A Design of the Inverted Index Based on Web Document Comprehending

Shaojun Zhong
Jiangxi University of Science and Technology, Ganzhou, China
Email:infor2000@qq.com

Min Shang and Zhijuan Deng
Yuxi Normal University, Yuxi, China
Jiangxi University of Science and Technology, Ganzhou, China
Email:bettysm@126.com, 66162815@qq.com

***Abstract*—A design of inverted index based on web document comprehending is induced and the algorithm is presented by combining the inverted index and the web documents understanding technology. Experimental results show that less time is cost than the traditional inverted index to query in the documents with the same size.**

***Index Terms*—inverted index, Web document comprehending, latent semantic analysis, correlation**

I. INTRODUCTION

The emergence of the Internet makes major changes of the information retrieval. Search engines have become a convenient way to query data, access to information. With the search engines' development the commercial engine has progressed several generations. A technical expert comes from Microsoft Research, advocated: "There are 75%'s contents can not search out by the general search engines [1]."

The birth of topic search engine is to improve the comprehensive and the accuracy of search in maximum. The topic search engine committed to provide more comprehensive and professional services in topics related. Using the traditional search method based on keywords matching constrained the search capabilities, and affected the search results. Although the traditional search strategy based on keyword matching has higher precision, but ignored a lot of words associated with the semantic, and limited the ability to provide the effective information.

The search model of the topic search engine works by comprehending the content of Webs so that it provides the strong professional and high correlation's search results. Therefore, study and exploration on the efficient inverted index technology and Web document comprehending technology addition to the combination of both advanced technology not only has theoretical value but also great commercial prospects.

Manuscript received December 15, 2010; revised January 5, 2011; accepted January 10, 2011.

II. INVERTED INDEX TECHNOLOGY

The inverted index is the most widely used index model at present. All words in the documents are indexed as key words in inverted index. The recording item of each word includes the document that contains the word, as well as its location in the document. Thus, when you search a word in the index, you can easily find the document which contains the word and its location in the document. For inverted index of search engine, since the number of web pages related to lexical items is dynamically changed, and so is the content of the web pages, it is more difficult to maintain the inverted index. However, inverted index has great advantage in query system. Inverted index is widely used in the system which has high demand for the response time of searching, since one may find all document information that contains the word in just one search. Many researchers have discussed the key technologies of inverted index [2].

The inverted index is an indexing mechanism for words. It can improve search speed. Each index entry in the inverted index composed of index term and its appearance, each index term has a posting list to record all information of the word appearing in the documents. This information contains the index ID, the position in the document and the index appearing frequencies, etc [3].

Since the number of documents related to every character and every word is changed dynamically, it is more difficult to make and maintain the inverted index. But the efficiency of inverted index is higher than that of forward index because one can get all documents that contain the key word in just one search. The quick response to the search is a critical property in full-text search. Since the index is run in the background, it will not affect the searching efficiency of the whole search engine.

The merge strategy of inverted index is to build index module according to the size of system memory in the running time. The index is divided into k groups such that each memory requirements are less than the maximum effective memory size that operating system provides. *k*

doi:10.4304/jcp.6.4.664-670

groups of indexes are created by the Spanning Tree Algorithm. By merging these indexes (merging the same indexes), The final inverted index file which takes the index terms as primary key is realized. The structure is shown in Figure1.

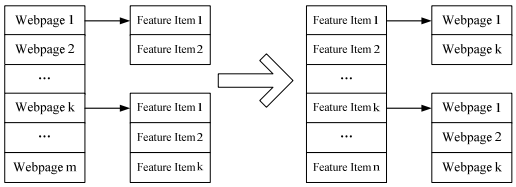


Figure 1. Diagram of establishing inverted index by the forward table.

A. Inverted File and Its Performance Model

The so-called inverted file is a kind of data structure which descripts the correspondence of elements between the index term and the documents set. Noted: Docs = {*d1, d2, ..., di, ..., dn*}, Terms = { *t1, t2, ..., tj, ..., tm*}.

The forward index is based on the document and can detect what lexical items are in the document *di*, or how many times a particular word *tj* appears in the document. But inverted index can not only detect where the word *tj* appears in the document, but also its location and occurrence number in the document. If we use PL indicates *tj*'s occurrence in the set of recording document, we call it *tj*'s inverted table.

As a data structure, the inverted file has two parts: The first part is called TERM LIST which is a index composed by different index items, the second part is called INVERTED FILE which is a documents set which contain the appeared index term. The counterpart of each term is called INVERTED LIST, also known as the RECORD LIST; it can be visited by term list. The Schematic diagram of inverted file is shown in Figure2.As shown in Figure2, the left part is the term list; the middle part is the log file.

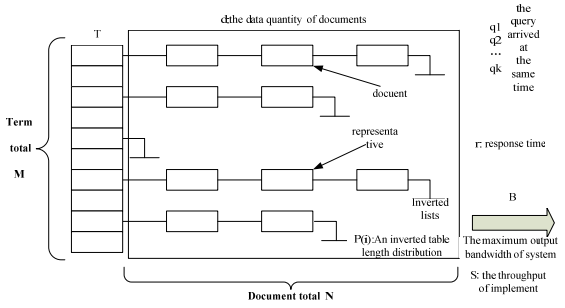


Figure 2. Schematic diagram of inverted file.

The so-called performance models is to give a relationship on the *N, M, p(i), d, B, r* and *k*, which can assess the throughput of its external behavior if the inner parameters in system are presented. A basic model of the inverted file can be presented as following.

$$D = d \times \alpha \times \frac{M \times [M^k - (M-1)^k]}{M^k} = d \times \alpha \times M \times (1 - (1 - \frac{1}{M})^k) \quad (1)$$

Where

$$d \times \alpha = \begin{cases} c \times \alpha & \text{Non - text index} \\ c \times (\alpha + \frac{T_n}{M}) & \text{Full - text index} \end{cases} \quad (2)$$

One can select the average frequency of query terms and I/O performances as a determinant of search efficiency, and assessed the relationships of system's throughput and data size by using it.

B. The Compression of Inverted Index

Compressing inverted files is conducive to the throughput capacity of searching because reading and compressing a compressed inverted index costs I/O less time than that of uncompressed inverted index. Previous researches focus more on compression ratio rather than the dynamics of inverted index. To balance the two aspects, a method which can not only support the dynamics of inverted index but also ensure compression ratio shall be applied.

Since inverted full-text index is to construct index for every non-stop word in the text, we must firstly learn the file structure of inverted index in order to analyze the compressing method. As for index term, its organization structure of posting list is:

$$(Term, docNums, (docId_1, frq_1, (pos_1, \cdots, pos_{frq_1})), (docId_2, frq_2, (pos_1, \cdots, pos_{frq_2})) \cdots)$$

Where *Term* is the index term (feature item), *docNums* indicates the number of documents that contain the word Term, $docId_i$ is the document's ID, frq_i is its frequency in the document, pos_i is the location information of the word. All the numbers above are positive integer. Each item in the posting list makes a record and the collection of posting list formed by every word makes the whole inverted index. The organization form by using inverted index can not only support Boolean Search but also Ranking Search according to the information of the word's frequency. In addition, it can also make phrase search according to location information. But additional location information will enlarge the size of inverted index, thereby, take up more storage space.

The dynamics of Web document is very important in the process of creating index, therefore we shouldn't neglect dynamic factors in regard of index compression technology. The dynamic orientation in the text can be replaced by two adjustments of textual level. That is, the change of a text is regarded as the deletion of original text or the addition of new text. Previous study only considered the dynamic synchronization adjustment of index. This paper proposed a hybrid coding scheme that can achieve high compression ratio with highest possibility as well as satisfying dynamics characters.

Position sequence records the location of a word in the text. Since the value of the inner sequence doesn't change regardless of the addition and deletion of textual level, it is static. Any compression method with high compression ratio can be used. Compared with document ID sequence and frequency sequence, position sequence occupies more index space than other storage content. Therefore, the compression of position sequence plays a decisive role in the compression ratio of the whole inverted index. To date, binary coding has the highest compression ratio. Though it takes much more time to compress and decompress, compared with the compressed I/O time, it can be ignored. For this reason, binary interpolative coding can be used for position sequence [4].

Unlike position sequence, document ID sequence (*d1,...,di,...,dn*) and word frequency sequence (*f1,...,fi,...,fn*) varies with the addition and deletion of documents. Therefore, dynamic-supported compression method must be used. Though binary interpolative coding has high compression ratio, it can not be used due to its nonsupport of dynamic updating of the sequence. Variable byte coding, γ coding and δ coding are three main compression methods that support dynamic sequence. δ coding has the highest compression ratio among the three. Therefore, δ coding is applied to both the two sequences.

Since document ID sequence (*d1,...,di,...,dn*) is an ascending sequence, usually we firstly convert it into intervening sequence (d_1,...,d_i-d_{i-1},..., d_n-d_{n-1}) and then compress, which increase the complexity of the subsequent operation of adding elements. For example, for document ID sequence（2,8,10,11,15,23,27）, its intervening sequence is（2,6,2,1,4,8,4). If we add a document 30, we must know the final value of the document ID sequence 27 so as to get the interval value 3=30-27. To get the final value 27 of ID sequence, we must uncompress the whole sequence from the beginning to the end and then get the sum. Thus, it requires additional I/O time and compressing time, especially for longer sequence. To avoid this problem, we can put the maximal value of document ID sequence into index term information in the inverted table. It's very easy to delete an item in document ID intervening sequence, just by modifying the subsequent value into the sum of the two. For the document ID sequence mentioned above, if we delete 11, we shall modify the intervening sequence to （2,6,2,5,8,4).

Frequency sequence (*f1,...,fi,...,fn*) is not an ascending sequence and there is no need to convert it into intervening sequence, so there is no such issue as in document ID sequence.

C. Inverted File Caching Mechanism

Caching technology is an important way of improving the performance and expandability of the index system and has drawn great concern in academic world recently. The effectiveness of caching technology is based on the existing local features of the cached object access sequence. The inverted file cache is in the index service node and cashes the access of inverted file data during user query implemented by query executer. A large number of studies have shown that user query word sequence has favorable locality and query executer can be expected to read these generated data by inverted query series with the same nature, which is a basic point of departure of studying inverted file cache.

If distributed architecture is adopted in indexing system, the data is organized to various index service nodes in a way of document classification and they carry out parallel implementation of user query independently and return their index results to users after summarizing through submitting them to query server. Locations of cache at all levels are shown in Figure3.

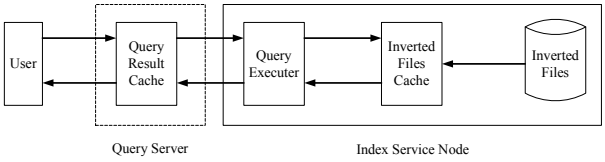


Figure 3. Cache structure of inverted index system of search engine.

The inverted file data need to be divided into two classes when the search engine invokes the query executer, one is document data, that is to query document label and document weight data related to query words in the inverted sheet; another is position data, that is to query the position of words existing in each file. During the implementation, each query word adopts descending treatment based on inverted file frequency. Firstly read the file data, and then implement Boolean calculation of the file set to get a small result set. Meanwhile weight the query correlation of each result file by the weight data of the file; read the corresponding position data again and sort the adjacent weight of the result set. Through compression technology to record the file data combined with bitmap for high-frequency words. It can control the length of file data effectively. Generally, the amount of position data is three or four times to the file data [5], and they are not necessarily to be read during query execution. Organize technologies through inverted indexing with random access can reduce the data quantity that can be read, which guarantee that caching mechanism becomes practical.

III. WEB DOCUMENT CONPREHENDING TECHNOLOGY

A. The Concept of PageRank and Its Formula

PageRank is a network link analysis algorithm proposed by Sergey Brin and Lawrence Page who are PhD students of Stanford University in 1998. The algorithm is based on the model of random surfer. It makes an assessment to the web pages, and gives a measurement of importance for each page, then applied to sort the search results [6].

Specifically, *PageRank* assumes that a surfer follows the link and browses several pages and then turn to a new random web page. After that the surfer follows the new link and browses, and then the value of a web page is determined by the access frequency of the random surfer.

The main idea of *PageRank* algorithm is: (1) If a page is referenced by many other pages, then it is probably important. (2) Although the page did not referenced by several times, but referenced by an important page, then it also probably important pages. (3) The importance of a page is divided equally and transported to the pages which it referenced to.

PageRank technology counts the importance of each page by the link structure of the entire Web; it supposes that the user can visit the entire network by the hyperlinks among web pages. *PageRank* in the Google application shows that it indeed can greatly improve the accuracy of search results by putting the *PageRank* value which is the result of web page links' analysis and comprehending into the sort of search results [7].

B. Defined PageRank Values

Assume there are $pageT_1$ ... $pageT_n$ which point to *pageA* (i.e., $pageT_1$ ... $pageT_n$ referenced *pageA*). The parameter *d*, the damping coefficient, is set between 0 and 1. *C (A)* is defined as the number of connections from *pageA*. The *pageA* 's *PageRank* value is:

$$PR(A) = (1-d) + d \times (\sum_{i=1}^{n} \frac{PR(T_i)}{C(T_i)}) \quad (3)$$

PageRank value is a probability distribution in the entire web pages' group, so the sum of all *PageRank* value of all the web pages is 1. Here *PR (A)* is the *PageRank* score of given *pageA*; *d* is a damping factor, $0<d<1$; $d*(PR(T_i)/C(T_i))$ represents that $pageT_i$ split its *d* share *PageRank* value to each external link pointing to itself in the random surfer model. Because of $pageT_i$ points to *pageA*, so *pageA* received one of $C(T_i)$*'s* value from $pageT_i$, usually set *d* to 0.85 [8].

PR (T_i) represents the page's own *PageRank* score which points to *pageA*; *C* (T_i) represents the page number of $pageT_i$*'s* links; *PR* (T_i) */ C* (T_i) represents ratio of the number of pages pointing to *pageA* and the number of pages which *pageA* points to.

The external l*inkB* of *pageA* can help *PageRank to* get scores and it is inversely proportional to outbound link numbers of *pageB*, that is to say, with the increasing outbound links of *pageB*, the score of *PageRank* of *pageA* decreases. This indicates that the *PageRank* score of a web page is a fundamental metric form of a web page for its page vote. One web page can vote for one or more outbound link, but its total vote right is certain and they are distributed to all outbound links. Assume that *PageRank* of *pageB* scores 5 and *pageB* has only one link pointing to *pageA*, then *pageA* will get all the *PageRank* scores of *pageB*. *PageB* doesn't lose anything and *pageA* gets all the *PageRank* scores of *pageB*. But if *pageB* has N links, then *pageA* will only get one equal part of N *PageRank* scores of *pageB*.

Totally, the *PageRank* value of *pageA* is determined by two scores and their proportion are *1-d* and *d* respectively. The first part of this formula can be regarded as the old score; the second part can be regarded as a partial score since it is pointed by other web pages [9].

C. Illustration of Operating Principle of PageRank

Diagrams are used to illustrate the operating principle of *PageRank*. Assume there are four web pages *pageA*, *pageB*, *pageC* and *pageD*, which interlink with each other, as shown in Figure4:

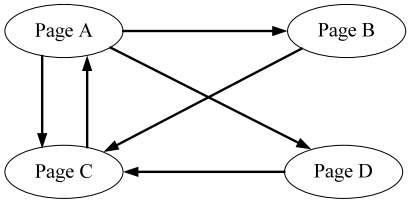


Figure 4. Schematic diagram of PageRank.

Assume the initial *PageRank* scores are all 0, and then their *PageRank* scores are all 0.15 in the first part according to the formula (3). Then the *PageRank* score of each one after interlink can be presented as follows.

1) pageA links to pageB, pageC and pageD. The initial PageRank score is 0.15, so the total PageRank score obtained by outbound link of pageA is 0.85×0.15=0.1275. The scores of pageB, pageC and pageD are all 0.0425.

2) The initial PageRank score of pageB linking to pageC and pageD is 0.15, so its only linking pageC can obtain the PageRank score: 0.85×0.15=0.1275.

3) pageC links to pageA, its PageRank score 0.1275 is transmitted to the only linked object pageA.

4) pageD links to pageC, its PageRank score 0.1275 is transmitted to pageC.

The scores of *PageA*, *PageB*, *PageC* and *PageD* are:

PR(A)=(1－0.85)+0.85PR(C)/C(C)

PR(B)=(1－0.85)+0.85PR(A)/C(A)

PR(C)=(1－0.85)+0.85(PR(A)/C(A)+PR(B)/C(B)
+PR(D)/C(D))

PR(D)=(1－0.85)+0.85PR(A)/C(A)

Now, the *PageRank* scores of all pages are as follows:

PageA:0.15+0.1275（from *pageC*）=0.2775；

PageB:0.15+0.0425（from *pageA*）=0.1925；

PageC:0.15+0.0425（from *pageA*）+0.1275（from *pageB*）+0.1275（from *pageD*）=0.4475；

PageD:0.15+0.0425（from *pageA*）=0.1925；

The *PageRank* scores of all pages after this analysis are presented in Figure5.

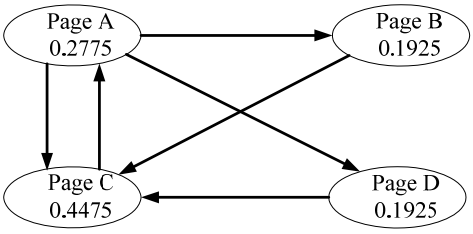


Figure 5. The PageRank scores of link analysis.

Continue such calculation until the value of each page approximates a constant value (*PageRank* is a convergent function). At last, it can be found that the *PageRank of pageC* is the highest. In addition, the quantity of external links has obviously changed the *PageRank* scores, as ATBLEI shows:

TABLE I.
COMPARISON TABLE OF EXTERNAL LINKS AND PAGERANK SCORES

	Inbound Links	Quantity	Outbound Links	Quantity	PageRank Scores
C	A/B/D	3	A	1	1.4860615
A	C	1	B/C/D	3	1.4131523
B	A	1	C	1	0.5503931
D	A	1	C	1	0.5503931

According to the above analysis, It is obvious that the *PageRank* technology obtain the importance of Web documents by simulating surfers browsing web pages along the huge page links and assigning *PR* score value to get the importance of Web pages.

Ⅳ. BUILDING INVERTED INDEX BASED ON WEB DOCUMENT COMPREHENDING

At present, the search mechanisms of all main commercial search engines are all based on inverted index. The traditional index creational patterns are word segmentation, extracting feature items and creating inverted index from forward index. Inverted files offer service to retrieval system in order to respond to the users' real-time search requests.

Inverted index of Web document comprehending is to integrate Web document comprehending technology into inverted index system, optimize the structure of inverted files, refine full-text search fineness, run with high recall ratio as well as precision ratio, and respect the query habits of users.

A. Related Inverted Index File

The development of Web document comprehending technology has made search technology transform from external feature of data information to full texts of the documents, its core technology is the full-text index Web documents in the index process. The process of Web documents is as follows, full-text word segmentation, analyzing word frequency, selecting feature item sets, creating index with feature items as index terms, and constructing vector space models. Retrieval system receives query string input by users, segments words and extracts key words, responds to users with the documents sets that contain key words obtained by retrieval of index term list and log files.

The search method of exact matching the keywords in the traditional inverted file system may have the following undesirable results: polysemy will lead to that many redundant content are listed in the search results, while the situation that one meaning can be expressed by multi words leads to the result that a lot of really needed content missed. For example, the indexing system stored an inverted index file in the way as shown in Figure6.The user enters a query word "PC", the search system try to find "PC" in the word list of the inverted index file terms after it received user's input and return *Document 1* to *Document n* to the user as search results by reading its inverted list. Obviously *Document 3* is not in the search results set which include "computer" that has a high similarity with the word "PC".

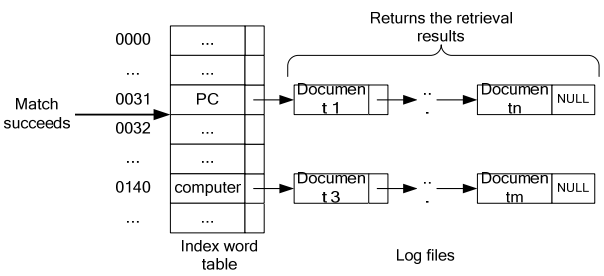


Figure6. The example of inverted index file.

According to the data organization form of index terms analyzed above, *(Term, docNums, ($docId_1$, frq_1, (pos_1,...,pos_{frq1})),($docId_2$, frq_2, (pos_1, ..., pos_{frq2}))...)*, the improved inverted index data structure can be get. In Figure7, the content stored in the index word is it self's content, the number of corresponding documents is *docNums*, and the max of the documents is *maxID*. The maximum document's ID is for adding documents in the inverted sequence when it is compressed. Each node in the document list should record the document number *docID*, the word frequency *frq*, and the position list that the word appears *($pos1$, …, pos_{frq})*.

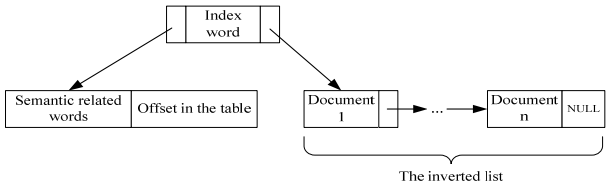


Figure 7. The improved inverted index data structure.

The position list should also be organized by list structure; the specific data structure is shown in Figure8.

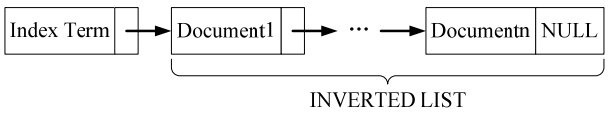


Figure 8. Data structure diagram of inverted files.

When the contents of Web document comprehending technology is applied to the inverted index file, the data structure is accordingly improved. The index term is still in the sequential storage structure, but a pointer domain should be added for each word. Therefore, the index word has two pointer fields, one points to the inverted list, while the other additional pointer fields points to the related index word term obtained in the comprehending. To search easily, the additional related index word term has two data fields: the semantically related words and the offset of the semantically related words in the inverted list. The data structure of related inverted file is shown in Figure9.

Index word(Term)	Document number(docNums)	Maximum number(maxID)	Pointer domain (Point to the inverted list)

(a) The data structure of index word item

Document numbers (docID)	word frequency(frq)	Pointer domain (Pointing to appear position list)

(b) The data structure of each index in the inverted list

Figure 9. Schematic diagram of the inverted file data structure.

The matching process of the query in the related inverted file is presented as an example above. The position of index word "PC" in the inverted file is "0031", and the position of word "computer" in the inverted file is 0140. It is known that the word "PC" and the word "computer" have high correlation by Web document comprehending technology and computing the *PageRank* value. Therefore, the Terms "computer" is added to the property of Semantic word of the word "PC" and "0140" is recorded to the offset in its list. The User enters a query

word "PC", the search system search for the word “PC” in the inverted file and match successful at the position “0031”. First, the search system read all documents which includes the query word "PC" in the inverted list, then read the contents of semantically related word "computer", the offset in list is “0140”, and finally we can get the word term information and the inverted word list item by directly visiting the storage unit which marked “0140” in memory. The process is shown in Figure10.

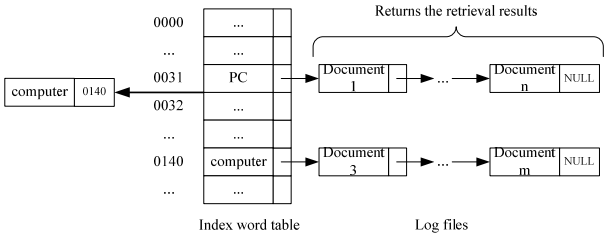


Figure10. The related inverted index file.

B. Experimental Results and Analysis

The retrieval system based on traditional inverted files realizes related search functions through second retrieval technology. That is, extracting query key words from the query strings input by users, then exactly matching the key words with the index term list of first retrieval. Finally, the related terms of key words are obtained according to the analysis results of word frequency. The second retrieval is done to search the related terms in inverted index files. The related web page result sets are finally got to meet the input query strings by users.

The analysis process of related terms of related inverted files is done in backstage, and the related terms are stored in the inverted file term list, which reduce the response time for the search request of users. Retrieval system can complete searching key words and their related words through retrieval only once.

Experimentally selected the Web documents set which include topic "60th anniversary”. Using the word "National Day", "parade" and other query words to search the traditional inverted file and the related inverted file, count the spent time to compare the efficiency of two kinds of indexes invited file. The experimental results are shown in TABLE Ⅱ and histogram of search time cost is shown in Figure11.

TABLE II.
THE TWO INVERTED FILE RETRIEVAL TIME BY COMPARISON WITH (UNIT: SECONDS)

Document total / Query words	Traditional inverted file			Relevant inverted file		
	200	*400*	*600*	*200*	*400*	*600*
National Day	4.98	16.81	37.46	3.91	10.46	25.00
The flag-raising	3.84	8.78	19.35	3.07	7.13	14.11
party	3.40	8.53	17.38	2.32	6.84	12.59
parade	5.38	19.65	40.63	4.16	15.86	30.22

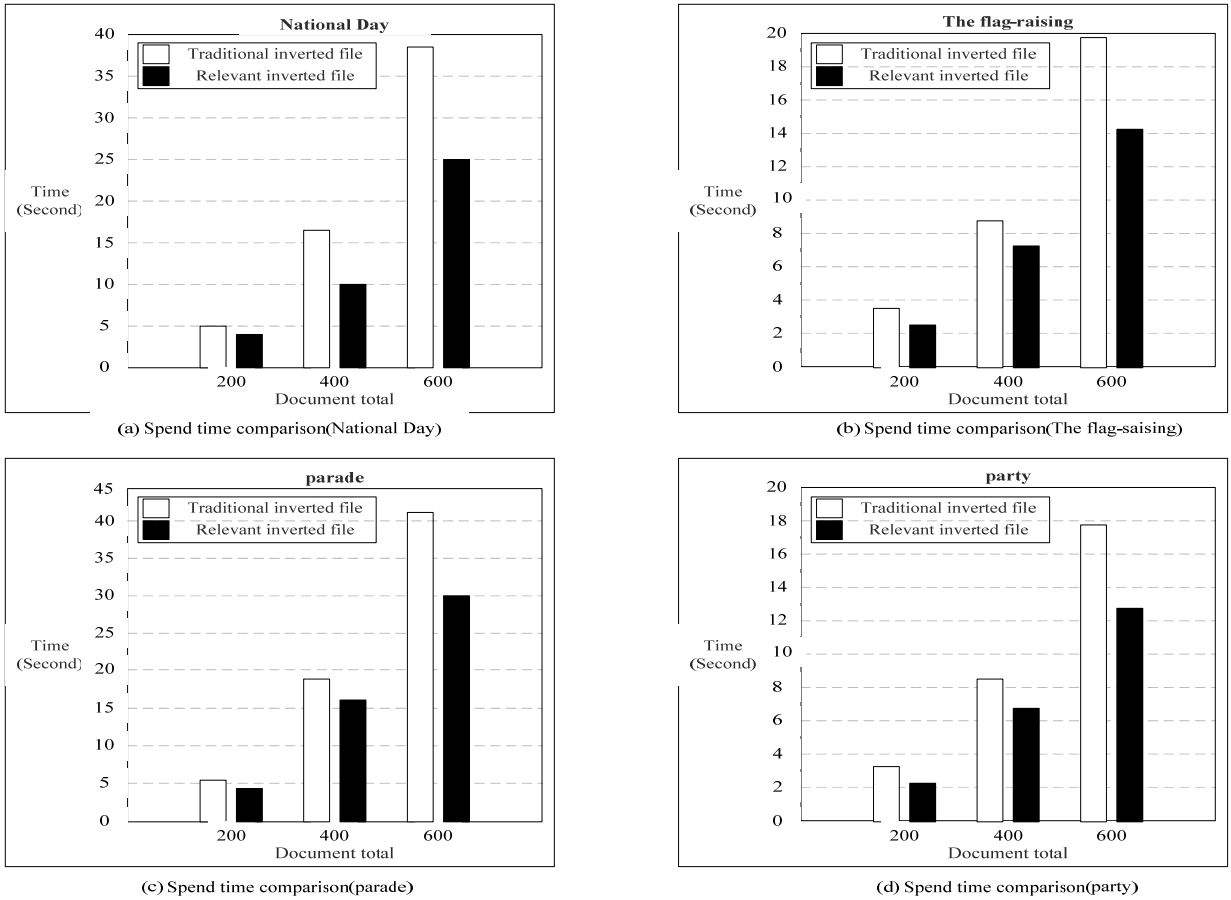


Figure 11. The comparison of two kinds of inverted files.

It can be seen from Figure11, the cost time increases with the count of documents. For the same query words to search in two kind index file at the same scale, the related inverted file spent less time than the traditional inverted file. Whether using the traditional inverted file or the related inverted file the overall work steps of search engine does not change, so the comprehensive of search remains unchanged. The web pages set associated with the user's query must be got by search in the traditional inverted file twice. While in the related inverted file, one can locate the query word by search the index word term only once to find the offset recorded in the index term list, and the time complexity of read the index term list is O (1). Therefore, the search engines with the related inverted file can raise the efficiency of search while the comprehensive keeps unchanged.

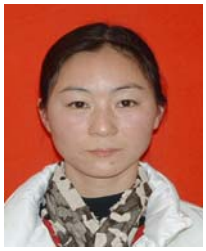
V. CONCLUSION

In this paper, the *related inverted files* is offered and analyzed which is improved according to traditional inverted files by adding attribute of semantic related words in traditional inverted index files, introducing the thought of comprehending Web documents into inverted index, and determining semantic correlation of index terms by calculating *PageRank* value, constructing attribute of *semantic related words* from lexical items of high correlation. The *related inverted files* ensure that the retrieval system has high efficiency. For example, it can offer users more documents related to users' interests and cost less time than traditional inverted files with the same quarry conditions.

REFERENCES

[1] Junxi Liu, Yu Sheng. Vertical and general search engines and case analysis of the differences [J], Modern Information, 2009, (3), 143 ~ 145.
[2] Nieholas, Lester,Justin, Zobel,Hugh, Williams. Effieient online index maintenance for contiguous inverted lists [J]. Information Proeessing and Management 2006, 42(4):916 ~ 933.
[3] Chiyoung Seoa, Sang-Won Leeb, Hyoung-Joo Kima. An efficient inverted index technique for XML documents using RDBMS [J]. Information and Software Technology, 2003,45(1):11 ~ 22.
[4] Xingyu Liu, Research on Full-text Retrieval Technology Based on Inverted Index [D]. Wuhan: Huazhong University of Science and Technology, 2004, 5: 20 ~ 21.
[5] Navarro G,et al. Adding compression to block addressing inverted indexes[J]. Information Retrieval, 2000, 3(1):49 ~ 77.
[6] L,Brin a1.The pagerank citation ranking: Bringing order to the web [R]. Stanford Digital Libraries SIDL-WP, 1999.
[7] Havelieala T H. Topic-sensitive PageRank [C].Proceedings of the 1lth International World Wide Web Conference, Hawaii, 2002: 517 ~ 526.
[8] Kleinberg J. Authoritative sources in a hyperlinked environment [C]. Proceedings of the Ninth Annual ACMSIAM Symposium on Discrete Algorithms, San Francisco, California, 1998: 668 ~ 677.
[9] Xing Wenpu, Ghorbani A. Weighted PageRank Algorithm [C]. Communication Networks and Services Research, Proceedings of Second Annual Conference, 2004: 305 ~ 3 14.

Shaojun Zhong was born in Guzhou, China, in Oct.1979.He is now a Lecturer in Faculty of Science, Jiangxi University of Science and Technology, China. His research interests include data mining, network technology, and Intelligence computation.



Min Shang was born in Yuxi, China, in Nor.1979.She is a lecturer in Resources and Environmental Sciences of Yuxi Normal University. She is the Cartography and GIS Master, hers research interests include GIS data quality, DEM application.

Zhijuan Deng, female, native place is Ganzhou, Jiangxi Province, born in Nov.1979, working in Faculty of Science, Jiangxi University of Science and Technology as an instructor. Research directions: Web information mining, software project management.