Hybrid Index Structures for Temporal-Textual Web Search

Peiquan Jin, Hong Chen, Sheng Lin, Xujian Zhao, and Lihua Yue

School of Computer Science and Technology,
University of Science and Technology of China, 230027, Hefei, China
jpq@ustc.edu.cn

Abstract. Most Web pages contain temporal information. However, most of previous studies only consider the update time of Web pages rather than fully exploit different temporal features in Web. In this paper, we propose a novel approach to fusing different temporal features in Web pages to build an efficient index structure for temporal-textual Web search. Specially, we focus on *update time* and *content time*, and propose to use a hybrid index structure to organize textual keywords, update time, and content time. In particular, we study three mechanisms to implement a hybrid index structure for temporal-textual Web search: (1) first inverted file then MAP21-tree and B+-tree, (2) first inverted file then MAP21-tree, (3) expanded inverted file. We conduct experiments on a real dataset to evaluate the performance of those hybrid index structures. The experimental results show that the *first inverted file then MAP21-tree* index structure has the best query performance.

1 Introduction

Recently, time has been a focus in the area of Web information extraction and Web search [1]. However, most of the studies only consider the update time of Web pages rather than fully exploit different temporal features in Web. Traditional commercial search engines, such as Google, Bing, and Baidu, only support the crawled dates of Web pages, i.e., users can only query Web pages towards their creation dates in database. To our knowledge, there are few search systems considering the temporal information embedded in Web pages [2].

The current temporal text indexing is mainly towards the versioned document collections such as Web archives [3, 4]. There have been some indexing approaches on directly addressing the issue of temporal-textual indexing. Anick and Flynn [5] have pioneered this research to support versioning in a full-text index on bitmaps for terms in current versions, and delta change records to track incremental changes to the index backward over time. The disadvantage is the costly recreation of previous states. Recent work in [6] and its earlier proposals [14-17] concentrate on the problem of supporting text-containment queries and neglect the relevance scoring of results. Stack [7] reports practical experiences made when adapting the open source search engine *Nutch* to search Web archives. Weikum et al. address the temporal dimensions completely by extending the inverted files index to make it ready for temporal search and implement the time-travel text search in the FluxCapacitor prototype [8, 9].

X. Du et al. (Eds.): APWeb 2011, LNCS 6612, pp. 271–277, 2011.

In contrast, research in temporal databases has produced several index structures tailored for time-evolving databases. A comprehensive overview of the state-of-art is available in [10]. Unlike the inverted file index, their applicability to text search is not well understood.

The index structure of most related work about temporal-textual indexing is usually based on inverted file index [11]. However, the main difference between our work and previous researches is that we consider to index both update time and content time for Web pages, while previous temporal-textual indexes are focused on indexing update time, because they are designed for Web archive system or document versioning.

In this paper, we propose a novel approach to fusing different temporal features in Web pages, i.e., update time and content time, to build an efficient index structure for temporal-textual Web search. Our basic idea is to develop an efficient hybrid index structure to cope with temporal-textual queries, which makes an integration of traditional temporal index and textual index. The most famous textual index is the inverted file structure, so in this paper we use this structure as the basic textual index structure. For temporal index, we adopt the MAP21-Tree [12], which is an efficient temporal index structure in temporal database area. However, there are many choices when integrating inverted file with MAP21-Tree, and in some case we need to introduce B+-Tree as the index structure for update time. Hence, we aims at making a comparison study on those different integration mechanisms, and finally find the best hybrid index structure which has the best performance for temporal-textual queries.

2 Index Structures for Temporal-Textual Web Search

We aim at building hybrid index structures to integrate text keywords and temporal information of Web pages for temporal-textual Web search. We adopt the *inverted file* as the basic index structure for text keywords in Web pages, due to its widely use in document search. The temporal information contains update time and primary time, in which the update time is regarded as a time instant and the primary time is modeled as a time period. Primary time refers to the most appropriate content time associated with a Web page. The time granularity is set to day. As the update time is a time instant, we can use B+-tree to organize them or directly put them into inverted files. For the primary time, we adopt the MAP21-tree as the basic index structure. MAP21-tree is designed towards time period and has better performance than other temporal indexes such as R-tree [13].

Basically, there are five hybrid methods when considering B+-tree, inverted file, and MAP21-tree to index temporal information and text keywords together, which are:

(a) *inverted file, B+-tree and MAP-21 triple index.*
(b) *expanded inverted file.*
(c) *first inverted file then MAP21-tree and B+-tree.*
(d) *first inverted file then MAP21-tree.*
(e) *first MAP21-tree then inverted file.*

However, the (a) and (e) methods are not suitable in temporal-textual Web search. The method (a) has to build multiple index files, which will cost more storage spaces and result in poor search performance. The method (e) performs poorly when no time predicates are given in the query. So in this paper we focus on the remaining three methods. In additional, the method (b), namely expanded inverted file, can be considered as the representative of the previous work on building temporal index, because most existing work about temporal text indexing are focused on indexing update time and keywords together and typically use a expanded inverted file [8, 9]. Figure 1 to 2 illustrate the index structures corresponding to (b), (c), and (d), in which the symbol UT refers to update time, while PT refers to primary time.

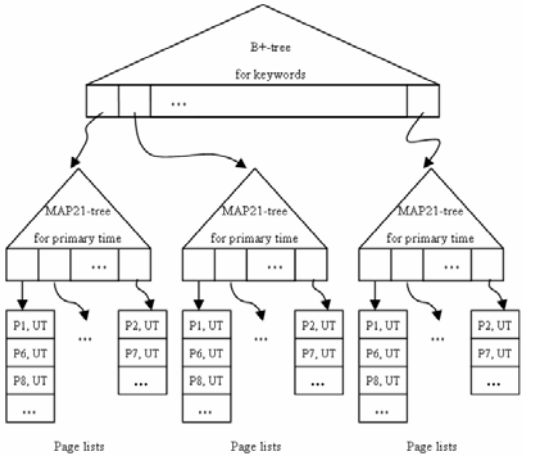
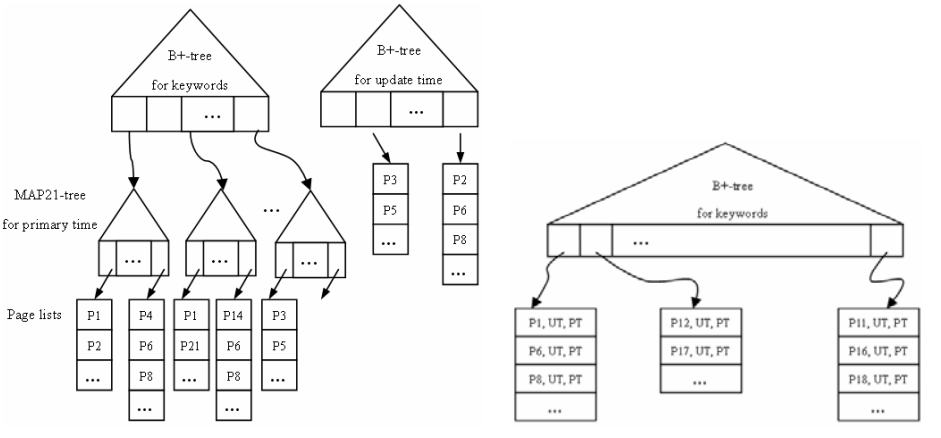


Fig. 1. Illustration of first inverted file then MAP21-tree



(a) First inverted file then MAP21-tree and B+-tree (b) Expanded inverted file structure

Fig. 2. Illustration of the other two index structures

3 Experiments

In this section, we will implemented the three hybrid index structures and make comparison experiment to show the different performance of those index structures under different workloads. For simplicity, we use the following notation to represent the three hybrid index structures discussed in the previous section:

***S*$_1$:** first inverted file then MAP21-tree and B+-tree.
***S*$_2$:** first inverted file then MAP21-tree.
***S*$_3$:** expanded inverted file.

We mainly evaluate the query performance of the three index structures. In order to get a comprehensive result, we use five types of queries in the experiment, which are:

Query 1: keywords only query
Query 2: keywords + update time instant
Query 3: keywords + primary time instant
Query 4: keywords + update time instant + primary time instant
Query 5: keywords + update time period + primary time period

Among those queries, Query 1 to 3 are partial queries, and Query 4 and 5 are complete temporal-textual queries.

In our experiments, we focus on the index size, query time, and rebuilt time of each index structure under given workloads and temporal-textual queries. Both query time and rebuilt time include disk I/O time and memory operation time.

3.1 Settings and Dataset

We choose the real dataset from the corpus provided by the SouGou lab (http://www.sogou.com/labs/) which records games, sports, IT, domestic and international news in May 2008 from some news sites.

In this experiment, we simply describe how to exact the update time and the primary time in one real Web page. In news Web pages, publish time is usually regarded as update time, and primary time is often appears in the first paragraph. The exaction of primary time is not the focus of this paper, so here we conduct a simple algorithm to extract primary time from Web pages. According to the simple algorithm, we always select primary time from the first paragraph of a Web page. If there is only time word in the first paragraph, we consider it as primary time. Otherwise, if there are two or more time words in the first paragraph, we choose the nearest time to the update time of the Web page as the primary time.

Keywords in Web page are exacted by a tool called ICTCLAS (http://ictclas.org/), which is the most efficient tool for the Chinese words segmentation. Each word is mapped a value in memory by ELFHASH function.

We use 250 thousand news Web pages as our real dataset and extract approximately 210 thousand different keywords. We run our experiment in a computer with an Intel Core 2.00 GHz CPU, 2 GB RAM, using Microsoft Window 7.

3.2 Comparison of Three Hybrid Index Structures

Firstly, we compare the index size (Mbytes) of three hybrid index structures. In our experiment, S1 has the smallest size, while S3 costs largest storage. The detailed size

of S_1, S_2, and S_3 are 1346.29 Mbytes, 1425.14 Mbytes, and 1528.07 Mbytes, as shown in Table 1. Generally, we have the following result: $size(S_1) < size(S_2) < size(S_3)$, and our experimental result has also indicated this truth.

Secondly, we measure the rebuilt time to compare the creation time of the three structures, as the creation time of index is crucial in Web search. In this experiment, we first drop the original index structure, and then insert all extracted items, which are formed as <URL, keywords set, update time, primary time>, into the index structure, and compute the insertion time for each index structure. The rebuilt time of S_1, S_2, and S_3 needs 29235.10, 31310.14, and 14170.90 seconds respectively, as shown in Table 1. Since S_1 and S_2 have to construct MAP21-trees to maintain primary time, while S_3 only needs to create a single B+-tree, both S_1 and S_1 consume much more time than S_3.

Table 1. Index sizes and rebuilt time for three index structures

	Index Size (MBytes)	Rebuilt Time (s)
S_1	1346.29	29235.10
S_2	1425.14	31310.14
S_3	1528.07	14170.90

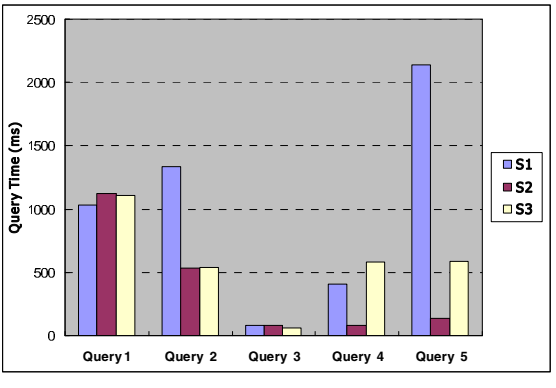


Fig. 3. Query time of the three index structures

Thirdly, we consider the query time of the three structures. Figure 3 shows the result of query time experiment. The query time includes three parts, i.e., looking-up on corresponding trees, reading page lists from disk, and merging page lists. The query time is calculated from query input to the output of the URLs results. Five types of queries are used in this experiment, which are denoted as Query 1 to Query 5, as described in the beginning of this section. We execute 300 queries and get the average run time as the experimental result of query time. Each query contains three keywords, each of which is a common keyword in the dataset, as indicated in the search log provided by the SouGou lab (http://www.sogou.com/labs/). Each query also

contains one update time and one primary time, which are randomly generated but falls into a certain range of time.

According to our experiment, S_2 has the best query performance with respect to the complete temporal-textual queries, i.e., queries involving keywords, primary time, and update time. It only costs about 14% to 20% of time to complete such queries, compared with S_1 and S_3. Besides, S_2 has a comparable query performance when performing partial temporal-textual queries. Thus, in general S_2 has shown best performance in the measurement of query time.

4 Conclusions

In this work we have designed and implemented three hybrid index structures for temporal-textual Web search and studied the performance of these index structures. We conduct an experiment on a real dataset, and use five temporal-textual query types to evaluate the index size, query time, and rebuilt time of each hybrid index structure. The experimental results show that the index structure "*first inverted file then MAP21-tree*" has the relative better query performance among the three index structures, whereas it has a comparable index size and rebuilt time with the other two competitors. In conclusion, the index structure "*first inverted file then MAP21-tree*" may be an acceptable choice for temporal-textual Web search engines.

Acknowledgements

This work is supported by the National Science Foundation of China (no. 60776801 and 70803001), the Open Projects Program of National Laboratory of Pattern Recognition (20090029), the Key Laboratory of Advanced Information Science and Network Technology of Beijing (xdxx1005), and the USTC Youth Innovation Foundation.

References

1. Alonso, O., Gertz, M., Yates, R.B.: On the value of temporal information in information retrieval. In: Proc. of SIGIR 2007, pp. 35–41 (2007)
2. Deniz, E., Chris, F., Terence, J.P.: Chronica: a temporal Web search engine. In: Proc. Of ICWE 2006, pp. 119–120 (2006)
3. Herscovici, M., Lempel, R., Yogev, S.: Efficient Indexing of Versioned Document Sequences. In: Amati, G., Carpineto, C., Romano, G. (eds.) ECiR 2007. LNCS, vol. 4425, pp. 76–87. Springer, Heidelberg (2007)
4. Grandi, F.: Introducing an Annotated Bibliography on Temporal and Evolution Aspects in the World Wide Web. SIGMOD Record 33(2), 84–86 (2004)
5. Anick, P.G., Flynn, R.A.: Versioning a Full-Text Information Retrieval System. In: Proc. of SIGIR (1992)
6. Nørvåg, K., Nybø, A.O.N.: DyST: Dynamic and Scalable Temporal Text Indexing. In: Proc. of TIME (2006)
7. Stack, M.: Full Text Search of Web Archive Collections. In: Proc. of IWAW (2006)

8. Berberich, K., Bedathur, S.J., Neumann, T., Weikum, G.: FluxCapacitor: Efficient Time-Travel Text Search. In: Proc. Of VLDB, pp. 1414–1417 (2007)
9. Berberich, K., Bedathur, S.J., Neumann, T., Weikum, G.: A Time Machine for Text Search. In: Proc. Of SIGIR, pp. 519–526 (2007)
10. Salzberg, B., Tsotras, V.J.: Comparison of Access Methods for Time-Evolving Data. ACM Comput. Surv. 31(2), 158–221 (1999)
11. Zobel, J., Moffat, A.: Inverted Files for Text Search Engines. ACM Comput. Surv. 38(2), 6 (2006)
12. Nascimento, M., Dunham, M.: Indexing Valid Time Databases via B+-Trees. IEEE Transactions on Knowledge and Engineering 11(6), 929–947 (1999)
13. Beckmann, N., Kriegel, H.P., Schneider, R., Seeger, B.: The R-tree: An efficient and robust access method for points and rectangles. In: Proc. Of SIGMOD, pp. 322–331 (1990)
14. Nørvåg, K.: Space-Efficient Support for Temporal Text Indexing in a Document Archive Context. In: Koch, T., Sølvberg, I.T. (eds.) ECDL 2003. LNCS, vol. 2769, pp. 511–522. Springer, Heidelberg (2003)
15. Nørvåg, K.: Supporting temporal text-containment queries in temporal document databases. Data Knowl. Eng. 49(1), 105–125 (2004)
16. Nørvåg, K., Nybø, A.O.: Improving Space-Efficiency in Temporal Text-Indexing. In: Zhou, L.-z., Ooi, B.-C., Meng, X. (eds.) DASFAA 2005. LNCS, vol. 3453, pp. 791–802. Springer, Heidelberg (2005)
17. Nørvåg, K., Nybø, A.O.: Albert Overskeid Nybø. DyST: Dynamic and Scalable Temporal Text Indexing. In: Proc. of TIME, pp. 204–211 (2006)