ISSN: 2321-7782 (Online)
Volume 3, Issue 1, January 2015
International Journal of Advance Research in Computer Science and Management Studies
Research Article / Survey Paper / Case Study
Available online at: www.ijarcsms.com

Inverted Index Schemes for Keyword Search: A Survey of Current Best Practices

Subin Omanakuttan
PG Scholar, Department of Computer Science
College of Engineering Perumon
Kerala, India

Abstract: ***Efficiency of indexing scheme is a crucial component in the design of a Document Retrieval Mechanism (DRM). DRM retrieves a document from a group of documents according to a set of keywords efficiently. Keyword search is also appropriate for document gatherings as well as for accessing structured data, xml documents and relational databases. Thus keyword search is one of the most important activities in the information retrieval system. Indexing scheme assigned for a particular keyword search system determines the storage efficiency and speed of document retrieval for that system. Based on this, various indexing schemes are proposed to enhance the performance of keyword search systems. Search Engines are another major area where these indexing schemes and keyword search systems together serve as a backbone. Current researches in search engine optimization field focuses on providing an indexing mechanism that supports semantic search. This paper conducts a comparative study on various existing inverted indexing mechanisms and concludes with a performance chart with efficiency indications.***

Keywords: Document Retrieval Mechanism ; Inverted Index ; Interval lists ; Inverted index variants ; exclusion list ; search engine indexing ; Keyword Search ; postings list.

I. INTRODUCTION

Inverted indexes are one of the important data structure used in many data storage and information retrieval applications. Inverted index has important role in the working behind search engines. Search engines keep the index of all available documents, resulted from web crawling, by the use of inverted index. For each unique word occurring in a document collection, the inverted index stores a list of the documents in which this word occurs. Each word in this collection is called a term and corresponding to each term it maintain a list, called inverted list, of all the documents in which this word appears. Along with each document in this list it may store some score which indicates how important the document is with respect to that word. Different variants of the inverted index sort the documents in the inverted lists in a different manner. For instance, the sorting order may be based on the document ids or the scores. But we know that real world dataset is very large and hence different compression techniques are necessary to manage inverted index. It reduces space cost as well as disk I/O time during query processing. Compressed inverted index is smaller than the original index and the system needs to decompress encoded lists during query processing. This leads to extra computational costs. To address this problem different variants of inverted index structure are proposed. All of these can be considered as an extension of the traditional inverted index.

In general, contributions of this paper are

- This paper shows the role of inverted index structure on the working behind search engines and its detailed design
- Discusses the problems and inefficiency of traditional inverted index structure and how it is solved in current systems.
- Discusses the need for new variants of inverted index structure for enhancing the performance and memory expense

- Conducts a comparative study of different variants of inverted index structure

II. Basic Concept of Inverted Index

A. Physical Representation

In the basic form, an inverted index is implemented by means of postings lists. Posting lists are associated with each term that appears in the collection. A postings list is comprised of individual postings, each of which consists of a document id and a payload information (P) about occurrences of the term in the document. For simple Boolean retrieval, no additional information is needed in the posting other than the document id that means the existence of the posting itself indicates the presence of the term in the document. The most common payload used is the term frequency or the number of times the term occurs in the document. More complex payloads include positions of every occurrence of the term in the document, properties of the term (such as if it occurred in the page title or not, to support document ranking based on notions of importance) etc. In the web context, anchor text information (text associated with hyperlinks from other pages to the page in question) is useful in enriching the representation of document. The figure given below represents the physical implementation of inverted index

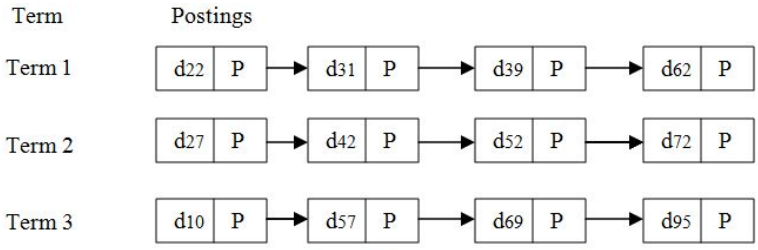

Fig. 1 Physical Representation of Inverted Index

B. Theoretical Representation

For more convenience, on the theoretical basis, it can be defined as follows. For each term t there is an inverted list that contains postings < fd,t , d > where fd,t is the frequency f of term t in the document d. Inverted lists of this form along with additional information such as the document length ld, and ft, the number of documents that contain the term t are sufficient to support efficient query modes. In short an inverted list posting can be mathematically represented as

< fd,t, d, [offset details] >

where fd,t is the frequency f of the term t in the document d and offset details shows the address position of that particular term in the document d. A simple example showing the implementation of inverted index can be given as below.

TABLE I

Term	Inverted Index	Explanation
Constitution	<2,9,[21,45]> <1,6,[5]>	Term occurs 2 times in the document 9 at offsets 21,45 and occurs 1 times in document 6 at the offset 5
Amendment	<2,9,[22,70]>	Term occurs 2 times in the document at the offsets 22 and 70
Constitution Amendment	<1,9,[21]>	By combining the inverted index of individual words

Fig. 2 Table Showing Theoretical Representation of Inverted Index

C. Application of Inverted Index in Web Search Engines

Current web search engines use the concept of inverted index for efficiently managing the index information of available web data and for retrieving the correct documents to the user. The main idea is to create a mapping from every token to a list of its positions in the documents, indexed by search engines. Inverted index is able to perform keyword search much faster than direct indexing methods. Various operations performed on inverted index determine the speed of document retrieval and keyword search. Simple working procedure of a search engine can be implemented as in Fig.3.First it remove the stop words from the user query. Then applies stemming procedure to identify keywords in the query. After that inverted index of each of

the keywords is considered and performs union operation on this data to get the documents that contain all the keywords in the query. Through this process relevant and irrelevant documents may be obtained. In order to eliminate the irrelevant documents search engines applies ranking procedure and algorithms to each of the documents. On doing so, the documents get an id based on the ranks they got. Finally relevant documents are retrieved to the user on the decreasing order of the document rank.

The importance of study of inverted index in the area of search engine can be explained with the help of following data

- In 1998 the first Google index had 26 million pages and reached the 1 billion mark in 2000
- In 2008 it reached 1 trillion and in 2012 Google claimed to have indexed over 30 trillion unique individual live URLs
- According to research results, more than 158 billion searches were performed world wide on each month in 2011.
- In current years, around 5.2 billion searches are performed on everyday ie 61000 searches were performed on one second

All these data shows that, web content and search for web contents increases day by day. The search engine has to face the herculean task of effective management of this web content. For this purpose it has to devise fast and memory efficient indexing mechanisms and in this way, study and experiments in the field of indexing schemes deserves research focus.

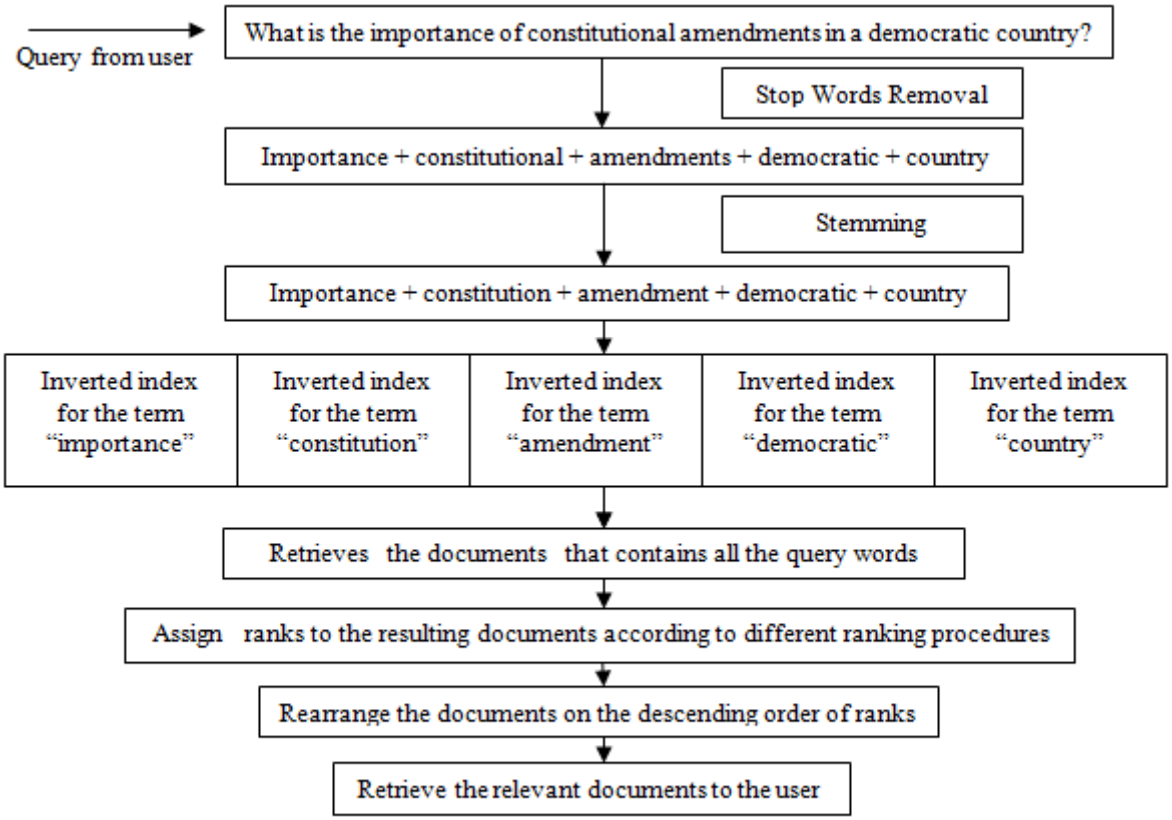

Fig. 3 Query processing of search engines

D. Performance Issues of Inverted Index

Various performance issues related to inverted index schemes are

- Storage of inverted index is a herculean task that requires efficient memory management as web data is extremely large. Different compression techniques are necessary to manage inverted index and complexity of these compression techniques is one of the important factor that can affect the speed of keyword search.

- Even though compressed inverted index is smaller than the original index structure , the system needs to decompress these encoded lists during query processing and after the use system again compresses it and stores in the same position. This leads to extra computational costs and cpu time.
- Insertion, deletion and modification of a single document results in the modification of multiple parts in the index structure.
- Inverted index structure destructs the semantic link between the words and thus doesn't support structured user queries. In other words by the simple use of inverted index we can only find the documents that contains user specified words.
- As web data increases day by day, we can't rely on normal inverted index structure as it explicitly stores each and every document ids in the index structure. This leads to extremely large memory expense.

Due to these reasons, different variants of the present inverted indexing scheme are proposed to enhance the performance. Researches in this field are still going on to build a powerful inverted indexing scheme for performing fast and memory efficient keyword search.

III. Performance Enhanced Variants

Different variants of inverted indexing scheme are proposed to face the existing performance issues. Even slight modifications ,with benefits, in the indexing scheme can save tremendous memory space as it deals with billions and trillions of documents (in case of search engines). Different proposed variants of inverted index are given below

A. Inverted Index with Interval Limits

In this variant instead of storing id list, interval list with upper and lower limits is used to create index. This method avoids the need for frequent compression and decompression of inverted index during query processing and thus saves cpu time and reduces computational cost. This method dramatically decreases the size of inverted index as it stores only integers in the upper and lower limits and avoids the need for explicitly storing all integer document ids. The memory saved in this way can be utilized for data storage.

Implementation: Index implementation with interval limits can be explained as follows. Consider a set of documents numbered from 1 to 7. For convenience each document is made with one simple sentence.

TABLE II

Document ID	Document Content
1	Indian constitution and its features , our duties with articles
2	Constitutional amendments with date of activation and article number
3	Expired amendments in the constitution with article number.
4	Amendments that affects personal freedom
5	Fundamental duties in the constitution
6	Power and duties of judiciary in the constitution to rule legislatives
7	Duties of legislatives in the constitution

Fig. 4 Table Showing 7 Sample Documents

TABLE III

Term	Document ID List
Constitution	1,2,,3,5,6,7
Amendment	2,3,4
Article	1,2,3
Legislative	6,7
Duty	5,6,7

Fig. 5 Table Showing Traditional Inverted Index

TABLE IV

Term	Interval List
Constitution	[1,3],[5,7]
Amendment	[2,4]
Article	[1,3]
Legislative	[6,7]
Duty	[5,7]

Fig. 6 Table Showing Inverted Index with Interval lists

Need for Document Ordering: If the interval lists contains fewer intervals the search algorithms will be faster and it requires only limited storage space. By reordering the available documents more and more documents can be included in the same interval lists. Document Reordering further enhances the memory efficient storage of index.

B. Interval List with Extended Features

Scheme for solving the Issue of single element interval: In the interval list version of inverted index storage, If an interval (l, u) is a single element interval, two integers are necessary for representing the interval. This can be considered as a disadvantage of interval lists. If there are many single element intervals, space cost will be large. It can be solved by avoiding the boundaries and taking the single id .It can be implemented as in the example given below.

TABLE V

Type of Scheme	Term	Interval List	Explanation
Interval List Scheme	Democracy	[1,1],[4,4],[7,7]	Two integers used for single element interval representation
Enhanced Version	Democracy	1,4,7	Replaces the need for 2 integers with 1 integer

Fig. 7 Table Showing Interval lists with and without Single Element Intervals

Scheme that uses the principle of exclusion: Sometimes we can furthermore reduce the size of interval list by using the principle of exclusion. The list of excluded integers is represented with the notation < int 1, int 2 ...int n.> and included integers are represented within square brackets. It can be explained with the help of following example

TABLE VI

Type of Scheme	Term	Interval List	Explanation
Interval List Scheme	Democracy	[1,3],[5,9],[11,55]	Three interval lists for storing the index
Enhanced Version With Exclusion Lists	Democracy	[1,55] <4,10>	Only one interval list and one exclusion list is needed and saved 2 integer spaces.

Fig. 8 Table Showing Interval lists with exclusion lists

C. Inverted Index with Hyperlinks

Research works are going on to create an inverted index scheme on which hyperlinks can be encoded with index information. This will enhance the searching speed and document retrieval can be made more semantic based as we are able to retrieve more related documents after query processing.

D. Inverted Index with Semantic Links

In this case the inverted index of each document is organized in a special manner for preserving the semantic links between the terms. This indexing scheme encodes semantic links between the terms along with the indexing information and thus creates a semantic network for supporting structural user queries. Both structural saving indexation and structural user search query allow to save semantic speech meaning of the text while search process. Researchers are still going on this field to develop a memory efficient indexing scheme along with semantic links and networks can be easily encoded.

IV. Comparative Study of Performance

Performance of different variants of inverted index can be measured by constructing separate indexing scheme for a given database with all the variants. The dataset is first given to traditional inverted indexing scheme and performance is measured. Then index created by each of the variants on the same dataset is compared with that of traditional inverted index. Based on this, a graph that showing the performance enhancement is plotted.

A. Details of Experiments

The performance is measured with the help of Pubmed dataset on Linux server. Pubmed dataset is a free dataset containing references and abstracts on life sciences and biomedical topics. This dataset contains more than 35000 medical documents. But

we selected a reduced version of Pubmed dataset with 4500 medical documents for the easiness of implementation .Inverted Index is build on this dataset based on the principles of different indexing schemes.

B. Experimental Results

Result of comparative study in terms of no: integers required to store in each indexing scheme on same dataset is given below

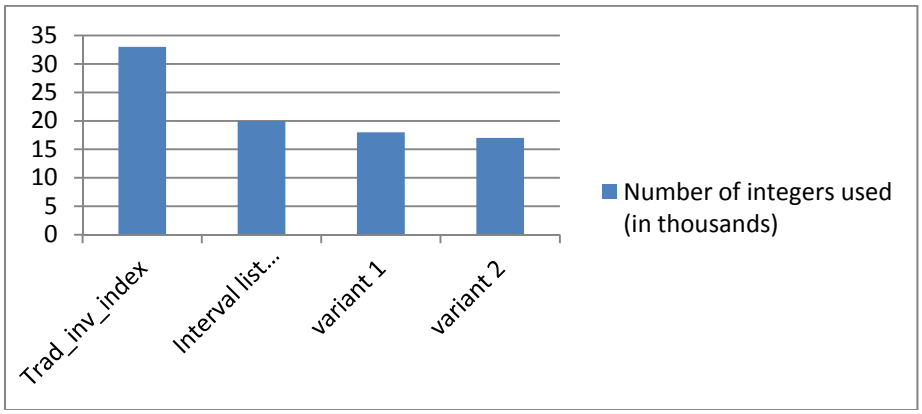

Fig. 9 Graph Showing the performance of various Inverted Indexing Schemes

Where variant 1 is the interval list scheme with provisions to handle single element intervals and variant 2 is the interval list scheme with exclusion lists. Thus the experiment shows that memory requirement for index storage reduces with enhanced versions of inverted index. From this result it can also interpret that, as the number of documents increases searching speed of traditional inverted index decreases sharply but for the performance enhanced variants of inverted index, searching speed doesn't decreases sharply with increase in number of documents.

V. CONCLUSION

This paper conducts a detailed study of inverted indexing schemes and their role in document retrieval and keyword search. Inverted index has wide variety of applications and major one is with search engines. A brief idea about how inverted index is used with search engines is given in this paper. By replacing id list with interval list, memory requirement can be reduced. Two enhanced versions of interval list indexing: one for handling single element interval list and other with exclusion list are discussed here. The comparative study shows that the performance of traditional inverted indexing scheme is get improved due to the concept of interval list. The performance is further improved through enhancements with exclusion lists and single element interval handling scheme.

We can extend this work for modifying the search algorithms of search engines so that performance enhanced versions of inverted index can be made compatible with them.

ACKNOWLEDGEMENT

Our sincere thanks go to all the teaching and non-teaching staff in Department of Computer Science and Engineering, College of Engineering Perumon, for their help and co-operation throughout the work.

References

1. Generalized Inverted index for keyword search(GINIX) by Hao Wu, Guoliang li and Lizhu Zhou, IEEE transactions on knowledge and data mining vol:8 no:1 year 2013.
2. Semantic Sentence Structure Search Engine by Nikita Gerasimov, Maxim Mozgovoy,alexey Lagunov,Proceedings of the 2014 Federated Conference on Computer science and Information Systems.
3. 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.
4. W. Shieh, T. Chen, J. J. Shann, and C. Chung, Inverted file compression through document identifier reassignment, Information Processing and Management, vol. 39, no. 1, pp. 117-131, 2003.

5. F. Silvestri, Sorting out the document identifier assignment problem, in Proc. of the 29th European Conference on IR Research, Rome, Italy, 2007, pp. 101-112.

6. H. Yan, S. Ding, and T. Suel, Inverted index compression and query processing with optimized document ordering, in Proc. of the 18th International Conference on World Wide Web, Madrid, Spain, 2009, pp. 401-410.

7. S. Ding, J. Attenberg, and T. Suel, Scalable techniques for document identifier assignment in inverted indexes. in Proc. of the 19th International Conference on World Wide Web, Raleigh, North Carolina, USA, 2010, pp. 311-320.

8. S. Agrawal, S. Chaudhuri, and G. Das, DBXplorer: Asystem for keyword-based search over relational databases, in Proc. of the 18th International Conference on Data Engineering, San Jose, California, USA, 2002, pp. 5-16.

9. V. Hristidis and Y. Papakonstantinou, DISCOVER: Keyword search in relational databases, in Proc. of the 28th International Conference on Very Large Databases, Hong Kong, China, 2002, pp. 670-681.

10. Optimize document identifier assignment for index compression by Chong Chen, Jing He Ent , Dongdong Shan, Hongfei Yan.

11. Search engine optimization :A survey of current best practices by Niko Solihin, Grand valley state university.

