# A Novel Information Search Approach for Languages Without Word Delimiters

# Lianlong Wu

School of Electronic Engineering and Computer Science, Peking University, Beijing, China

#### Summary

In many languages there are no word delimiters among the text. It is very difficult to index articles in those languages. For example, Chinese information search engines always encounter a difficulty in segmentation of Chinese words from an article. In this paper, a suffix tree based searching approach is proposed to avoid the difficulty in segmentation of Chinese words. The suffix tree algorithms are studied and a set of optimal algorithms for index build are proposed. Based on the algorithms, a prototype of Chinese information search system is developed and applied to the Chinese Web Test collection with 100 GB Web pages (CWT-100g). The experimental results show that the system is capable of searching Chinese information without segmentation of Chinese words and the speed of index build is reduced to the theoretical limitation. part of summary.

#### Key words:

Search engine, segmentation of Chinese words, suffix tree, information system.

# 1. Introduction

An index mechanism is at the core of an information search system. Inverted list is a technique that is widely used as an index mechanism [1]. However, there are constraints on the use of such word-based lists in Chinese information systems. Since there are no delimiters to separate Chinese words, it is very difficult to segment words from a text. If one wants to apply inverted list, Chinese word segmentation has to be done at first. Firstly, it takes time to segment Chinese words. This affects the efficiency of index build. Secondly, the quality of word segmentation severely affects the quality of index.

In this paper, a full-text index mechanism based on suffix trees[2], [3] is applied to develop a new system, so the segmentation of Chinese words is not necessary. Based on properties of suffix trees, the search for phrases, sentences, and even more complicated searches, such as a whole paragraph search, are amazingly feasible. Moreover, accurate results can be achieved.

As index build is very time consuming, a set of optimal algorithms are studied in Section 3. The functionalities and interface of the new search system is described in Section 4. The experimental results are provided in Section 5. A conclusion is given in Section 6.

# 2. Problem of Inverted Files

Inverted files have been widely used as an index technique [1] in information search systems. Inverted files are a text index composed of a vocabulary and a list of occurrences. Inverted indices assume that the text can be seen as a sequence of words. In English, there are spaces to delimit words, but in Chinese text there is no space between words to separate them. Therefore, before an index is built, text information has to be segmented into Chinese words first. The segmentation of Chinese words not only takes longer time than indexing itself, but also may make mistakes. For example, the Chinese phrase 'developing country' can be segmented in two sets of words shown in Figure 1. The

segmented in two sets of words shown in Figure 1. The segmentation at the right column is correct. The segmentation in the left column is wrong. The meaning is changed to 'to develop China'.

Chinese vocabulary 1.发展 2.国家 3.中国 The Chinese phrase '发展中国家' can be segmented

Segment	发展 □ 1	中国家 2	发展 1	中国 3	家	
means	developin	g country	to develop China			
	Cor	rect	Wrong			

Figure 1. Difficulty of Chinese word segmentation.

# 3. Optimal Algorithms

#### 3.1 Suffix Trees and Suffix Arrays

A suffix tree[2] is a data structure built over all the suffixes of a text. Suffix is a string that goes from a position to the end of the text. Each suffix is thus uniquely identified by the position. For example, *Suffix* (8) of "mississippi" is "ippi" as shown in Figure 2 (this example is from [4]).



Figure 2. Suffix Tree of string "mississippi"

The data structure in a suffix tree can be easily applied to solve many problems on string operations such as those longest repeated substrings (2), (3), and (4). Any long substrings can easily be searched in the text. In order to take these advantages, a full-text index technique based on suffix trees is applied to develop the new search system. The system does not need to take time to do the segmentation of words. Therefore, a full-text index based on suffix trees can be built much faster than the index based on inverted files.

As suffix arrays provide the same function as suffix trees and occupy much less space, suffix arrays are proposed to implement the search system. A suffix array is simply an array containing all the pointers to the text suffixes listed in lexicographical order. For example, the suffix tree of string "mississippi" can be represented as a suffix array shown in Figure 3.

The Unicode is applied to represent Chinese characters. A content extraction module is developed to extract the contents from the Web pages. The text contents are stored in a Content File in the Unicode. The suffix array is stored as integers in the Index Array. The system produces files with file types as follows:

type TIdx= Integer; // 32Bits Integer

*TCot* = *WideChar;* // *Unicode Character* 

var

CotFile : file of TCot; // Content File IdxFile : file of TIdx; // Index File

Suffixes in		Suffix Array			
lexicographical		Index	Value		
order:		1	11		
11: i		2	8		
8: ippi		3	5		
5: issippi		4	2		
2: ississippi		5	1		
1: mississippi		6	10		
10: pi		7	9		
9: ppi		/	,		
7: sippi		8	7		
4: sissippi		9	4		
6: ssippi		10	6		
3: ssissippi		11	3		

Figure 3. Suffix Array of string "mississippi"

## 3.2 Suffix Array Building

Array  $Rank_k$  is used for building the suffix array.  $Rank_k[i]$  is the rank number of the Suffix[i] according to the first k characters of each suffix.  $Index_k$  is the suffix array corresponding to the  $Rank_k$ .

An optimal algorithm for building a suffix array is proposed as follows:

*k* ← 1;

repeat

*Quick* Sort (Index<sub>k</sub>); //according to first k characters of each suffix

Count the Rank<sub>2k</sub> Array;

 $k \leftarrow 2k$ 

until k≥n;

Where *n* is the length of the text, the time complexity for string comparisons in the Quick Sort is O (1), so the time complexity of Quick Sort is  $O(n \cdot \log n)$ . The variable *k* is increasing exponential; the time complexity of loop is log *n*. The total time complexity is  $O(n \cdot \log^2 n)$ .

For long query strings, it is not necessary to do a complete sort. The loop can be stopped when  $k \ge m$ , where *m* is the maximum length of query strings. This length is enough for the binary search. The time complexity is reduced to O ( $n \cdot \log n \cdot \log m$ ), where *m* is a small constant (m=16 or 32, in most cases), so the time complexity is O( $n \cdot \log n$ ).

61

The Quick Sort can be replaced by Bucket Sort and Radix Sort. A optimal algorithm is proposed as follows:

 $k \leftarrow 1;$ 

Bucket Sort; //according to the first character only

Count Rank<sub>1</sub>Array;

repeat

*Radix Sort (Index<sub>k</sub>); //according to first k characters of each suffix* 

Count the Rank<sub>2k</sub> Array;

 $k \leftarrow k \cdot 2;$ 

until  $k \ge n$ ;

Both the Bucket Sort and the Radix Sort are linear, so the total time complexity is reduced to O(n) that is theoretic lowest bound.

Additionally, an approach for rolling array is applied to recycle array memory. Scanning a static array is applied to replace dynamic queue so that the spaces for pointers are saved. Therefore, a high stable system is obtained. Because the Web pages are scanned only once to perform the extraction, the time complexity is O (n).

## 4. Development of Search System

A prototype of search system is developed in three separate modules and runs on single PC. First module is for extraction of content from Web pages. The CotFile is created. Second module is for index build. The optimal algorithms are applied in this module to build a suffix tree based index. A efficient search algorithm is applied to third module to search the information on the suffix tree based index. The index building interface is shown in Figure 4. The content files created by first module are listed in the left box. Select the content files and click bottom 'Suffix'. A suffix tree can be generated. Click bottom 'Index'. An index based on suffix tree is built. Click bottom 'Auto'. An index is automatically built for the selected content. Based on this interface, the information of the index can be shown. For example, a list of positions of URL is shown in the right box.

Figure 5. Shows searching interface of the prototype system. Multiple indexes are listed in the left box. The key words, phrase, or sentence can be input in the box beside the bottom 'search'. Results are shown in the middle boxes. The right box are listed the test collection provided in The CWT-100g[5].

🞽 Index Building		– – ×
List G:\Content\ 9_041109012229.c	Load Suffix Index Auto Record	http://www.whxx.net/whxx/zyk/l 🔳 Show 🗌 Loop
No. FileName Size (k) 📥	0% Stop	No. URL Staart End 🔥
2057 9_041109012025.cot * 6600	中国科普博览 冰雪馆 贫而 冰雪馆 热闹的冰 🔍	1 http://www.whxx.ne 1 449
2058 9_041109012108.cot * 3220	天雪地一一北极 顽强的北极生命 鸟类王国 绒 🤗	2 http://www.whxx.ne 450 1163
2059 9_041109012123.cot * 2162	鸭 绒鸭 鸭类通常被认为是一种低能动物,除了一个游泳之外,其生存其能似乎乎乎 然而,也许	3 http://www.whxx.ne 1164 2009
2060 9_041109012133.cot * 1981	是环境所迫的缘故吧,北极的绒鸭看上去智商似	4 http://www.whxx.ne 2010 2930
2061 9_041109012144.cot * 2204		5 http://www.whxx.ne 2931 3303
2062 9_041109012155.cot * 2497	"vds-sonne"). http:	6 http://www.whxx.ne 3304 3484
2063 9_041109012208.cot * 2419	┃ "忉相"号超音速客机 "忉相"号超音速	7 http://www.whxx.ne 3485 3957
2064 9_041109012219.cot * 2278	"多址接入技术"蜂窝系统中是以信道来	8 http://www.whxx.ne 3958 4804
2065 9_041109012229.cot *1172	"蜂窝"用户的未来(二) 为此,我们把	9 http://www.whxx.ne 4805 5746
2066 9_041109012233.cot * 3773	■ 「「「「「「「」」」 「「「」」」 「「「」」」 「「」」 「「」」 「「	10 http://www.whxx.ne 5747 5796
2067 9_041109012246.cot * 2175	多址接入技术 Disk Memory	11 http://www.whxx.ne 5797 5847
2068 9_041109012257.cot * 2195	Disk	12 http://www.whxx.ne 5848 5892
2069 9_041109012308.cot * 2562	Pos: 358417 Order: 306080	13 http://www.whxx.ne 5893 6512
2070 9_041109012319.cot * 3187	Por: 358417	14 http://www.whxx.ne 6513 7091
2071 9_041109012333.cot * 2567	多址接入技术"蜂窝系统中是以信道来区分	15 http://www.whxx.ne 7092 7961
2072 9_041109012345.cot * 2148	Url: http://www.whxx.net/whxx/zyk/dianxin/	16 http://www.whxx.ne 7962 8389
2073 9_041109012354.cot * 2562	Wireless/Wrizif.html	17 http://www.whxx.ne 8390 8431
2074 9_041109012405.cot * 2557		18 http://www.whxx.ne 8432 8484
2075 9_041109012415.cot * 3356	多址接入技术"在移动通信中,我们常常听	19 http://www.whxx.ne 8485 8988
2076 9_041109012429.cot * 2618		20 http://www.whxx.ne 8989 9456
2077 9_041109012441.cot * 2725	Get Text 358332 多址接入技术"任移	21 http://www.whxx.ne 9457 9829
2078 9_041109012451.cot * 1802	Get Index 306080 358417	22 http://www.whxx.ne 9830 10265 😒
Time: 2.68389s	Time: 0.08890s Speed: 13185k/s	

Figure 4. Interface of index build in the prototype system

<sup>™</sup> Search – □ ×									
List	G:\Content\ 4_04110	9094702.co	t	《商业周刊》	Search	Show	Go	Loop NPHP1-285.txt Ge	et
No.	FileName Size (k)	Index 🧧	^	0%	Load	🔳 Show	Number	Keyword	^
459	3_041109103935.cc 2251	Yes		_			NP1	宁夏社会保障人员联系	
460	3_041109103942.cc 2868	Yes		《商业周刊》评选) 贝尔纳德	•j•埃贝	Q斯 🚔	NP2	香港电台全球华语歌曲排行机	
461	3_041109103949.cd 2264	Yes		http://www.an-gao.com/book/z	:z/jj/q/		NP3	国际服装博览会组委会地址	
462	3_041109103955.cc 291	Yes		40/4379 218.75.52.36		=	NP4	租车车型报价	
463	3_041109103956.cc 2663	Yes			主众北西	<del>*</del> *	NP5	对外经济贸易社会团体管理刻	
464	3_041109104005.cd 2563	Yes		http://www.an-gao.com/book/z	±1£1£⊠ z/ii/a/	秋 📃	NP6	浪潮英信np50	
465	3_041109104013.cc 1653	Yes		qiaosi/who/007.htm			NP7	《思教课实践教学》大纲	
466	66 3_041109104019.cc 1486 Yes			202/20022 218.75.52.36		NP8	前沿培训网报名须知		
467	467 3_041109104025.cc 961 Yes			《商业周刊》评选) 克雷格・巴雷特 英特 http://www.an-gao.com/book/zz/jj/q/		NP9	搜狐近期招商项目		
468	468 4_041109094524.cd 1352 Yes					NP10	长春大学党政领导信箱		
469	4_041109094531.cc 3349	Yes					NP11	无机材料学报	
470	4_041109094544.cc 1803	Yes		NP291 93 http://www.an-gao.com giaosi/who/026 htm	1/book/zz	z/  /q/ 🔼	NP12	华中科技大学硕士招生	
471	71 4_041109094552.cc 2177 Yes			(Jassi wind) z 255 http://www.an-gao.com/book/zz/jj/q /qiaosi/who/007.htm NP29 3 525 http://www.an-gao.com/book/zz/jj/q /qiaosi/who/026.htm NP29 4 2796 http://www.an-gao.com/book/zz/jj/		NP13	徐州市城市绿化管理办法		
472	72 4_041109094600.cc 1425 Yes					NP14	中国电工技术协会常务理事会		
473	3 4_041109094607.cc 1980 Yes					NP15	免费自助法律咨询文件		
474	74 4_041109094616.cc 2572 Yes					NP16	金融采购网供应商注册		
475	4_041109094625.cd 2327	Yes		s/shixiaochuan/fanbai/008.htm		"	NP17	赞助计划《企业信息登记表》	
476	4_041109094633.cc 2479	Yes		NP29 5 9080 http://www.an-gao.c	:om/book	/zz/jj/	NP18	国际焊接工程师可选途径培训	
477	_041109094642.cc 1731 Yes			q/qiaosi/who/UU/.htm		NP19	各地彩票游戏规则		
478	8 4_041109094651.cc 2082 Yes				NP20	神舟软件升级			
479	4_041109094702.cc 11848	Yes				~	NP21	第九届中国专利技术博览会	-
400	4 041109094729 - 9521	Yoo	<b>*</b>	L			MD00	北方子山中学硕士切开	
Time	Time: 4.57161s 5 Time: 0.05737s Speed: 206493k/s								

Figure 5. Searching interface in the prototype system



Figure 6. Indexing speed vs. hard disk speed



Figure 7. Indexing time shows linear increase with data size

## **5. Experimental Results**

The new system was applied to the benchmark data set --Chinese Web Test collection with 100 GB Web pages (CWT-100g) proposed by Network Group, Peking University [5]. The CWT-100g is a collection like TREC. It is composed of three parts: the documents, the queries, and relevance judgments. The CWT100g is designated as the test collection of SEWM-2004 Chinese Web Track.

Based on the benchmark data set, 285 items of home/named page finding tasks were done. Lists of URL were obtained corresponding to different tasks respectively. The total output for the 285 items of tasks is 4395 lines of URL. The running speed of the system was tested. The content was extracted from the 100GB Web pages with a speed of 3MB/s. Index build worked at a speed of 1MB/s. This is better than the speed of indexing based on inverted lists. As the fastest word segmentation speed is only 0.6MB/s, the indexing speed for inverted list based system is less than 0.6MB/s. The speed of indexing is highly dependent on hard disk read/write speed. The test results are shown in Figure 6. It can be seen that indexing speed is very close to hard disk speed. If a high-speed hard disk is applied, the higher indexing speed will be obtained. The relationship between indexing time and data size is shown in Figure 7. It can be seen that the indexing time shows linear increase with data size. For some other index mechanism, indexing time usually increases exponentially with data size. Based on this relationship, the indexing time for a huge number of Web pages can be predicted easily.

# 6. Conclusion

In this paper, a suffix tree was applied to index Chinese information in the information search system. As a result, the difficulties of segmentation of Chinese words can be avoided, and the system is independent of vocabulary library. The information can be searched not only by key words but also by any long substrings. The system supports searches with phrases, sentences, and even a paragraph. A set of improved algorithms was applied to the system so that the system can build index and search information fast. The index module was developed based on Unicode and is applicable to information in various languages, for example, Korea and Japanese languages. Based on this prototype system, a lot of further studies can be done. For example, if gene data set is input to the prototype system, frequency of gene sequence can be calculated by using the system.

## Acknowledgments

I would like to thank the Computer Networks and Distributed Systems Laboratory in Peking University for providing the Chinese Web Test collection with 100 GB Web pages (CWT-100g).

## References

- [1] R. Baeza-Yates and B. Ribeiro-Neto, "Modern Information Retrieval", Essex: ACM Press, 1999.
- [2] E. M. McCreight, "A Space-economical Suffix Tree Construction Algorithm", Journal of the ACM, 23(2): pp. 262--272, 1976.
- [3] Ukkonen E, "On-line Construction of Suffix Trees", Algorithmica, 14(3): pp. 249-260, 1995.
- [4] http://www.csse.monash.edu.au/~lloyd/tildeAlgDS/Tr ee/Suffix/
- [5] http://www.cwirf.org/en\_index.html



Lianlong Wu is a student in the School of Electronic Engineering and Computer Science, Peking University, China. He is a student member of ACM. His research interesting includes information retrieval, P2P network, dynamic programming and network flow algorithm.