An Efficient Random Access Inverted Index for Information Retrieval

Xiaozhu Liu
State Key Lab of Software Engineering
Wuhan University
Wuhan 430072, China
lxz_h@163.com

Zhiyong Peng
School of Computer
Wuhan University
Wuhan 430072, China
peng@whu.edu.cn

ABSTRACT
To improve query performance and space efficiency, an efficient random access blocked inverted index (RABI) is proposed. RABI divides an inverted list into blocks and compresses different part of each block with the corresponding encoding method to decrease space consumption. RABI can provide fast addressing and random access functions on the compressed blocked inverted index with the novel hybrid compression method, which can provide both block level and inner block level skipping function and further enhance both space and time efficiencies without inserting any additional auxiliary information. Experimental results show that RABI achieves both high space efficiency and search efficiency, and outperforms the existing approach significantly.

Categories and Subject Descriptors
H.3.1 [Information Storage and Retrieval]: Content Analysis and Indexing - *Indexing methods*.

General Terms
Algorithms, Measurement, Performance.

Keywords
Information Retrieval, Inverted Index, Random Access.

1. INTRODUCTION
The inverted index technique has been comprehensively studied in recent years [1, 2]. An inverted index consists of an index file (vocabulary) and a postings file (a set of inverted lists). Compressing inverted lists is the most popular technique used to increase query throughput [1, 3, 4]. Although the disk access time can be reduced greatly, the compressed index for each query term must be completely decompressed, which will degrade query performance to some extent, especially for a huge amount of text [5, 6, 7].

Some works [2, 4, 7] show that the blocked inverted index with skipping mechanism is a promising way to improve query performance on the compressed inverted index, which can provide fast addressing function with inserting some additional auxiliary information. However those blocked index mechanisms can incur high storage overheads with auxiliary information, and the increase in disk I/O time outweighs the reduction in decompression time for a huge amount of data. Hence how to design a good index to balance the tradeoff between time and space performance is an important and challenge task for large scale information retrieval systems.

In this paper, a novel random access blocked inverted index (RABI) is proposed following our previous work on compressed inverted index [7]. Compared with the existed schemes, RABI can achieve both block level and inner block level fast addressing and random access functions on the compressed index without inserting any additional auxiliary information, which can decrease both space and time consumption.

2. RANDOM ACCESS INVERTED INDEX

2.1 Index Structure and Compression Method
For a given inverted list L_i of term w_i, containing n postings (d_j, fq_j), where d_j is the document ID, $d_j < d_{j+1}, j \in [1, n-1]$, fq_j is the frequency of term w_i in d_j. In order to guarantee that the frequency is in the ascending order without changing the original order of frequency, the frequency fq_j is replaced with the cumulative frequency f_j:

$$f_j = \sum_{i=1}^{j} fq_i, j \in [1, n]. \quad (1)$$

Thus the cumulative frequency f_j has the same ascending order with document ID d_j, which conduces to select appropriate compression method to support fast addressing and random access functions. The structure of the proposed index RABI is shown in Figure 1, where p_i is the address of the blocked inverted list $L_{i,block}$.

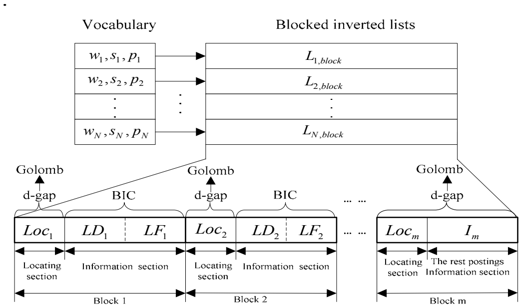


Figure 1: Structure of the random access inverted index

In RABI, each blocked inverted list $L_{i,block}$ consists of m blocks. Every block SB_r includes two sections: locating section Loc_r and information section I_r. Loc_r is a posting (d_i, f_i). For the information sections, except the I_m in the last block SB_m is the residual postings, other information section I_r is made up of list LD_r and list LF_r, where LD_r is the ascending list of document IDs, and LF_r is the ascending list of cumulative frequency. To decrease space cost of $L_{i,block}$ as possible, each section of $L_{i,block}$ is compressed with the corresponding encoding method as shown in Figure 1. For locating section Loc_r and the last block SB_m, RABI firstly compresses them with d-gap scheme. Considering that there is only one posting in the locating section Loc_r, we choose the good compression ratio Golomb coding (actual any other good encoding scheme can use) to compress Loc_r and SB_m.

In order to implement fast locating and random access of the compressed index without inserting any additional auxiliary information, the key to the success of this mechanism is to find efficient encoding methods with accurate addressing and random access functions for compressing the document IDs and the cumulative frequencies in the information section within a block. The compression method should meet two conditions: the first condition is that it can achieve block level skipping without inserting any auxiliary information, and the second is that it can support inner block skipping, namely directly random access any element in a block with only decompressing the element. Since the binary interpolative coding (BIC) [3] method can efficiently compress the ascending order integer set, and BIC is also easy to calculate the space cost of any element in the compressed set if the first and last integers are known. If we know the postings of Loc_r and Loc_{r+1}, the BIC method will meet the two necessary conditions mentioned above. Hence we adopt BIC to compress LD_r and LF_r of I_r.

2.2 Decoding and Random Access

For the convenience of decoding, we need to know the locating sections Loc_r and Loc_{r+1} to calculate the space cost (bits) of I_r, so we slightly adjust the physical storage order of $L_{i,block}$ as:

$$L_{i,block} = Loc_1, Loc_2, I_1, Loc_3, I_2, \cdots, Loc_m, I_{m-1}, I_m. \tag{2}$$

Let $P(Loc_1)$ denote the address of Loc_1. Then we can get the addresses of Loc_1, LD_r and LF_r in any block:

$$P(Loc_r) = \begin{cases} p_i, r = 1, \\ P(Loc_1) + B_{Golomb,k}(Loc_1), r = 2, \\ P(Loc_1) + \sum_{j=1}^{r-1} B_{Golomb,k}(Loc_j) \\ \quad + \sum_{j=1}^{r-2} [B_{BIC,k}(LD_j) + B_{BIC,k}(LF_j)], r \in [3, m] \end{cases} \tag{3}$$

$$P(I_r) = P(LD_r) = \begin{cases} P(Loc_{r+1}) + B_{Golomb,k}(Loc_{r+1}), r \in [1, m-1], \\ P(I_{m-1}) + B_{BIC,k}(LD_{m-1}) + B_{BIC,k}(LF_{m-1}), r = m, \end{cases} \tag{4}$$

$$P(LF_r) = P(I_r) + B_{BIC,k}(LD_r), r \in [1, m], \tag{5}$$

where $B_{Golomb,k}(Loc_r)$ is the space cost (bits) of Loc_r, $B_{BIC,k}(LD_r)$ is the space cost of LD_r, $B_{BIC,k}(LD_r)$ is the space cost of LF_r, k is the number of postings per block. According to the principle of BIC, we have:

$$B_{BIC,k}(LD_r) = B_{BIC,k}(d_{r \cdot k+1} - d_{(r-1) \cdot k+1} - 1), \tag{6}$$

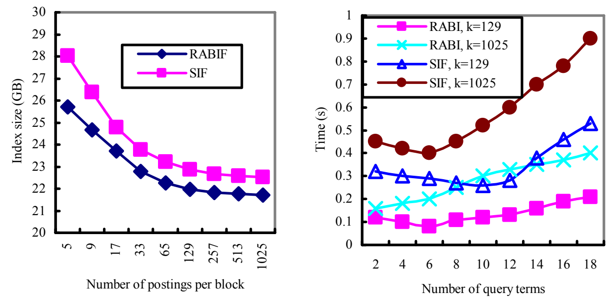
$$B_{BIC,k}(LF_r) = B_{BIC,k}(f_{k \cdot r+1} - f_{k(r-1)+1} - 1), \tag{7}$$

$$B_{BIC,k}(D) = \begin{cases} 0, if \quad D = k - 1, \\ (k-1) \cdot \lceil \log_2 D \rceil, otherwise. \end{cases} \tag{8}$$

Hence, with the known p_i and k, we can obtain the address of any element in the compressed list $L_{i,block}$ by the expressions mentioned above. Then RABI can provide both block level and inner block level fast addressing and random access on the compressed index without inserting any additional auxiliary information.

3. EXPERIMETAL RESULTS

To evaluate the efficiency of various inverted file organizations, the skipped inverted file (SIF) [1, 2] and RABI were implemented with C++. All experiments were run on an Intel P4 3.0GHz PC with 1GB DDR memory system. We crawled a huge amount of real data from the Internet, and there were approximate 1,000,000 documents. We gave the actual space cost and conjunctive Boolean query processing time with varying number k of postings per block in Figure 2.



(a) Space cost (b) Query time performance

Figure 2: Performance of two schemes

4. CONCLUSIONS

In this paper, we have studied compression and query processing of an inverted index to improve time and space performance for information retrieval systems. Our proposed RABI divides the inverted list into blocks and employs a novel hybrid compression method to support fast addressing and random access functions. Compared with the existed mechanisms, RABI can support both block and inner block levels skipping function with less storage overhead. Experimental results show that, compared with SIF, our proposed RABI averagely reduces space cost by 5.3%, conjunctive Boolean query time by 25.8%. This provides a very simple and attractive way to build a fast and space-economical information retrieval system.

5. ACKNOWLEDGMENTS

This work is supported by the National Natural Science Foundation of China under Grant No.90718027, and the National Key Basic Research and Development (973) Plan of China under Grant No. 2007CB310806.

6. REFERENCES

[1] J. Zobel, A. Moffat. Inverted Files for Text Search Engines. *ACM Computing Surveys*, 38(2): 1-56, 2006.

[2] A. Moffat, J. Zobel. Self-Indexing Inverted Files for Fast Text Retrieval. *ACM Transactions on Information Systems*, 14(4): 349-379, 1996.

[3] A. Moffat, L. Stuiver. Binary interpolative coding for effective index compression. *Information Retrieval*, 3(1): 25-47, 2000.

[4] D. K. Blandford, G. E. Blelloch. Compact representations of ordered sets. In *Proc. of the 15th annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 11-19, 2004.

[5] S. Buttcher, Charles L. Clarke. A. Index compression is good, especially for random access. In *Proc. of the 16th ACM CIKM*, pages 761-770, 2007.

[6] J. Zhang, X. Long, T. Suel. Performance of compressed inverted list caching in search engines. In *Proc. of the 17th International Conference on World Wide Web*, pages 387-396, 2008.

[7] X. Liu, Z. Peng. Time and Space Efficiencies Analysis of Full-Text Index Techniques. *Journal of Software*, 20(7): 1768-1784, 2009.