Automatic Indexing for Thai Text with Unknown Words using Trie Structure *

Witoon Kanlayanawat and Somchai Prasitjutrakul
Department of Computer Engineering
Chulalongkorn University
Bangkok 10330, Thailand
Phone : (66-2)-218-3743, Fax : (66-2)-215-3554, E-Mail : somchaip@computer.org

Abstract

In this paper, we present an automatic indexing for Thai text retrieval system where given documents can have words that are unknown to the system's dictionary. Trie structure is used as a main file structure for indexing which supports word, pattern, and approximate searching. Since there is no explicit inter-word delimiter in Thai text, we also propose a word segmentation algorithm that segments a given text to a set of words and *sistrings* (semi-infinite strings) of unknown words for adding to the trie. The algorithm first finds a set of words maximally matching all the sistrings of a given text. Then it constructs an overlapping graph whose shortest paths represent a smallest list of words minimizing unknown strings of the text. By using our proposed dictionary-based word segmentation algorithm which can deal with unknown words, along with the use of trie structure to store the index, precision and recall of the retrieval can be enhanced.

1. Introduction

Text retrieval has become one of the most essential tools for managing information as computer-generated documents get published and computers get connected locally and globally, especially with the advent of the World Wide Web and CD-ROM. Traditionally, there are four major techniques, full text scanning, inversion, signature file, and clustering for text retrieval. A detail survey of these technique can be found in [2], [3], [10], [14]. Full text scanning locates the documents by searching through all documents for the specified string pattern. The technique does not need additional space for index but the search time is linearly proportional to the size of text. In inversion technique, keywords for all the documents are stored in alphabetical order in an index file (which can be implemented using sorted array, B-tree, or trie structures). For each keyword, there is a list of pointers (kept in a posting file) to associated documents. The technique is easy to implement and gives fast response time. However, it does suffers from the high storage overhead for the index. In signature file technique, each document has an associated *signature* which is a bit string created by using hashing on its words and superimposed coding. Then signatures of all the documents are stored in a separate file called signature file which is much smaller than the original document files and thus a search for signature pattern is much faster than a search for string pattern. Since it uses a signature as a representation of a document, it introduces a notion of *false hit* where a matched signature does not always mean that the corresponding string pattern matches. In clustering technique, similar documents are grouped together to form clusters to improve the efficiency and effectiveness of retrieval.

* This research was supported by the Thai Government Research Fund.

When building the database for text retrieval, text in each document is segmented into words which are optionally compared against a stoplist (a list of words having no index value). Non-stoplist words are then stemmed so that variations of the same words are represented with only one pattern. Then, each of the stemmed word is assigned a weight used during the search to rank retrieved documents. Finally, stemmed words along with their weights and locations are kept into the database.

Segmenting a given text into words is a nontrivial task in Thai (and other Asian languages) text processing since there is no explicit inter-word delimiter. There are currently two approaches in Thai word segmentation, rule-based and dictionary-based approaches. The rule-based approach uses a set of extensively studied rules for Thai syllable [1],[13],[15]. Although the technique achieves high precision in syllable segmentation, it is not suitable for indexing application since it is word not syllable that is required to create the index. The dictionary-based approach matches words in dictionary or lexicon with a given text. The matching can use longest-word-matching greedy strategy [16], least-number-of-matched-words strategy [17], or statistical data of word tags [7]. The word segmentation gets more complicated if the text to be segmented contains some unknown words (we define the unknown words to be words not kept in the dictionary being used) which can be proper names, transliterated words, words with spelling error, etc.

In this paper, we present an automatic indexing for Thai text retrieval system. By using our proposed dictionary-based word segmentation algorithm which can deal with unknown words, along with the use of trie structure to store the index, precision and recall of retrieval can be enhanced. The word segmentation algorithm constructs an overlapping graph whose shortest path represents a smallest list of words minimizing unknown strings of the given text. The words and unknown strings obtained then determines corresponding locations of sistrings (semi-infinite strings) to be kept in a trie structure. Trie structure is reviewed in Section 2. Detail of our word segmentation algorithm is described in Section 3. Section 4 summarizes the entire automatic indexing and retrieval. Conclusion of the paper is given in Section 5.

2. Trie Structures

Tries have been used for indexing large texts [4],[8],[9]. Tries are trees whose edges represent letters of the alphabet encoding the data. Therefore a word is represented as a path from root to leaf. Tries support efficient prefix searching (prefix searching is a searching for any word matching a given prefix) since all words sharing a prefix share the same path from root to an internal nodes. In addition, approximate matching can also be efficiently performed in trie structure which is applicable for searching text with error [12]. Implementations of trie structures on secondary storage can be found in [9],[11]. In this work, we use trie structures to store both the dictionary and index of all of the documents.

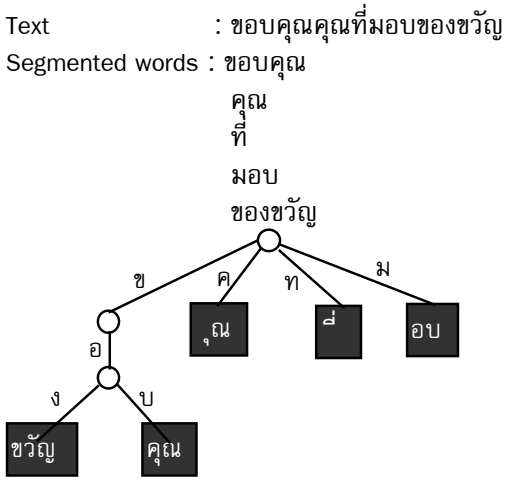

Figure 1. An example of a trie

Figure 1 shows an example of a trie consisting of segmented word from a text. Locations of the words are stored in its corresponding external node of the trie. When searching for a word, we follow branches determined from characters of the word. For example, to search for ขอบคุณ, we first go to the left branch (ข), go down (อ), and then go to the

right branch (บ) where we hit an external node whose content matches characters of the rest of the word (คุณ). Therefore the search time is proportional to the word length.

Gonnet [5] defined a semi-infinite string, or *sistring*, to be a suffix of the text starting at some position. Figure 2 shows the first four sistrings of a given text. A text of length *n* can have at most *n* sistrings. All of the sistrings can be stored in a trie where the search procedure remains the same as described before. However, it is obvious that we can eliminate many useless sistrings whose starting locations are not the beginning of the word. In [6], sistrings of Thai text whose starting location can potentially be words (using word formation rule-based approach) are stored in a PAT tree [5] (which is a special binary trie). Although, the concepts of storing sistring gives high recall (i.e., it is unlikely to miss any occurrences of the given search word), but the precision of retrieval will be low if we keep too much useless sistrings (i.e., many retrieved documents are not what we actually want) since the search acts like a pattern matching without knowing the notion of word. For example, the phrase in Figure 2 will match the search word อบ in spite that it is not a word in the phrase.

Text : ขอบคุณคุณที่มอบของขวัญ
Sistrings : ขอบคุณคุณที่มอบของขวัญ
: อบคุณคุณที่มอบของขวัญ
: บคุณคุณที่มอบของขวัญ
: คุณคุณที่มอบของขวัญ

Figure 2 Examples of sistrings

However, sistring can be used where we can not determine word boundaries as will be presented in the next section.

3. Word Segmentation Algorithm

Given a text, our word segmentation algorithm uses dictionary-based approach to find a smallest list consisting of known words and minimal unknown words. The known words are directly stored in the trie whereas the unknown words are stored in the trie as a set of sistrings whose starting locations are determined to potentially be syllable boundary using rule-based approach.

Let T be a given text to be segmented, T_i be a sistring of T starting at the i-th character, $T_{i,j}$ be a substring of T consisting of the i-th thru j-th characters, and D be a dictionary.

The algorithm consists of four steps as follows :

1. For each T_i , i = 1,...,*n*, find a word, w_i , in D satisfying the following conditions :

   - w_i maximally matches T_i . Let i′= i–1+ length of w_i. Therefore $w_i = T_{i,i'}$.
   - w_i is not a substring of any w_j where j< i

   For example, T = นายเจมส์มาร์ตินต้องการผลิตรายการโทรทัศน์ we have w_i's as shown in Figure 3. (Other sistrings of T not shown in the figure have their w_i's equal to null strings.)

i	T_i	w_i
1	นายเจมส์มาร์ตินต้องการผลิตรายการโทรทัศน์	นาย
4	เจมส์มาร์ตินต้องการผลิตรายการโทรทัศน์	เจ
5	จมส์มาร์ตินต้องการผลิตรายการโทรทัศน์	จม
9	มาร์ตินต้องการผลิตรายการโทรทัศน์	มาร
13	ตินต้องการผลิตรายการโทรทัศน์	ติ
16	ต้องการผลิตรายการโทรทัศน์	ต้องการ
20	การผลิตรายการโทรทัศน์	การผลิต
24	ลิตรายการโทรทัศน์	ลิตร
26	ตรายการโทรทัศน์	ตรา
27	รายการโทรทัศน์	รายการ
33	โทรทัศน์	โทรทัศน์

Figure 3 Example shows w_i's for sistrings

2. Construct an overlapping graph (which is a weighted directed graph) G =(V,E) where

- V = {w_i | w_i obtained from step 1, $w_i \neq \varnothing$, $1 \leq i \leq n$ }

- E = { (w_i, w_j) | w_i is adjacent to or overlap with w_j , i < j } weight of an edge (w_i, w_j) is determined from cases shown in Table 1 :

The first case is the most favorable one where two words are adjacent. In case 2, we have two overlapping words which can be further segmented yielding segmentations with no unknown strings. For example, ต้องการผลิต (w_1 = ต้องการ and w_4 = การผลิต) can be segmented to ต้อง + การผลิต or ต้องการ + ผลิต. Case 3 represents cases where two overlapping words can be segmented yielding some segmentations having unknown strings . For example, เพื่อนำ (w_1 = เพื่อน and w_5 = อนำ) can be segmented to เพื่อ + นำ, เพื่อน + ำ, or เพื่ + อนำ. The last case (with the highest weight) represents cases where two overlapping words can be segmented yielding segmentations with unknown strings. Notice that cases 3 and 4 are used to handle text with errors. For example, เพื่อนำ may actually be เพื่อนทำ where the ท is missing in the text. In this situation, case 3 keeps the เพื่อน in addition to เพื่อ and นำ.

Table 1. Edge weighting for overlapping and adjacent words

#	Weights of (w_i, w_j)	Conditions
1	1	if i′+1 = j, i.e., w_i is adjacent to w_j
2	10	if $j \leq i'$ and $T_{i, j-1}$ and $T_{i'+1, j}$ are all in the dictionary.
3	100	if $j \leq i'$ and there exists a position k, $j-1 \leq k \leq i'$, such that both $T_{i, k}$, and $T_{k+1, j'}$ are in the dictionary.
4	1000	otherwise

From the example in step 1, we can construct the corresponding overlapping graph as shown in Figure 4.

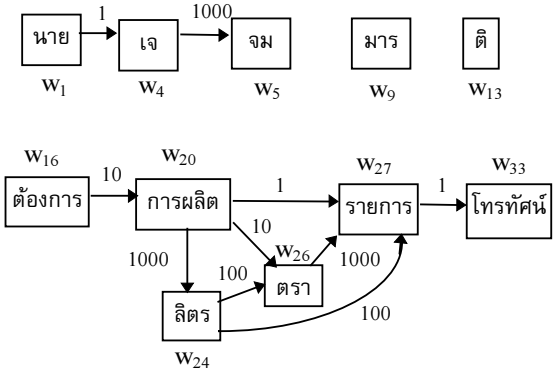

Figure 4 The overlapping graph for a text นายเจมส์มาร์ตินต้องการผลิตรายการโทรทัศน์

3. For each component of the graph, find a shortest path from the leftmost to the rightmost nodes of the component. Let W = {w_i | w_i has its corresponding node on the shortest paths obtained }. From the above example, we get W = {w_1, w_4, w_5, w_9, w_{13}, w_{16}, w_{20}, w_{27}, w_{33} }

4. In this step, we determine W′, a set of segmented words, and U′, a set of sistrings for unknown words as follows.

   4.1 Let U be a set of unknown strings. U can be determined as follows :

      4.1.1 For each edge (w_i, w_j) with weight of 1000 on the shortest path (these are the two unknown strings discussed in case 4 of Table 1), we add $T_{i,j-1}$ and $T_{i'+1,j'}$ to U. From the above example, they are เ and ม (determined from edge (w_4, w_5)).

      4.1.2 For each pair of nodes w_i and w_j belonging to different components of G, i < j, and there is no $w_k \in W$, where i < k < j, we add $T_{i'+1, j-1}$ to U. From the above example, they are ส์, ์, and น.

4.1.3 We concatenate each string $T_{j,k}$ obtained from the two steps above to a word $w_i \in W$ which is adjacent to the left of $T_{j,k}$ in T. From the above example, U = { นายเ, เจม, จมส์, มาร์, ติน }.

4.1.4 This step combines any strings in U that overlap or adjacent to each others. From the above example, U= {นายเจมส์มาร์ติน }.

Then, U′ consists of sistrings T_i , T_{i+1},..., T_k of string $T_{i,k} \in U$. This step deals with unknown words which can be proper names, transliterated words, words with spelling error, etc. These words usually can be segmented to known words with some unknown strings in between. Since we do not know the exact word boundary of these unknowns, they are kept as sistrings in the trie. (Actually, we also apply syllable segmentation rules to eliminate some useless sistrings from the set U′.)

4.2 Let W* = W – {w_i | w_i being used to form unknown string in step 4.1.3}. Starting with W′ = W* , for any w_i and w_j of W*

- satisfying case 2 in Table 1, we add $T_{i,j-1}$ and $T_{i'+1,j'}$ to W′. From the above example, they are ต้อง and ผลิต determined from w_{16} and w_{20}.

- satisfying case 3 in Table 1, we add $T_{i,k}$, and $T_{k+1,j'}$ to W′ where k is defined in Table 1.

From the word segmentation algorithm just described, we obtained W′, a set of known segmented words, and U′, a set of sistrings of unknown words, to be kept in the trie. From the above example given a text T = นายเจมส์มาร์ตินต้องการผลิตรายการโทรทัศน์, we obtain W′ = {ต้องการ, การผลิต, รายการ, โทรทัศน์, ต้อง, ผลิต} and U′ = { T_i | T_i is sistring of string นายเจมส์มาร์ติน in T } to be stored in the index trie.

Performances of the four steps of the algorithm are as follows. Step 1 maximally matches each sistring of T to words in dictionary. There are *n* sistrings of T (*n* is the number of characters in T). Since our dictionary is also kept in a trie structure. Then it takes O(*nk*) where *k* is the maximum length of word in the dictionary. Step 2 constructs graph G=(V,E) which takes O(|V|+|E|). The worst case (happens when there are *n* w_i's where all of them overlap each other which is very unlikely) is O(n^2). Step 3 finds a shortest path for each component which is O(n^2). And the last step, re-scanning the string for unknown strings, takes O(*n*). Therefore the word segmentation algorithm takes worst case time of O($nk + n^2$) or O(n^2) where $k < n$ for a long sentence.

4. Thai Text Retrieval

From the trie structure along with the word segmentation algorithm presented, documents can be indexed by first parsing the document for explicit sentence or phrase delimiters e.g., space, then these sentences or phrases are segmented to obtain a set of words and a set of sistrings of unknown words. The two sets can be fed to other text processing modules such as word filtering, stoplist, stemming, and weight assignment before adding to the index trie.

For the retrieval phrase, user's query words must be segmented using the same algorithm so that an obtained set of words and sistrings are used for querying the trie structure.

5. Conclusion

An automatic indexing for Thai text retrieval system was presented in this paper where given documents can have words that are unknown to the system's dictionary. Unknown words can be misspelled words, proper words, transliterated words, etc. The fundamental file structure used for keeping index is trie which supports both

word, pattern, and approximate matching. A word segmentation algorithm was also proposed for segmenting a given text to a set of words and sistrings of unknown words for adding to the trie. The algorithm finds a smallest set of words and sistrings with the objective of minimizing number of sistrings to enhance retrieval precision. While the use of sistring for unknown words enhances the retrieval recalls.

References

[1] S. Charnyapornpong, "A Thai Syllable Separation Algorithm," M.Eng. Thesis, Asian Institute of Technology, Aug. 1983.

[2] C. Faloutsos and D.W. Oard, "A Survey of Information Retrieval and Filtering Methods," Technical Report CS-TR-3514, University of Maryland, College Park, August 1995.

[3] W. B. Frakes and R. Baeza-Yates eds., *Information Retrieval : Data Structures and Algorithms*, Englewood Cliffs, N.J. : Prentice-Hall.

[4] G. Gonnet, "Unstructured Data Bases or Very Efficient Text Searching," *ACM PODS*, vol. 2, pp. 117-124, 1983.

[5] G. Gonnet, R. Baeza-Yates, and T. Snider "New Indices for Text: PAT Trees and PAT Arrays," in *Information Retrieval : Data Structures and Alforithms*, ed., W. B. Frakes and R. Baeza-Yates, Englewood Cliffs, N.J. : Prentice-Hall

[6] P. Jindavimonlert, "A Thai Text Retrieval System using the PAT tree," M.Sc. Thesis, Department of Computer Engineering Chulalongkorn University, 1996.

[7] A. Kawtrakul, C. Thumkanon, and S. Seriburi, "A Statistical Approach to Thai Word Filtering," Proc. of the second Symposium on Natural Language Processing, pp. 398-406, 1995

[8] U. Manber and G. Myers, "Suffix Arrays: A New Method for On-line String Searches," *First ACM-SIAM Symp. on Discrete Algorithms*, pp. 319-327, San Francisco, 1990.

[9] T.H. Merrett and H. Shang, "Trie Methods for Representing Text," *Proc Fourth Int'l Conf., FODO'93*, LNCS 730, pp. 130-145, Chicago: Springer-Verlag, Oct. 1993.

[10] G. Salton and M.J. McGill, *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983

[11] H. Shang, "Trie Methods for Text and Spatial Data on Secondary Storage," Ph.D. Dissertation, School of Computer Science, McGill University, Nov. 1994.

[12] H. Shang and T.H. Merrett, "Tries for Approximate String Matching," *IEEE Trans. on Knowledge and Data Eng.*, Vol. 8, No. 4, pp. 540-547, Aug. 1996.

[13] D. Sintupunpratum and C. Bandhitanont, "Thai Word Processing (in Thai)", Proc. of the second Symposium on Natural Language Processing in Thailand, pp. 322-376, March 1993.

[14] I.H. Witten, A. Moffat, and T.C. Bell, *Managing Gigabytes : Compressing and Indexing Documents and Images*, N.Y., Van Nostrand Reinhold.

[15] D. Sawamibhadhi, "Implementation of Thai Grammar Analysis Software under UNIX system (in Thai)", Thammasart Univ., 1990.

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

[17] V. Sornlertlamvanich, "Thai Word Segmentation in Language Translation System," *Computerized Language Translation* (in Thai), p. 50-55, 1993.