Character Cluster Based Thai Information Retrieval

Thanaruk Theeramunkong[1] **Virach Sornlertlamvanich**[2]

Thanasan Tanhermhong[1] **Wirat Chinnan**[1]

[1] Information Technology Program, Sirindhorn International Institute of Technology, Thammasat University
P.O. Box 22 Thammasat Rangsit Post Office, Pathumthani 12121, Thailand
[2] Software and Engineering Laboratory, National Electronics and Computer Technology Center (NECTEC)
National Science and Technology Development Agency (NSTDA)
Gypsum Metropolitan Tower 22nd Floor 539/2 Sriyudhya Rd., Rajthevi, Bangkok 10400 Thailand
Email: ping@siit.tu.ac.th, virach@nectec.or.th

Abstract

Some languages including Thai, Japanese and Chinese do not have explicit word boundary. This causes the problem of word boundary ambiguity that results in decreasing the accuracy of information retrieval. This paper proposes a new technique so-called character clustering to reduce the ambiguity of word boundary in Thai documents and hence improve searching efficiency. To investigate the efficiency, a set of experiments using Thai newspapers is conducted in both non-indexing and indexing searching approaches. The experimental results show our method outperform the traditional methods in both non-indexing and indexing approaches in all measures.

Keywords: Character Cluster, Thai Document, Indexing and Non-indexing information retrieval

1 Introduction

Recently, along with the fast progress of information systems, there has been rapid growth of building large electronic text documents, such as electronically-published newspapers, on-line distributed documents (hypertext) on the World Wide Web and so forth. This motivates the need of developing efficient searching algorithms used in information retrieval systems to find required information from large-scaled documents. While there have been a lot of various efficient search algorithms developed for English documents [1], these algorithms may be directly used for other languages. However, due to idiosyncrasies of each individual language, directly applying such algorithms may not be suitable for the language considered.

Many languages including Chinese, ancient Greek, Chinese, Japanese and Thai have no explicit word boundary delimiters. To build an information retrieval system for these languages, the process of word segmentation is needed. However, processing these languages suffers with word boundary ambiguity. This causes the reduction of searching accuracy, such as during full-text search.

To solve this problem in Thai language, this paper proposes a new technique to improve searching efficiency by grouping Thai contiguous characters into inseparable units, so called Thai Character Clusters (TCCs), based on the Thai language spelling features. A TCC is an unambiguous unit that is smaller than a word and cannot be 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 words 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. However, TCC is a general concept that can be applied in any other information retrieval (IR) algorithms. In this research, both non-indexing and indexing full-text searching approaches are explored. As the non-indexing approach, brute-force and Boyer-Moore methods [2] are explored. One-character-indexed inverted files [3] and sorted sistring arrays (also called, PAT array) [4][5], are investigated as the indexing approach. The proposed framework is evaluated using a pre-segmented corpus, called ThaiTax, and Thai newspapers.

Proceedings of the 5th International Workshop Information Retrieval with Asian Languages

Copyright ACM 1-58113-300-6/00/009 … $5.00

In the rest of this paper, section 2 explores some previous works in Thai IR and full-text searching algorithms. Section 3 illustrates the TCC concept. In section 4, full-text search with TCC and the merits of TCC in this task are described. Experimental results and discussions are shown in section 5. The last section gives the conclusion.

2 Previous Works

2.1 Information Retrieval for Thai Documents

At present, there are still very few researches on information retrieval (IR) for Thai language. Kanlayanawat et al. [6] proposed a method for indexing Thai text with unknown words using a trie structure. Due to no explicit word boundary delimiter in Thai language, the method applied a word segmentation algorithm to segment a given text to a sequence of words, and to sistrings (semi-infinite strings) in the case of unknown words. The index of the obtained words was kept in an original trie structure. Although a way to deal with unknown words was suggested in this research, only theoretical evaluation was conducted. As another work, Mitrapiyanuruk et al. [7] developed a full-text search engine for large-scaled Thai text database. The approach used an inverted file where the index was kept in the form of a double array trie [8]. The index was constructed by keeping the words resulted from a word segmentation algorithm in a trie structure. In the case of segmentation ambiguity, all of the possible combinations were kept. Both of these works applied dictionary-based word segmentation to divide a contiguous string into words. They could deal with the problems of unknown words by keeping their sistrings, and segmentation ambiguity by storing their all possibilities. Although these methods seem to work well in Thai full-text search, there are a number of disadvantages: (1) it is impossible to search, from the text database, for words that are not found in the dictionary, (2) some irrelevant texts are retrieved when the search keyword is a substring across the word boundary of that string. For instance, if the search keyword is 'ลิตร' (litre), the string 'การ-ผลิต-ราย-การ'[1] (program production) are found. This result is incorrect because the last character of the keyword 'ร' cannot treat separately from the next character 'า' in the text, (3) some irrelevant texts are also retrieved when a search keyword is a substring of an unknown word. For example, when 'เป้า' (target) is an unknown word in the text database and the user wants to find 'ป้า'(aunt), texts including 'เป้า' (target) will be retrieved. This is also incorrect because the first character 'เ' of 'เป้า' (target) cannot be separated from the rest 'ป้า'.

The first problem prevents you from getting some needed documents while the next two problems provide you some excess irrelevant documents. Among these, the first problem can be solved automatically if one applies full-text search without using a dictionary, e.g., string search, but the next two problems still remain. To cope with all problems, this paper proposes a framework to group Thai contiguous characters into inseparable units. These units are called Thai Character Clusters (TCCs) and can be generated by a set of simple rules based on types of Thai characters as shown in the next section.

2.2 Full-Text Search with/without Index

Although there are a lot of merits in using keywords extracted from documents as indices for information retrieval (IR) systems, full-text search is still demanded. It can be used as both direct search for keywords in large amounts of text, and a complementary part of IR systems for filtering of potential matches or for searching retrieval terms that will be highlighted in the output. The non-indexing approach, e.g. the brute-force algorithm, the Knuth-Morris-Patt algorithm [9] and the Boyer-Moore algorithm [2], conducts the search through the original text. Merits of this approach are that there is no additional storage needed to keep index and no additional process to perform when the original text is modified. On the other hand, the indexing approach, such as inverted file using a trie structure, B-trees or sorted array [3], requires a process to build an index and needs some additional storage to keep the generated index. However, in general, it can conduct the search much faster than the non-indexing approach does.

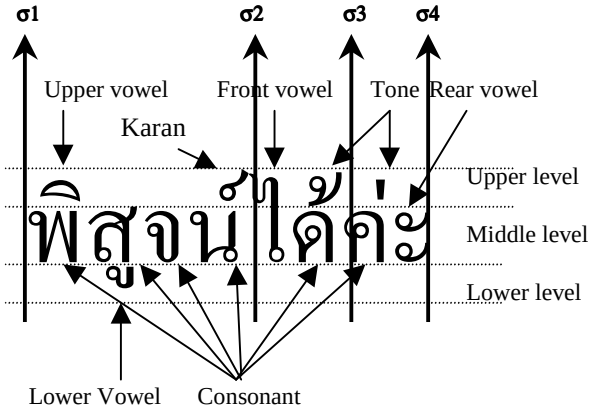

Figure 1: An example of Thai characters

3 Thai Character Cluster (TCC)

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

[1] For clarity's sake, a meta symbol '-' is inserted to indicate the word boundary. It does not exist in the actual

character). σ1, σ2, σ3 and σ4 indicate word boundaries. 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 [6][10][11]. 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 rules for segmenting into TCCs can be represented in EBNF form.

<TCC> → 'กี่' | 'อื' | 'หื่'

| <Cons> 'รร' ,<Cons> '◌์'

| <Cons> <BCons> <Cons> '◌์'

| <Cons> <TCC1> <Karan>

| <FSara><Cons> <TCC2> <Karan>

<TCC2> → <Cons>'◌ะ'

| '◌็'<BCons>

| <USara>{<Tone>}<BCons> ['า' | 'ะ']

| {<Tone>}['า' | 'าะ'| 'ะ']

<TCC1> → <DSara> {<Tone>}

| {<Tone>} '◌ำ'

| ['เ◌' | 'แ◌'] {<Tone>}<BCons>

| '◌ั'{<Tone>['◌ุ' | '◌ิ']}

| '◌็' <BCons>

|<Tone>[<TSara>|<DSara>]{'ว'}<BCons>

| '◌ิ'{<Tone>{<BCons>}}

| '◌็'<Tone>

| {<Tone>} <Bsara>

| NULL

<Karan> → <Cons>{<Cons>}{[<DSara>| '◌ิ']}'◌์'

| NULL

Figure 2: All rules for TCC Segmentation

Figure 2 shows all rules used for TCC segmentation. Here, <FSsara>, <TSara>, <USara> and <DSara> represent a front vowel, a rare vowel, a upper vowel, and a lower vowel, respectively. <Cons> is a consonant that can be the first consonant of a word, <BCons> is a consonant that can be the second consonant of a word, <Tone> is a tonal mark, <Karan> is a special character named 'karan'. All the characters within ' ' are Thai characters. 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.

4 Approaches in TCC-based IR

In this paper, the TCC framework is applied to full-text search algorithms in both non-indexing (i.e., string search) and indexing approaches. This section briefly describes the algorithms.

4.1 Non-Indexing Approach

As the non-indexing approach, brute-force and Boyer-Moore algorithms are explored. The brute-force algorithm is the simplest approach for string searching. 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 Boyer-Moore algorithm [2] 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.

4.2 Indexing Approach

Here one-character-indexed inverted files and sorted sistring (semi-infinite) arrays are investigated as the indexing approach. The concept of inverted file type index is to collect all keywords (attributes) and form a sorted list (or index) of the keywords (attributes), with each keyword having links (positions) to the documents (texts) containing that keyword. The inverted file concept applied here is to use a single character as an index to all positions where the character is located in the documents. Retrieving a pattern from the documents is to (1) look up each character in the pattern from the index table, (2) get the positions for each character, (3) find the minimum positions among those of the characters in the pattern, and (4) then compare the pattern with the strings at those positions in the original documents.

In the second method, the sorted sistring array is applied as a searching index to the documents. The sorted sistring array is a sort listed of all sistrings in the documents. This array is similar to a PAT array (also called a suffix array) [5]. To find a specified pattern in the documents is to

perform an indirect binary search over the array with the results of the comparisons being less than, equal, and greater than. This process takes time $O(\log_2 n)$, where n is the length of the document.

4.3 TCC-based IR

The concept of TCC can be applied directly to both non-indexing and indexing approaches. For the non-indexing approach, the original document is processed by the rules described in section 3 to group characters in the document together to form character clusters. Instead of including characters, the newly generated document, for sake called a TCC document, is composed of character clusters. The original search algorithm can directly be applied on the TCC document. As the second option, the search process performs on the original document but later the searching results will be checked by the rules in section 3.

In the same way, for the indexing approach, the original document is processed by the rules to form a TCC document, and then the original index creation algorithm can be directly applied on the TCC document. The searching process is the same as the original one. Exploiting the TCC concept, the three following problems of traditional methods (refer to section 2.1) can be solved.

1. The relevant documents are not retrieved when the query word A is a substring of the word B, which is found in the dictionary.

   $A = \beta$

   $B = \alpha\beta\gamma$

   where α, β and γ are substrings.

2. The irrelevant documents are retrieved in the case that the query word A is a substring of a string B in the document but B is not morphologically breakable.

   $A = \beta$

   $B = \alpha\beta\gamma$

   where α, β and γ are substrings, and $\beta\gamma$ or $\alpha\beta$ is morphologically unbreakable.

3. The irrelevant documents are retrieved in the case that the query word A is a substring of an unknown word B in the document.

   $A = \beta$

   $B = \alpha\beta\gamma$

   where α, β and γ are substrings and $\alpha\beta\gamma$ is morphologically unbreakable word.

The first case can be solved by full-text search. The second and third cases are very similar except that A and B in the third case are both TCCs but both in the second case are not. The concept of TCCs can solve all of the three problems because TCCs are helpful to suggest that A cannot be a substring of B.

5 Experimental Results

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

1. The accuracy of IR with/without TCC
2. The improvement of searching time when TCC is introduced, and
3. The time and storage overhead that the indexing approach has to compensate comparing with the non-indexing one.

In the first experiment, the Thai Royal Institute Word dictionary (RIWord) and the ThaiTax corpus (Thai Revenue Code) 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, inverted file, sorted sistring are compared in the dimension of IR with and without TCC. The indexing approach, i.e., inverted file and sorted sistring, requires some additional storage to keep an index. The last experiment is done to examine the time and memory consumption in creating and modifying (inserting and deleting) the index.

	RIWord	ThaiTax
No. of TCCs (N0)	1731	402
No. of smaller TCCs (N1)	290	24
No. of larger TCCs (N2)	592	28
No. of errors (N3)	759	29

Table 1: The number of TCCs in RIWord and ThaiTax

5.1 Evaluation of IR Accuracy

A word, which is equivalent to a TCC and also a substring of another TCC words, frequently causes an error in IR. 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 3. 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. Table 1 shows the total number of distinct TCCs (N0), the number of distinct TCCs which can be substrings of other TCCs (N1), the number of distinct TCCs some substrings of which are TCCs (N2) and the number of errors that occur when N1 are used for searching but N2 are retrieved if TCCs are not applied (N3).

From 32,558 words in the RIWord, 1731 words (N0) are TCCs. Among this number, 290 words (N1) can be substrings of some other words and 592 words (N2) have some other words as their substrings. 759 combinations (N1,N2) are the errors occurred when N1 are used for searching but N2 are retrieved as their results. The ThaiTax consists of much fewer words, i.e. nearly 2743 distinct words, but only 402 words are TCCs. That is, the ThaiTax is roughly segmented. Its N1, N2 and N3 are 24, 28 and 29, respectively. This result shows the amount of the third problem shown in section 4.3.

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 2: Searching accuracy with/without TCC

Table 2 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 in N1 and N2.

5.2 Searching Time

The second experiment shows the improvement in searching speed when TCC is introduced. 'BF' stands for the brute-force algorithm, 'BM' for the Boyer-Moore algorithm, 'INV' for the one-character-indexed inverted file and 'SSA' for the sorted sistring array. The average time is taken using all words with any length.

Method	TCC	Pattern Length (chars.)			Avg. Time (sec)
		1-10	11-20	21 up	
BF	No	29.60	29.83	29.68	4.00
	Yes	15.30	15.33	15.29	2.01
BM	No	23.28	7.06	4.70	1.51
	Yes	17.75	6.54	3.77	1.40
INV	No	4.38	2.24	2.28	0.22
	Yes	0.41	0.34	0.02	0.03
SSA	No	.0015	.0039	.0072	.0024
	Yes	.0012	.0029	.0054	.0023

Table 3: The number of string matching times (10^6 times) and average searching time (seconds)

Table 3 indicates that TCC can improve all algorithms. Both the number of string matching times and the average searching time are shown. BF is the slowest algorithm, gaining 2 times improvement when TCC is applied. This is because the size of TCC document is a half of that of the original. The BM algorithm depends directly on the length of the search pattern. Applying TCC, the speed of BM improves 1.0-1.3 times. TCC helps INV in improving its index structure. Instead of using one character each, the unit of TCC is used as an index. Consequently, searching speed improves about 7-100 times. SSA gives the best searching performance among the four algorithms. TCC improves 1.0-1.4 times for SSA.

5.3 Time and Memory Consumption in Index Creating

Though the indexing approach, i.e., INV and SSA yields the far better searching time performance than the non-indexing approach, it needs an additional step to create the index for the documents and additional storage to keep the index. Moreover when there are some modifications in the text database, the index has to be updated. The last experiment shows the performance of creating the index for the document.

Methods	TCC	Time (sec)	Memory (bytes)
INV	No	71	110M
	Yes	60	58M
SSA	No	3753	110M
	Yes	3122	58M

Table 4: Time and memory consumption in creating the index

Table 4 shows the time and memory consumption in creating the index for INV and SSA. TCC helps in reducing the indexing time and memory consumption in both INV and SSA. It is about 1.2 times faster when TCC is introduced. The memory consumption decreases to be about 50%. Though SSA requires much more time to create the index than INV does, SSA requires about the same size of memory for creating the index.

Modifying an index of INV is quite simple in both insertion and deletion while the insertion in SSA is quite complicated because the costly sorting process is needed. In one additional experiment, the Thansetthakij newpapers (35 MB) is used to examine the index insertion. To add this document to the existing index (Thanweek newspapers), it needs around 1,200 seconds with 67MB for TCC. In the approach without TCC, it requires up to 2,400 seconds for the same document.

6 Conclusion

This paper proposes a technique called character clustering to reduce the ambiguity of word boundary in Thai documents and improve searching efficiency. A Thai character cluster (TCC) is an inseparable unit. It can be detected by using a set of few simple rules. Supported by some experiments using a set of well-known algorithms, such as brute-force, Boyer-Moore, inverted file and sort sistring array, TCC is shown to be very helpful to reduce searching errors. It also improves the searching time and memory consumption. Our experiments show the improvement of using TCC for all searching algorithms in all measures.

The TCC framework also shows the merits in both non-indexing and indexing approaches. The combination of TCC and a dictionary will be explored to improve IR speed and accuracy in our future work. TCC can also be applied to improve the accuracy of word segmentation and spell checking.

7 Acknowledgement

This research is partly supported by National Electronics and Computer Technology Center (NECTEC). We would like to thank NECTEC for permitting us to access several useful Thai language resources including RIWord dictionary and ThaiTax Corpora.

References

1. Frakes, W. B. and Baeza-Yates, R. Eds. Information Retrieval: Data Structures & Algorithms. Prentice Hall, 1992.

2. Boyer, R. and Moore, S. A fast string searching algorithm. CACM, 1977, 20, pp. 762-772.

3. Harman, D., Fox, E., Baeza-Yates, R. and Lee, W. Inverted Files. In *Information Retrieval: Data Structures & Algorithms*, Eds. Frakes W.B. and Baeza-Yates R. Prentice Hall, 1992, pp. 28-43.

4. Gonnet, G. Pat 3.1: An Efficient Text Searching System. User's Manual. UW Centre for the New OED, University of Waterloo, 1987.

5. Manber , U. and Myers, G. Suffix Arrays: A New Method for On-line String Searches. In *Proceedings of the first ACM-SIAM Symposium on Discrete Algorithms*. 1990, pp. 319-327.

6. Kanlayanawat, W. and Prasitjutrakul, S. Automatic Indexing for Thai Text with Unknown Words using Trie Structure. In *Proceedings of the Natural Language Processing Pacific Rim Symposium (NLPRS'97)*, 1997, pp. 115-120.

7. Mitrapiyanuruk, P., Puvanich, C., Meknavin, S. and Boriboon, M. *A.* Development of Full-Text Search Engine for Large Scale Thai Text Database. In *the 1999 National Science and Technology Development Agency (NSTDA) Annual Meeting*. in Thai, 1999, pp. 247-257.

8. Jun'ichi, A. Quick Digital Search for Double Array Trie. Bit, 21/6, March 1989, pp. 776-784.

9. Knuth, D., Morris, J., and Pratt, V. Fast Pattern Matching in Strings. In *Journal of SIAM on computing.* 1977, 6, pp.323-350.

10. Kawtrakul, A., Thumkanon, C. Poovorawan, Y., Varasrai P. and Suktarachan, M. Automatic Thai Unknown Word Recognition. In *Proceedings of the Natural Language Processing Pacific Rim Symposium (NLPRS'97)*. 1997, pp. 341-346.

11. Mekanavin, S., Charoenpornsawat, P. and Kijsirikul, B. Feature-based Thai Word Segmentation. In *Proceedings of the Natural Language Processing Pacific Rim Symposium (NLPRS'97)*. 1997, pp. 41-46.