An Efficient Dictionary Mechanism Based on Double-Byte

Lei Yang[1], Jian-Yun Shang[1], and Yan-Ping Zhao[2]

[1] Dept. of Computer Science, Beijing Institute of Technology
[2] School of Management and Economics, Beijing Institute of Technology
Beijing 100081, P.R. China
jeffy2008@gmail.com, shangjia@bit.edu.cn, zhaoyp@bit.edu.cn

Abstract. Dictionary is an efficient management of large sets of distinct strings in memory. It has significant influence on Natural Language Process, Information Retrieval and other areas. In this paper, we propose an efficient dictionary mechanism, which is suitable for Double-Byte coding languages. Compared with other five popular dictionary mechanisms, this mechanism performs the best of all. It improves the search performance greatly and reduces the complexity of the construction and maintenance of the dictionary. It can be well applied in large-scale and real-time processing systems. Since Unicode is a typical double-byte code which can represents all kinds of characters in the world, this dictionary will be applicable for multi-language dictionaries.

Keywords: Dictionary, Double-Byte, Information Retrieve, multi-language.

1 Introduction

Dictionaries play an important part in improving efficiency of information retrieval, text filtration, semantic analysis, Chinese word segmentation and other areas. The pioneers have done a lot of work in studying dictionary mechanism and proposed many efficient dictionary mechanisms[1]-[7]. Summarizing the characteristics of the previous mechanisms, we design an efficient dictionary mechanism, the Double-Byte, with experiments to show the efficiency of our algorithm. The time complexity of the algorithm is O(n) and the space complexity of that is O(C^n), n being the length of a word and C being a constant. This mechanism is suitable for various double-byte languages and mixed multi-languages without extra modifications. Therefore, it has broad prospects in future.

2 Related Work

There has been a lot of work such as Binary-Seek-by-Character[3], TRIE indexing tree[1][4], Binary-Seek-by-Word mechanism[2][3] , Double-Array TRIE[4] and Double Coding[5][6]. Binary-Seek-by-Word consists of a binary search after the location of the first character; TRIE indexing tree comprises multiple chains to save dictionary and a

D.H.-L. Goh et al. (Eds.): ICADL 2007, LNCS 4822, pp. 420–423, 2007.

Hash table for location; Binary-Seek-by-Character combines the previous two mechanisms; Double array TRIE introduces the DFA(Definite Finite Automata), and Double Coding exploits the serial code of Chinese words to obtain higher efficiency.

3 Double-Byte Based Dictionary Mechanism

3.1 Motivation

There are 6768 popular characters in Chinese and are coded using GB2312[8] which is a double-byte code and the same as Unicode. If we treat each character as a node in tree, then each node may connect to at least 6768 nodes(out-node). This will waste a lot of space. But we find that each double-byte character can be split into two bytes and the range of each byte in Chinese Area Code is about $(\sqrt[2]{6768}) = 82$. If we divide each character into two bytes and use two nodes to represent one character in Chinese tree, we can cut down the number of out-nodes from 6768 to 82 and this will make Chinese tree similar to English tree.

3.2 Dictionary Mechanism Based on Double-Byte

With a group of Chinese words, we define our dictionary as follows:

(1) We split each character of these words into two bytes.
(2) Construct a tree with each byte being a node in the tree.
(3) Define a hash function for each node in the tree for state transition.

a) The range of each byte of a character is from 0xa1 to 0xfe. We change the range into 0-94 by subtracting 0xa1.

b) The index of the Hash function is the decimal value of each byte subtracting 0xa1.

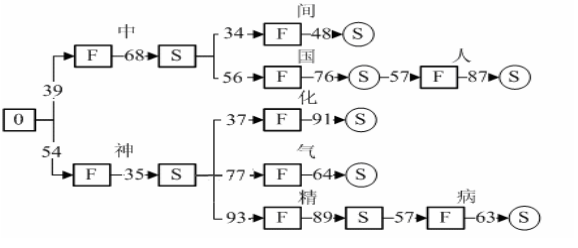

Fig. 1. Double-Byte based dictionary tree

In figure 1, node F represents the first byte of a character and S represents the second byte and (S) represents the final state.

(4) The last node of each branch is the final state. A node in middle of a branch represents any word is set from the first to the second state.

4 Experiments and Discussions

To evaluate the performance, we conduct experiments on comparing the mechanisms mentioned previously. Experiments are run in the same environment (Windows XP, Intel(R) Pentium(R) 4 CPU 1.80GHz, 512MB memory).

4.1 Experiments Data

Our data comes from half-year's corpus of People's Daily, containing 49500 Chinese words that include single-character, double-character, three-character and four-character words with a frequency affixed to each word. The total frequency of 2595 single-character words is 2062393 by adding up each single-character word's frequency, the total frequency of 31838 double-character words is 2860757, the total frequency of 7858 three-character words is 165044 and 7209 four-character words have the total frequency of 62829. Totally, there are 5151023 words in the corpus.

4.2 Dynamic Performance

We use these algorithms to search all 5151023 words in the half-year's corpus and count the number of different operations defined in Table 1, then divide the number of each operation by 5151023 to get the average value. Table 1 gives the results.

Table 1. The Dynamic Performance (by average number of operations per word)

NO.	Numerical Computation	String-Length Computation	Reading Array	String Comparison
(1) Double-Byte	23.18	0	10.6	0
(2) Binary-Seek-by-Character	38.244	0	8.268	4.268
(3) Binary-Seek-by-Word	46.24	0	14.48	8.232
(4) Double-Array TRIE	30.44	1	1.857	0

From Table 1, we can see that algorithm (1) performs the best on numerical computation which is the most important factor. Actually, most of algorithm (1)'s numerical operation is assigning but not computation. In only one case algorithm (4) is better than algorithm (1) on reading array but worse in other aspects. Algorithm (2) and (3) increases the complexity because of the extra string comparison operation.

4.3 Search Performance

We compare the search performance of the new algorithm with other three typical algorithms. Table 2 below shows the results of the time consumption of each algorithm by searching 5151023 words (average time per word in ms).

Table 2. Time Consumption of the Four Algorithms

Algorithm	Time(s)
Double-Byte	0.150
Binary-Seek-by-Characters	0.688
Binary-Seek-by-Word	1.282
Double-Array TRIE	0.484

We can see that algorithm Double-Byte performs the best of all. It improves search performance by nearly 10 times which has gone far beyond the other three algorithms.

5 Conclusion

This paper proposes an efficient dictionary mechanism whose performance and efficiency meet the requirement of large-scale and massive processing systems with highly real-time requirements. Furthermore, this mechanism is suitable to other double-byte coding languages, and wide application aspects.

Acknowledgments. This research is supported by the National Science Foundation of China, the project code: 70471064, and the National Innovation Base in Philosophy and Social Science, the project of National Defense Science and Technology, of the second phase of "985 Project", code 107008200400024.

References

1. Aoe, J.: An Efficient Digital Search Algorithm by Using a Double-Array Structure. IEEE Transactions on Software Engineering (9) (1989)
2. Li, X., Yang, W., Chen, G.: PATRICIA-tree based Dictionary Mechanism for Chinese Word Segmentation. Journal of Chinese Information Processing (2001.03)
3. Sun, M., Zuo, Z., Huang, C.: An Experimental Study on Dictionary Mechanism for Chinese Word Segmentation. Journal of Chinese Information Processing 14(1) (2000)
4. Karoonboonyanan, T.: An Implementation of Double-Array TRIE, http://linux.thai.net/~thep/datrie/datrie.html
5. Li, J., Zhou, Q., Chen, Z.-s.: A Study on Fast Algorithm for Chinese Dictionary Lookup. Journal of Chinese Information Processing 20(5), 31–39 (2003.4)
6. Li, J., Zhou, Q. , Chen, Z.-s.: A Study on Rapid Algorithm for Chinese Dictionary Query. In: Proceedings of Large-Scale Information Retrieval and Content Security. Beijing, 9, pp. 380–390 (2005)
7. Morrison, D.: PATRICIA2Pratrical Algorithm to Retrieve Information Coded in Alphanumeric. JACM (15) (1968)
8. GB 2312-1980. Code of chinese graphic character set for information interchange, Primary set, http://www.csres.com/detail/1417.html