Application of Bitmap Index to Information Retrieval

Kengo Fujioka Yukio Uematsu Makoto Onizuka

NTT CyberSpace Laboratories, NTT Corporation
1-1 Hikarinooka, Yokosuka
Kanagawa 239-0847, Japan

fujioka.kengo@lab.ntt.co.jp uematsu.yukio@lab.ntt.co.jp onizuka.makoto@lab.ntt.co.jp

ABSTRACT

We developed the *HS-bitmap index* for efficient information retrieval. The HS-bitmap index is a hierarchical document-term matrix: the original document-term matrix is called the *leaf matrix* and an upper matrix is the summary of its lower matrix. Our experiment results show the HS-bitmap index performs better than the inverted index with a minor space overhead.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Search process

General Terms

Algorithms, Performance, Experimentation

Keywords

information retrieval, bitmap index

1. INTRODUCTION

Inverted indices are widely used in information retrieval (IR) systems. However, inverted indices degrade Boolean query processing performance when queries and lists of results are long [1]. To deal with this performance issue, we apply bitmap indices, which carry out Boolean operations efficiently [2]. To apply bitmap indices to IR, we built a document-term matrix by recording whether or not a document contains a term [3]. Figure 1 shows an example of a document-term matrix. If the document contains a term, its field in the matrix is marked 1. If not, its field is marked 0. A Boolean query with multiple terms can be processed by bitmap-joining the column vectors of the terms in the document-term matrix.

	Terms			
Doc. num.	dog	apple	cat	orange
1	1	1	0	0
2	1	0	1	0
3	0	1	1	1
4	0	0	0	0

Figure 1. Document-term matrix

However, bitmap indices in IR, specifically document-term matrixes, require too much space to be handled efficiently, even where 1s in bitmaps are very sparse. This problem is a result of IR having to handle huge amounts of records and attribute values, e.g., Web pages and terms.

To solve this problem, we devised the *HS-bitmap index*, which is hierarchically comprised of compressed data of *summary bits*. A summary bit in an upper matrix is obtained by logical OR of the n bits in its corresponding lower matrix. Let n denote the *summary unit*. Summary units are constant and determined by an appropriate rule in the document-term matrix. The hierarchical summary bits provide bitmap indices with efficient bit operations, usage of space for large sparse bitmaps, and partial decompression. To evaluate the efficiency of the HS-bitmap index, we implemented it on top of PostgreSQL, compared the query performance of the HS-bitmap index with that of the inverted index, and found that the HS-bitmap index performs better especially when queries are long.

2. HS-BITMAP INDEX

2.1 Data Structure

Figure 2 shows the hierarchical structure of the summary bitmaps in the HS-bitmap index. This example shows that a summary bit in the first matrix is representative of two bits in the second matrix. We call a document-term matrix a *leaf matrix* and column vectors in this matrix *raw bit vectors*. First, to construct the HS-bitmap index, we horizontally divide the leaf matrix by summary unit n. Second, we obtain summary bits from n bits in the raw bit vectors. We call column vectors in the upper matrix *summary bit vectors*. Here, 1s in the summary bit vector have methods to access their corresponding parts in their lower matrix. The HS-bitmap index construction repeats this sequence until the summary bit vector size becomes small enough to be efficiently handled. Third, we compressed all bit vectors in the matrix and stored them. Finally, we create a term index to process the look-up for the summary bit vectors.

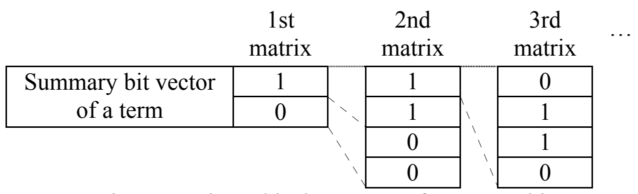


Figure 2. Hierarchical structure of summary bits

2.2 Query Execution

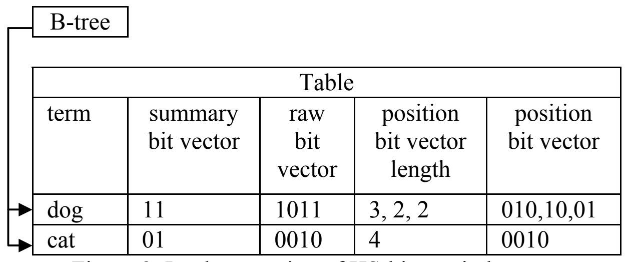
Information retrieval systems can execute a conjunctive Boolean query using the HS-bitmap index, as explained below. In step 1, the summary bit vectors of query terms in the first matrix are obtained using the term index. In step 2, these vectors are joined by bitwise operations. In step 3, the summary bit vectors of query terms in the second matrix are partially obtained using 1s in the result bit vector in step 2. Steps 2 and 3 are then repeated until reaching the leaf matrix. Query answers are indicated by 1s in the result bit vector.

2.3 Word Position

Information retrieval systems often use word position, i.e., where the word appears in a document [4]. The HS-bitmap index stores the word position in the form of the position bit vector. This vector is obtained by recording whether or not the word appears in a position in a document. The length of the position bit vector is the number of words in the document. The HS-bitmap index executes phrase queries by processing bit shifts and Boolean operations.

3. EXPERIMENT

We implemented the HS-bitmap index on top of PostgreSQL. To simplify implementation, we used the B-tree index for the term index. Then we stored the bit vector part in the table. An outline of how we implemented the HS-bitmap index is shown in Figure 3. This HS-bitmap index has two layers. The stored data was compressed by TOAST, which is the PostgreSQL compression algorithm. This algorithm does not implement a partial decompression mechanism.



term	summary bit vector	raw bit vector	position bit vector length	position bit vector
dog	11	1011	3, 2, 2	010,10,01
cat	01	0010	4	0010

Figure 3. Implementation of HS-bitmap index

We measured the HS-bitmap index performance in experiments in which Linux 2.6.18-5 and PostgreSQL 8.2.4 were running with Intel Xeon 2.66 GHz of 64-bit CPU, 16.4 Gbytes of memory, and 1.4 Tbytes of disk. We constructed an HS-bitmap index for the data set issued by IREX, which is a Japanese language IR and information extraction contest. The IREX test data set consists of about two hundred thousand documents and its size is 866 Mbytes. The summary unit is one of the tuning parameters of the HS-bitmap index, and we set the summary unit to 512. Test queries are made from the terms extracted from the documents in the data set. To avoid the effects of disk access, we set the PostgreSQL shared buffer size to 2 Gbytes. As a reference, we ran Ludia in the same conditions. Ludia is the PostgreSQL binding of Senna, which is a text search engine with an inverted index [5].

We measured the retrieval times of bag-of-words queries while varying the query length. The results are shown in Figure 4. The time in Figure 4 is the average of processing three queries three hundred times. Our results show that the HS-bitmap index retrieved faster than the inverted index, particularly when the query length was long. The HS-bitmap index size was about 400 Mbytes, and the inverted index size of Ludia was about 300 Mbytes. These sizes include the term index and word position data. They were compressed by TOAST.

We also measured retrieval times for bag-of-words queries while varying query terms. When document frequencies (Df) of terms are large, e.g., stop words, the HS-bitmap index is not efficient. One reason for this is that summary bits do not work well when almost all of the summary bits are 1. For phrase queries with two terms, the retrieval time of the HS-bitmap index is 19 msec, and that of the inverted index Ludia is 11 msec. The HS-bitmap index performance worsens when the phrase query length is increased. One reason for this is that the word position data is large so copying the data from the buffer to the work memory is very expensive.[1]

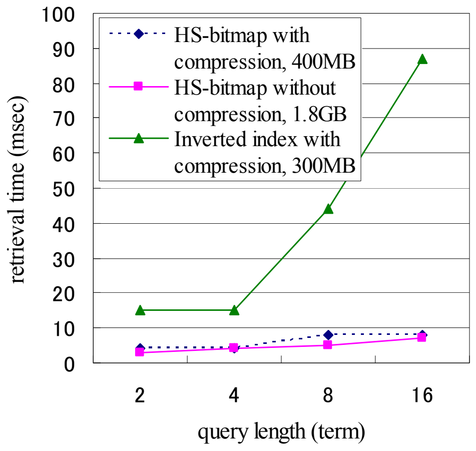


Figure 4. Retrieval time vs. query length

4. CONCLUSION AND FUTURE WORK

We developed the HS-bitmap index for IR systems. Its size was slightly larger than inverted index. Also, the HS-bitmap index executed conjunctive Boolean queries faster than the inverted index, particularly when the query length was long. In the future, we will improve the HS-bitmap index. We will use sentence bit vectors for within-same-sentence queries. Sentence bit vectors are obtained by recording whether or not a word appears in a sentence in a document. We will also cluster raw bits semantically in a document-term matrix. We expect that semantic clusters gather 1s in matrixes and reduce the population of 1s in summary bitmaps.

5. REFERENCES

[1] T. Strohman and W. Bruce. Efficient Document Retrieval in Main Memory. In 30th annual international ACM SIGIR conference, pages 175–182, 2007.

[2] H. Garcia-Molina, J. D. Ullman and J. Widom. Database Systems: The Complete Book. Prentice Hall, 2001.

[3] C. D. Manning, P. Raghavan and H. Schutze. Introduction to Information Retrieval (Preliminary draft). http://www-csli.stanford.edu/~hinrich/information-retrieval-book.html.

[4] J. Zobel and A. Moffat. Inverted Files for Text Search Engines. In ACM Computing Surveys archive Volume 38, Issue 2, Article No. 6, 2006.

[5] Senna: An Embeddable Fulltext Search Engine. http://qwik.jp/senna/FrontPage.html

[1] In our experiments, the HS-bitmap index size was about 100 Mbytes without word position data.