An automatic indexing technique for Thai texts using frequent max substring

Todsanai Chumwatana, *Member, IEEE,* Kok Wai Wong, *Senior Member, IEEE*, Hong Xie, *Member, IEEE*

Abstract—**Thai language is considered as a non-segmented language where words are a string of symbols without explicit word boundaries, and also the structure of written Thai language is highly ambiguous. This problem causes an indexing technique has become a main issue in Thai text retrieval. To construct an inverted index for Thai texts, an index terms extraction technique is usually required to segment texts into index term schemes. Although index terms can be specified manually by experts, this process is very time consuming and labor-intensive. Word segmentation is one of the many techniques that are used to automatically extract index terms from Thai texts. However, most of the word segmentation techniques require linguistic knowledge and the preparation of these approaches is time consuming. An n-gram based approach is another automatic index terms extraction method that is often used as indexing technique for Asian languages including Thai. This approach is language independent which does not require any linguistic knowledge or dictionary. Although the n-gram approach out performs many indexing techniques for Asian languages in term of retrieval effectiveness, the disadvantage of n-gram approach is it suffers from large storage space and long retrieval time. In this paper we present the frequent max substring mining to extract index terms from Thai texts. Our method is language-independent and it does not rely on any dictionary or language grammatical knowledge. Frequent max substring mining is based on text mining that describes a process of discovering useful information or knowledge from unstructured texts. This approach uses the analysis of frequent max substring sets to extract all long and frequently-occurred substrings. We aim to employ the frequent max substring mining algorithm to address the drawback of n-gram based approach by keeping only frequent max substrings to reduce disk space requirement for storing index terms and to reduce the retrieval time in order to deal with the rapid growth of Thai texts.**

I. INTRODUCTION

The amount of electronically stored information in Thai language has rapidly grown in the past few years and the number of these documents is still continually increasing. This makes information extraction (IE) as one of the more essential techniques for extracting index terms from Thai texts [1] - [3] in information retrieval [4], [5]. IE refers to the extraction of index terms from the texts, which is prerequisite to information retrieval. The basic method of information retrieval [5] is to use the index terms to retrieve documents that are likely to be relevant to the user's query. The frequencies of index terms are also used to compute their relevance scores for ranking the retrieved documents. For European languages, the process of text indexing is straightforward. This is because European texts are explicitly segmented into word tokens by word delimiter such as space or other special characters. These tokenized words can then be indexed into the inverted index for efficient retrieval. Unlike European languages, words in many Asian languages such as Chinese, Japanese and Korean are not explicitly delimited. A sentence consisting of several words is a string of symbols without explicit word boundary delimiters to separate these words. Thai language is also one of the non-segmented languages that written continuously as a sequence of characters without using word boundary delimiters. Due to this problem, IE must be applied first to extract index terms for Thai texts. IE refers to a process of discovering important keywords or index terms from unstructured texts for indexing. Many information extraction techniques have been proposed to perform the index terms tokenization. After which, these segmented index terms will be stored into the inverted index structure. Most techniques are based on word segmentation which usually relies on any dictionary or requires the linguistic knowledge of the language [3], [6]. However, there is some other techniques which do not rely on language analysis.

Manuscript received May 2009

C. Todsanai is with School of Information Technology Murdoch University, South St, Murdoch Western Australia 6150; email: T.chumwatana@murdoch.edu.au

K.W. Wong is with the School of Information Technology, Murdoch University, Australia (corresponding author – phone: +61 8 9360 6100; fax: +61 8 9360 2941; e-mail: k.wong@murdoch.edu.au).

H. Xie is with the School of Information Technology, Murdoch University, Australia (e-mail: h.xie@murdoch.edu.au).

978-1-4244-4139-6/09/$25.00 ©2009 IEEE

II. THAI INDEX TERMS EXTRACTION

In indexed Thai texts using an inverted index [5], word segmentation [6] – [10] is one of the most widely used information extraction techniques in Natural Language Processing (NLP). The word segmentation technique is used to perform the index terms tokenization. The exploitation of word segmentation techniques for IE is not new and there are several approaches to Thai word segmentation. The techniques for Thai word segmentation can be broadly classified into three approaches: Dictionary based [11], [12], Rule based [13] - [15] and Machine learning based approaches [16], [17]. Most of these approaches are language-dependent, they rely on language analysis or on the use of dictionary. Also, the preparation of these methods is very time consuming. Therefore, word segmentation has become a challenging task and one of the main issues in Natural Language Processing for Thai texts due to its non-segmented nature.

Of the language-independent approaches for indexing technique, an n-gram based approach [18], [19] is an alternative method for extracting index terms from Thai texts [20]. Since the n-gram based approach does not require any language knowledge and does not concern the meaning of word, this method is most widely used in many Asian languages [21] – [26] such as Chinese, Japanese and Korea (CJK) because these Asian languages share the similar difficulties in segmenting texts and specifying index terms. This technique is acknowledged by many Asian researchers as a workable solution to information retrieval problem for Asian languages. It outperforms other word segmentation techniques in retrieval effectiveness, but suffers from high storage space and retrieval time.

III. FREQUENT MAX SUBSTRING

In this paper, we propose a substrings mining technique to classify terms called Frequent Max substrings [27] or FM from the input string where the word boundary and characteristic are not clearly defined. We also allow the construction of the index file using trie data structure. We aim to extend Vilo's algorithm [28] for mining all frequent substrings by constructing only the subtrees of suffix trie that correspond to the frequent substrings of the string. We selected Vilo's algorithm as the starting point for this research because it is more efficient and scalable compared with other similar algorithms. However, if we can construct only subtrees that correspond to the frequent max substrings which contain all frequent substrings, it uses less space for storing all frequent substrings and more efficient for mining these substrings.

In order to explain the concept, we first define the frequent max substring (FM).

A. Definition of frequent max substrings (FM)

Frequent max substrings mining or FM mining is to mine frequent max substrings through a given threshold having no superstrings which have equal frequency.

Let a string $T = (s_1, s_2, \ldots, s_n)$ where $s_i \in \sum$ for $i \in \{1, 2, 3, 4, 5, 6, \ldots, n\}$ and $\sum$ is a finite set of characters that has N symbols [30]. An integer $\theta \geq 2$ is a threshold.

Let $T = (s_1, s_2, \ldots, s_n)$ be a string of length n. A set of substrings of string T, or SP_T, is a set of substrings $t = \langle s_j, \ldots, s_k : f \rangle$ of the string T ; where $1 \leq j \leq k \leq n$, f is the frequency of t and $m = k - j + 1$ is the length of the substring.

Let α and β be members of SP_T. If α is a substring of β, we call β a superstring of α. This relation is denoted

$$\alpha \sqsubseteq \beta.$$

If α *is a true substring of* β, ie, $\alpha \neq \beta$, we denote

$$\alpha \subset \beta.$$

A max substring of string T is a member of SP_T that has no superstrings which have equal frequency. A set of max substrings is denoted M_T :

$$M_T = \{\alpha \in SP_T | \nexists \beta \in SP_T, \alpha \subset \beta \text{ and } frequency(\beta) = frequency(\alpha)\}$$

A set of frequent substrings of string T or $FSP_{T\theta}$ is a set of substrings of string T that has frequency at least θ; where $FSP_{T\theta} \in SP_T$, as denoted

$$FSP_{T\theta} = \{\alpha \in SP_T | frequency(\alpha) \geq \theta\}$$

A set of frequent max substrings of string T or $FM_{T\theta}$ is a subset of $FSP_{T\theta}$ whose frequent substrings having no superstrings with equal frequency:

$$FM_{T\theta} = \{\alpha | \alpha \in FSP_{T\theta} \text{ and } \alpha \in M_T\}$$

Trie data structure is employed to find FM and to create the index file at the same time. The basic strategy for constructing FM is to enumerate substrings with their frequencies. These substrings will then be correctly selected based on pre-defined frequency or threshold. The enumeration has to give correctly both all substrings and their frequencies. This requires an efficient enumeration method. Suffix trie structure [29] is an efficient enumeration method but it enumerates only substrings without their frequency information. If suffix trie could also keep frequencies of substrings, it could be exploited in finding FM. The *wotd* (write-only top-down) suffix trie proposed by Vilo [28] keeps the starting position of substrings without their frequencies. In this paper we propose the concept of frequent suffix trie (FST). The proposed FST is constructed by extending from Vilo's suffix trie to find FM. The FST structure inherits the following properties from suffix trie structure: A) the frequency of parent substrings are always greater than or equal to the frequencies of its child substrings in the same path, because the parent substrings are distributed to child substrings. B) A set of frequent substrings can be covered by a set of frequent max substrings. These properties are exploited to reduce the number of substrings in FM mining. As a result, all FM are showed on the resulting FST structure. Therefore, all possible frequent substrings can be derived from the FM because the FM contains all possible frequent substrings while less space is required to keep the FM. In addition, the index file can be built from the resulting FST structure that shows all FM with their frequencies and list of positions. This technique is easy to implement and uses less space. The frequencies and list of positions in FST will be important in ranking text documents or web pages. Such information enables fast and accurate information retrieval.

For our proposed FST, the definition is as follows.

B. Definition of FST structure

A set of all suffixes of an *n*-length string *T* or $S[s_i...s_n]$; where $1 \leq i \leq n$, is a set of substrings of string *T* that starts at position *i* and ends at position *n* [29].

The FST structure of *n-length* string *T* is tree structure that represents all suffixes of string *T* starts with root node and ends with *n* leaf nodes. Also '$' is appended at the end of string *T*. $, the terminating symbol, is added to show the end of string *T* and to make all suffixes of string *T* different from each other. Therefore, all suffixes of string *T* contain $ at the *n* different ends of the FST structure.

FST structure also shows all substrings with their frequencies and positions in string *T*.

Edge is a symbol or a character that is an element of the character set. Each edge starts with the same character, and then an extra character is added at each edge.

A node is used to represent a substring with frequency and list of positions (or .pos). The position is the end position of each substring of string *T*. The depth of each node represents the increased length of the substrings. All leaf nodes keep suffixes with their frequencies and positions of suffixes.

IV. ALGORITHMS

A. The traditional algorithm

Suffix trie structure [30] is the traditional method to enumerate all possible substrings from a given string but it only records substrings without their frequencies of occurrence. On the other hand, the pattern trie [18] keeps the starting position of substrings but it does not keep their frequencies. Therefore, this paper uses frequent suffix trie or FST and frequent max substring mining technique [14] to find the long substrings that are likely to be the key terms of the web page contents as well as their frequencies and list of positions. This is mainly based on the assumption that the substrings that occur frequently in the web page contents should be the important keys of web pages.

The following example illustrates the construction of the FST structure representing all substrings of Thai string using the traditional algorithm to enumerate all suffixes of the string.

Let string *T* =

ก า ร ป ร ะ ก อ บ ก า ร $

1) Append $ to the string and define the position of each character in the string.

String *T*:	ก	า	ร	ป	ร	ะ	ก	อ	บ	ก	า	ร	$
Position (.pos):	1	2	3	4	5	6	7	8	9	10	11	12	13

2) Enumerate all suffixes of the string

1. การประกอบการ$
2. ารประกอบการ$
3. รประกอบการ$
4. ประกอบการ$
5. ระกอบการ$
6. ะกอบการ$
7. กอบการ$
8. อบการ$
9. บการ$
10. การ$
11. าร$
12. ร$
13. $

3) All suffixes are used to create the FST structure, as shown figure 1.

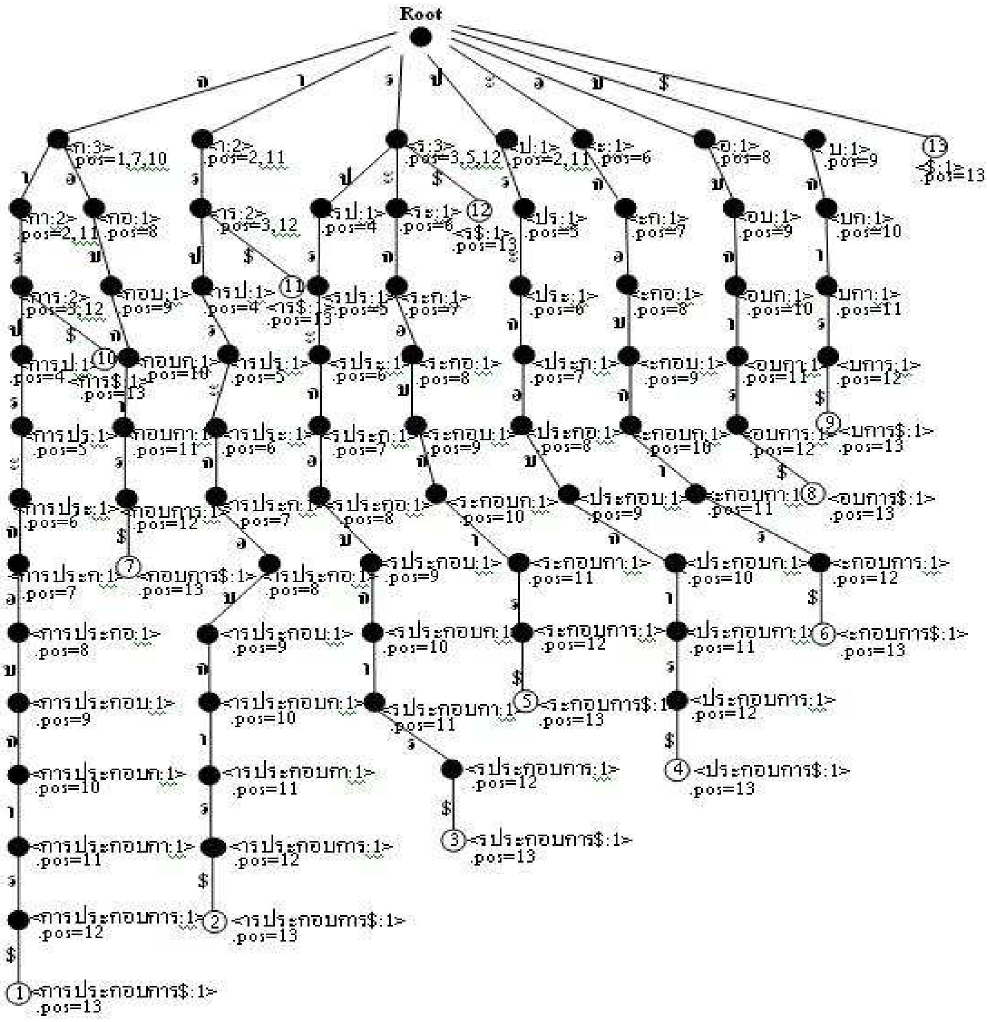

Fig. 1. The FST structure for string T = ก า ร ป ร ะ ก อ บ ก า ร $

From Figure1, frequent max substrings are obtained by the following steps.

1. Enumerate all substrings of string *T* using the FST structure.
2. Search the set of frequent substrings whose frequencies are greater or equal to the frequency threshold from the FST structure.
3. Find the frequent max substrings from frequent substrings set.

From the above algorithm, it is clear that to find all frequent max substrings, we must firstly enumerate all substrings, follow by searching the set of frequent substrings and frequent max substrings. To do this, a large memory has to be used to keep all substrings. In order to reduce the amount of memory used, we reduce the number of substrings by using the following two reduction rules: (1) reduction path rules using the given frequency to check search termination, (2) reduction path rules using super-substring definition.

B. *Efficient algorithm [27]*

This algorithm uses the two reduction path rules to reduce the number of substrings. It also uses heap data structure to support computation. As an example, the steps for finding frequent max substrings for a given string T with the frequency 2 using this algorithm is given below.

Let string T = ก า ร ป ร ะ ก อ บ ก า ร $

And the given frequency= 2

1. Enumerate the *1*-length substrings with their frequencies, and then select frequent substrings. Substrings, frequency and position transaction (.pos) are kept in Min Heap structure sorted by order of occurrence in the string. The prior substrings can have more frequent max substring than later substrings.

2. Enumerate the child substrings of a substring in Min Heap to process, and select only frequent child substrings. Min Heap structure will then be updated by using deletion rule. The deletion rule checks which child substrings are super-substrings of substrings in the Min Heap structure. If the frequency of substring in Min Heap minus the frequency of super-substring is less than defined frequency, the substring will be deleted from the Min Heap structure and frequent child substrings are inserted in Min Heap by considering two rules; (1) substring will be inserted to Min Heap structure ordered by the occurring position on string *T*, (2) if the first position of the substring is equal to the first position of an existing substring in Min Heap, a substring is inserted in the last position in the same group. The processed substrings are deleted from Min Heap. The other substrings will be processed until Min Heap is empty.
3. Mine frequent max substring by selecting substrings having no super-substrings from set of substrings in Min Heap.

From above steps, the FST structure is shown figure 2

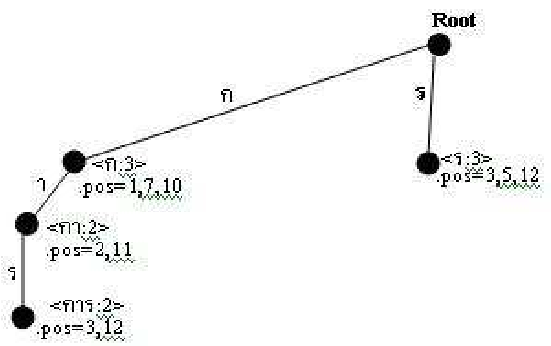

Fig. 2. The FST structure using the efficiency algorithm.

Figure 2 shows the FST structure that contains some frequent substrings and all frequent max substrings.

V. Conclusion

This paper proposes using FM algorithm to extract index terms from Thai texts. As the amount of electronically stored information in Thai language has been growing rapidly in the past few years, information extraction (IE) and information retrieval (IR) for Thai language are becoming more and more important. IE has become one of the most essential techniques for extracting index terms from Thai texts in information retrieval. Our algorithms based on information extraction that will be able to specify index terms using less storage space to support constructing the index. The algorithm is aimed for Thai language, but it can also be used with any language in which words are not explicitly delimited. Our algorithm uses an improved suffix trie structure, called FST structure. FST structure is exploited to reduce the number of computations using two rules. This has improved over the original algorithm in term of memory storage space, computing efficiency and scalability.

References

[1] V. Sornlertlamvanich, T. Potipiti and T. Charoenporn. Automatic Corpus-Based Thai Word Extraction with the C4.5 Learning Algorithm. In forthcoming Proceedings of COLING 2000.

[2] RattasitSukhahuta and Dan Smith. Information Extraction for Thai Documents. International Journal of Computer Processing of Oriental Languages (IJCPOL), 2001, 14(2):153–172.

[3] ChoochartHaruechaiyasak, PrapassSrichaivattana, SarawootKongyoung and Chaianun Damrongrat. Automatic Thai Keyword Extraction from CategorizedText Corpus, 2008.

[4] Asanee Kawtrakul, Chalathip Thumkanon, and Paul McFetridge. Automatic multilevel indexing for Thai text information retrieval. In IEEE Asia Pacific Conference on Circuits and Systems, Chiangmai, Thailand, December 1998.

[5] G. Salton. and M. J. McGill. Introduction to Modern Information Retrieval. McGraw Hill Book Co., New York, 1983.

[6] Haruechaiyasak, C., Kongyoung, S., and Dailey, M.N., A Comparative Study on Thai Word Segmentation Approaches. In Proceedings of Electrical Engineering/Electronics, Computer, Telecommunications, and Information Technoly, 2008.

[7] SuraphanMeknavin, Paisarn, Charoenpornsawat, and BoonsermKijsirikul. Feature-based Thai Word Segmentation. In Proceedings of the Natural Language Processing Pacific Rim Symposium (NLPRS'97), Phuket, Thailand (1997).

[8] Wirote Aroonmanakun. 2002. Collocation and Thai word segmentation. In Proceedings of the 5th SNLP & 5th Oriental COCOSDA Workshop, pages 68–75.

[9] Krit Kosawat, Méthodes de segmentation et d'analyse automatique de textes thaï Automated methods of segmentation and analysis of Thai texts, Thése pour obtenir le grade de Docteur de l.Université de Marne-La-Vallée, le 8 septembre 2003.

[10] Luksaneeyanawin S. A Thai Text to Speech System. In Proceedings of the Conference on Electronics and Computer Research and Development. NECTEC. 1992.

[11] V. Sornlertlamvanich, "Word Segmentation for Thai in Machine Translation System," Machine Translation, National Electronics and Computer Technology Center, Bangkok.

[12] Y. Poovorawan and V. Imarom, "Dictionary-based Thai Syllable Segmentation (in Thai)," 9th Electrical Engineering Conference, 1986.

[13] Y. Thairatananond, "Towords the design of a Thai text syllable analyzer". Master Thesis Asian Institute of Technology.

[14] S. Charnyapornpong , "A Thai Syllable Separation Algorithm. Master Thesis Asian Institute of Technology", 1983.

[15] Theeramunkong, T., Sornlertlamvanich, V., Tanhermhong, T., Chinnan, W., Character-Cluster Based Thai Information Retrieval, Proceedings of the Fifth International Workshop on Information Retrieval with Asian Languages, September 30 - October 20, 2000, Hong Kong, pp.75-80.

[16] ChoochartHaruechaiyasak, SarawootKongyoung and ChaianunDamrongrat, "LearnLexTo: A Machine-Learning Based Word Segmentation for Indexing Thai Texts", In ACM 17th Conference on Information and Knowledge Management, 2008.

[17] C. Kruengkrai and H. Isahara, "A conditional random field framework for thai morphological analysis," Proc. of the Fifth Int. Conf. on Language Resources and Evaluation (LREC-2006), 2006.

[18] Cavnar, W. and Trenkle, J. 1994. N-gram based text categorization. In Proceedings of the 3rd Annual Symposium on Document Analysis and Information Retrieval (Las Vegas, NV, 1994), 161—175.

[19] PMajumder, M Mitra, B.B. Chaudhuri, "N-gram: a language independent approach to IR and NLP", International conference on Universal Knowledge-2002.

[20] Jaruskulchai C., An Automatic Indexing for Thai Text Retrieval, Ph.D. Thesis, George Washington University, U.S.A., Aug 1998.

[21] Joon Ho Lee and JeongSooAhn, "Using n-Grams for Korean Text Retrieval," In Proc. Int'l Conf. on Information Retrieval, ACM SIGIR, Zurich, Switzerland,pp. 216–224, 1996.

[22] Kwok, K.L. 1997. Comparing representations in Chinese information retrieval. Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Philadelphia, Philadelphia, July 1997, pp. 34-41.
[23] Fujii, Hideo and W. Bruce Croft. A Comparison of Indexing Techniques for Japanese Text Retrieval.In Proceedings of ACM SIGIR International Conference on Research and Development in Information Retrieval, 237-246, 1993.
[24] P. McNamee, "Knowledge -light Asian. Language. In Proceedings of the Third NTCIR Workshop on research in information Retrieval, Automatic Text Summarization and Question Answering Text Retrieval, at the NTCIR-3. Workshop, 2002.
[25] M. S. Kim, K. Y. Whang, J. G. Lee, and M. J. Lee, "n-Gram/2L: A Space and Time Efficient Two-Level n-Gram Inverted Index Structure," in VLDB, Trondheim, Norway, 2005, pp. 325-336.
[26] M.M. Hasan and Y. Matsumoto, "Chinese-Japanese Cross Language Information Retrieval: A Han Character Based Approach", In Proceedings of the SIGLEX Workshop on Word Senses and Multi-linguality, pp.19-26, ACL-2000, Hong Kong, 2000.
[27] T. Chumwatana, , KokWai Wong and Hong Xie, Frequent max substring mining for indexing. International Journal of Computer Science and System Analysis (IJCSSA), India, 2008.
[28] J. Vilo, "Discovering Frequent Patterns from Strings: Department of Computer Science. University of Helsinki, Finland," Technical Report C-1998-9, p. 20, May 1998.
[29] D. R. Morrison, "PATRICIA - practical algorithm to retrieve information coded in alphanumeric," Jrnl. A.C.M., pp. 15(4):514-534, 1968.
[30] D. Gusfield, Algorithms on Strings, Trees and Sequences Computer Science and Computational Biology. Cambridge: Cambridge University Press, 1997.