FULL-TEXT SEARCH FOR THAI INFORMATION RETRIEVAL SYSTEMS

THANARUK THEERAMUNKONG* WIRAT CHINNAN*
THANASAN TANHERMHONG* VIRACH SORNLERTLAMVANICH**

* Information Technology Program, Sirindhorn Institute of Technology, Thammasat University Rangsit Campus
** Software and Language Engineering Laboratory, National Electronics and Computer Technology Center (NECTEC)

ABSTRACT

While there have been a lot of efficient full-text search algorithms developed for English documents, these algorithms can be directly used for other languages, e.g. Chinese, Japanese, Thai and so on. However, due to idiosyncrasies of each individual language, directly applying such algorithms may not be suitable for the language considered. This paper proposes a simplification of Boyer-Moore algorithm, called BMT, in order to reduce computation and makes it appropriate for Thai full-text. To investigate the efficiency, the comparison of BMT with other search algorithms is evaluated. Moreover, we applied syllable-like segmentation, called Thai character clusters (TCCs), to improve searching efficiency in Thai documents by grouping Thai characters into inseparable units. The TCC is based on the spell features of Thai language. Comparing with traditional full-text searching methods, this approach can improve not only searching time and memory consumption but also searching accuracy. The experimental results evidence that searching methods using TCC outperform the traditional methods in full-text search algorithm.

Keywords: Full-text Search, Boyer-Moore algorithm, BMT, TCC

INTRODUCTION

As the number of Thai documents grew, searching algorithms have become one of the most essential tools for managing information of Thai documents. Most conventional information retrieval systems (Frakes and Baeza-Yates eds 1992) do exact match, in both non-indexing and indexing approach. The non-indexing approach finds occurrences of a specified text string within a large body of the text. For the indexing approach, it searches with an index file to improve computation time for searching huge documents. The advantages of non-indexing approach are need no additional storage to keep indices, called *an index file*, and suitable for one-time search. Then this paper discussed the non-indexing approach as potential points and presents a simplification of the Boyer-Moore algorithm, called *BMT*. BMT improves the efficiency of Boyer-Moore tables, d and dd table, by reduce the comparison between this 2 tables. We show its effectiveness through a number of experiments.

According to Thai language have no spaces between words. This feature causes the problem of word boundary ambiguity. The word boundary ambiguity may be solved by using a dictionary or a set of rules (Ngamwiwit et al.1995; Suchaichit et al.1995). In this paper, we apply a framework to improve searching efficiency and reduce word boundary ambiguity by grouping Thai contiguous characters into inseparable units, called *Thai character clusters (TCCs)*, based on the Thai language spelling features. A TCC is an unambiguous unit, which is smaller than a word and cannot further divided. Comparing with word segmentation there is no ambiguity in determining the TCC and can be realized by utilizing only a set of simple rules based on types of Thai characters, while determining word is not an easy task due to word boundary ambiguity. To investigate the merits of TCC, the framework is applied to full-text search for Thai documents. In this research, as non-indexing approach, brute-force, Knuth-Morris-Pratt (1977) and Boyer-Moore (1977) method are explored. The proposed framework is evaluated using a pre-segmented corpus, called Thai Tax[1], and Thai newspaper.

In the rest of this paper, section 2 illustrates the TCC concept. Experimental results are shown in section 3. The last section gives the discussion and conclusion.

1 THAI CHARACTER CLUSTER

Like some oriental languages such as Pali and Sangskrit, the Thai language has a various types of characters comparing with English, i.e., vowels, consonants, tones and some other special characters. In addition, Thai characters are located in three levels: upper, middle and lower levels. Figure 1 illustrates an example of a Thai phrase consisting of three words. In this example, there are three levels and seven types of characters: upper/lower/front/rear vowels, consonant, tone and karan (a pronunciation deletion character). $\sigma 1$, $\sigma 2$, $\sigma 3$ and $\sigma 4$ indicate word boundaries.

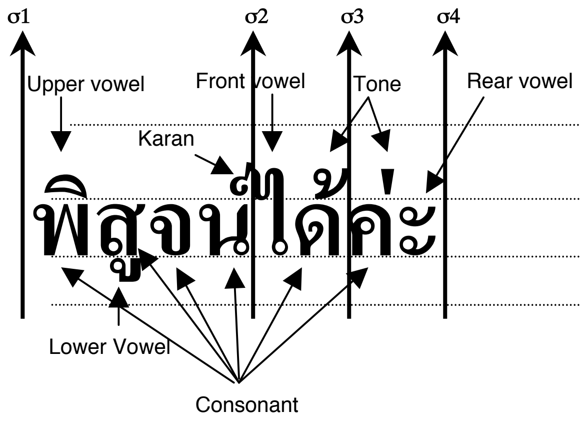

FIGURE 1. An example of Thai characters

[1] Thai Revenue Code documents, supported by NECTEC

Note that there is no space between the words. That is, a word segmentation algorithm is needed. Most algorithms are based on exploiting a dictionary (Kawtrakul A. et al., 1997; Meknavin, S. et al., 1997; Kanlayanwat and Prasitjutrakul, 1997). The fragile case for word segmentation is when it needs semantic background for making decision. For example, 'ตากลม' can be segmented as 'ตา-กลม' or 'ตาก-ลม'. The correct segmentation depends on the context. Moreover, if the sequence on focus is composed of unknown words, it is very hard to segment it into words, and only sistrings of the sequence can be kept instead. To overcome the problem, this research proposes a concept of character cluster, which is a unit smaller than a word but larger than a character. We call this 'Thai Character Cluster (TCC)'. The composition of TCC is unambiguous and can be defined by a set of rules. For example, a front vowel and the next character have to be grouped into a same unit. A tonal mark is always located above a consonant and cannot be separated from the consonant. A rear vowel and the previous character have to be grouped into a same unit.

The rule for segmenting into TCCs can be represented by a context-free grammar. In the proposed framework, there are rules for constructing the TCC. Figure 2 shows some example rule.

```
<TCC>->
        |'ก็่'
        |'อื่'
        |'หึ่'
        |<Cons>'รร<Cons>'์'
        |<Cons><bCons><Cons>'์'
        |<Cons><TCC1><Karan>
        |<fSara><Cons><TCC2><Karan>
;
<TCC2>->
        <Cons>'าะ'
        |'็'<bCons>
        | <uSara>{<tone>}<bCons>{['า'|'ะ']}
        | {<tone>} {'า'|'าะ'|'ะ'}
;

<TCC1>
        <dSara>{<tone>}
        | {<tone>} 'ำ'
        | ['ิ่'|'ี่'] {<tone>}<bCons>
        | 'ั'{<tone><'ุ'|'ิ'>}
        | '็'<bCons>
        | <tone>[<tSara>|<dSara>] {'ว'} <bCons>
        | 'ิ' {<tone>{<bCons>}}
        | 'ี่'<tone>
        | {<tone>}<bSara>
        | NULL
;
<Karan>->
        <Cons>{<Cons>} {[<dSara>|'ิ']}'์'
        |NULL
;
```

FIGURE 2. Some example rules of TCC.

They are expressed in the EBNF form that can be expanded to CFG rules, for example, in figure 1, 'พิสูจน์ได้ค่ะ' can be divided into 'พิ-สูจน์-ได้-ค่ะ' by these rules. Note that the string is almost correctly separated into words.

2 SEARCHING METHODOLOGY

2.1 Brute-force, Knuth-Morris-Pratt and Boyer-Moore

As the non-indexing approach, brute-force, Knuth-Morris-Pratt(1997) and Boyer-Moore algorithm (1977) are explored. The brute-force algorithm is the simplest approach for string searching, which involves comparing each and every substring in the text that is being searched against the text that is being searched for. The idea consists of simply trying to match any substring of length m in the text with the given m-length pattern from left to right.

The Knuth-Morris-Pratt String Searching (Knuth, Morris and Pratt 1977), discovered in 1970, is the first algorithm for which the constant factor in the linear term, in the worst case, does not depend on the length of pattern. The basic idea behind this algorithm is that each time a mismatch is detected, the "false start" consists of characters that we have already examined. We can take advantage of this information instead of repeating comparisons with the known characters.

2.2 BMT

The Boyer-Moore algorithm (Boyer and Moore 1977) positions the pattern over the leftmost characters in the text and attempts to match it from right to left. If no mismatch occurs, then the pattern has been found. Otherwise, the algorithm computes a shift; that is, an amount by which the pattern is moved to the right before a new matching attempt is undertaken. This shift action makes it possible to avoid a lot of unnecessary comparisons which may occur in the brute-force algorithm.

There are many variations on the basic Boyer-Moore algorithm (Horspool 1980; Baeza-Yates 1989a, 1989b; Sunday 1990) that lend added efficiency to search process. This paper introduces a simplification of Boyer-Moore algorithm, which reduce the comparisons between Boyer-Moore tables by improving the d and dd table. Comparing BMT with Boyer-Moore algorithm, this approach can improve the searching efficiency as well as ideal Boyer-Moore algorithm. BMT is *a simplification of Boyer-Moore algorithm*. This algorithm improves the efficiency of Boyer-Moore tables, d and dd table. Initially, BMT has changed the d and dd table to reduce the comparisons between these two tables in Boyer-Moore algorithm. The different formula between the Boyer-Moore tables and BMT tables are shown as follow.

Boyer-Moore Table:

$d[x] = \min \{ s | s = m \text{ or } (0 \le s < m \text{ and pattern } [m-s] = x)\}.$

$dd[j] = \min \{ s+m-j \mid s \ge 1 \text{ and } ((s \ge i \text{ or pattern } [i-s] = \text{pattern } [i]) \text{ for } j < i \le m)\}$

BMT table:

$d [x] = \min \{s \mid s = m \text{ or } (1 \le s < m \text{ and pattern } [m-s] = x)\}.$

$dd'[j] = \min \{ s+m-j \mid s \ge 1 \text{ and } (s \ge j \text{ or pattern } [j-s] \ne \text{pattern}[j]) \text{ and } ((s \ge i \text{ or pattern}[i-s] = \text{pattern}[i]) \text{ for } j< i \le m) \}$

$dd[j] = \max\{s | s = dd'[j] \text{ or } s = d[\text{pattern } [m]]+m-j\}$

The idea of BMT is setting positions the pattern over the left most character in the text and attempts to match it from right to left. If mismatch occurs, the algorithm computes a shift equal to d (text [m]) from d table. Otherwise, this will compare character m-1 from the right end of the character strings instead of the left and after a mismatch occurs, shifting right will equal to dd (number of already match character) from dd table then continue. When two characters are the same, it will compare again until the pattern is matched. If the number of successful search texts is more than n-m+1, then the pattern is found.

3 EXPERIMENTAL RESULTS

To evaluate of the proposed framework, TCC and BMT, three kinds of experiments are conducted. These experiments are designed to check the following performance:

1. The evaluation of TCC
2. The improvement of string searching when TCC is introduced, and
3. Evaluation the efficiency of BMT algorithm

In the first experiment, the Thai Royal Institute Word dictionary (RI Word[2]) and the ThaiTax are used for evaluating the searching accuracy. There are 32,558 words in RIWord. ThaiTax is a manually-word-segmented corpus, which composes of 594,526 words with 2,743 distinct words.

In the last two experiments, the corpus used is taken from Thai newspaper named "Thanweek". The corpus is constructed by excluding English words or symbols from the original in order to test the efficiency on a pure Thai document. The corpus is totally 28 MB large. In the second experiment, the results of searching speed of brute-force, Boyer-Moore and Knuth-Morris-Pratt are compared in the dimension of Information Retrieval with and without TCC. The last experiment is done to examine the time and memory consumption in creating and modifying (inserting and deleting) the index.

[2] Pochananukrom Chabab Rachabanditayasathan BE.2525

3.1 Evaluation of TCC

A word, which is equivalent to a TCC and also a substring of another TCC words, frequently causes an error in Information Retrieval. Firstly, we examine the number of words of this type. To do this, the RIWord dictionary and ThaiTax corpora are segmented into TCCs by the rules described in section 2. As a result, a list of segmented units (TCCs) is obtained. Among TCCs in the list, this experiment focuses on the words, which are equivalent to TCCs in RIWord or ThaiTax.

TCC	Precision (%)					
	RIWord			ThaiTax		
	All	N0	N1	All	N0	N1
No	58.63	42.39	15.43	85.69	77.00	13.21
Yes	79.37	100.0	100.0	94.04	100.0	100.0

TABLE 1. Searching accuracy with/without TCC

Table 1 shows the searching accuracy of IR with and without TCC. Here, the ThaiTax corpus is used as a test corpus for testing the searching accuracy. The table shows only the precision because in all cases the recall is 100%. 'All' means all of words extracted from RIWord (or ThaiTax) are used for the search and the numbers in the column show the averages of the searching results. 'N0' means only the words, which are TCCs, are used for searching. 'N1' means only the words, which are TCCs and also substrings of some other words, are used for searching. The result indicates that the searching precision is improved when TCCs are applied. It is 100% in the case of N0 and N1. Without TCCs, the precision is very low, especially in the case of words, which are TCCs and also substrings of some other TCCs (N1). The numbers in 'All' column indicate the amount of the second and third problems in section 4.3. The concept of TCCs improves the precision from 58.6 % to 79.4 % for RIWord, and from 85.7 % to 94.0 % for ThaiTax. The ThaiTax gets higher precision than the RIWord because it has fewer words for N1 and N0.

3.2 The improvement of string searching when using TCC

The main objective of this experiment is to compare the performance of a number of string search algorithms with and without TCC. Initially, we test the efficiency of string search algorithm by applying syllable-like segmentation, TCC, into string search algorithms. 'BF' stands for the brute-force algorithm, 'BM' for the Boyer-Moore algorithm. The average time is taken using all words with any length. It is calculated by using the set of test patterns taken from RIWord.

Searching Algorithm		**Pattern Length**				
		1-5	6-10	11-15	16-20	21-up
BF	No TCM	29.49	29.71	29.81	29.84	29.68
	TCM	15.28	15.30	15.32	15.32	15.29
KMP	No TCM	29.46	29.65	29.74	29.77	29.60
	TCM	15.28	15.30	15.32	15.32	15.28
BM	No TCM	34.23	12.34	7.99	6.14	4.70
	TCM	23.54	11.96	7.36	5.71	3.77

TABLE 2. The number of comparisons (x10^6)

Table 2 shows the result of string search algorithm within a given range of the pattern length. In our experiment, we consider only the pattern lengths, which are less than 50 characters. This result shows that the combination of string search algorithm and TCC can improve searching speed approximately one to two times from the original. Hence, we can infer that TCC improves the efficiency of string search algorithm. Considering three kinds of string search algorithm, we find that BM algorithm is the best of string search algorithms in this experiment and it has the fewest comparisons with the highest searching speed. So we will focus on the BM algorithm in our next experiment. For KMP, the comparison result is not satisfied enough because Thai words do not have a lot of repeating pattern. The KMP algorithm has almost the same performance as the naïve (or brute-force) algorithm. Then it can infer that KMP might not be suitable for Thai documents.

3.3 Evaluate the efficiency of BMT

From the second experiment, we found that BM algorithm was the best among other string search algorithms. Therefore, we will mainly use BM algorithm in this experiment, which concentrates on our simplification of BM, so-called *BMT*. Comparing BMT with the original BM algorithm and the best-case BM to examine the performance of BMT by using Thai newspaper corpus. For the best-case BM, the computation complexity is N/M where N is text size and M is pattern length. This formula achieves from shifting the pattern (length M) by not matching the character at the end of the pattern until the end of text (length N). It is not necessary to compare any other characters inside the pattern. The graph in Figure 3 show the comparison result of BM algorithm, BMT and the ideal BM for Thai newspaper, Thanweek, corpus.

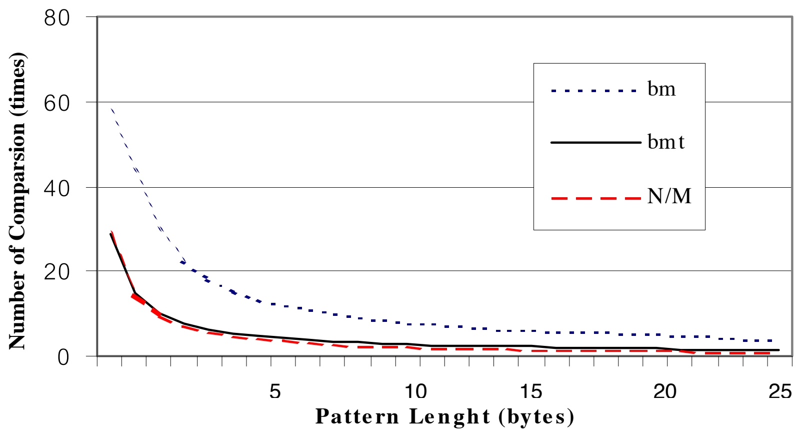

FIGURE 3. Comparison between BM algorithms

The graph shows that BMT performs almost as well as the best-case BM, and BMT also increases searching speed approximately 1-3 times compared with BM. As expected, the result concludes that BMT acquires better performance than BM algorithm.

4 DISCUSSION

This section discusses about information retrieval methods and a comparison between non-indexing and indexing approach. Although indexing approach can search faster than non-indexing approach, indexing approach needs more time and space for creating an index file. Moreover, for some applications where only one time the search is held, it is not good to waste time for creating index file before searching. In this case, we should use non-indexing approach instead of indexing approach. This means that even the slowness of non-indexing approach; the non-indexing approach is still needed. Toward an improvement of searching speed to the non-indexing approach, we have developed a simplification of BM algorithm, so-called BMT. BMT surpasses the complicated BM algorithm and is proved to be suitable for Thai documents in both accuracy and computation time by changing the BM tables as shown in section 3. The searching speed of BMT is sufficient for general search, especially when applying TCC into the algorithm. TCC is helpful for not only improving the searching speed but also reducing error rate, due to less word boundary ambiguity as seen in the section 3.1.

Currently, there are still very few researches on full-text search in Thai IR system. Moreover, tools for retrieving Thai information are still insufficient. Focused on this, this paper proposes a number of the string search algorithms and their implementation for Thai documents.

5 CONCLUSION

This paper proposes a set of search algorithms for Thai language and evaluates their comparison. To improve searching efficiency Thai documents, this paper applies a syllable-like segmentation, named TCC. TCC is shown to be very helpful to reduce searching error. It also improves the searching time and memory consumption. The experimental result shows that the combination of search algorithms and TCC can improve all searching algorithms in all measures.

As non-indexing approach, This paper presents a simplification of BM algorithm, called BMT, which improves the efficiency of BM tables, d and dd table. BMT outperform the traditional string search algorithms in all measures. It is shown that BMT can get the result as well as to the best-case BM.

ACKNOWLEDGEMENT

We would like to thanks to NECTEC for supporting all information in this research. Special thanks to Ms. Sasiporn Usanavasin who gives a good comments during writing this paper.

REFERENCES

BAEZA-YATES, R 1980a. Efficient Text Searching Ph.D. thesis Dept. of Computer Science, University of Waterloo Also as Research Report, CS-89-17.

BAEZA-YATES, R. 1989b. Imrproved String Searching Software-Practice and Experience, 19 (3) pp.257-271.

BOYER, R., and MOORE, S. 1977. A fast string searching algorithm CACM, Vol. 20, pp. 762-772.

FRAKES, W. B. and BAEZA-YATES, R eds 1992 Information Retrieval: Data Structures & Algorihtms, Prentice Hall.

HORSPOOL, R. N. 1980. Practical Fast Searching in Strings. Software-Practice and Experience. 10, pp. 501-506

KANLAYANAWAT, W. and PRASITJUTRAKUL, S. 1997. Automatic Indexing for Thai Text with Unknown Words using Trie Structure Proceedings of the Natural Language Processing Pacific Rim Symposium 1997 (NLPRS'97), pp 341-346.

KAWTRAKUL, A., THUMKANON, C. POOVORAWAN, Y., VARASRAI P. and SUKTARACHAN, M. Automatic Thai Unknown Word Recognition In Proceedings of the Natural Langague Processing Pacific Rim Symposium 1997 (NLPRS'97), pp. 341-346.

KNUTH, D., MORRIS, J., and PRATT, V. 1977. Fast Pattern Matching in Strings. Journal of SIAM on computing, 6, pp. 323-350.

MEKNAVIN, S., CHAROENPORNSAWAT, P. and KIJSIRIKUL, B. Feature-based Thai Word Segmentation In Proceedings of the Natural Language Processing Pacific Rim Symposium 1997 (NLPRS'97), pp. 41-46.

NGAMWIWAT, J., VARAKULSIRIPUNTH, R., JUMWUN, S., CHIWATTAYAKUL, S., THIPCHAKSURAT, S. 1995 Word Segmentation In Thai Sentence by Longest Word Mapping. Papers on Natural Language Processing 1995. NECTEC, pp 279-290.

SUCHAICHIT, W., VARAKULSIRIPUNTH, R., JUMWUN, S., THIPCHAKSURAT, S. 1995. An analysis on Correct Sentence Selection by Word's General Usage Frequency. Papers on Natural Language Processing 1995. NECTEC, pp. 291-300.

SUNDAY, D. 1990. A Very Fast Substring Search algorithm. Communications of the ACM, 33(8) pp. 132-142.