinates and their associated texts.

For example, it would be useful if a search engine can be used to find the nearest restaurant that offers "steak, spaghetti and brandy" all at the same time. Note that this is not the "globally" nearest restaurant, but the nearest restaurant among only those providing all the demanded foods. In this paper, we design another form of inverted index that is optimized for multidimensional points and it is named as spatial inverted index (SI-index). This access method incorporates point coordinates into a conventional inverted index with small extra space. An SI-index preserves the spatial locality of data points and comes with an R-tree built on every inverted list at little space overhead. As a result, it offers two competing ways for query processing. We can (sequentially) merge multiple lists very much like merging traditional inverted lists by ids. Alternatively, we can also influence the R-trees to browse the points of all relevant lists in ascending order of their distances to the query point.

We present a method to efficiently answer top-k spatial
Keyword queries, which is based on the tight integration of data structures and algorithms used in spatial database search and Information Retrieval (IR).In particular, our method consists of building an Information Retrieval R-Tree (IR2-Tree), which is a structure based on the R-Tree . At query time an incremental algorithm is employed that uses the IR2-Tree to efficiently produce the top results of the query. The IR2-Tree is an R-Tree where a signature (Fallouts and Christodoulakis) is added to each node v of the IR2-Tree to denote the textual content of all spatial objects in the sub tree rooted at v. Our top-k spatial keyword search algorithm, which is inspired by the work of Hjaltason and Samet , exploits this information to locate the top query results by accessing a minimal portion of the IR2-Tree.
This work has the following contributions:
• The problem of top-k spatial keyword search is defined.
• The IR2-Tree is proposed as an efficient indexing Structure to store spatial and textual information for a set of objects. Efficient algorithms are also presented to maintain the IR2-Tree, that is, insert and delete objects.
• An efficient incremental algorithm is presented to answer Top-k spatial keyword queries using the IR2 -Tree. Its performance is evaluated and compared to current approaches. Real datasets are used in our experiments that show the significant improvement in execution times. Note that our method can be applied to arbitrarily-shaped and multi-dimensional objects and not just points on the two dimensions, which are used in our running examples for clarity.



(a)

p	W_p
p_1	$\{a, b\}$
p_2	$\{b, d\}$
p_3	$\{d\}$
p_4	$\{a, e\}$
p_5	$\{c, e\}$
p_6	$\{c, d, e\}$
p_7	$\{b, e\}$
p_8	$\{c, d\}$

(b)

Fig.. (a) Shows the locations of points and (b) gives their associated texts.

II. RELATED WORK

A. k Nearest Neighbor Queries

In spatial databases, k nearest neighbor (kNN) and range queries are fundamental query types. These two types of spatial queries have been extensively studied and applied in various location-based service (LBS) applications. The solutions for nearest neighbor queries are designed in the context of spatial databases. In addition, a Voronoi diagram-based solution for kNN searches in spatial databases is presented in by partitioning a large regions to small Voronoi regions and pre-computing distances both within and across the regions. Because most Dijkstra's algorithm-based kNN solutions have been shown to be efficient only for short distances, Hu et al. proposed an efficient index (distance signature) for distance computation and query processing over long distances. Their technique discretizes the distances between objects and network nodes into categories and then encodes these categories to execute the kNN search process. In order to speed up kNN searches, Samet et al. designed an algorithm to compute the shortest paths between all the vertices in the network and employing a shortest path quadtree to capture spatial coherence. With the algorithm, the shortest paths between all possible vertices can be computed to answer various kNN queries on a given spatial network. Nevertheless, all the above mentioned techniques mainly focused on the distance metric. They did not consider text description (keywords) of spatial objects in their query evaluation processes

B. Text Retrieval

Text retrieval is another important topic related to spatial Keyword queries. There are two main indexing techniques, inverted files and signature files, widely utilized in text retrieval systems. According to experiments made by Zobel et al., signature files require a much larger space to store index structures and are more expensive to construct and update than inverted files. The inverted files outperform Signature files in most cases. Although these methods Perform quite well in text retrieval applications, none of them can efficiently process spatial keyword queries. In other words, it is impractical to answer spatial keyword queries by simply employing approaches introduced in the previous subsection. An effective way to handle spatial keyword queries is to combine the two groups means Nearest neighbor queries and Text retrieval.

C. Spatial Keyword Query

Many solutions have been developed to evaluate spatial keyword queries. Location-based web search is studied by Zhou et al. to find web pages related to a spatial region. They described three different hybrid indexing structures of integrating inverted files and R*-trees together. According to their experiments, the best scheme is to build an inverted index on the top of R*-trees. In other words, the algorithm first sets up an inverted index for all keywords, and then creates an R*-tree for each keyword. This method performs well in spatial keyword queries in their experiments, but its maintenance cost is high. When an object insertion or deletion occurs, the solution has to update the R*-trees of all the keywords of the object. Cong et al. illustrated a hybrid index structure, the IR-tree, which is a combination of an R-tree and inverted files to process location-aware text retrieval and provide k best candidates according to a rank system. It minimizes areas of enclosing rectangles and maximizing text into account during construction procedures. Felipe et al. developed a novel index, IR2-Tree which integrates an R-tree and signature files together, to answer top- k spatial keyword queries. They record signature information in each node of R-trees in order to decide whether there is any object which satisfies both spatial and keyword constraints simultaneously. However, the size of space for storing signatures in each node is decided before IR2-Tree construction. Once the IR2-Tree has been built, it is impossible to enlarge the space unless the tree is reconstructed. If the number of keywords grows quickly, a system will spend a lot of time repeatedly rebuilding the IR2-Tree. Hariharan et al. proposed an indexing mechanism, KR*-tree, which combines an R*-tree and an inverted index. The difference between their solution and is that they only store related keywords in each node of an R*-tree in order to avoid merging operations to find candidates containing all keywords. If the number of keywords that appear in each node varies. However, such a complicated indexing technique has a high maintenance cost as well. Although there are a number of previous studies on spatial keyword queries, most of their solutions can only evaluate queries in Euclidean spaces. This limitation is due to the adoption of the R-tree (or its variants), which cannot index spatial objects based on network distances, into their hybrid index structures.

III. EXISTING SYSTEM

Existing works mainly focus on finding top-k nearest Neighbors, where each node has to match the whole Querying keywords .It does not consider the density of data objects in the spatial space. Also these methods are low efficient for processing query. The existing data structure called the IR2-tree was used for processing the query. But IR2-tree has a drawback of signature files: false hits. That is, a signature file, due to its conservative nature, it searches to some objects, even though they do not have the keywords. The penalty thus caused is the need to verify an object whose satisfying a query or cannot be resolved using only its signature, but requires loading its full text description, which is expensive due to the resulting random accesses.

IV. PROPOSED SYSTEM

A spatial database manages dimensional objects (such as points, rectangles, etc.) and provides quick access to those objects. The importance of spatial databases is, it represents entities of reality in geometric manner. For example, locations of restaurants, hotels, hospitals are described as points in map, whereas larger extents like parks, lakes and landscapes as a mix of rectangles. In this paper we design a proposed system called spatial inverted index (SI-index).SI-preserves the spatial location of data points and builds R-tree on every inverted list at little space overhead.

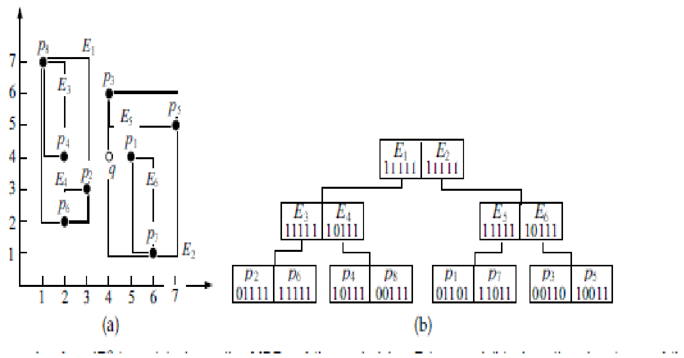


Fig. Example of an IR2 -tree. (a) shows the MBRs of the underlying R-tree and (b) gives the signatures of the entries.

V. SPATIAL INVERTED INDEX ALGORITHM

The kNN spatial keyword query process is shown in Algorithm. The inputs to the algorithm are the query point q, the boundary object BO , the parameter k and keyword Kw. The kNN result returned by the server is retrieved by calling BO.result () on line 2. H is a min-heap which sorts points according to their distances to query q. First (lines 3-12), the algorithm constructs the boundary cell (BC) of the first object p1 and checks whether q falls inside BC (p1). If not, p1 is not the first NN and the verification process fails. Otherwise, p1 is verified as the first NN and is added to the Visited set. The subsequent for loop (lines 13-22) iterates through all objects in L (kNNs from the BO) and performs the following operations: 1) if the neighbor of the last verified object (L[i]) has not been visited yet, it is inserted into the min-heap H and the Visited set (lines 14-18) and 2) it compares the next object in the result set (L[i+1]) with the top of H (lines 19-21). If they are identical, L[i+1] is verified as the next NN. Otherwise, verification fails and the program returns false.

Algorithm 1: KNN (q, BO, K, Kw)

```
1: H ← ∅; visited ← ∅
2: L ← BO.result ( ); p1=L[1];
3: BCP ← compute BC( p1 );
4: if ( q ∉ BCP ) then
5: return false; {the first NN fails}
6: else
7: if ( Kw ϵ p1 ) then
8: visited.add ( p1);
9: else
10: return false;
11: end if
12: end if
13: for i=1 to k-1 do
14: for all ( n ϵ L[i]. Neighbors) do
15: if ( n ∉ visited) then
16: visited.add (n);
17: end if
18: end for
19: if (L [i+1].location ≠ H.pop( ) ) then
20: return false; {the ( i+1) th NN fails }
21: end if
22: end for
23: return true;
```

VI. FEATURES OF THE MECHANISMS

A. *Approximate string search*

We refer to the problem of conjunctive keyword search with relaxed keywords as *approximate keyword search*. An important subproblem of approximate *keyword* search is that of approximate *string* search, defined as follows. Given a collection of strings, find those that are similar to a given query string. There are two main families of approaches to answer such queries. (1) Trie based method: The string collection is stored as a trie (or a suffix tree). For answer an approximate string query, we traverse the trie and prune subtrees that cannot be similar to the query. Popular pruning techniques use an NFA or a dynamic programming matrix with backtracking. (2) Inverted-index method: Its main idea is to decompose each string in the collection to small overlapping substrings (called grams) and build an inverted index on these grams. More details on fast indexing and search algorithms can be found in.

B. Spatial approximate-keyword search

Yao et al proposed a structure called MHR-tree to answer spatial approximate-keyword queries. They enhance an R-tree with min-wise signatures at each node to compactly represent the union of the grams contained in objects of that subtree. They then use the concept of set resemblance between the signatures and the query strings to prune branches in the tree. The main advantage of this approach is that its index size does not require a lot of space since the min-wise signatures are very small. However, the method could miss query answers due to the probabilistic nature of the signatures. Sattam Alsubaiee. et.al, we study how to support approximate keyword search on spatial data. Answering such queries efficiently is critical to these Web sites to achieve a high query throughput to serve many concurrent users. Although there are studies on both approximate keyword search and location-based search, the problem of supporting both types of search simultaneously has received little attention. To answer such queries, a natural index structure is to augment a tree-based spatial index with approximate string indexes such as a gram-based inverted index or a trie-based index. The main focus of this work is a systematic study on how to efficiently combine these two types of indexes, and how to search the resulting index (called LBAK-tree) to find answers.

C. Nearest Neighbor and Top-k Queries

The processing of *k*-nearest neighbor queries (*k*NNs) in spatial databases is a classical subject. Most proposals use index structures to assist in the *k*NN processing. Perhaps the most influential *k*NN algorithm is due to Roussopoulos et al. In this solution, an R-tree indexes the points, potential nearest neighbors are maintained in a priority queue, and the tree is traversed according to a number of heuristics. Other branch-and-bound methods modify the index structures to better suit the particular problem addressed. Hjaltason and Samet propose an incremental nearest neighbor algorithm based on an R*-tree.

D. Location-Aware Text Retrieval Queries

Profitable search engines such as Google and Yahoo! have introduced local search services that appear to focus on the retrieval of local content, e.g., related to stores and restaurants. However, the algorithms used are not publicized. Much attention has been given to the problem of extracting geographic information from web pages. The extracted information can be used by search engines. McCurley covers the notion of geo-coding and describes geographic indicators found in pages, such as zip codes and location names. Recent studies that consider location-aware text retrieval constitute the work most closely related to this study. Zhou et al.tackle the problem of retrieving web documents relevant to a keyword query within a pre-specified spatial region. They propose three approaches based on a loose combination of an inverted file and an R*- tree. The best approach according to their experiments is to build an R*-tree for each distinct keyword on the web pages containing the keyword. As a result, queries with multiple keywords need to access multiple R*-trees and to intersect the results. Building a separate R*-tree for each keyword also requires substantial storage.

Ian De Felipe.et.al, the problem of top k spatial keyword search is defined. The IR2-Tree is proposed as an efficient indexing structure to store spatial and textual information for a set of objects. Efficient algorithms are also presented to maintain the IR2-Tree, that is, insert and delete objects. An efficient incremental algorithm is presented to answer top-k spatial keyword queries using the IR2-Tree. We present a method to efficiently answer top-k spatial keyword queries, which is based on the tight integration of data structures and algorithms used in spatial database search and Information Retrieval (IR). In particular, this method consists of building an Information Retrieval R-Tree (IR2- Tree), which is a structure based on the R-Tree. At query time an incremental algorithm is employed that uses the IR2- Tree to efficiently produce the top results of the query.

VII. Testing

Testing is a process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an as-yet –undiscovered error. A successful test is one that uncovers an as-yet- undiscovered error. System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently as expected before live operation commences. It verifies that the whole set of programs hang together. System testing requires a test consists of several key activities and steps for run program, string, system and is important in adopting a successful new system. This is the last chance to detect and correct errors before the system is installed for user acceptance testing.

The software testing process commences once the program is created and the documentation and related data structures are designed. Software testing is essential for correcting errors. Otherwise the program or the project is not said to be complete. Software testing is the critical element of software quality assurance and represents the ultimate the review of specification design and coding. Testing is the process of executing the program with the intent of finding the error. A good test case design is one that as a probability of finding a yet undiscovered error. A successful test is one that uncovers a yet undiscovered error. An1y engineering product can be tested in one of the two ways:

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each type addresses a specific testing requirement.

VIII. TYPES OF TESTS

Unit testing: Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration testing: Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Functional test: Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals. Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing.

Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Test: System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing**:** White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing: Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

Unit Testing: Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach: Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

Integration Testing: Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects. The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results**:** All the test cases mentioned above passed successfully. No defects encountered.

Acceptance Testing: User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements**.**

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

IX. Conclusions

In this paper we introduced the problem of spatial keyword search and explained the performance limitations of current approaches.

We proposed a solution which is dramatically faster than current approaches and is based on a combination of R-Trees and signature files techniques. In particular we introduced the IR2-Tree and showed how it is maintained in the presence of data updates. An efficient incremental algorithm was presented that uses the IR –Tree to answer spatial keyword queries. We experimentally evaluated our technique, which proved its superior performance. We have seen plenty of applications calling for a search engine that is able to efficiently support novel forms of spatial queries that are integrated with keyword search. The existing solutions to such queries either incur prohibitive space consumption or are unable to give real time answers. In this paper, we have remedied the situation by developing an access method called *the* spatial inverted index (SI-index). Not only that the SI-index is fairly space economical, but also it has the ability to perform keyword-augmented nearest neighbor search in time that is at the order of dozens of milliseconds. Furthermore, as the SI-index is based on the conventional technology of inverted index, it is readily incorporable in a commercial search engine that applies massive parallelism, implying its immediate industrial merits.

References:

[1] S. Agrawal, S. Chaudhuri, and G. Das. Dbxplorer: A system for keyword-based search over relational databases. In *Proc. Of International Conference on Data Engineering (ICDE)*, pages 5–16, 2002.

[2] N. Beckmann, H. Kriegel, R. Schneider, and B. Seeger. The R*-tree: An efficient and robust access method for points and rectangles. In *Proc. of ACM Management of Data (SIGMOD)*, pages 322–331, 1990.

[3] G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, and S. Sudarshan. Keyword searching and browsing in databases using banks. In *Proc. of International Conference on Data Engineering (ICDE)*, pages 431–440, 2002.

[4] 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.

[5] X. Cao, G. Cong, and C. S. Jensen. Retrieving top-k prestige-based relevant spatial web objects. *PVLDB*, 3(1):373–384, 2010.

[6] X. Cao, G. Cong, C. S. Jensen, and B. C. Ooi. Collective spatial keyword querying. In *Proc. of ACM Management of Data (SIGMOD)*, pages 373–384, 2011.

[7] B. Chazelle, J. Kilian, R. Rubinfeld, and A. Tal. The bloomier filter: an efficient data structure for static support lookup tables. In *Proc. of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 30–39, 2004.

[8] Y.-Y. Chen, T. Suel, and A. Markowetz. Efficient query processing in geographic web search engines. In *Proc. of ACM Management of Data (SIGMOD)*, pages 277–288, 2006.

[9] E. Chu, A. Baid, X. Chai, A. Doan, and J. Naughton. Combining keyword search and forms for ad hoc querying of databases. In *Proc. of ACM Management of Data (SIGMOD)*, 2009.

[10] 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.

[11] Y.-Y. Chen, T. Suel, and A. Markowetz. Efficient query processing in geographic web search engines. In *SIGMOD*, 2006.

[12] S. Alsubaiee, A. Behm, and C. Li. Supporting location-based approximate-keyword queries. In *GIS*, pages 61–70, 2010.

[13] A. Arasu, S. Chaudhuri, K. Ganjam, and R. Kaushik. Incorporating string transformations in record matching. In *SIGMOD*, pages 1231– 1234, 2008.

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

[15] I. D. Felipe, V. Hristidis, and N. Rishe. Keyword search on spatial databases. In *ICDE*, 2008.

[16] A. Guttman. R-trees: A dynamic index structure for spatial searching. In *SIGMOD*, 1984.

[17] X. Cao, G. Cong, and C. S. Jensen. Retrieving top-k prestigebased relevant spatial web objects. *Proc. VLDB Endow.*, 3:373– 384, 2010.

[18] K. Chakrabarti, S. Chaudhuri, V. Ganti, and D. Xin. An efficient filter for approximate membership checking. In *SIGMOD*, pages 805–818, 2008.

[19] S. Chaudhuri, K. Ganjam, V. Ganti, and R. Motwani. Robust and efficient fuzzy match for online data cleaning. In *SIGMOD*, pages 313– 324, 2003.

[20] S. Chaudhuri, V. Ganti, and L. Gravano. Selectivity estimation for string predicates: Overcoming the underestimation problem. In *ICDE*, pages 227–238, 2004.

[21] S. Chaudhuri, V. Ganti, and R. Kaushik. A primitive operator for similarity joins in data cleaning. In *ICDE*, pages 5–16, 2006.

[22] M. Hadjieleftheriou, A. Chandel, N. Koudas, and D. Srivastava. Fast indexes and algorithms for set similarity selection queries. In *ICDE*, 2008.

[23] 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.

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

[25] C. Li, J. Lu, and Y. Lu. Efficient merging and filtering algorithms for approximate string searches. In *ICDE*, 2008.

Authors Biography

First Author: K.Shivakrishna

M.Tech in CSE from JNTUH,
Malla Reddy College of Engineering and Technology, Hyderabad

Second Author: M.Jayapal

M.Tech in CSE from JNTUH,
Associate Professor, Dept of CS&E,
Malla Reddy College of Engineering and Technology, Hyderabad