ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 6, Issue 10, October 2017

A Survey on Nearest Keyword Set Search in Multi-Dimensional Datasets

Twinkle Pardeshi, Prof. Pradnyakulkarni

Department of Computer Engineering, Maharashtra Institute of Technology, Pune, Maharashtra, India

ABSTRACT- A spatial database is a database that store data that represents objects defined in a geometric space. This paper considers that objects are tagged with keyword and are embedded in a vector space. Due to growing data, Nearest Keyword Set(NKS) queries are used in many applications. These queries are studied that retrieves the tightest groups of points that contains a given set of keywords. Various methods for fast processing of queries are been studied.

KEYWORDS: Querying, Multi-dimensional data, Indexing, Hashing, Nearest Keyword Set (NKS).

1. INTRODUCTION

Keyword based search in multi-dimensional datasets facilitates many novel applications and tools. Objects are considered to be tagged with keywords and are embedded in vector space. Objects are the collections of various features and are represented points in multi-dimensional feature space.Current research on queriesgoes well beyond pure spatial queries such as nearest neighbor queries, range queries, and spatial joins . Queries on spatial objects are represented by sets of keywords that are beginning to receive significant attention from the spatial database research community and the industry. Queries on spatial objects are associated with textual information that are represented by a set of keywords, have received significant attentionThe main focus is on the Nearest Keyword Set(NKS) queries text-rich multi-dimensional datasets. In NKS query, user provides set of keywords and query retrieves set of points that contains those keywords.These queries are used in the applications such as in photo sharing social network, where photos are tagged with people names and even location. NKS query is used to find similar photos based on tagged data. In graph pattern search, query can be used to find the subgraph with set of specified labels. In geographical patterns, query can be used to find the patterns by finding set of similar regions with respect to provide user query keywords. Various techniques have evolved for fast processing of queries. There are various types of queries and for the same various indexing techniques are also been developed. Location specific queries are answered using R-Tree and inverted index. Later on, IR^2 –Tree use to rank the objects from spatial dataset based on the distance relevancy and relevance of their text descriptions to query keywords. Various algorithms were also developed to retrieve group of spatial web objects such that the retrieved group of objects covers all the keywords in the query been requested. Other related queries include aggregate nearest keyword search in spatial database, top-k sites in spatial data based on their influence on feature points, and optimal location queries.

II. NEAREST NEIGHBOR SEARCH TECHNIQUES

- **IR-Tree, Approximation Algorithm and ExactAlgorithm**

To retrieve group of spatial objects, satisfying the submitted user query IR- Tree method is used. It returns the objects nearer to the user query and have lowest inter object distance. There are total two phases that are included in nearest neighbor search techniques. The first phase finds the group of objects such that sum of their distances is minimized. The second phase includes to find the group of objects that covers the keywords such that sum of the maximum distance among an object in group of objects and query and maximum distance among two objects in group of objects is minimized. Both of these sub problems are NP-complete. Greedy algorithm is used to provide an approximation solution to the problem that utilizes the spatial keyword index IR-tree to reduce the search space. But in some

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 6, Issue 10, October 2017

application query does not contain a large number of keywords, for this exact algorithm is used that uses the dynamic programming.

- **IUR-Tree (Intersection union R-tree)**

Geographic objects associated with descriptive texts are becoming common. This gives importance to spatialkeyword queries that take both the location and text description of content. This technique is used to analyzethe problem of reverse spatial and textual k nearest neighbor search i.e. finding objects that takes the query object as one of their spatial textual similar objects. For this type of search hybrid index structure is used thatsuccessfully merge the location proximity with textual similarity. For searching, branch and bound algorithm isused. In addition to increase the speed of query processing a variant of IURtree and two optimization algorithm isused. To enhance the IUR-tree text clustering is used, in this objects of all the data base is group into clustersaccording to their text similarity. Each node of the tree is extended by the cluster information to create a hybrid treewhich is called as cluster IUR-tree. To enhance the search performance of this tree two optimization methods isused, first is based on outlier detection and extraction and second method is based on text entropy.

- **BR*-Tree**

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

- **IR^2-Tree**

The growing number of applications requires the efficient execution of nearest neighbor queries which is constrained by the properties of spatial objects. Keyword search is very popular on the internet so these applicationsallow users to give list of keywords that spatial objects should contain. Such queries called as a spatial keyword query. This is consisted of query area and set of keywords. The IR2-tree is developed by the combination of R-treeand signature files, where each node of tree has spatial and keyword information. This method is efficientlyanswering the top-k spatial keyword queries. In this signature is added to the every node of the tree. An ablealgorithm is used to answer the queries using the tree. Incremental nearest algorithm is used for the tree traversal and if root node signature does not match the query signature then it prunes the whole subtrees. But IR2-tree has some drawbacks such as false hits where the object of final result is far away from the query or this isnot suitable for handling ranking queries. Spatial Inverted IndexesA new access method spatial inverted access method is used to remove the drawbacks of previous methods suchas false hits. This method is the variant of inverted index using for multidimensional points. This index stores thespatial region of data points and on every inverted list Rtree is built. Minimum bounding method is used fortraversing the tree to prune the search space.

III. APPROXIMATION ALGORITHM

Weighted Set Cover (WSC) problem is been studied under this section. Basically , greedy algorithm is used to decompose the given query q dynamically into sequence of partial queries each contains different set of keywords depending upon preceding partial query and then each partial query is evaluated independently.This process continues till all the keywords from the provided user query are been covered. Due to this process, it requires to scan the whole dataset. The overcome this drawback method of IR- tree is used.

Algorithm 1: Type1Greedy(*q, irTree)*

1 1 $U \leftarrow$new min-priority queue;
2 U.Enqueue($irTree.root$,0);
3 $V \leftarrow \emptyset$; $Cost \leftarrow 0$;
4 $mSet \leftarrow q.\psi$; $pSet \leftarrow q.\psi$: ;

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 6, Issue 10, October 2017

5 while $mSet \neq \emptyset$ do
6 while *U is* not *empty* do
7 $e \leftarrow U$:Dequeue();
8 $Cost \leftarrow Cost + e.Key$;
9 if *e is an object* then
10 $V \leftarrow V \cup e$;
11 $pSet \leftarrow mSet$;
12 $mSet \leftarrow mSet / e.\psi$;
13 for *each entry e′ in U* do
14 if $e'.\lambda \cap e..\lambda \neq \emptyset$ then
15 e'.key = $\frac{e'.key * |e'.\lambda \cap mSet|}{|e'.\lambda \cap pSet|}$;
16 else remove *e* from *U*;
17 reorganize priority queue *U* using new key values;
18 break;
19 else
20 read the posting lists of *e* for keywords in *mSet*;
21 for *each entry e′ in node e* do
22 if $mSet \cap e'.\psi \neq \emptyset$ then
23 if *e is a non-leaf node* then
24 $U.\text{Enqueue}(e', \frac{\text{minDist}(q,e')}{|mSet \cap e'.\psi|})$;
25 else
26 $U.\text{Enqueue}(e', \frac{\text{Dist}(q,o)}{|mSet \cap o.\psi|})$;

Algorithm uses min-priority queue for best-first search with cost as key. *msetconatins the* current partial query and pSet contains set of preceding partial query. Best first search method is used to find out the overlapping objects and lowest cost. The algorithm computes the cost of an object and also cost of an non-object. Whenever an object is pop from U, it makes sure that the text description of object overlaps with mSet and the cost of object is lowest. Hence, it becomes part of the result.

IV. EXACT ALGORITHM

1. Exact algorithm without an index

The exact algorithm enumerates every subset of spatial objects whose text descriptions will be overlapping with the user provided query keyword. It finds the spatial objects that will contain all the query keywords and then evaluates its cost. The algorithm will compute cost for each subset. minValue contains the objects with the lowest cost. The algorithm scans whole dataset and then finds the best group. Therefore, it runs exponentially in terms of the number of query keywords and linearly in terms of the size of the dataset.

Algorithm 2: Type1ExactNoIndex (*q,D*)

1. $n \leftarrow |q.\psi|$;
2. for *i from* 1 *to* $2^n - 1$ do Cost[*i*]$\leftarrow \infty$, Group[i] $\leftarrow \emptyset$;

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 6, Issue 10, October 2017

3. for *each object* o_i *in D* do
4. if $o_i.\psi \cap q.\psi \neq \emptyset$then
5. Dist[o_i]←Dist (o_i q);
6. for*each subset* s_i *in* $o_i.\psi \cap q.\psi$do
7. I ←MapToInteger(s_i);
8. if Cost[i] >Dist [o_i] then
9. Cost[i]←Dist [o_i];
10. Group[i]←{o_i};
11. for *i from* 1 *to* $2^n - 1$ do
12. *minValue*← ∞, *bestSplit*←0;
13. for *j from* 1 *to* $i/2$ do
14. if *j* &*i* = *j* then
15. *S* ←Group[*j*] ∩Group[$i - j$];
16. *oDist*←0;
17. for *each object o in S* do
18. *oDist*←*oDist*+ Dist [*o*];
19. *cost*←Cost[*j*] + Cost[$i - j$] − *oDist*;
20. if *cost* <*minValue*then
21. *minValue*← *cost*;
22. *bestSplit*← *j*;
23. if Cost[*i*] >*minValue*then
24. Cost[*i*]←*minValue*;
25. Group[*i*]←Group[*bestSplit*] ∪Group[$i - bestSplit$];
26. return Cost[$2^n - 1$] and Group[$2^n - 1$]

2. Exact algorithm using an index

The dynamic programming algorithm needs to scan whole dataset with two drawbacks those are it computes unnecessary objects that does not contain the query keyword and all the objects whose text descriptions overlaps with the query keywords are scanned to obtain the lowest costs. To overcome the first drawback, the IR-tree method is used that retrieves only the objects that contain some query keywords while avoiding checking the unnecessary objects. For the second drawback, it is not always necessary to scan all the objects. The objects are process in the ascending order of their distance. However, if no such object exists in the dataset, the algorithm is still required to scan to the furthest object containing some query keywords before it can stop. The algorithm keeps a track of upper bounds of the subsets whose cost is not known. The IR-tree method is used for retrieving the nearest object that covers query keywords. Min-priority queue *U is use*to store the IR-tree nodes and objects, where their distances to the query are the keys. U is initialized to root node. An element is dequeue from U and keyword intersection is been computed. If the retrieved element is an non leaf index node, each of its child is checked to check whether it contains the query keywords or not. If it contains the query keyword it is inserted into U with it lowest cost. If the element is the leaf node it is been handled in the way as child nodes. If all the query keywords are been retrieved with it possible minimum cost the algorithm terminates.

Algorithm 3: **Type1ExactWIndex(*q*, *irTree*)**

1 *markedSet*← ∅, *valuedSet*← ∅ ;
2 *n*←|*q*.ψ| ;
3 for *i from* 1 *to* $2^n - 1$ do Cost[*i*]←∞, Group[*i*]←∅ ;
4 *U* ←new min-priority queue;
5 *U*.Enqueue(*irTree.root*, 0);
6 while *U is* not *empty* do
7 *e* ← *U:*Dequeue();

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 6, Issue 10, October 2017

8 $ks \leftarrow q.\psi \cap e.\psi$;
9 if $ks \notin markedSet$ then
10 if *e is a non-leaf node* then
11 foreach *entry e′ in node e* do
12 if $q.\psi \cap e'.\psi \neq \emptyset$ and $q.\psi \cap e'.\psi \notin markedSet$ then U.Enqueue(e', minDist(q, e'));
13 else if *e is a leaf node* then
14 foreach *object o in leaf node e* do
15 if $q.\psi \cap o.\psi \neq \emptyset$ and $q.\psi \cap o.\psi \notin markedSet$ then U.Enqueue(o, Dist(q, o));
16 else
17 foreach *set* $S \in valuedSet$ do
18 $i \leftarrow$ MapToInteger(S);
19 if Cost[i] <Dist(q, e) then
20 if $i = 2^n - 1$ then
21 return Cost[$2^n - 1$] and Group[$2^n - 1$];
22 $valuedSet \leftarrow valuedSet \setminus S$;
23 $markedSet \leftarrow markedSet \cup S$;
24 $tempSet \leftarrow \emptyset$;
25 foreach *subset* $ss \subseteq ks$ do
26 if $ss \notin markedSet$ then
27 $I \leftarrow$ MapToInteger(ss);
28 $markedSet \leftarrow markedSet \cup ss$;
29 $tempSet \leftarrow tempSet \cup ss$;
30 if $ss \in valuedSet$ then
31 $valuedSet \leftarrow valuedSet \setminus ss$;
32 Cost[i] $\leftarrow$ Dist (e, q);
33 Group[i] $\leftarrow$ {e};
34 if $j = 2^n - 1$ then
35 return Cost[$2^n - 1$] and Group[$2^n - 1$];
36 foreach *set* $ts \in tempSet$ do
37 $j \leftarrow$ MapToInteger(ts);
38 for i *from* 1 *to* $2^n - 1$ do
39 if Cost[i] = ∞ then continue;
40 $unionKey \leftarrow i|j$;
41 if $unionKey = i \wedge unionKey = j$ then
42 continue ;
43 $D \leftarrow$ Cost[i] + Dist (e, q);
44 if Cost[$unionKey$] > D then
45 Cost[$unionKey$] $\leftarrow D$;
46 Group[$unionKey$] $\leftarrow$ Group[i] $\cap$ {e};
47 eturn Cost[$2^n - 1$] and Group[$2^n - 1$];

V. DESIGN OF THE SEARCH ENGINE

System is always been designed to enable the user to search and browse the resources easily. There are three main components in the design framework.

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 6, Issue 10, October 2017

Index Engine

Index Engine supports tag matching on the data resources derived from other applications. Data retrieval is the vital steps to be carried out in processing the data. Indexing is the best way to retrieve the data.

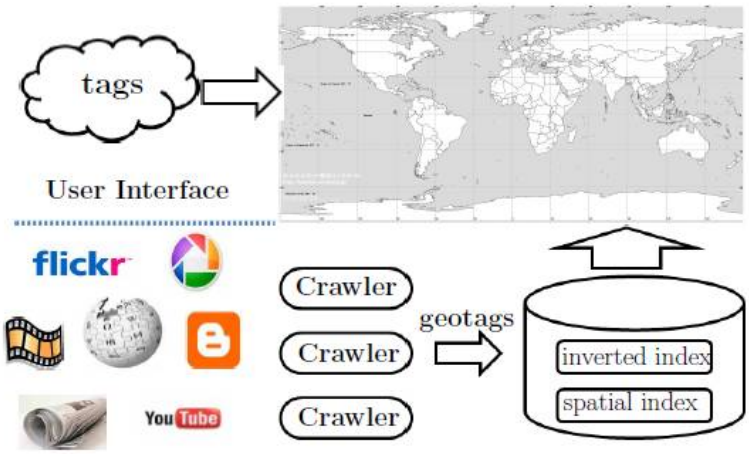

Fig 1.The framework of location detecting in Web 2.0 applications

The Query Engine

The query engine will allow the user to submit the query to find appropriate location based resource. Query is nothing but collection of tags associated with the resources. It also supports resource locating. The area or region can be explicitly specified by the user manually or can be automatically detected. index is actually inverted list, all the elements that do not lie in the query region will be eliminated to ensure all the candidates are within the boundary.

User Interface

The search engine is designed to be simple and friendly. The whole interface is map-based. Users can freely explore geo-resources in the area they are interested in. The main search entry is only a simple textbox to accept query tags. All the related resources will be displayed directly
in the map for further refinement.

VI. BOTTOM-UP SEARCH ALGORITHM

The m Closest Keyword query i.emCK is introduced in this section. This query helps to find m closest keywords in the database. Various indexing techniques are been used to make this happen. R*- Tree and bR* -Tree are one of the indexing used here. In bR* -Tree, each node consists of two information those are keyword bitmap and keyword MBR(Minimum Bounding Rectangle). Keyword bitmap indicates whether the keyword is present or not and MBR bounds all resources associated with keywords. Lowest distance is been calculated among each keywords. To deal location associated tags R* -Tree is used. Location which is nearest to each other is retrieve as an output. Inverted index is been used along with R* Tree. Therefore, only those nodes are computed in which query keyword is present.The virtual bR* -Tree is built level by level. m inverted list be retrieve and then list is merged in one. The list is traverse and data points with same label are fetched. Then these data points are use to create a new virtual node.

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 6, Issue 10, October 2017

Algorithm 4 Bottom-Up Search Strategy

Input: *m* query tags, inverted index
Output: Distance and location of *m*closest keywords

1. Retrieve m inverted lists for each query tag
2. Merge the lists into one list *L* ordered by the label
3. Initialize a virtual node *cur node*
4. **while***L.level*<*tree.height***do**
5. **for** each element *vnode*in *L* **do**
6. **if***vnode*has the same label with its previous element
then
7. add*vnode*into *cur node*
8. **else**
9. SubsetSearch(*cur node*)
10. add*cur node* to List L'
11. move$L'$to L

Each virtual node created is treated as subset and pruning algorithm SubsetSearch is been proposed.Bottom-up method is given more importance because it can earlier access the leave nodes than top-down approach. If leave nodes are access initially computation is faster and even lowest distance is computed at early stage which solves the drawback of scanning whole dataset.

VII. SEARCH ALGORITHMS

To execute mCK queries various search algorithms are studied here. Algorithms are used to find nearest keyword among all the keywords in dataset with lowest distance. Query keywords to be retrieved can be located at child nodes, hence the mechanism much be implemented so that the child nodes are executed first.

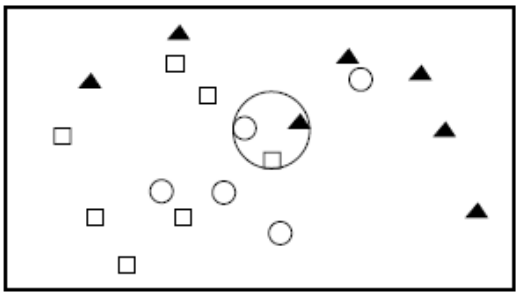

Fig.2. *m*CK query for three keywords obtained from placemarks

Algorithm 5 : Finding *m* Closest Keywords

Input: *m* query keywords, bR*-tree
Output: Distance of *m*closest keywords
1. Find an initial $\delta*$
2. returnSubsetSearch(*root*)

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 6, Issue 10, October 2017

The search starts with root node. There are two algorithms introduced here one for search in one node and other in another node. To denote set of nodes NodeSet is used in the proposed algorithm. The function in the above algorithm is i.e. SubsetSearch is used because it travels in depth-first manner so as to access the leaf nodes earlier. Due to this keywords with the smallest distance is found with less amount of time.

A. Searching In One Node

While dealing with search in one, major task is to enumerate all the subsets of its child node. Subsets are derived which contains all the m query keywords and among child nodes, nearest child nodes are retrieve. One of the constraint is been introduces here is that the number of candidates much not exceed m. The priori algorithm is introduced here to find frequent itemsets using candidate generation via lattice structure .Two more constraints used those are keyword mutex and distance mutex. If the distance between the two nodes is larger than it is called as distance mutex and if the distance between the keywords is larger than it is called as keyword mutex.

Algorithm 6 :SubsetSearch: Searching in a Subset ofNodes

Input: current subset *curSet*
Output: Distance of *m*closest keywords

1. **if***curSet*contains leaf nodes **then**
2. δ= ExhaustiveSearch(*curSet*)
3. **if**$\delta < \delta^*$**then**
4. $\delta^* = \delta$
5. **else**
6. **if***curSet*has only one node **then**
7. *setList= SearchInOneNode*(*curSet*)
8. **for**each $S \in$*setList***do**
9. δ^*= SubsetSearch(*S*)
10. **if***curSet*has multiple nodes **then**
11. *setList= SearchInMultiNodes*(*curSet*)
12. **for**each $S \in$*setList***do**
13. δ^*= SubsetSearch(*S*)

Algorithm 7:SearchInOneNode: Searching in One Node

Input: A node *N* in bR*-tree
Output: A list of new NodeSets

1. L_1= all the child nodes in N
2. **for***i* from 2 to *m* **do**
3. **for**each *NodeSet*$C_1 \in L_{i-1}$**do**
4. **for**each *NodeSet*$C_2 \in L_{i-1}$**do**
5. **if**C_1 and C_2 share the first $i-1$ nodes **then**
6. C = *NodeSet*(C_1, C_2)
7. **if***C* has subset not appear in L_{i-1}**then**
8. continue
9. **if***C* is not distance mutex**then**
10. **if***C* is not keyword mutex**then**
11. $L_i = L_i \cup C$
12. **for**each *NodeSet S* $\in \cup_{i=1}^{m} L_i$**do**

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 6, Issue 10, October 2017

13. **if***S* contains all the query keywords **then**
14. add*S* to *cList*
15. return*cList*

B. Searching In Multiple Nodes

Exploring the search space requires to find all the possible combinations of child nodes with lowest distance. Two constraints are even applied here, one is that the subset of possible combination much not exceed m and constraint of distance and keyword mutex. Only the bottom level of lattice is been built for given m query keywords. To generate possible candidate set, enumeration of all the possible child node subset is required. The algorithm is recursive. Each time an candidate is generated it is checked for whether it contains all the query keyword or not. This check is done to take decision for pruning the unnecessary nodes.

Algorithm 8 :SearchInMultiNodes: Search In Multiple Nodes

Input: A set of $\{N_1, \ldots, N_n\}$ in bR*-tree
Output: A list of new NodeSets
1. **for** each node N_i**do**
2. L_i= SearchInOneNode(N_i)
3. perform an initial filtering on L_i
4. return Enumerate($L_1, \ldots, L_i$, *n*, NULL)

Algorithm 9: Enumerate: Enumerate All Possible Candidates

Input: *n* lists of sets of child nodes *L*1, . . . , *Ln*, count and *curSet*
Output: A list of new NodeSets

1. **if**count = 0 **then**
2. **if***curSet*contains all the query keywords **then**
3. push*curSet*into the candidate list *cList*
4. return
5. **if**count = *n* **then**
6. **for**each *NodeSet S* $\in L_n$**do**
7. *curSet*= *S*
8. Enumerate(*L*1, . . . , L_n, count−1, *curSet*)
9. **else**
10. **for**each *NodeSet S* $\in L_n$**do**
11. *newSet*= *NodeSet*(*curSet*, *S*)
12. **if***newSet*contains more than *m* nodes **then**
13. break
14. **if***newSet*has any illegal subset candidate **then**
15. continue
16. **if***newSet*is not distance mutex**then**
17. **if***newSet*is not keyword mutex**then**
18. Enumerate($L_1, \ldots, L_n$, count−1, *newSet*)
19. return*cList*

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 6, Issue 10, October 2017

VIII. DATASETS

Various datasets were used during experiments. Few of those are listed below:

- **TIGER**

TIGER (Topologically Integrated Geographic Encoding and Referencing system) (downloadable from *http://www.census.gov/geo/www/tiger*) is one to the real dataset used. The database consists of numerous complicated geographic and cartographic information of the entire United States. Landmark data was extracted. Each point in the data set is associated with a census feature class code
to identify its noticeable characteristic. For example, *D85* is the class code for keyword *Park*.

- **High Performance Database Research Center**

Two datasets were provided by High Performance Database Research Center (http://hpdrc.fiu.edu/). Datasets of hotel and restaurants were used. Both datasets are plain text files (tab delimited) where each spatial object occupies a row.

- **Flickr**

Images and their meta data were collected from flickr using flickr API service. Timestamp, user ID, geotag, and text tags fo images were also collected.

- **Web**

The Web dataset contains each point associated with large set of keywords. Efficiency and accuracy is been computed on this dataset.

- **.GOV data**

It is a collection of real web resources of major USA government sites whose top domain is .gov. These data are mainly crawled in the year 2002 and used by TREC2003. The dataset covers a wide geographical range of USA.

- **GIS database**

The dataset is downloaded from www.mapdex.org that catalogs vast amounts of GIS databases.

IX. CONCLUSION

The mCK query is been studied that is possible with the help of bR*- trees. We investigated the other methods for graph based keyword search so that the smallest distance is been captured, which can be done by graph mapping methods. Thus the same methods can also be used to process graph based keyword search. From our survey it is concluded that single bR*-tree is used for most frequent keywords and multiple R*- tree for infrequent keywords.Survey of methodologies to manipulate multi-dimensional spatial data is been done.we proposed a Bloom-filter-based R-treestructure, i.e., BR-tree, for supporting multiple queries ofitems having multidimensional attributes. The proposedBR-tree can efficiently support point, range, cover, andbound queries. To the best of our knowledge, we are the firstto address the bound query. Note that bound query could bewidely applied into real applications that do not have theexact-matching requirement. The BR-tree structure makes itpossible for fast point query and accurate bound query sinceBR-tree keeps the correlated consistency between querieddata and their attribute bounds in an integrated structure.

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 6, Issue 10, October 2017

REFERENCES

1. Vishwakarma Singh, Bo Zong, and Ambuj K. Singh, "Nearest Keyword Set Search in Multi-Dimensional Datasets" ,in IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 28, NO. 3, MARCH 2016.
2. W. Li and C. X. Chen, "Efficient data modeling and querying system for multi dimensional spatial data," in Proc. 16th ACM SIGSPATIAL Int. Conf. Adv. Geographic Inf. Syst., 2008, pp. 58:1-58:4.
3. D. Zhang, B. C. Ooi, and A. K. H. Tung, "Locating mapped resources in web 2.0," in Proc. IEEE 26th Int. Conf. Data Eng., 2010, pp. 521–532.
4. V. Singh, S. Venkatesha, and A. K. Singh, "Geo-clustering of images with missing geotags," in Proc. IEEE Int. Conf. Granular Comput., 2010, pp. 420–425.
5. V. Singh, A. Bhattacharya, and A. K. Singh, "Querying spatial patterns," in Proc. 13th Int. Conf. Extending Database Technol.: Adv. Database Technol., 2010, pp. 418–429.
6. X. Cao, G. Cong, C. S. Jensen, and B. C. Ooi, "Collective spatial keyword querying," in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2011, pp. 373–384.
7. D. Zhang, Y. M. Chee, A. Mondal, A. K. H. Tung, and M. Kitsuregawa, "Keyword search in spatial databases: Towards searching by document," in Proc. IEEE 25th Int. Conf. Data Eng., 2009, pp. 688–699.
8. R. Hariharan, B. Hore, C. Li, and S. Mehrotra, "Processing spatialkeyword (SK) queries in geographic information retrieval (GIR) systems," in Proc. 19th Int. Conf. Sci. Statistical Database Manage., 2007, p. 16.
9. A. Khodaei, C. Shahabi, and C. Li, "Hybrid indexing and seamless ranking of spatial and textual features of web documents," in Proc. 21st Int. Conf. Database Expert Syst. Appl., 2010, pp. 450–466.
10. A. Guttman, "R-trees: A dynamic index structure for spatial searching," in Proc. ACM SIGMOD Int. Conf. Manage. Data, 1984, pp. 47–57.
11. I. De Felipe, V. Hristidis, and N. Rishe, "Keyword search on spatial databases," in Proc. IEEE 24th Int. Conf. Data Eng., 2008, pp. 656–665.
12. 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.
13. B. Martins, M. J. Silva, and L. Andrade, "Indexing and ranking in Geo-IR systems," in Proc. Workshop Geographic Inf., 2005, pp. 31–34.
14. Z. Li, H. Xu, Y. Lu, and A. Qian, "Aggregate nearest keyword search in spatial databases," in Proc. 12th Int. Asia-Pacific Web Conf., 2010, pp. 15–21.
15. M. L. Yiu, X. Dai, N. Mamoulis, and M. Vaitis, "Top-k spatial preference queries," in Proc. IEEE 23rd Int. Conf. Data Eng., 2007, pp. 1076–1085.
16. N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger, "The R*- tree: An efficient and robust access method for points and rectangles," in Proc. ACM SIGMOD Int. Conf. Manage. Data, 1990, pp. 322–331.
17. R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases," in Proc. 20th Int. Conf. Very Large Databases, 1994, pp. 487–499.
18. J. M. Kleinberg, "Two algorithms for nearest-neighbor search in high dimensions," in Proc. 29th Annu. ACM Symp. Theory Comput., 1997, pp. 599–608.
19. A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing," in Proc. 25th Int. Conf. Very Large Databases, 1999, pp. 518–529.
20. D. Papadias, N. Mamoulis, and Y. Theodoridis, "Processing and optimization of multiway spatial joins using r-trees," in Proc. 18th ACM SIGMOD-SIGACT-SIGART Symp. Principles Database Syst., 1999, pp. 44–55.
21. Y. Tao, K. Yi, C. Sheng, and P. Kalnis, "Quality and efficiency in high dimensional nearest neighbor search," in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2009, pp. 563-576.