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

International Journal of Innovative Research in Science, Engineering and Technology

Volume 3, Special Issue 3, March 2014

2014 International Conference on Innovations in Engineering and Technology (ICIET'14)

On 21st&22nd March, Organized by

K.L.N. College of Engineering, Madurai, Tamil Nadu, India

A Novel Approach for Semibulk Loading In IR-Tree

Amruta Joshi , Prof. U. M, Patil

ME Dept Of Computer Science and Engineering, North Maharashtra University, RCPIT, Shirpur, India.

Dept Of Computer Science and Engineering, North Maharashtra University, RCPIT, Shirpur, India.

Abstract— A geospatial query is made up of two things, viz. keywords and a location, geographic search engine retrieves the documents which are textually as well as spatially relevant to keyword in query and location of it. The index should concurrently handle both the textual aspects and spatial aspects of the documents and query. Existing geographic search engines are inefficient in answering geographic queries in this way. In this paper, we propose an efficient index, called IR-tree, that together with a top-k document search algorithm facilitates four major tasks in document searches, spatial filtering, textual filtering, relevance computation, and document ranking in a fully integrated manner.IR Tree also allows to have different values for spatial and textual relevance.

Keywords— Geospatial Index, Textual Filtering, Spatial Filtering, Semibulk Loading.

I. INTRODUCTION

According to wikipedia, there are 25 billion indexable webpages and over 100 million websites recorded in 2009, and these numbers continue to grow. Due to the large number of webpages, search engines that search the documents based on their relevance are essential for information seeking for users. Search engines have to determine relevant webpages within a short period of time. So for search engines, high search efficiency is one of the important factor in design and implementation of search engines. Thus, efficient indexing techniques are required that organize webpages according to the contents in it. Although webpages are accessible all over the earth, users are only interested in information (such as business listings or news) related to certain locations,or sometimes containing certain keywords e.g., Hotels in Delhi" , "Mumbai's weather report" .These queries are referred as geographic queries which consist of both textual and spatial conditions on documents. some search engines specialized for answering these geographic queries are called as geographic search engines. In the past few years, geographic search engine has been receiving a lot of attention due to increasing application demands and rapid technological changes in geographical information systems, [12], [15], [21], [22]. A geographic search engine should quickly return documents of high relevance in both textual and spatial aspects to a given geographic query. Index structures serves as the core of search engines and are very important. While designing an efficient index structure for both textual and spatial information four major to be fulfilled.
In the paper we will have an overview of basic concepts, related work done on the topic and finally we will study Ir tree structure and Semibulk loading in it.

II. OVERVIEW OF BASIC CONCEPTS

Geographic document search and document ranking based on textual relevance and spatial relevance should be known first. Then only we can understand measurements of textual relevancies and spatial relevance, and review existing works proposed for geographic search engines

A. Searching And Ranking Of Geographic Document

It is assumed here that, assume each document d in a given document set D is composed of a set of words Wd, and is associated with a location Ld. With a query q that specifies a set of query keywords Wq and a query spatial scope Sq, the textual relevance and spatial relevance of a document d to q are as follows:

- Textual filtering: A document d is textually relevant to a query if d contains some or all of the keyword that are queried.
- Spatial filtering: A document d is spatially relevant to a query q if the location of d and the query spatial scope of q have at least some overlapping.

B. Document relevance measurement

The most importance factor for the quality and performance of search engines is the accurate estimation of the relevance between documents and user queries.

1) Textual filtering :

TF-IDF offers a term in a document based on term frequency (tf) and inverse document frequency (idf) [16]. A term frequency $tf_{w;d}$ provides us the number of times a word w appears in a document d, which provides the importance of the word within the document. The inverse document frequency, $idf_{w;D}$ quantifies the importance of a word w in a document set D. However, under the context of geographic document search, the idf of a word w, denoted by $idf_{w;D;S}$, is termed corresponding to a candidate document set D_S instead.

2) Location Intent in Documents :

The spatial relevance of a document d, is dependent on the types of the spatial relationships defined between a document location Ld and a spatial scope S. The relationships can either be Enclosed, Overlapping or Proximity.

C. Milestones In Previous Work

Here, we will review some existing works in textual filtering, spatial filtering, and geographic document search engines.

1) Textual Filtering :

To facilitate the calculation of TF/IDF of documents, we propose inverted files, which are collections of inverted list. In each inverted list l_w serves one word w. An element $tf_{w;di}$ in l_w records a document d with $tf_{w;d} > 0$; and the lists are arranged in descending order of documents' tf values.

2) Filtering by Location Intent :

Spatial indexes [9] have been extensively studied in the spatial database community [21]. Among all the existing spatial indexes, R-tree [11] is very well-received. In an Rtree, spatial objects are first collected as minimum bounding boxes (MBBs). Those spatial objects whose MBBs are closely located are gathered together in leaf nodes. This grouping is done iteratively until the root node is formed.

TABLE I

FONT SIZES FOR PAPERS

Author	**Indexing data structure**	**Publication**
A. Guttman	R- Trees	"R-Trees: A Dynamic Index Structure for Spatial Searching
K.S. McCurley,	Grid index	Geospatial Mapping and Navigation of the Web
Y.-Y. Chen, T. Suel, and A. Markowetz	Quad-tree	Efficient Query Processing in Geographic Web Search Engines
RamaswamyHariharan, BijitHore	KR* trees	Processing Spatial-Keyword (SK) Queries in Geographic Information Retrieval (GIR) Systems
I.D. Felipe, V. Hristidis, and N. Rishe	IR2-tree [8]	Keyword Search on Spatial Databases
Zhisheng Li, Ken C.K. Lee,	IR-Trees	IR-Tree: An Efficient Index for Geographic Document Search

III STRUCTURE OF IR-TREE

In this section, we present IR-tree, an index that provides the following required functions for geographic document search and ranking: 1) spatial filtering: all the spatially irrelevant documents have to be filtered out to narrow the search space; 2) textual filtering: all the textually irrelevant documents have to be discarded earlier to lessen the search cost; and 3) relevance computation and ranking: so only the top-k documents are returned and k may be smaller than the total number of relevant documents .It is better to have an incremental search process that integrates the computation of the joint relevance and document ranking concurrently so that the search process can stop when the top-k documents are identified. In addition, IR-tree is designed by considering the storage and access overheads since a document set is very large in terms of numbers of documents and their words.

A. Basic IR-tree Structure

The basic IR-tree can be manipulated with three operations, bulk loading of all documents to form tree, inserting documents in the tree, and deleting documents from the tree. It clusters documents based on their geographic locations into leaf-level entries, and then clusters the nodes in a bottom-up fashion to form the root . The structure of IR-tree can be easily adapted. Each document is associated to one location Ld, and documents with the same location are collected in a set of entries. And an inverted file is created for each entry in the set to keep the term frequencies of different words .The entries in sets are clustered according to their locations to form IR-tree . Nodes are also associated with a document summary. When a new document d is inserted, based on Ld, an IR-tree is to be traversed to reach a leaf node that provides the smallest expanded area after Ld is included. Then, the document summaries of all nodes on the path are updated according to changes.

Semibulk loading in IR Tree:

We have used two different data structures to store the geocodes and their importance in the document. It may happen that a document can contain two different location names in it. But it has to belong to a single location only. For this purpose we have calculated the importance of the location name by using its frequency of occurring in the document.

Also we have stored the documents along with the maximum appearing location name associated with it. This will prevent us from each time geocoding the documents before construction the tree.This will reduce the construction time required for IR Tree.

Pseudocode for Construction :

```
1: Ne <- NULL;
2: for each d in D do
3: Check if the geocoding for d already exists
 Else, geocode d and represent Ld with n1,n2,w1,w2// n1,n2,w1,w2 are the limits for longitude and latitude for location Ld;
4: if for any e, n1<n<n2 and w1<w<w2 then
5: add d to e's document set De
6: else
7: create a new entry e;
8: set e's long. And lat.
9: Ne <- Ne U e
10: end if
11: end for
12: for each e in Ne do
13: build inverted file with each list lw w.r.t. every word w in d belonging to De;
14: end for
15: Cluster nodes in Ne according to its long and lat.
16. store document summeries;
17: Store geocoded documents along with location names with highest ocurrance
18: end while
19: create the root node to covering all beneath nodes and include document summaries;
20: output the root node;
```

Top-k document search

The size of the document set is much larger than the required number of result documents, k,so top-k document search algorithm is needed based on Irtree. First determines IDF for query keywords; and the second computes the relevance of candidate documents and returns k most relevant documents.

1) IDF Calculation :

The idf, is a fraction between |DS| and $df_{w;DS}$. Firstly find out all of the documents that are located inside the query scope Sq for a query q. This search traverses the nodes i with Ai = Sq. If node i is already fully covered by Sq, it will not visit i's child/descendant nodes to find |DS| and $df_{w;DS}$. Then identify some candidate nodes that contain result documents being spatially and textually relevant to a query. Documents beneath those nodes are spatially relevant to the query but may have very less textual relevance. $df_{w;Di}$ values are sufficient to find whether a node i contains any textually relevant document or not.

2) Top-k Document selection :

As result from above is stored in buffer B Top-k Document Retrieval algorithm runs to identify the result documents. Here the candidate set might contain far more documents than k. So this algorithm tries to avoid examining nonresult documents. Strategy is to evaluate the documents based on their joint spatial and textual relevances and to terminate the process once the top-k result documents are obtained.

IV PERFORMACE EVALUATION

A. Experiment Setup

While preparing the sample documents for experiments, extraction of all the location names from all individual documents is done and then geocoding the location names into spatial regions. We used the google maps to assign the MBR and longitude and latitude. It focuses on the locations in United States of America. It is used to locate one location for each document downloaded from LATimes.com. We have downloaded news paper clips from LATimes.com and total size of collected database is 1.27 GB.

Storage cost :

Total cost for storing the IR Tree is calculated by following formula.:

$C_{tot} = C_{inv} + C_t + C_s$ [22]

Where C_{inv} is the storage needed to store the inverted files, C_t storage for newly built tree and C_s is sorage for documents.

Construction Time :

We constructed tree repeatedly by adding some more documents each time. The basic version of IR tree creates itself each time from scratch so requires highest time.

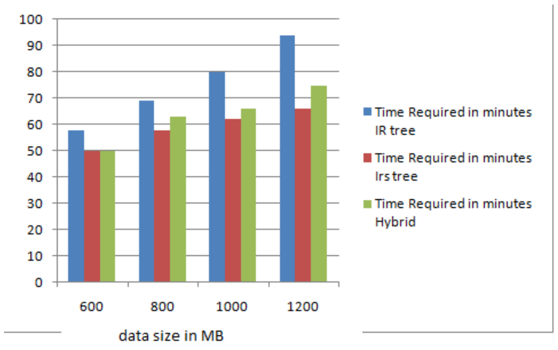

Fig. 1 A sample graph of construction time in milliseconds versus data size in MB

The effects of various parameters on the performance is described briefly in following sections. The major impacting factors are number of locations |L|, the number of requested documents k and number of documents in document set D.

A. Effect of Number of Locations |L|

The same trends for HybridR are also observed. IR-tree performs the best as its cost increases slightly with |L|. Here, HybridR produces the smallest indexes but it incurs the largest I/O cost. We use |Lq | to represent the size of query spatial scope Lq .

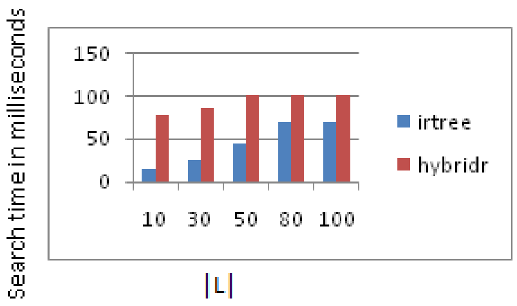

Fig. 2 A sample graph of search time in milliseconds versus number of locations |L|

Performance in Search time versus |L|

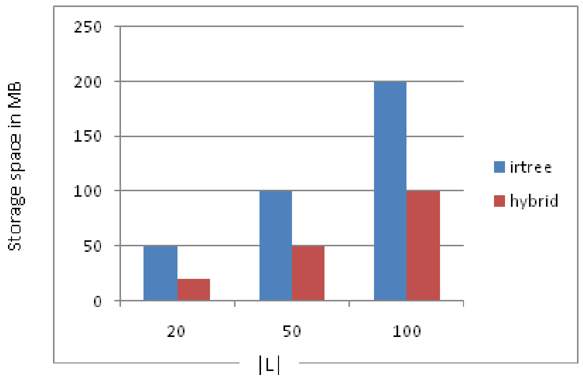

Fig. 3 A sample graph of storage space in MB versus number of locations |L|

B. Effect of Number of Documents |D|.

We vary the |D| from 500 to 2000 and study the impacts of different factors on the search performance which are the size of the query spatial scope |S|; the number of requested documents k.. As HybridI maintains R-trees under inverted files, all of the documents that contain any query keyword have to be accessed, while the majority of them are located outside the query spatial scope.

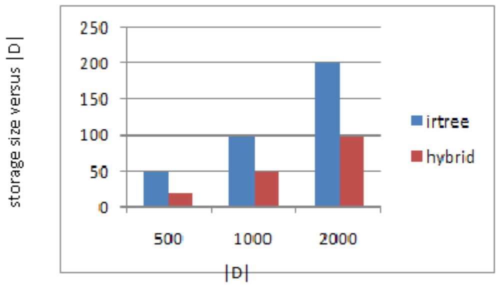

Fig. 4 A sample graph of storage space in MB versus number of documents |D|

C. Effect of search results |k|

First, we change the requested number of document, k, from 10 to 50 but |S| is fixed at 100 km by 100 km. The first finding is that HybridR performs the worst while IRtree performs the best in all the cases. The performance trend for IR-tree is consistent with that under various k values. It works superiorly over HybridR becomes more significant when |S| increases.

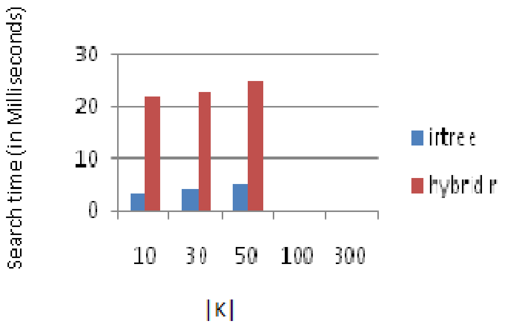

Fig. 5 A sample graph of search time in milliseconds versus number of searched documents |K|

V CONCLUSIONS

In this paper the focus is on the efficiency issue of geographic document search and an efficient indexing structure, IR-tree, along with a top-k document search algorithm is studied. IR-tree is performing well in comparison with its similar structures like hybrid trees. We also have further enhanced the IR-tree index based on semibulk loading. We observed that construction time is significantly reduced by Semibulk loading.

REFERENCES

[1] E. Amitay, N. Har'El, R. Sivan, and A. Soffer, "Web-a-Where: Geotagging Web Content," Proc. ACM SIGIR '04, pp. 273-280, 2004.
[2] V.N. Anh, O.d. Kretser, and A. Moffat, "Vector-Space Ranking with Effective Early Termination," Proc. ACM SIGIR '01, pp. 35-42, 2001.
[3] V.N. Anh and A. Moffat, "Pruned Query Evaluation Using Pre-computed Impacts," Proc. ACM SIGIR '06, pp. 372-379, 2006.
[4] R. Baeza-Yates and B. Ribeiro-Neto, Modern Information Retrieval.Addison-Wesley, 1999.
[5] Y.-Y. Chen, T. Suel, and A. Markowetz, "Efficient QueryProcessing in Geographic Web Search Engines," Proc. ACM SIGMOD '06, pp. 277-288, 2006.
[6] Dow Jones Factiva, http://www.factiva.com, 2010.
[7] S. Shekhar, S. Chawla, S. Ravada, A. Fetterer, X. Liu, and C.-T. Lu, "Spatial Databases—Accomplishments and Research Needs,"IEEE Trans. Knowledge and Data Eng. (TKDE), vol. 11, no. 1, pp. 45-55, Jan./Feb. 1999.
[8] I.D. Felipe, V. Hristidis, and N. Rishe, "Keyword Search on Spatial Databases," Proc. IEEE 24th Int'l Conf. Data Eng. (ICDE '08), pp. 656-665, 2008.
[9] V. Gaede and O. Gu¨ nther, "Multidimensional Access Methods," ACM Computing Survey, vol. 30, no. 2, pp. 170-231, 1998.
[10] U. Ga¨untzer, W.-T. Balke, and W. Kiessling, "Optimizing MultiFeature Queries for Image Databases," Proc. Int'l Conf. Very Large Data Bases (VLDB '00), pp. 419-428, 2000.
[11] A. Guttman, "R-Trees: A Dynamic Index Structure for Spatial Searching," Proc. ACM SIGMOD '84, pp. 47-57, 1984.
[12] R. Hariharan, B. Hore, C. Li, and S. Mehrotra, "Processing Spatial-Keyword (SK) Queries in Geographic Information Retrieval (GIR) Systems," Proc. 19th Int'l Conf. Scientific and Statistical Database Management (SSDBM '07), pp. 16-25, 2007.
[13] Z. Li, C. Wang, X. Xie, X. Wang, and W.-Y. Ma, "Indexing Implicit Locations for Geographical Information Retrieval," Proc. Third Workshop Geographic Information Retrieval (GIR '06), 2006.
[14] "Los Angeles Times," http://www.latimes.com, 2010.
[15] Y. Zhou, X. Xie, C. Wang, Y. Gong, and W.-Y. Ma, "Hybrid Index Structures for Location-Based Web Search," Proc. 14th ACM Int'l Conf. Information.
[16] Zhisheng Li, Ken C.K. Lee, Baihua Zheng, Wang-Chien Lee, Dik Lun Lee, and Xufa Wang , "IR-Tree: An Efficient Index for Geographic Document Search",IEEE transactions on knowledge and data engineering, vol. 23, no. 4, april 2011