IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 2 Issue 6, June 2015.
www.ijiset.com
ISSN 2348 – 7968

Data mining strategy for fast record searching

Ms. Sonali B. Gosavi[1], Dr. S. V. Gumaste[2]

[1]M.E. Second Year Student, Department of Computer Engineering,
SPCOE, Otur, Pune, India
sonalibgosavi@gmail.com
[2]Professor, Department of Computer Engineering,
SPCOE, Otur, Pune, India
svgumaste@gmail.com

Abstract: Web users and content are increasingly being geo-positioned, and increased focus is being given to serving local content in response to web queries. This development work calls for spatial keyword queries that take into account both the locations and textual descriptions of content .Authors of the paper propose a novel algorithm and index structure for the query processing of top-*nearest neighbor* spatial keyword queries. Empirical studies show that the proposed solution is efficient on real datasets. Authors of the paper also offer analytical studies on synthetic datasets to demonstrate the efficiency of the proposed solution.. *To satisfy the requirement of such queries the solution that is been used is the IR2-tree. By considering the disadvantages of IR2-tree an alternative to this is to use WIBR- Tree which is capable to handle multidimensional data and process nearest neighbor retrieval queries that can search the user's records efficiently.*

Keywords: Closest item search**,** nearest neighbor search, IR2-tree., keyword search, WIBR-tree, R-tree, spatial inverted index.

1. Introduction

Web users searching for information about locations, institutions and many other topics often require information that is geographically specific. It has been suggested [1] that users will focus their Web query by using geographic terminology such as place names and spatial prepositions (e.g. "near", "between" and "north of") to associate a topic with a location. When the name of a place is typed into a typical search engine, Web pages that include that name in the text will be retrieved but most likely, not places that are within or close to that specified place. In order to understand the potential of improving the functionality of search engines in relation to geographic search, it is necessary to understand what people search for and how they structure their queries. There are an increasing number of studies available of how people formulate Web queries and how they modify those queries during a search. However, to the best of knowledge, no study exists on the use of geographic terminology within Web search queries. It is acknowledged by [3] that the potential benefit of Web query log studies to IR system developers, users, and Web site classifiers and designers could be considerable. It is therefore of interest to assess the manner in which users formulate their queries to find information on geographic and related topics, in order to gauge whether the interpretation of queries by search engines could be improved.

2. Literature Review

First techniques used is Signature files. Signature files were introduced by Faloutsos and Christodoulakis [2][3][4]] as a way to with efficiency search a group of text documents. Lee et al. [5] gift ways to make structures on high of a signature file. during this work one tend to read the document describing a abstraction object as a text block in their notation and build similar structures on high of this set of objects. specifically, one tend to adopt the thought of AN indexed descriptor file structure may be a variant of AN , that may be a tree wherever all-time low level consists of block signatures. These area unit superimposed codes obtained from the text blocks. a gaggle of b signatures at the i-th level is superimposed along to create a signature at the (i-1)-th level. The signatures of every level have a similar length. Similarly, in IR2-Tree, the parent encompasses a signature that superimposes (binary ORs) the signatures of the kids. Finally, once building AN indexed descriptor file, one tend to expect the highest levels to possess a lot of 1's because of the larger range of words in their subtrees, that successively results in a lot of false positives. The principle of the multi level superimposed committal to writing was projected as an answer to the current downside, wherever higher levels have longer signatures. This principle permits fewer false positives by acquisition an area overhead. However, this makes updates on the underlying documents costly to keep up.

Second technique used is IR2 tree. IR2 tree [1] is that the combination of the R-tree and signature files. Signature enter general refers to a hashing-based framework, whose internal representation in [6] is understood as superimposed committal to writing (SC), that is shown to be simpler than different instantiations. The IR2-tree is associate degree R-tree wherever every (leaf or nonleaf) entry E is increased with a signature that summarizes the union of the texts of the objects within the sub tree of E. On standard R-trees, the best-first algorithmic rule could be a well-known answer to NN search. it's simple to adapt it to IR2-trees. Specifically, given alphabetic characteruery|a question |a question} purpose q and a keyword set Wq, the custom-made algorithmic rule accesses the entries of associate degree IR2-tree in ascending order of the distances of their MBRs to letter of the alphabet (the MBR of a leaf entry is simply the purpose itself), pruning those entries whose signatures indicate the absence of a minimum of one word of Wq in their subtrees. Whenever a leaf entry, say of purpose p, can't be cropped, a random I/O is performed to retrieve its text description Wp. If Wq could be a set of Wp, the algorithmic rule terminates with p because the answer; otherwise, it continues till no additional entry remains to be processed.

A disadvantage of the IR2-Tree delineate on top of is that constant signature length is employed for all levels that ends up in a lot of false positives within the higher levels, that have a lot of 1's (since they're the superimpositions of the

ISSN 2348 – 7968

lower levels). to deal with this drawback, one have a tendency to use variable signature lengths for various levels. The IR2-tree is that the 1st access methodology for respondent NN queries with keywords. like several pioneering solutions, the IR2-tree conjointly incorporates a few drawbacks that have an effect on its potency. The foremost serious one amongst all is that the amount of false hits may be extremely massive once the article of the ultimate result's off from the question purpose, or the results merely empty. In these cases, the question algorithmic rule would wish to load the documents of the many objects, acquisition overpriced overhead as every loading necessitates a random access.

3. System Design Overview

System uses the technique called word partitioning. As a first step in presenting the index structure, authors consider the partitioning of a dataset according to keywords. One hypothesizes that a keyword query will often contain a frequent word (say *w*). This inspires us to partition dataset *D* into the subset *D*+ whose objects contain *w* and the subset *D*− whose objects do not contain *w* and that one need not examine when processing a query containing *w*. Authors aim at partitioning *D* into multiple groups of objects, such that the groups share as few keywords as possible. However, this problem is equivalent to, e.g., the clustering problem and is NP-hard. Hence, authors propose a heuristic to partition the objects. Let the list *W* of keywords of objects sorted in descending order of their frequencies be: *w*1, *w*2, . . . , *wm*, where *m* is the number of words in *D*. Frequent words are handled before infrequent words. One start by partitioning the objects into two groups using word *w*1: the group whose objects contain *w*1, and the group whose objects do not. Authors then partition each of these two groups by word *w*2. This way, the dataset can be partitioned into at most 2, 4, . . . , 2*m* groups. By construction, the word overlap among groups is small, which will tend to reduce the number of groups accessed when processing a query.

Algorithm for Word Partition (Dataset *D*, Sorted list of words *W*, Integer *B*, List of tree nodes *L*):

```
1. if B|2<=|D|<=B then
2.       add D as a node to L.
3. else if |D|<B/2 then
4.       return D;
5. else
6.     if W is empty then
7.         insert D into a main memory R-tree with fanout B;
8.         add leaf nodes of main memory R-tree to L;
9.     else
10.        w ← first word in W; W ←W\{w};
11.        D+←{p €D | w €p.ψ};
12.        D-←{p €D | w €p.ψ};
13.        T +← WordPartition(D+,W,B,L);
14.        T- ← WordPartition(D-,W,B,L);
15.        T ‘ ← T + U T-
16.        If  B/2<=| T ‘ |<= B then
17.            add T ‘ as a node to L;
18.        else if  | T ‘ |<=B/2 then
19.              Return T ‘ ;
```

Example: Consider the 8 objects in Figure 2b and let the node capacity *B* be 3. Words are sorted by their frequencies in the dataset, and ties are broken according to alphabetic order. Thus, authors obtain the list: *W* = _(*a*, 5), (*d*, 4), (*f*, 3), (*b*, 2), (*e*, 2), (*c*, 1)_. Using word *a*, the objects are partitioned into the groups *D*+ = {*p*1, *p*2, *p*3, *p*5, *p*9} and *D*− = {*p*4, *p*6, *p*7, *p*8}. Both contain more than 3 objects and are partitioned according to word *d*, resulting in *D*++ = {*p*1, *p*2, *p*5}, *D*+− = {*p*3, *p*9}, *D*−+ = {*p*6, *p*8}, and *D*−− = {*p*4, *p*7}. All groups but *D*+− are added to result list *L*, as they contain between 1.5 and 3 objects. Group *D*+− is passed to the first call of the algorithm and is finally added to *L*.

Tree Construction Using Word Partitioning.:

The W-IR-tree uses the same data structures as the IR-tree, but is constructed differently by using word partitioning (thus
the prefix ‘W’). Instead of performing insertions iteratively, one build the W-IR-tree bottom-up. Authors first use the word partitioning (Algorithm 3) to obtain the groups that will form the leaf nodes of the W-IR-tree. For each leaf node *N*, authors compute *N*.ψ as the union of the words of the objects in node *N*, and *N*.λ as the MBR of the objects in *N*. Next, one regard the leaf nodes as objects and apply Algorithm 3 to partition the leaf nodes into groups that form the nodes at the next level in the W-IR-tree. One repeat this process until a single W-IR-tree root node is obtained. Figure 4a illustrates the W IR-tree for the 8 objects in Figure 4b. Following Example 3, leaf nodes *R*1 ← {*p*3, *p*9}, *R*2 ← {*p*1, *p*2, *p*5}, *R*3 ← {*p*6,*p*8}, and *R*4 ← {*p*4, *p*7} are first formed. Figure 4b shows the MBRs of those leaf nodes and *R*1.ψ = {*a*, *d*}, *R*2.ψ = {*a*, *b*, *c*}, *R*3.ψ = {*d*, *e*, *f*}, *R*4.ψ = {*e*, *f*}. Next, Algorithm is used to partition *R*1, *R*2, *R*3, and *R*4, since they are 4 (the node capacity is 3) nodes and cannot be put into one node at the next level. Using word *a*, two partitions are obtained, i.e., *R*5 ← {*R*1,*R*2} and *R*6 ← {*R*3,*R*4}. Since *R*5 and *R*6 contain between 1.5 and 3 nodes, there is no need to further partition them. Finally, *R*5 and *R*6 can be put into one node at the next level, resulting the root node of the W-IR-tree.

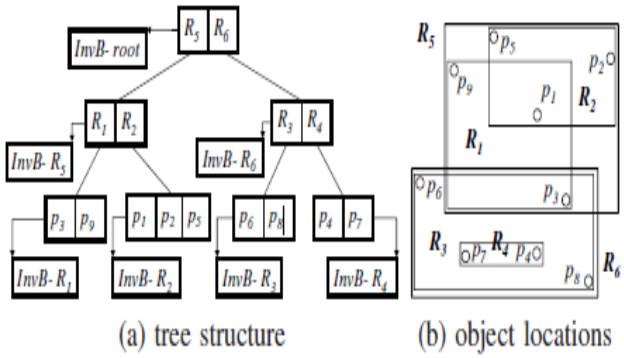

Fig.1: WIR-tree

To enhance it more next echnique used is W-IBR tree. Which contain Inverted Bitmap Optimization. Each node in the W-IR-tree contains a pointer to its corresponding inverted file. By replacing each such inverted file by an inverted bitmap, One can reduce the storage space of the W-IR-tree and also save I/O during query processing. Authors call the resulting tree the W-IBR-tree. Table 1 illustrates the inverted bitmaps that correspond to the nodes of the W-IBR-tree in Figure 1. A bitmap position corresponds to the

relative position of an entry in its W-IBRtree node. The length of a bitmap is equal to the fanout of a node. For example, the node *R*3 stores the points *p*6 and *p*8. The inverted list for item *f* of *R*3 is *p*8 (which is the second entry in *R*3). Thus, the inverted bitmap for item *f* of *R*3 is '01'.

Table 1: Content of Inverted Bitmaps of the W-IBR-Tree

InvB-root	R5	R6	R1	R2	R3	R4
a:10	a:11	c:01	a:11	a:111	d:11	c:11
b:10	b:01	d:10	d:11	b:101	e:10	f:11
c:10	c:01	e:10		c:010	f:01	
d:11	d:10	f:11				
e:01						
f:01						

4. Conclusion

This paper introduces the spatial keyword query and presents efficient means of computing the query. This solution consists of: (i) the W-IBR-tree that exploits keyword partitioning and inverted bitmaps for indexing spatial keyword data. In addition, it describe how to adapt the solution to existing index structures for spatial keyword data. Empirical studies show that W-IBR-tree is the most efficient combination for processing.

Acknowledgement

The authors would like to thank the researchers as well as publishers for making their resources available and teachers for their guidance. We also thank the college authority for providing required infrastructure and support.

References

[2] Yufei Tao and Cheng Sheng, "Fast Nearest Neighbor Search with Keywords," National Research Foundation of Korea, GRF 4166/10,4165/11, and 4164/12 from HKRGC .

[3] Christos Faloutsos: Signature files: Design and Performance Comparison of Some Signature Extraction Methods. In S IGMOD Conference 1985.

[4] Christos Faloutsos, Stavros Christodoulakis: Signature Files: An Method for Documents and Its Analytical Performance Evaluation. In ACM Trans. Inf. Syst. 2(4): 267-288(1984).

[5] Christos Faloutsos, Stavros Christodoulakis: Design of a Signature File Method that Accounts for Non-Uniform Occurrence and Query Frequencies. In VLDB 1985: 165-170.

[6] DikLun Lee, Young Man Kim, Gaurav Patel: Efficient Signature File Methods for Text Retrieval. Pages 423-435.TKDE Vol 7, Number 3, June 1995

[7] 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

[8] 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 a 16, 2002.

[9] 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 a 331, 1990.

[10] 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 a 440, 2002.

[11] 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 a 29, 2012.

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

[13] 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 a 384, 2011.

[14] 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 a 39, 2004.

[15] 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 a 288, 2006.

[16] D. Wu, M. L. Yiu, G. Cong, and C. S. Jensen, "Joint top-k spatial keyword query processing". IEEE TKDE, 24(10):18891903, 2012.

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