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

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 \((d_j, f_q)\), where \( d_j \) is the document ID, \( d_j < d_{j+1}, j \in [1, n−1] \), \( f_q \) 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 \( f_q \) is replaced with the cumulative frequency \( f_j \):

\[
f_j = \sum_{i=1}^{j} f_q, j \in [1, n].
\]

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 \( L_{i, \text{block}} \).

![Figure 1: Structure of the random access inverted index](image)

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

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 process 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 and Loc_m, the BIC method will meet the two necessary
conditions mentioned above. Hence we adopt BIC to compress
LD_i and LF_i of I_i.

2.2 Decoding and Random Access

For the convenience of decoding, we need to know the locating
sections Loc, and Loc_m to calculate the space cost (bits) of I_i,
so we slightly adjust the physical storage order of I_i as:

\[ I_{block} = Loc, Loc_{m}, Loc_{m}, Loc_{m}, ..., Loc_{m} \]

Let \( P(\text{Loc}) \) denote the address of \( \text{Loc} \). Then we can get the
addresses of \( \text{Loc} \), \( \text{LD} \), and \( \text{LF} \) in any block:

\[ P(\text{Loc}) = P(\text{Loc}) + \sum_{r=1}^{m} B_{\text{Loc},r}(\text{Loc}) \]

\[ P(\text{Loc}) = \sum_{r=1}^{m} B_{\text{Loc},r}(LD_i) + B_{\text{Loc},r}(LF_i) \]

\[ P(\text{Loc}) = \sum_{r=1}^{m} B_{\text{Loc},r}(LD_i) + B_{\text{Loc},r}(LF_i) \]

where \( B_{\text{Loc},r}(\text{Loc}) \) is the space cost (bits) of \( \text{Loc} \), \( B_{\text{Loc},r}(\text{LD}) \) is
the space cost of \( \text{LD} \), \( B_{\text{Loc},r}(\text{LF}) \) is the space cost of \( \text{LF} \), \( k \)
is the number of postings per block. According to the principle of
BIC, we have:

\[ B_{\text{Loc},r}(\text{LD}) = B_{\text{Loc},r}(d_{i,1} - d_{i,k+1}) \]

\[ B_{\text{Loc},r}(\text{LF}) = B_{\text{Loc},r}(f_{i,1} - f_{i,k+1}) \]

\[ B_{\text{Loc},r}(D) = \begin{cases} 0, & \text{if } D = k+1 \\ (k-1) \cdot \log_2 D, & \text{otherwise} \end{cases} \]

Hence, with the known \( p_r \) and \( k \), we can obtain the address of
any element in the compressed list \( I_{block} \) 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. EXPERIMENTAL 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.

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

In Proc. of the 15th annual ACM-SIAM Symposium on Discrete
[5] S. Butcher, Charles L. Clarke. A. Index compression is good, especially
caching in search engines. In Proc. of the 17th International Conference