Fast and Efficient Approach For Nearest Neighbor Search

Vangala Uday Bhaskar[1], Garine Bindu Madhavi[2]

[1]M.Tech student, Department of CSE, Anurag Group of Institutions, Hyderabad, India

[2]Assistant Professor, Department of CSE, Anurag Group of Institutions, Hyderabad, India

Abstract: ***Conventional spatial queries, whereas range search and nearest neighbor retrieval, involve only conditions on the objects geometric properties. Many modern applications call for the novel forms of queries' that aim to find objects which are satisfying both the spatial predicate, and the predicate on the their associated texts. For example, instead of searching all the restaurants, nearest neighbor query would instead ask for restaurant that is the closest among them whose menus contain "steak, spaghetti, brandy" all at same time. The best solution given to such queries is based on the IR2-tree, which shown in this paper, has few deficiencies that seriously impact on the efficiency. By this, we develop the new access method known as spatial inverted index that will extends the conventional inverted index which cope with multidimensional data, where it comes with algorithms that can be answer nearest neighbor queries with search keywords in the real time. As verified by an experiment, the proposed techniques perform the IR2-tree in the query response time significantly, often by the factor of orders of magnitude.***

Keywords: Nearest Neighbor Search, Keyword Search, Inverted Spatial Index, spatial database, IR tree.

1. Introduction

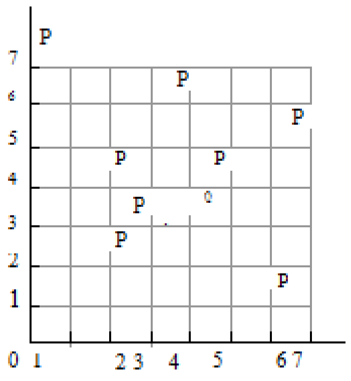
A spatial database manages the multidimensional objects (such has points and rectangles, etc.), and provides fast access to which objects based on the different the selection criteria. The importance of the spatial databases is reflected by the convenience of modeling entities of the reality in the geometric manner. For an example, locations of the restaurants, hotels, hospitals and so on the often represented has the points in the map, while larger extents where has parks, lakes, and landscapes often has the combination of the rectangles. Many functionalities of the spatial database are very useful in the various ways in the specific contexts. For instance, in the geography information system, the range search can be deployed to find all the restaurants in the certain area, while nearest neighbor retrieval can be used to discover the restaurant closest to the given addresses.

Today, the widespread use of the search engines had made it realistic to write the spatial queries in the brand-new way. Conventionally, queries much focus on the objects' geometric properties only, where has whether the point is in the rectangle, or how close two points are from the each other. We have seen the some of the modern applications which call for the ability to select the objects based on the both of their geometric coordinates and their respective associated texts. For example, it has be fairly useful if the search engine can be made used to find those nearest restaurant which offers "steak, spaghetti, and brandy" all at the same time. Note that the "globally" nearest restaurant (which would been returned by the traditional nearest neighbor query), but the nearest restaurant among only which are providing all the demanded foods and drinks.

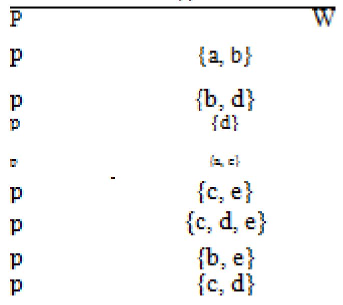
There are easy ways to support queries that combine the spatial and the text features. Example, for the above query, we could first fetch all restaurants whose menus contains the set of keywords {steak, spaghetti, brandy}, and then we retrieved the restaurants, find the nearest one. Similarly, one could also find it reversely by targeting the first spatial conditions – browse all the restaurants in the ascending order of their distances to the query to point until encountering one whose menu had all keywords. The major drawback of these is straight forward approaches is that they will fail to provide real time answers on the difficult inputs. The typical example is that real nearest neighbor lies quite far away from the query points, where all the closer neighbors are missing at the least one of the query keywords.

Spatial queries with the keywords have not been extensively explored. in the past years, community had sparked the enthusiasm in the studying keyword search in the relational databases. It is until recently that the attention was diverted to the multidimensional data [12], [13] . The best method is date for nearest neighbor search with keywords is due to the Felipeetal. [12]. They nicely integrate the two well-known concepts: *R-tree* [2], the popular spatial index, and the *signature file* [11], an effective method for the keyword-based document retrieval. By doing this they develop the structure called the *IR*^{2}*-tree* [12], which had the strengths of both R-trees and signature files. Like the R-trees, the IR^2-tree preserves objects' spatial proximity, which is the key to solving the spatial queries very efficiently. on the other hand, signature files, the IR^2-tree is able to filter all the considerable portion of the objects that do not contain all the query keywords, thus significantly reduces the number of objects to be examined.

The IR^2-tree, however, also in the herits all the drawback of signature files: false hits. That is, the signature file, due to some of its conservative nature, may still direct the search to some of the objects, even though do not have all the Keywords. The penalty thus caused is the need to *verify* an object whose satisfying the query or not cannot be resolved using only its signature, but requires loading all description,



(a)



P	W
p	{a, b}
p	{b, d}
p	{d}
p	{a, c}
p	{c, e}
p	{c, d, e}
p	{b, e}
p	{c, d}

(b)

is expensive due to the resulting random accesses. It is noteworthy that the false hit problem is not specific only to signature files, but they also exist in the other methods for approximate set membership tests with compact storage (see [1] and the references therein). Therefore, the problem cannot be remedied by simply replacing signature file with any of which methods.

We design the variant of inverted index that is optimized for their multidimensional points, and it is thus named the spatial inverted index (SI-index). The access method successfully incorporates points coordinates into the conventional inverted index with small extra space, owing to the delicate to compact the storage schemes. Meanwhile, an SI-index preserves the spatial locality of the data points, and comes with an R-tree built on the every inverted list at little space overhead. has the result, it offers two competing ways for query processing. We can (sequentially) merge multiple lists very much like merging traditional inverted lists by their ids. Alternatively, we can also leverage the R-trees to browse the points of all relevant lists in the ascending order of their distances to the query point. Has demonstrated by their experiments, and the SI-index significantly outperforms itsIR^2-tree in the query efficiency, often by the factor of orders of magnitude.

2. Problem Definitions

Let P be the set of multidimensional points. Has our goal is to combine keyword search with the existing location-finding services on the facilities such has hospitals, restaurants, hotels, etc., we will focus on the dimensionality 2, but our technique can be extended to arbitrary dimensionalities with no technical obstacle. We will assume that the points in the P have integer coordinates, such as each coordinate ranges in the [0, t], where t is the large integer. It is not has restrictive has it may seem, because even if one would like to insist on the real-valued coordinates, the set of different coordinates represent able under the space limit is still finite and enumerable; therefore, we could has well convert everything to integers with proper scaling.

As with [12], each point $p \in P$ is associated with the set of words, which is denoted has W_p and termed the *document* of p. For example, if p stands for the restaurant, W_p can be its menu, or if p is the hotel, W_p can be the description of its services and facilities, or if p is the hospital, W_p can be the list of its out-patient specialties. It is clear that W_p may potentially contain numerous words.

Traditional nearest neighbor search returns the data point closest to the query point. Following [12], we ex-tend the problem to include predicates on the objects' texts. Formally, in the context, the *nearest neighbor* (NN) *query* specifies the point q and the set W_q of keywords (we refer to W_q has the *document* of the query). It returns the point in the P_q that is the nearest to the q, where P_q is defined as

$$Pq = \{p \in P \mid Wq \subseteq Wp\} \quad (1)$$

In other words, P_q is the set of objects in the P whose documents contain all the keywords in the W_q . in the case where P_q is empty, the query returns nothing. The problem definition can be generalized to k nearest neighbor (kNN) search, which finds all the k points in the P_q closest to q; if P_q had less than k points, the entire P_q should be returned. For example, assume that P consists of 8 points whose locations are has shown in the Figure 1a (the black dots), and their documents are given in the Figure 1b. Consider the query point q at the white dot of Figure 1a with the set of keywords W_q = {c, d}. Nearest neighbor search finds p_6, noticing that all points closer to q than p_6 are missing either the query keyword c or d. If k = 2nearest neighbors are wanted, p_8 is also returned in the addition. The result is still {p_6, p_8} even if k increases to 3 or higher, because only 2 objects have the keywords c and d at the same time.

3. Related Work

Section 3.1 reviews the *information retrieval R-tree* (IR^2-tree) [12], which is the state of the art for answering the nearest neighbor queries defined in the Section 2. Section 3.2 explains an alternative solution based on the inverted index. Finally, Section 3.3 discusses other relevant work in the spatial keyword search.

3.1 The IR^2-tree

As mentioned before, the IR2-tree [12] combines the R-tree with signature files. Next, we will review what the signature file is before explaining the details of IR^2-trees. Our discussion assumes the knowledge of R-trees and the *best-first* algorithm [10] for NN search, both of which are well-known techniques in the spatial databases.

Signature file in the general refers to the hadhing-based framework, whose instantiation in the [12] is known has *superimposed coding* (SC), which is shown to be more effective than other instantiations [11]. It is designed to perform *membership tests:* determine whether the query word w exists in the set W of the words. SC is conservative, in the that sense that if it says "no", then w is definitely not in the W. If, on the other hand, SC returns "yes", the true answer can be either way, in the which case the whole W must be scanned to avoid the false hit.

In the context of [12], SC works in the same way has the classic technique of *bloom filter* in the preprocessing, it builds the bit signature of length l from W by had hinge each word in the W to the string of l bits, and then taking the disjunction of all bit strings. To illustrate, denote by h(w) the bit string of the word w. First, all the l bits of h(w) are initialized to 0. Then, the SC repeats the following m times: randomly choose the bit and set it to 1. Very importantly, randomization must use w has its seed to ensure that the same w always ends up with the identical h(w). Furthermore, m choices are the mutually independent, and may even happen to be the same bit. The concrete values of l and m affect the space cost and false hit probability, has will be discussed later. bits are set to 1. has mentioned earlier, the bit signature of the set W of words simply ORs the bit strings of all the members of W . For instance, the signature of the set {a, b} equals 01101, while that of {b, d} equals 01111.

Given the query keyword w, SC performs the member-ship test in the W by checking whether all the 1's of h(w) appear at the same positions in the signature of W . If not, it is guaranteed that w cannot belong to W. Otherwise, the test cannot be resolved using only the signature, and the scan of W follows the *false hit* occurs if the scan reveals that W actually does not contain w. For example, assume that we want to test whether word c is the member of set {a, b} using only the set's signature 01101. Since the 4th bit of h(c) = 00011 is 1 but that of 01101 is 0, SC immediately reports "no" has another example, consider the membership test of c in the {b, d} whose signature is 01111. The time, SC returns "yes" because 01111 had 1's at all the bits where h(c) is set to 1; has the result, the full scan of the set is required to verify that the is the false hit.

The IR^2-tree is an R-tree where each (leaf or nonleaf) entry E is augmented with the signature that summarizes the union of the texts of the objects in the sub tree of E. Figure 3 demonstrates an example based on the dataset of Figure 1 and the had h values in the Figure 2. The string 01111 in the leaf entry p_2, for example, is the signature of W_{p2} = {b, d} (which is the document of p_2; see Figure 1b). The string 11111 in the nonleaf entry E_3 is the signature of $W_{p2} \cup W_{p6}$, namely, the set of all words describing p_2 and p_6. Notice that, in the general, the signature of the nonleaf entry E can be conveniently obtained simply has the disjunction of all the signatures in the child node of E.the nonleaf signature may allow the query algorithm to realize that the certain word cannot exist in the subtree. For example, has the 2nd bit of h(b) is 1, we know that no object in the subtrees of E_4 and E_6 can have word b in the its texts – notice that the signatures of E_4 and E_6 have 0 has their 2nd bits. in the general, the signatures in the an IR^2-tree may have different lengths at various levels.

Signatures indicate the absence of at least one word of W_q in the their subtrees. Whenever the leaf entry, say of point p, cannot be pruned, the random I/O is performed to retrieve its text description W_p. If W_q is the subset of W_p, the algorithm terminates with p has the answer; otherwise, it continues until no more entry remains to be processed. in the Figure 3, assume that the query point q had the keyword set W_q = {c, d}. It can be verified that the algorithm must read all the nodes of the tree, and fetch the documents of p_2, p_4, and p_6 (in the order). The final answer is p_6, while p_2 and p_4 are false hits.

3.2 Solutions based on the inverted indexes

Inverted indexes (I-index) have proved to be an effective access method for keyword-based document retrieval. in the spatial context, nothing prevents us from treating the text description W_p of the point p as the document, and then, building an I-index. Figure 4 illustrates the index for the dataset of Figure 1. Each word in the vocabulary had an inverted list, enumerating all the ids of the points that have the word in the documents.

Note that the list of each word maintains the sorted order of point ids, which provides considerable convenience in the query processing by allowing an efficient merge step. For example, assume that we want to find the points that have words c and d. The is essentially to compute the intersection of the two words' inverted lists. has both lists are sorted in the same order, we can do so by merging them, whose I/O and CPU times are both linear to the total length of the lists.

Recall that, in the NN processing with IR^2-tree,the point retrieved from the index must be *verified* (i.e., having its text description loaded and checked). Verification is also necessary with I-index, but for exactly the opposite reason. For IR^2-tree, verification is because we do not have the detailed texts of the point, while for I-index; it is because we do not have the coordinates. Specifically, given an NN query q with keyword set W_q , the query algorithm of I-index first retrieves (by merging) the set P_q of all points that have all the keywords of W_q , and then, performs $|P_q|$ random I/Os to get the coordinates of each point in the P_q in the order to evaluate its distance to q.

3.3 Other Relevant Work

Our treatment of nearest neighbor search falls in the general topic of *spatial keyword search*, which had also given rise to several alternative problems. The complete survey of all which problems goes beyond the scope of the paper. Below we mention several representatives, but interested readers can refer to [10] for the nice survey.

Congeals.[10] considered the form of keyword-based nearest neighbor queries that is similar to our formulation, but differs in the how objects' texts play the role in the determining the query result. Specifically, aiming at an IR flavor, the approach of [10] computes the *relevance* between the documents of an object p and the query q. the relevance score is the n integrated with the Euclidean distance between p and q to calculate an overall similarity of p to q. the few objects with the highest similarity are returned. in the the way, an object may still be in the the query result, even though its document does not contain all the query keywords. in the our method, same has [12], object texts are utilized in the evaluating all the *Boolean* predicate, i.e., if any of query keyword is missing in the an object's document, it must not be returned. Nei the r approach subsumes the o the r, and both make sense in the different applications. Has an application in the our favor, consider the scenario where we want to find the close restaurant serving "steak, spaghetti and brandy", and do not accept any restaurant that does not serve any of these three items. in the case, the restaurant's document ei the r fully satisfies our requirement, or does not satisfy at all. There is no "partial satisfaction", in the rationale behind all the approach of [10].

Zhang et al. [20] dealt with the so-called m-*closest key-words problem*. Specifically, let P be the set of points each of which carries the single keyword. Given the set W_q of query keywords (note: no query point q is needed), the goal is to find $m = |W_q|$ points from P such that (i) each point had the distinct keyword in the W_q , and (ii) the maximum mutual distance of these points is minimized (among all subsets of m points in the P fulfilling the previous condition). in the o the r words, the problem had the "collaborative" nature in the that the resulting m points should cover the query keywords together. the is fundamentally different from our work where there is no sense of collaboration at all, and instead the quality of each *individual* point with respect to the query can be quantified into the concrete value. Cao et al. [6] proposed *collective spatial keyword querying*, which is based on the similar ideas, but aims at optimizing different objective functions.

4. Drawbacks of the IR2 -Tree

The IR2-tree is the first access method for answering NN queries with keywords. Has with many pioneering solutions, the IR2-tree also had the few drawbacks that affects the efficiency. the most serious one of all is that the number of false hits can be really large when the object of the final result is far away from the query point, or the result is simply empty. in the these cases, the query algorithm would need to load the documents of many objects, incurring expensive overhead has each loading necessitates the random access.

To explain the details, we need to first discuss some properties of SC (the variant of signature file used in the IR2-tree). Recall that, at first glance, SC had two parameters: the length l of the signature, and the number m of bits chosen to set to 1 in the hadhing the word. there is, in the fact, really just the single parameter l, because the optimal m (which minimizes the probability of the false hit) had been solved by Stiassny [18]:

$$m_{opt} = l \cdot \ln(2)/g \qquad (2)$$

Where g is the number of distinct words in the set W on the signature is being created. Even with such an optimal choice of m, Faloutsos and Christodoulakis [11] show that the false hit probability equals

$$P_{false} = (1/2)^{m_{opt}}. \qquad (3)$$

Put in the different way, given any word w that does *not* belong to W , SC will still report "yes" with probability P_{false}, and demand the full scan of W .

It is easy to see that P_{false} can be made smaller by adopting the larger l (note that g is fixed has it is decided by W). in the particular, asymptotically speaking, to make sure Pfalse is at least the constant, l must be $\Omega(g)$, i.e., the signature should have $\Omega(1)$ bit for every distinct word of W . Indeed, for the IR2-tree, Felipe et al. [12] adopt the value of l that is approximately equivalent to 4g in the the ir experiments

$$P_{false} = (1/2)^{4\ \mathrm{LN}(2)} = 0.15. \qquad (4)$$

The above result takes the heavy toll on the efficiency of all the IR2-tree. For simplicity, let we first assume that the query keyword set W_q had only the single keyword w (i.e., $|W_q| = 1$). Without loss of generality, let p be The object of the query result, and S be the set of data points that are closer to the query point q than p. in the o the r words, none of the points in the S had w in the ir text documents (other wise, p cannot have been the final result). By Equation 4, roughly 15% of the points in the S *cannot* be pruned using the ir signatures, and thus, will become false hits. the also means that the NN algorithm is expected to perform at least 0.15|S| random I/Os. So far we have considered $|W_q| = 1$, but the discussion extends to arbitrary $|W_q|$ in the straightforward manner. It is easy to observe (based on the Equation 4) that, in the general, the false hit probability satisfies

$$P_{false} \geq 0.15^{|W_q|}. \qquad (5)$$

When $|W_q| > 1$, there is ano the r negative fact that adds to the deficiency of the IR^2-tree: for the greater $|W_q|$, the we expected size of S increases dramatically, because fewer and fewer objects will contain all the query keywords. The effect is so severe that the number of random accesses, given by $P_{false}|S|$, may escalate has $|W_q|$ grows (even with the decrease of P_{false}). In the fact, has long has $|W_q| > 1$, S be the entire dataset when the user tries out an uncommon combination of keywords that does not exist in the any object. in the case, the number of random I/Os would be so prohibitive that the IR^2-tree would not be able to give real time responses.

5. Merging and Distance Browsing

Since verification is the performance bottleneck, we should try to avoid it. There is the simple way to do so in the I-index: one only needs to store the coordinates of each point together with each of its appearances in the inverted lists. the presence of coordinates in the inverted lists naturally motivates the creation of an R-tree on the each list indexing the points the rein (a structure reminiscent of the one in the [21]). Next, we discuss how to perform keyword-based nearest neighbor search with such the combined structure.

The R-trees allow us to remedy awkwardness in the way NN queries are processed with an I-index. Recall that, to answer the query, currently we have to first get all the points carrying all the query words in the W_q by merging several lists (one for each word in the W_q). It appears to be unreasonable if the point, say p, of the final result lies fairly close to the query point q. It would be great if we could discover p very soon in the all the relevant lists so that the algorithm can terminate right away.

The would become the reality if we could browse the lists synchronously by *distances* has opposed to by ids. in the particular, has long as we could access the points of all lists in the ascending order of the ir distances to q (breaking ties by ids), such the p would be easily discovered has its copies in the all the lists would definitely emerge *consecutively* in the our access order. So all we have to do is to keep counting how many copies of the same point have popped up continuously, and terminate by reporting the point once the count reaches $|W_q|$. At any moment, it is enough to remember only one count, because whenever the new point emerges, it is safe to forget about the previous one.

Distance browsing is easy with R-trees. in the fact, the best-first algorithm is exactly designed to output data points in the ascending order of the ir distances to q. How-ever, we must coordinate the execution of best-first on the $|W_q|$ R-trees to obtain the global access order. the can be easily achieved by, for example, at each step taking the "peek" at the next point to be returned from each tree, and output the one that should come next globally. The algorithm is expected to work well if the query keyword set W_q is small. For sizable W_q , the large number of random accesses it performs may over when all the gains over the sequential algorithm with merging.

6. Spatial Inverted List

The spatial *inverted list* (SI-index) is essentially the compressed version of an I-index with embedded coordinates has described in the Section 5. Query processing with an SI-index can be done either by merging, or together with R-trees in the distance browsing manner. Furthermore, the compression eliminates the defect of the conventional I-index such that an SI-index consumes much less space.

6.1 The compression scheme

To attack the problem, let us first leave out the ids and focus on the coordinates. Even though each point had 2 coordinates, we can convert the m into only one so that gap-keeping can be applied effectively. The tool needed is the space filling curve (SFC) such has Hilbert- or Z-curve. SFC converts the multidimensional point to the 1D value such that if two points are close in the original space, then their 1D value also tend to be similar. Has dimensionality had been brought to 1, gap-keeping works nicely after sorting the (converted) 1D values.

Let us put the ids back into consideration. Now that we have successfully dealt with the two coordinates with the 2D SFC, it would be natural to think about using the 3D SFC to cope with ids too. Has far has space reduction is concerned, the 3D approach may not be the bad solution. The problem is it will destroy locality of the points in the their original space. Specifically, the converted values would no longer preserve the spatial proximity of the points, because ids in the general have nothing to do with coordinates.

7. Experiments

In the sequel way we experimentally evaluate the practical efficiency of our solutions to NN search with key-words, and compare them against the existing methods.

a) **Competitors.** The proposed SI-index comes with two query algorithms based on the merging and distance browsing respectively. We will refer to the former has *SI-m* and the o the r has *SI-b*. Our evaluation also covers the state-of- the -art *IR^2-tree*; in the particular, our IR^2-tree implementation is the fast variant developed in the [12], which uses longer signatures for higher levels of tree. Furthermore, we also include the method, named *index file R-tree* (*IFR*) henceforth, which, has discussed in the Section 5, indexes each inverted list (with coordinates embedded) using an R-tree, and applies distance browsing for query processing. *IFR* can be regarded has an uncompressed version of *SI-b*.

b) **Data.** Our experiments are based on the both synthetic and real data. The dimensionality is always 2, with each axis consisting of integers from 0 to 16383. The synthetic category had two datasets: *Uniform* and *Skew*, which differ in the distribution of data points, and in the whether there is the correlation between the spatial distribution and objects' text documents. Specifically, each dataset had 1 million points. Then their locations are uniformly distributed in the *Uniform*, whereas in the *Skew*, the y follows the Zipf distri-bution^3. For both datasets, the vocabulary had 200 words, and each word appears in the text documents of 50k points. The difference is that the association of words with points is completely random in the *Uniform*, while in the *Skew*, there is the pattern of "word-locality": points that are spatially close have almost identical text documents.

8. Conclusions

We have seen plenty of applications calling for the search engine that is able to efficiently support novel forms of spatial queries that are integrated with the keyword search. The existing solutions to such queries ei ther incur prohibitive space consumption or are unable to give real time answers. in the 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 had the ability to perform keyword-augmented nearest neighbor search in the time that is at the order of dozens of milli-seconds. Fur the rmore, has the SI-index is based on the conventional technology of inverted index, it is readily incorporable in the commercial search engine that applies on massive parallelism,on implying its immediate industrial merits

References

[1] C. Faloutsos and S. Christodoulakis as. Signature files: An access method for documents and its analytical performance evaluation.ACM Transactions on the Information Systems (TOIS), 2(4):267–288, 1984.

[2] E. Chu, A. Doan, and J. Naughton. The Combining keyword search and forms for ad hoc querying of

databases. In Proc. of ACM Management of Data (SIGMOD), 2009.
[3] G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, and S. Sudar-shan. Keyword searching and the browsing in the databases using banks. in the *Proc. of International Conference on the Data Engineering (ICDE)*, pages 431–440, 2002.
[4] G. R. Hjaltason and H. Samet. Distance browsing in the spa-tial databases. *ACM Transactions on the Database Systems (TODS)*, 24(2):265–318, 1999.
[5] J. Lu, Y. Lu, and G. Cong. Reverse spatial and textual k nearest neighbor search. in the *Proc. of ACM Management of Data (SIGMOD)*, pages 349–360, 2011.
[6] J. S. Vitter. Algorithms and data structures for external memory.
[7] X. Cao, G. Cong, and C. S. Jensen. Retrieving the top-k prestige-based relevant on the spatial, 2010.
[8] X. Cao, G. Cong, C. S. Jensen, and B. C. Ooi. Collective spatial keyword querying. in the *Proc. of ACM Management of Data (SIG-MOD)*, pages 373–384, 2011.
[9] X. Cao, L. Chen, G. Cong, C. S. Jensen, Q. Qu, A. Skovsgaard, D. Wu, and M. L. Yiu. Spatial keyword querying. in the *ER*, pages 16–29, 2012.
[10] Y.-Y. Chen, T. Suel, and A. Markowetz. Efficient query processing in the geographic web search engines. in the *Proc. of ACM Management of Data (SIGMOD)*, pages 277–288, 2006.
[11] B. Chazelle, J. Kilian, R. Rubinfeld, and the A. Tal. the bloomier filter: efficient data structure for static support lookup tables. In the *Proc. of the Annual ACM-SIAM Symposium on the Discrete Algorithms (SODA)*, pages 30–39, 2004.
[12] D. Zhang, Y. M. Chee, A. Mondal, A. K. H. Tung, and M. Kitsure-gawa. Keyword search in the spatial databases: Towards searching by document. in the *Proc. of International Conference on the Data Engineering (ICDE)*, pages 688–699, 2009.
[13] G. Cong, C. S. Jensen, and D. Wu. Efficient retrieval of top-k.
[14] I. D. Felipe, V. Hristidis, and N. Rishe. Keyword search on the spatial databases. in the *Proc. of International Conference on the Data Engineering (ICDE)*, pages 656–665, 2008.
[15] I. Kamel and C. Faloutsos. Hilbert R-tree: An improved r-tree using fractals. in the *Proc. of Very Large Data Bases (VLDB)*, pages 500–509, 1994.
[16] N. Beckmann, H. Kriegel, R. Schneider, and the B. Seeger. the R*-tree: efficient and the robust access method for points and rectangles. in the *Proc. of ACM Management of the Data (SIGMOD)*, pages 322–331, 1990.
[17] R. Hariharan, C. Li, and S. Mehrotra. For Processing spatial-keyword (SK) queries in the geographic information retrieval (GIR) systems. in the *Proc. of Scientific and Statistical Database Management (SSDBM)*, 2007.
[18] S. Agrawal, S. Chaudhuri, and the G. Das. Dbxplorer: the system for keyword-based search over relational databases. in the *Proc. of International Conference on the Data Engineering (ICDE)*, pages 5–16, 2002.
[19] S. Stiassny. mathematical analysis of various superimposed on the coding. *Am. Doc.*, 11(2):155–169, 1960.
[20] V. Hristidis and Y. Papakonstantinou. Discover: Keyword search in the relational databases. in the *Proc. of Very Large Data Bases (VLDB)*, pages 670–681, 2002.

Author Profile

Vangala Uday Bhaskar received the B.Tech degree in computer science and Engineering Degree from JNTU Hyderabad in 2012 and pursuing M. Tech. Degree in Computer science and Engineering from Anurag Group Of Institutions Venkatapur (V), Ghatkesar (M), Ranga Reddy District, JNTU Hyderabad-500088, Telangana State.

Garine Bindu Madhavi working as Assistant Professor in Computer Science and Engineering from CVSR College Of Engineering from Anurag Group of Institutions Venkatapur(V), Ghatkesar(M), Ranga Reddy District, Hyderabad-500088, Telangana State.