Searching Nearest Neighbor Based on Keywords using Spatial Inverted Index

Shilpa B. Patil [1], Sushma S. Nandgaonkar [2]

[1] ME-II Computer Engineering, Savitribai Phule Pune University,Vidya Pratishthan's College of Engineering, Baramati,

[2] Professor, Department of Computer, Savitribai Phule Pune University, Vidya Pratishthan's College of Engineering, Baramati,

Abstract: ***With rapid growth of mobile users it is essential to optimize nearest neighbor retrieval. Even though many approaches available for searching the spatial query, modern technology requires some enhancements for the optimality. The results based on the keywords, and spatial query are combinations of a location and set of features. System returns the nearest neighbor of users current location based on specified query keyword and for speeding up the process of spatial queries, IR2-tree is used. There are some drawbacks of this data structure. In order to overcome this spatial inverted index can be used which improves query processing significantly. However, it works with single level gap-keeping. In this paper we use two level gap-keeping to save space cost.***

Keywords: Keyword Search, Spatial Data Mining, Nearest Neighbor Search, Index Compression

1. Introduction

A spatial database manages the multidimensional objects (such as 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 as 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 general 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, whereas 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 will be fairly useful if the search engine can be 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 retrieve 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.

To overcome the drawbacks of IR2-tree, recently, Tao and Sheng [1] proposed a spatial inverted index based on conventional inverted index. As mention above it uses spatial inverted index which is essentially compressed version of an I-index. Compression eliminates the defect of inverted index such that an SI-index consumes much less space and also it is widely used to minimize the size of an inverted index where each inverted list contains only IDs. In such cases, an effective method is used to record gap between IDs called gap-keeping method. In this paper we proposed a two level gap-keeping method. The space cost is less when two level gap-keeping method is used to build SI-index. We built a spatial inverted index using compression method two level gap-keeping that consumes less space than single level gap-keeping and also the space cost is less when SI-index is used. It demonstrates the usefulness of the two level gap-keeping methods for fast nearest neighbor search. Our contributions in this paper is building a spatial inverted index using two level gap-keeping approach based on the concepts proposed by Tao and Sheng [1].

2. Literature Survey

A. Keyword search on spatial databases

This method focuses on searching top-k nearest neighbor query. They present an efficient method to answer top-k spatial keyword queries, it introduce an indexing structure called IR2-Tree (Information Retrieval R-Tree). IR2-tree is the most related work which is similar to our work. Here incremental algorithm is used to answer Top-k spatial keyword queries using the IR2 -Tree. The IR2-tree combines the features of R-tree with signature files. Signature file is nothing but a hashing-based framework that is based on the concept of superimposed coding as explored in this method. The problem with this technique is that as the number of words grows in size, scanning the entire list become tedious. When the list is not scanned it may result in false hits. As a solution to this problem inverted indexes were introduced in [5].

B. Spatial Keyword Query

This hybrid index structure is used to search m-closest keywords. This technique finds the closest tuples that matches the keywords provided by the user. This structure combines the R*-tree and bitmap indexing to process the m-closest keyword query that returns the spatially closest objects matching m keywords.To reduce the search space a priori based search strategy is used. Two monotone constraints are used as a priori properties to facilitates efficient pruning which is called as distance mutex and keyword mutex. But this approach is not suitable for handling ranking queries and in this number of false hits is large.[2]

C. Processing Spatial-Keyword (SK) Queries in Geographic Information Retrieval (GIR) Systems

Location based information is stored in GIS database. These information entities of such databases have both spatial and textual descriptions. This method introduces a framework for GIR system and focus is on indexing strategies that can process spatial keyword query. It introduces two index structures to store spatial and textual information.1) Separate index for spatial and text attributes 2) Hybrid index. But by using first structure that is separate index for spatial and text attributes, if filtering is done first, many objects may lie within a query is spatial extent, but very few of them are relevant to query keywords. This increases the disk access cost by generating a large number of candidate objects. The subsequent stage of keyword filtering becomes expensive. And by using second structure that is hybrid index there are high overhead in subsequent merging process. Idea of geographical web search was illustrated in [2], [4] [5] and [6].

D. Hybrid Index Structures for Location-based Web Search

There is more and more research interest in location-based web search, i.e. searching web content whose topic is related to a particular place or region. This type of search contains location information, it should be indexed as well as text information, text search engine is set-oriented where as location information is two-dimensional and in Euclidean space. In previous technique we see same two indexes for spatial as well as text information this creates new problem, i.e. how to combine two types of indexes. This method uses hybrid index structure, to handle textual and location based queries, with help of inverted files and R*-trees. It considered three strategies to combine these indexes namely:1) Inverted file and R*-tree double index.2) First inverted file then R*-tree.3) First R*-tree then inverted file. It implements search engine to check performance of hybrid index structure, that contains four parts:(1) an extractor which detects geographical scopes of web pages and represents geographical scopes as multiple MBRs based on geographical coordinates. (2) An indexer which builds hybrid index structures to integrate text and location information. (3) A ranker which ranks results by geographical relevance as well as non-geographical relevance. (4) An interface which is friendly for users to input location-based search queries and to obtain geographical and textual relevant results. [2]

E. Collective Spatial Keyword Querying

This technique use new type of queries, called spatial group keyword queries, that find groups of objects that collectively satisfy a query. Here two instantiations of the problem and show that both are NP-complete. This method introduce algorithms that offer approximate solutions to the two sub problems with provable approximation bounds. And also present exact algorithms for the two sub problems. All algorithms exploit a spatial keyword index to prune the search space. The results demonstrate that the proposals offer scalability and are capable of excellent performance.[8]

3. Implementation Details

A user enters query for search, then query preprocessing is performed to remove stop-words and non-words from that query. Then SI-indexer builds index. To compress SI- index gap-keeping method is used where each element of a list, a point p, is a triplet. But gap-Keeping applied on only one attribute of the triplet. So, first 2D Z-curve values are considered then 3D Z-curve values are recorded values with single level gap-keeping. To make more compressed SI-index again generate index with two level gap-keeping, then search query and return results. The existing system uses SI-index and compressed index using gap keeping in single level. But further compression to second level consumes much less space as compared to single level and save space cost.

The proposed system, process query in two ways that is dynamic search and static search. When users go through the first way i.e. dynamic search then it uses the results retrieved using web mapping service known as “Google Maps” powered by Google.

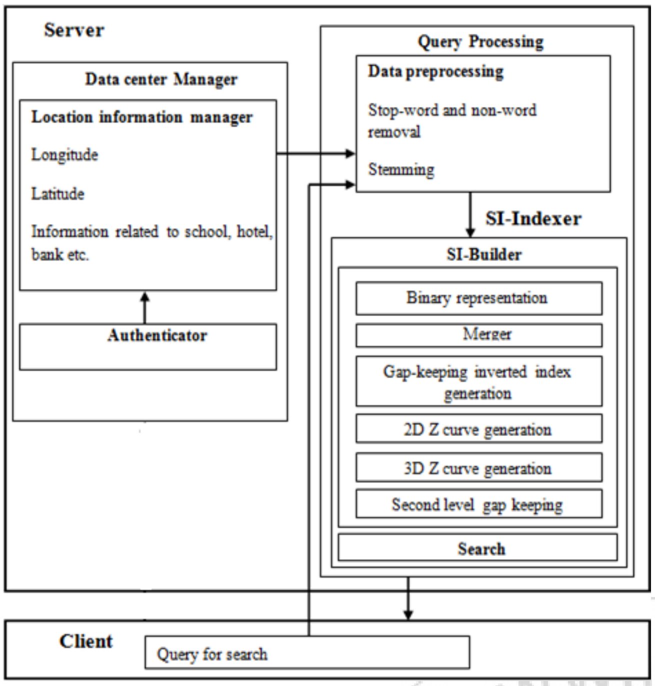


Figure 1: System architecture

A . Data center manager

In first module location information manager use "Google Map" of particular area, by selecting any location on that map it gives longitude and latitude. The retrieved longitude and latitude saved into the database.It also includes information related to school, bank, hotel, restaurant, college etc. Data preprocessing is applied on the data that is managed by information manager.

B. Query preprocessing

1. Data preprocessing

When user enter the query to search then it is passed to the data preprocessing, first preprocessing of the query is stop-words removal from that query. Stop-words are frequently occurring and unimportant words in a language that helps to construct sentences but do not represent any content of the documents. Stop-words in include: a, about, an, are, as, at, be, by, for, from, how, in, is, of, on, or, that, the, these, this, to, was, what, when, where, who, will, with. Such words should be removed before documents are indexed and stored. For example if user query is "search for nearest school" then remove "for" word from that query. Next step is stemming which is the process of reducing words to their stems or roots. A stem is the portion of a word that is left after removing its prefixes and suffixes. For example if user query is "search nearest engineering colleges" then remove "s" from colleges it remains stem that is "college".

2. SI-Indexer

The proposed system is based on the spatial inverted index. SI-index is an compressed version of I-index with compressed coordinates. Compression eliminates the defect of I-index such that SI-index consumes much less space. Compression methods are used in order to reduce size of index where each inverted index contains only IDs. To build SI-index we have used an effective method to record gaps between consecutive IDs when compared with precise ids. For example, consider a set of integers {2,3,6,8}, the gap-keeping method stores values such as {2,1,3,2} illustrate in [1]. To calculate gap between ids and coordinates of points, first it takes binary form of id and coordinates of points. Then merge binary numbers to get converted values of points based on Z-curve.

Let (x , y) be the point, then first apply $f(x)$ and $f(y)$on x and y values, where f is the function which converts decimal to binary. For example, p_2 $(3,3)$as shown in Fig.1 that is $f(3) = 011$, $f(3) = 011$.Then let consider output of $f(x)$or $f(y)$ as set of binary bits $\{x_1 x_2 \dots \dots x_n\}$ where x_i is bit having value output.

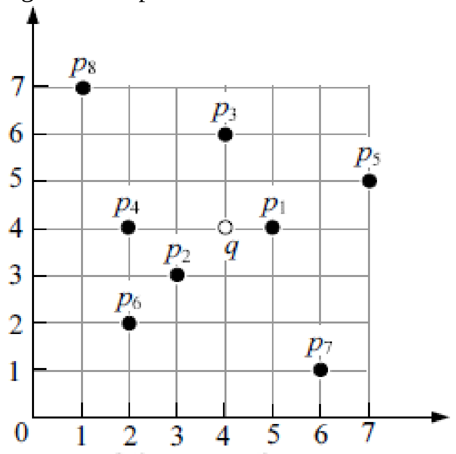


Figure 2: Shows locations of points

Next we merge $f(x)$and $f(y)$

$m(f(x), f(y))$ ………………………………........ $eq(1)$

Where, m is the merger function will do bit by bit merging of $f(x)$ and$f(y)$ output.

So if $f(x) = \{x_1 x_2 \dots \dots x_n\}$, $f(x) = \{y_1 y_2 \dots \dots y_n\}$

Therefore,

$m(f(x), f(y)) = \{x_1 y_1 x_2 y_2 \dots \dots x_n y_n\}$…….. $eq(2)$

Then $d(m)$ be the function which converts bits to decimal. For example,

$f(3) = 011$

$f(3) = 011$

$m(f(3), f(3)) = 001111$

Therefore $d(m) = 15$

p_6	p_2	p_8	p_4	p_7	p_1	p_3	p_5
12	15	23	24	41	50	52	59

Fig.3 Converted values of the points in Fig.4 based on Z-curve.

The compressed spatial inverted index has a triplet to represent both ID and coordinates of each point. But gap-keeping is applied on only attribute of triplet. So it is not simple solution. To tackle this problem we first focuses on the coordinates. Even though each point has two coordinates, we can convert that coordinates into only one attribute so that gap-keeping can applied effectively. A space filling curve (Z-curve) is constructed for efficient processing, this is called 2D Z-curve generation. Pseudo ids are used instead of real id and form 3D Z-curve. This is done for better sorting procedure. Values in Z-curve are sorted.

Let element in set $e = \{e_1, e_2, \dots, e_n\}$ integer values

Inverted index at e,

$e = e^{th}\ value - \ (e-1)^{th}\ value$ .... $eq\,(3)$

Except first element.

$e = e^{th}\ value + (e-1)^{th}\ value$ .... $eq\,(4)$

This gives first level gap-keeping values. Fig. 3 that contains p2, p3, p6, p8, whose Z-values are 15,52,12,23 respectively, and pseudo-ids being 1,6,0,2 respectively. After sorting Z-values automatically the pseudo ids are kept in ascending order. Gap-keeping is applied on both Z-values and pseudo-ids, it results in 12,3,8,29 and 0,1,1,4 respectively. We can capture the 4 points with 4 pairs: {(0,12),(1,3),(1,8),(4,49)}.

Two level gap-keeping method applied on values obtained in first level gap-keeping that is both values Z-values and pseudo-ids like 12,3,8,29 and 0,1,1,4 respectively. It gives result as 3,5,4,17 and 0,1,0,3 respectively. We can capture the 4 points with 4 pairs:{(0,3),(1,5),(0,4),(3,17)}. Second level gap between values is calculated using same formula that are used to calculate first level gap between values.

Let elements in set $i = \{i_1, i_2, \dots, i_n\}$ integer values.

Inverted index as i,

$i = i^{th}\ value - \ (i-1)^{th}\ value$ .... $eq\,(4)$

Except first element.

$i = i^{th}\ value + (i-1)^{th}\ value$ .... $eq\,(5)$

In our contribution we have built a spatial inverted index using compression method two level gap-keeping that consumes less space than single level gap-keeping and also the space cost is less when SI-index is used.

C. Searching

ID	Point name	Words
0	p0	a, b, c
1	p1	b, d, f
2	p2	d
3	p3	e, g, c
4	p4	c, d, e, f, g
5	p5	a, g
6	p6	b, f
7	p7	c, e
8	p8	d, g

Figure 4: Example of inverted index

When user enter query, then that query contains keyword and two coordinates that is (x, y). We have points (i.e two co-ordinates) and related keywords in our database which are taken from Google Map. Then search query into inverted index, it gives us points that contain query keyword. After that search for nearest one from that points. Above example shows example of inverted index that have points and related words.

D. Mobile App

User can enter a query through Mobile App.GPS location tracker can track current location information. And, finally results are stored on server.

E. Mathematical Module

Let q be the set of query keywords, these keywords are input to the system for search.

$q = \{q_1, q_2, \dots\dots q_n\}$

Let r be the dataset which is either static or dynamic, it gives preprocessing result after searching query keywords into dataset.

$r = \{r_1, r_2, \dots\dots\dots r_2\}$

Let d_i be the distance, it is preprocessing result getting after searching query in dataset.

$d_i = \{d_1, d_2, \dots\dots\dots\dots\dots\dots d_n\}$

$d_i = \text{search}(r, q)$

Let d_{ni} be the nearest distance, which is calculated by applying k-nearest neighbor algorithm on distance d_i, which is preprocessing result of search.

Where,

$d_{ni} = \text{kNN}(d_i)$

The d_i gives distances which are result of our query but it returns number of results, form that we have to calculate minimum distance from current location so, the algorithm kNN apply on preprocessing results to calculate minimum distance from current location.

F. Algorithm

Algorithm 1 SI index Building Algorithm

Output: Semantic nearest neighbor.
Description:

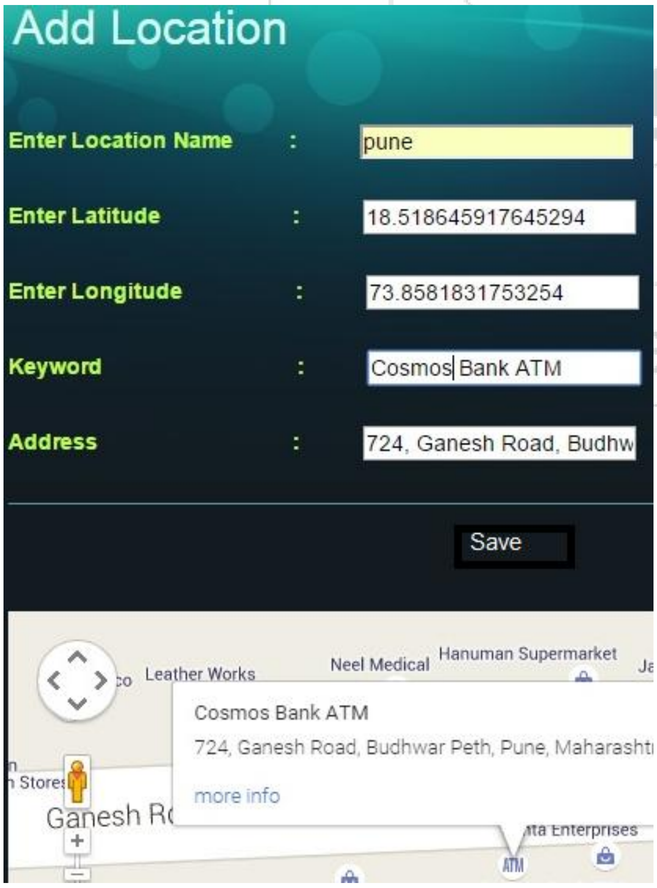
```
1: Read three values as (id, x, y)
        Where, id - id of place or word
        x - Position on x-axis of place
        y - Position on y-axis of place

        //2D gap-keeping
2: Apply gap-keeping on x and y first as following,
        a. Read x
        b. Let 2D = {b1b2b3.......bn}
        Convert x to binary values as,
        b1= binary(x);
        c. Repeat step a and b for y and create b2.
        d. Merge b1and b2bit by bit and store in b3
                For each bit in b1and b2,
            B = b1&b2;
        e. Convert B to decimal.
        f. Generate Z-curve using gap-keeping.
3: Repeat step 1 and 2 for all places.
4: Generate sorted 2D Z-curve using gap-keeping.


//3D gap keeping
5: Repeat 1 and 2 for id as x and value from set 2D as y
   will store merge results
        Let, 3D = {c1, c2.......cn}
6: Generate 3D Z-curve using gap-keeping on set 3D.
7: Apply 2-level gap-keeping on set 3D.
```

4. Experimental Setup

To search nearest neighbor, system requires to track current location of the user. It can be tracked by using GPS location tracker.

5. Result and Dataset

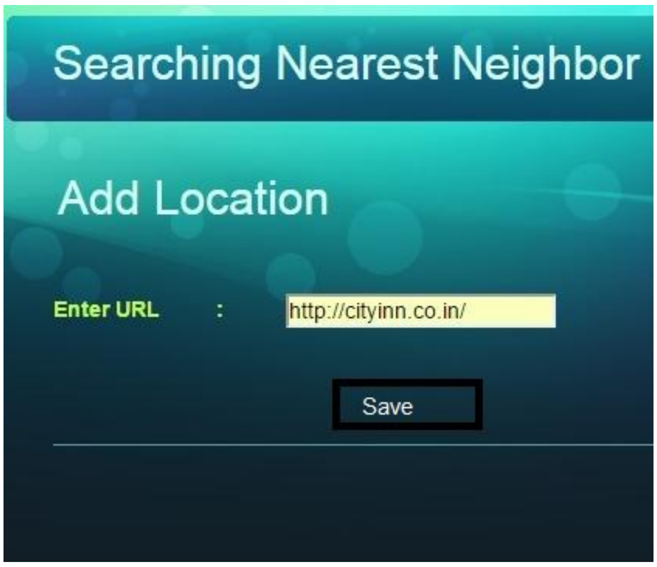
We have used "Google Map" of particular area, by selecting any location on that map it gives longitude and latitude. That longitude and latitude is saved into the database. It also includes information related to hospital, school, bank, hotel, restaurant, college etc. Fig 5 shows that we can add location information using "Google map". In Fig 6. we can add different categories like hotel from their websites. And make list.



Enter Location Name : pune
Enter Latitude : 18.518645917645294
Enter Longitude : 73.8581831753254
Keyword : Cosmos Bank ATM
Address : 724, Ganesh Road, Budhw

Cosmos Bank ATM
724, Ganesh Road, Budhwar Peth, Pune, Maharasht

Figure 5: Add Location Information using Google Map



Enter URL : http://cityinn.co.in/

Figure 6: Add Location Information using Website

6. Conclusions

In this paper we mainly focus on spatial data mining technique. Spatial inverted Index structure is used to deal with the problem of IR2-tree. Compression of SI-index has done using Gap-keeping method. This method can't be applied on triplet. So firstly consider 2D Z-curve values and then 3D Z-curve values. And to calculate these Z-curve values there are two steps that is binary representation and merging. In this paper we used two level gap-keeping to save space cost.

7. Acknowledgement

I express great many thanks to Prof. S. S. Nandgaonkar for his great effort of supervising and leading me, to accomplish this fine work. Also to college and department staff, they were a great source of support and encouragement. To my friends and family, for their warm, kind encourages and loves. To every person gave us something too light my pathway, I thanks for believing in me.

References

[1] Yufei Tao and Cheng Sheng, "Fast Nearest Neighbor Search with Keywords," IEEE Transactions on Knowledge and Data Engineering, 2013, p1-13.
[2] D. Zhang, Y.M. Chee, A. Mondal, A.K.H. Tung, and M. Kitsuregawa, "Keyword Search in Spatial Databases: Towards Searching by Document," Proc. Int'l Conf. Data Eng. (ICDE), pp. 688-699, 2009.
[3] Y. Zhou, X. Xie, C. Wang, Y. Gong, and W.-Y. Ma, "Hybrid index structures for location-based web search," In Proc. of Conference on Information and Knowledge Management (CIKM), pages 155–162, 2005
[4] R. Hariharan, B. Hore, C. Li, and S. Mehrotra, "Processing spatial- keyword (SK) queries in geographic information retrieval (GIR) systems," In Proc. of Scientific and Statistical Database Management (SSDBM), 2007.

[5] I. D. Felipe, V. Hristidis, and N. Rishe, “Keyword search on spatial databases,” In Proc. of International Conference on Data Engineering (ICDE), pages 656–665, 2008.

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

[7] 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.

[8] X. Cao, G. Cong, C.S. Jensen, and B.C. Ooi, “Collective Spatial Keyword Querying,” Proc. ACM SIGMOD Int’l Conf. Management of Data, pp. 373-384, 2011.