International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
Volume: 3 Issue: 1 62 - 66

Overview of GINIX and Top-k Method

Nikhil B. Kedare[1]
Student of BE Computer
BVCOE & RI, Nasik, Maharashtra, India
University of Pune
nikhilkedare@gmail.com

Harshada P. Wagh[2]
Student of BE Computer
BVCOE & RI, Nasik, Maharashtra, India
University of Pune
harshadawagh15@gmail.com

Sayali D. Pingle[3]
Student of BE Computer
BVCOE & RI, Nasik, Maharashtra, India
University of Pune
nishapingle@gmail.com

Nayana P. Khode[4]
Student of BE Computer
BVCOE & RI, Nasik, Maharashtra, India
University of Pune
nainakhode1@gmail.com

Prof. Kavita S. Kumavat[5]
ME Computer Engineering
BVCOE & RI, Nasik, Maharashtra, India
University of Pune
kavitakumavat26@gmail.com

***Abstract*—** In today's life more applications are web based and peoples may communicate with each other by using Internet. It involves more and more data retrieval from database system as per user demand. Inverted Index is a system use for searching in which searching is takes place as per index sequentially. So it require more time for searching. While Ginix can search as per word in which all files or related document that word is search appropriately. But it only search documents file which are save in database system but not search multimedia files. Hence the more competent technique for searching is top-k method in which all database is scan for finding appropriate result for given data. Also data is search on web pages. It provides more perfect result within less time as compare to Ginix.

Keywords- *keyword search, index compression, document reordering, top-k list, Ginix.*

*****

I. INTRODUCTION

A central amount of the world's enterprise data resides in relational databases. It is important that users can perfectly search and browse information stored in these databases also. Due to large information, keyword search becomes hard for user to access text datasets.The primary focus of designers of computing systems and data mining has been on the step up of the system performance. According to this point, the performance has been steadily growing driven by more efficient system design and civilizing complexities of the system. Mostly users uses keyword search to get the likely document. A current keyword system generally uses an inverted index. An inverted index is a method that maps each word in the datasets to a catalog of IDs, by using specific ID the word can be found which will be capable to retrieve the documents.

Basically, the datasets are very big and the keyword search system uses various compression techniques to cut the space complexity and time complexity of storing inverted index. Although a compressed inverted index is minor than the original index, the system needs to decompress resolute lists during query dealing out, which leads to extra computational tempo. For this trouble, this paper presents Generalized Inverted Index (Ginix), in which inverted index is extended to support keyword search [3].

The World Wide Web is currently the largest source of information. Though, most of the information on the web is unstructured text in typical languages, and extracting information from natural language text is very difficult. Still, some information is present on the web in structured or semi-structured types, for example, as lists or web tables coded with specific tags such as <ul>, <li>, and <table> on html pages. As a result, a lot of recent work has focused on acquiring knowledge from structured information on the web, in particular, from web tables [4]. However, it is doubtful how much valuable information we can extract from lists and web tables. It is fact that the numbers of web tables are huge in the entire corpus but only a very small percentage of them contains useful information. An even smaller percentage of them contain information interpretable without context. Specifically, based on our experience, more than 90% of the tables are used for content layout on the web. Furthermore, a majority of the remaining tables are not "relational."

It is clear that understanding the context is extremely important in information extraction. Unfortunately, in most cases, context is expressed in unstructured text that machines cannot interpret. In this paper, instead of focusing on structured data (such as tables) and ignoring context, we focus on that context which can be understood, and then after we use the context to interpret less structured or almost free-text information, and guide their extraction.

Specifically, we focus on a rich and valuable source of information on the web, which we call top-k web pages. A top-k web page describes k items of a particular interest. In most cases, the description is in natural language text which is not directly machine interpretable, although the description has the same format or style for different items. But most importantly, the title of a top-k page often clearly discloses the context, which makes the page interpretable and extractable.

II. TERMS AND DEFINITIONS

Keyword search

Keyword search is used by search engine to reveal and examine definite search conditions. People go through the

search engines while conducting a search. Search engine optimization professional's research keywords in order to get better rankings in search engines. Once the position of keyword is found, it is extended to find related keywords. The process is normally done by using keyword suggestion tools, which offer word list and alternate keyword suggestion functionality. Normally various search engines provide their own keyword suggestion tools which also include the number of searches made for every keywords. This information is then used in order to select the correct keyword depending on the SEO goals of the website.

Inverted index

An inverted index (also referred to as postings file or inverted file) is an index data structure storing a mapping from content, like words or numbers, to its locations in a database file, or in a document or a set of documents. The cause of an inverted index is to permit fast full text searches, at a cost of increased handing out when a document is added to the database. The inverted file might be the database file itself, somewhat than its index. It is the most well-liked data structure used in document recovery systems, used on a large scale for example in search engines. Several major general-purpose mainframe-based database management systems have used inverted list architectures, plus ADABAS, DATACOM/DB, and Model 204.

Generalized Inverted Index

An inverted listing of file is an index data configuration storing a mapping from satisfied, such as digits, to its place in a database file, or in a file or a group of documents. The main principle of an inverted index is to authorize fast whole text searches, at a outlay of improved processing when a manuscript is supplementary to the innovative database. The inverted data may be the database file itself, quite than its index. It is the majority popular data structure worn in document storing and accessing the systems, used on a large scale for illustration in Google.

Top-k list

We first introduced the concept of "top-k" list in a demo paper [5]. In that demo, we proposed the top-k list extraction problem and designed a prototype system. We presented this prototype as a web GUI on the project website [6]. One of the potential use of the extracted top-k lists is to act as background knowledge for a Q/A system [7] to answer top-k related queries. To prepare for such knowledge, we need techniques to aggregate a number of similar or related lists into a more comprehensive one, which is in the space of top-k query processing. One of the most well known algorithms there is TA (threshold algorithm) [8]. TA utilizes aggregation functions to combine the scores of objects in different lists and computes the top-k objects based on the combined score. Later, Chakrabarti et al. [9] introduced the OF (object finder) query, which ranks top-k objects in a search query exploring the relationship between TOs (Target Objects, e.g., authors, products) and SOs (Search Objects, e.g., papers, reviewers). Bansal et al. [10] utilize a similar framework but elevate terms at a higher level by taking advantage of a taxonomy, in order to compute accurate rankings. Angel et al. [11] considered the EPF (entity package finder) problem which is concerned with associations, relations between different type of TOs. Some of these techniques can serve as the basis for comprehensive integration of top-k lists.

III. LITERATURE SURVERY

Beyond asking for unambiguous user input, earlier work focused on handling recent queries, which are queries that are after recent events or breaking news. The time sensitive loom processes a regency query by computing usual topic similarity scores for each document, and then "boosts" the scores of the most recent documents, to benefit recent articles over older ones. In dissimilarity to traditional models, which suppose to be a uniform prior probability of significance for each document d in a collection, define the former to be a function of document d's creation date. The prior prospect decreases exponentially with time, therefore the recent documents are ranked higher than older documents. Li and Croft's approach is developed for queries that are after recent documents, but it not handle other types of time-sensitive queries, such as [Madrid bombing], [Google IPO], or even that absolute target one or more past time periods. System structure is based on compactness of data using intervals of indexes. Every time when we stack up the data, index file get generated which will include the lexicons, indexes as well as the frequency of each word from that database. An efficient indexing is necessary for storing the data in order to increase the searching performance. Searching is ended by queries, and queries must be processed fast if the data is properly stored and managed. Then using top k look for method it will and the exact identical data to save the time and space. When there is lots of information, keyword search is companionable for user to access text data. Text data includes web pages, XML documents. Now a days keyword search systems usually use inverted index, a data structure maps each word in the dataset to a list of IDs of documents in which the word appears to efficiently get back documents. The inverted index for a document collection consists of a set of inverted lists .Each inverted list corresponds to a word, which stores all appears in rising order. In reality, datasets are so large that keyword search systems normally use various compression techniques to reduce the space cost of storing inverted indexes. Compression of inverted index reduces the space, and also leads to less disk I/O time during query processing. As a result, these techniques have been extensively studied in recent years. While IDs in inverted lists are sorted in ascending order, such as Variable-Byte Encoding (VBE)[1] and PForDelta [2], store the differences among IDs, called d-gaps, and then use various approaches to encode these d-gaps using shorter binary representations. Though a compressed inverted index is smaller than the original index, the system wishes to decompress encoded lists during query processing, which leads to additional computational costs.

IV. SYSTEM OVERVIEW

Many compression techniques have been designed to reduce the storage space and disk I/O time. Nevertheless, these techniques generally carry out decompression techniques on the fly, which greater the CPU time. This paper presents a

 ISSN: 2321-8169
62 - 66

well-organized index structure, the Generalized Inverted Index (Ginix), which merges successive IDs in inverted lists into intervals to save storage space. The problem of document reordering is correspondent to making similar documents stay near to each other. Silvestri projected a simple method that sorts web pages in lexicographical order based on their URLs as an suitable solution to the problem. This method is sensible because the URLs are typically good indicates of the web page content. The presentation of Ginix is also improved by reordering the documents in datasets using two measurable algorithms. Developing the performance and scalability of Ginix on real datasets show that Ginix not only requires less storage space, but also increases the keyword search performance.

Figure 1 shows basic client server cycle in which server can compressed data using id and client decompressed data for use.

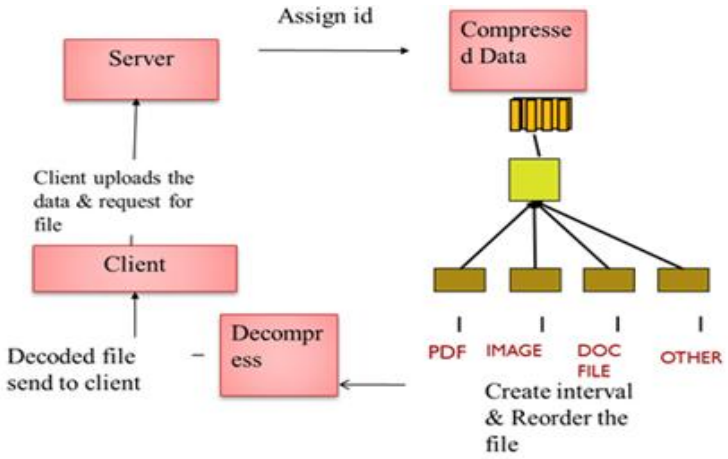

Figure 1. Client Server Cycle

V. ALGORITHMIC STRATEGY

Algorithmic strategy involve algorithms for scan line union as follow-

Search Index Algorithms

Any keyword search method generally uses union and the intersection operations on inverted lists. The union operation is a vital operation to support OR query in which every data file that contain at least one of the query keywords is displayed as an output. The intersection operation is mostly used to support AND query in which only those data files that include all the query keywords are displayed. Typical search algorithms are all based on Identifier lists. This method introduces more computing costs for decode, and Identifier list based search functions can be even more costly because ID lists are generally very large in size.

Below Table 1 shows all information about searching records.

Table 1: Searching Records

Id	Content
1	Keyword ranking in databases
2	Keyword searching in databases
3	Keyword search in relational databases
4	Required fuzzy type-ahead search
5	Navigation system for any item search
6	Keyword search on relational databases
7	Searching for web databases

Table 1 (a1) Dataset content

(b1) Inverted Index		(c1) Generalized Inverted Index	
Word	IDs	Word	Intervals
Keyword	1,2,3,6	Keyword	[1,3],[6,6]
Databases	1,2,3,6,7	Databases	[1,3],[6,7]

1] Union operation

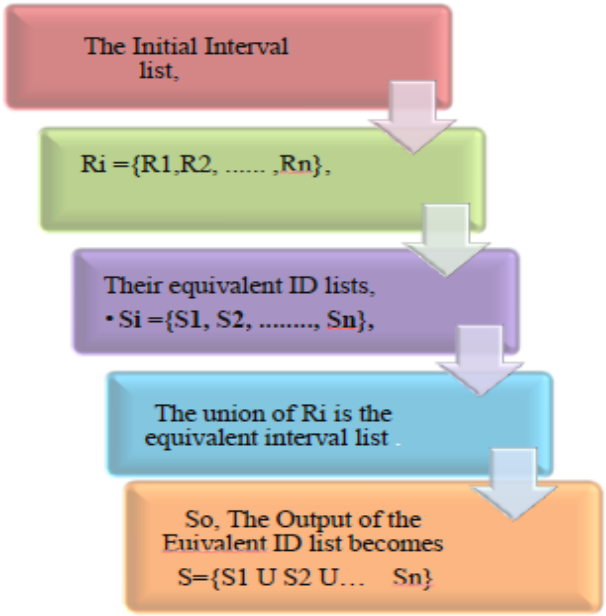

Figure 2: Union operation

Figure 2 shows union operation performed for searching records from database system.

The union of a set of ID lists, denoted by S ={S1, S2, …… , Sn}, is another ID list, in which every ID is restricted in at least one ID list in S. Therefore the union of a set of interval lists can be explained as follows in figure 1. For example, consider the subsequent three interval lists: {[2, 7], [11, 13]}, {[5, 7], [12, 14]}, and {[1, 3], [6, 7], [12, 15]}. Their corresponding ID lists are {3, 4, 5, 6, 7, 11, 12, 13} {5, 6, 7, 12, 13, 14}, and {1, 2, 3, 6, 7, 12, 13, 14, 15}. The union of these three ID lists is {1, 2, 3, 4, 5, 6, 7, 9, 11, 12, 13, 14, 15}, thus, the union of the three interval lists is the equivalent interval list of this ID list, i.e.{[1, 7], [11, 15]}.

2] Intersection operation

The intersection operation, calculates the intersection list of a set of structured lists. As by means of the union of interval lists, the intersection of interval lists can be explained as follows in the figure 2. Consider the three interval lists that we have used earlier: {[2, 7], [11, 13]}, {[5, 7], [12, 14]}, and {[1, 3], [6, 7], [9, 9], [12, 15]}. Their corresponding ID lists are {2, 3, 4, 5, 6, 7, 11, 12, 13} {5, 12, 13, 14} and, {2, 3, 9, 12, 13, 14, 15}, respectively. The intersection list of these ID lists is{ 12, 13}, thus the intersection of the interval lists is the equivalent interval list of this ID list, i.e., {[12, 13]}.

Figure 3 shows intersection operation performed on R and S for searching records from database system.

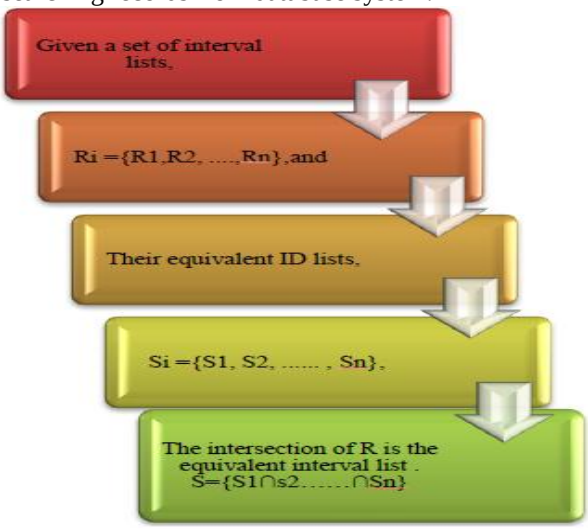

Figure 3: Intersection operation

Algorithm: - Scan line Union (R)

Input: *R* A set of interval lists.
Output: *G* The resulting interval list.

1: **for all** $k \in [1, n]$ **do**
2: Let r_k be the first interval of R_k
3: Insert $\texttt{lb}(r_k)$ and $\texttt{ub}(r_k)$ to min-heap $\mathcal{H}$
4: $a \leftarrow 0, b \leftarrow 0, c \leftarrow 0$
5: **while** $\mathcal{H} \neq \emptyset$ **do**
6: Let t be the top element in $\mathcal{H}$
7: Pop t from $\mathcal{H}$
8: **if** t is a lower-bound **then**
9: $c \leftarrow c + 1$
10: **if** $c = 1$ **then** $a \leftarrow t$
11: **if** t is an upper-bound **then**
12: $c \leftarrow c - 1$
13: **if** $c = 0$ **then** $b \leftarrow t$ and append $[a, b]$ to G
14: Let $r \in R_j$ be the corresponding interval of t
15: Let r' be the next interval (if any) of r in R_j
16: Insert $\texttt{lb}(r')$ and $\texttt{ub}(r')$ to $\mathcal{H}$
17: **return** G

VI. CONCLUSION

This paper presents a generalized inverted index for keyword search in text database to improve the storage space as compared to old method, i.e., inverted index. Along with Ginix this paper also presents a technique of extracting top-k list from web pages. Also, Ginix is suitable with list compression techniques which lead to better performance and on other hand top-k lists are clear, easy to recognize, hence it is an important source for data mining and knowledge discovery. We show a algorithm that automatically retrieve more than 1.7 million such lists from the a web snapshot and also discovers the structure of each list. Our evaluation results show that the algorithm achieves 92% correctness and 72% recall.

REFERENCES

[1] F. Scholer, H. E. Williams, J. Yiannis, and J. Zobel, "Compression of inverted indexes for fast query evaluation", in Proc. of the 25th Annual *International ACM SIGIR Conference on Research and Development in Information Retrieval, Tammpere, Finland, 2002, pp. 222-229.*

[2] M. Zukowski, S. Hman, N. Nes, and P. A. Boncz, "Superscalar RAM-CPU cache compression", in Proc. of the *22nd International Conference on Data Engineering, Atlanta, Georgia, USA, 2006, pp. 59.*

[3] Hao Wu, Guoliang Li, and Lizhu Zhou, "Ginix: Generalized Inverted Index for Keyword Search" *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA MINING VOL:8 NO:1 YEAR 2013*

[4] M. J. Cafarella, E. Wu, A. Halevy, Y. Zhang, and D. Z. Wang, "Webtables: Exploring the power of tables on the web," in *VLDB*, 2008.

[5] Z. Zhang, K. Q. Zhu, and H. Wang, "A system for extracting top-k lists from the web," in *KDD*, 2012.

[6] "Top-k web demo," http://202.120.38.145:8088/TopTenSite/.

[7] X. Cao, G. Cong, B. Cui, C. Jensen, and Q. Yuan, "Approaches to exploring category information for question retrieval in community question-answer archives," *TOIS*, vol. 30, no. 2, p. 7, 2012.

[8] R. Fagin, A. Lotem, and M. Naor, "Optimal aggregation algorithms for middleware," in *PODS*, 2001, pp. 102–113.

[9] K. Chakrabarti, V. Ganti, J. Han, and D. Xin, "Ranking objects based on relationships," in *SIGMOD*, 2006, pp. 371–382.

[10] N. Bansal, S. Guha, and N. Koudas, "Ad-hoc aggregations of ranked lists in the presence of hierarchies," in *SIGMOD*, 2008, pp. 67–78.

[11] A. Angel, S. Chaudhuri, G. Das, and N. Koudas, "Ranking objects based on relationships and fixed associations," in *EDBT*, 2009, pp. 910–921.

ISSN: 2321-8169
62 - 66

Nikhil B. Kedare He is graduate student of Computer Engineering at BVCOE & RI, Nasik under University of Pune. His areas of interest include Data Mining.

Harshada P. Wagh She is graduate student of Computer Engineering at BVCOE & RI, Nasik under University of Pune. Her areas of interest include Data Mining.

Sayali D. Pingle she is graduate student of Computer Engineering at BVCOE & RI, Nasik under University of Pune. Her areas of interest include Data Mining.

Nayana P. Khode she is graduate student of Computer Engineering at BVCOE & RI, Nasik under University of Pune. Her areas of interest include Data Mining.

K. S. Kumavat, ME, BE Computer Engg. Was educated at Pune University. Presently she is working as Head Information Technology Department of Brahma Valley College of Engineering and Research Institute, Nasik, Maharashtra, India. She has presented papers at National and International conferences and also published papers in National and International Journals on various aspects of Computer Engineering and Networks. Her areas of interest include Computer Networks Security and Advance Database.