THAI TEXT SEGMENTATION USING VOWEL- CENTERED RULES AND LEARNING

Patrawadee Tanawongsuwan*

National Institute of Development Administration
118 SeriThai Road, Bangkapi, Bangkok 10240 Thailand

ABSTRACT

A vast majority of text processing algorithms make one common assumption that input text is a sequence of *words*. In some language in which word boundaries are not always explicit, such as, Thai, text segmentation is an issue of interest. This work presents a two-step algorithm for Thai text segmentation. The first step chops the input text into pieces centered around the vowels. In the second step, the algorithm defines a set of features that might help determine whether or not two consecutive pieces from the previous step belong together as a unit (word, syllable, etc). It then uses learning algorithms to build a model out of these features. Given an input text, applying this model will result in a sequence of units, each small (few syllables) yet useful enough for further processing by other word-based algorithms.

KEYWORDS: Thai, Text, Segmentation, Learning, Decision Trees, C4.5

1. INTRODUCTION

In this so-called information age, an enormous amount of information is discovered, stored, transferred, and processed in various ways. In order to take advantages of such information, of which a vast portion is in text format, a number of text processing algorithms have been proposed. One assumption often made by many of these algorithms is that input text is a string of words, each delimited by space. In some written languages, such as, Chinese, Japanese, Korean, and Thai, there are not always explicit word boundaries. For instance, in Thai language, the string นายแดน consists of two words, นาย (Mr.) and แดน (Dan). An indexing algorithm that makes the space-delimit assumption will treat นายแดน as a single index term, and therefore will not be able to match the one-word query แดน. To resolve this problem by indexing individual characters might be too expensive. An alternative solution is to break input text into individual words or some other units larger than individual characters but yet small enough to be useful. This paper describes an algorithm that performs such task.

Previous work

One of the earlier works in Thai word segmentation is [1]. Observing how frequent a given pair of characters occurs adjacently in Thai words, a set of rules is proposed for segmenting given input text into individual units, which is either one syllable or few consecutive ones that would be grammatically incorrect or least probable when written separately. [2] outlines another rule-based method inspired by characteristics of individual characters.

A dictionary-based method is proposed in [3]. Starting from the first character of the input text, the algorithm picks up one additional character at a time as long as the resulting substring appears in the dictionary; otherwise, it marks a segmentation point. The algorithm backtracks when the remaining input does not make a word in the dictionary. [2] also briefly discusses dictionary-based methods.

Corpus-based methods take advantages of the vast amount of text by using it to build a model. [4], [5], [6], and [7] are examples of this approach.

* Corresponding author.
E-mail: patrawadee@as.nida.ac.th

2. TEXT SEGMENTATION USING VOWEL-CENTERED RULES AND LEARNING

2.1 Thai Characters

Thai character set consists of vowels, tonal marks, consonants, numbers, and special symbols.

(a)	-ะ	-ั-	-า	-ำ	-ิ	-ี	-ึ	-ื	-ุ	-ู
	เ-	แ-	โ-	ใ-	ไ-	-็	-์			
(b)	เ-ะ	แ-ะ	โ-ะ	เ-า	เ-็-	แ-็-	เ-ิ-	เ-าะ		
(c)	-ัวะ	เ-ีย	เ-ียะ	เ-ือ	เ-ือะ	เ-อะ				
(d)	- -	-อ	-ว-							

Figure 1: (a) Vowel character set ('-' indicates a consonant placeholder); (b) Examples of vowels with two or more vowel characters; (c) Examples of vowels with mandatory consonants; (d) Examples of implicit vowels

There are thirty-two vowels, made of characters from the set shown in figure 1(a). Each character in 1(a) can be used in isolation as a complete vowel. Some combinations of them make up other vowels, shown in 1(b). Some vowels, shown in 1(c), require certain consonant characters. And some are made up of only consonant characters as shown in 1(d). Some vowels take different forms depending on the existence of the trailing consonant. For instance, \mä\ and \mäb\ are spelled แมะ and เมือบ, respectively. Note the differences between **vowel character** and **vowel** in this paper. While the former refers to individual characters, the latter refers to one or more of them that together make a sound. While those in figure 1(a) are vowel characters, each is a single-character vowel as well. Those in figure 1(b), 1(c), and 1(d) are vowels, made up of vowel characters and, in some cases, consonant characters.

Though the special symbol -์ is technically not a vowel, it is treated as one by this algorithm since its usage is similar to that of vowel characters'.

Thai pronunciation has five different tones. A syllable may have one or none of the four tonal marks (◌่ ◌้ ◌๊ ◌๋).

Forty-four characters comprise the consonant set. In Pali, a language by which Thai is partially influenced, consonants are systematically arranged in rows and columns. Using similar arrangement for Thai characters, thirty-three **row consonants** are arranged into five rows (rows 1-5 in figure 2) and five columns. The rows differ by places of articulation. In pronunciation of consonants from rows 1 through 5, the air stream coming from your vocal folds is altered by the throat, roof of mouth, alveolar, teeth, and lips, respectively. The columns differ by articulation, tone, and tonal mark usage. Column-1 consonants have exclusive use of two particular tonal marks, ◌๊ and ◌๋. Column-2 consonants have higher tone in their basic pronunciation. Consonants in columns 3 and 4 differ by their usage. In particular, in strict Pali spelling, those in column 3 can be used as trailing consonants while those in column 4 can not. Those in column 5 are nasal consonants.

The other eleven **non-row consonants** are not categorized by the criteria above. However, for the purpose of this work, they are arranged into rows (rows 6-9 in figure 2), based on their sound as trailing consonants (d/n, y, w, none), and columns, based on the criteria used for row consonants.

Though the characters ฦ and ฦๅ are technically vowel characters, they are treated as consonant characters by this algorithm since their usage is similar to that of consonant characters'. They are thus arranged into this table in row 10, column 0.

	1.med	2.hi	3.low	4.low	5.low
1. guttural (throat)	ก	ข, ฃ	ค, ฅ	ฆ	ง
2. palatal (roof of mouth)	จ	ฉ	ช, ซ	ฌ	ญ
3. dental (alveolar)	ฏ, ฎ	ฐ	ฑ	ฒ	ณ
4. dental (teeth)	ด, ต	ถ	ท	ธ	น
5. labial (lips)	บ, ป	ผ, ฝ	พ, ฟ	ภ	ม
6. d/n		ศ, ษ, ส			ร, ล, ฬ
7. y					ย
8. w					ว
9. (none)	อ	ห		ฮ	
10. special characters ฤ, ฦ					

Figure 2: Consonant character set

Numbers and special symbols (๑, ๒, ๓, ๆ, ฯ, ., ?, -, etc) are least relevant to the understanding of this paper.

Thai characters are written left to right in four levels (figure 3). Level 1 is for tonal marks. Vowels are at levels 2 through 4. Consonants are always at level 3. Tonal marks, upper, and lower vowels must share a column with exactly one consonant.

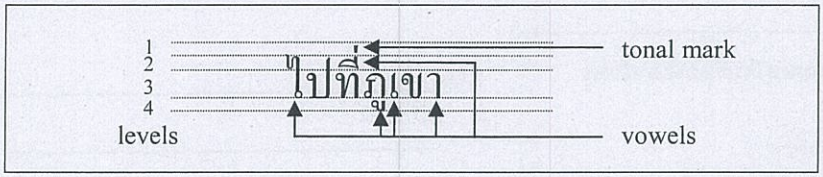

Figure 3: Thai characters written left to right in four levels

A Thai syllable is made up of at least one vowel and one consonant. In other words, a vowel can only be pronounced when coupled with a consonant, and vice versa (with very few exceptions).

2.2 Text Segmentation Algorithm

Given a string of characters, how might it be divided into sensible and meaningful units? This work proposes an algorithm that segments given input text into units, each having one or few syllables. The algorithm performs two passes through the text, each having its own steps and rationale behind.

Preprocessing

This step takes an input text and marks all **hard boundaries**. We define hard boundary as a location where it is impossible for items on its both sides (characters, in this case) to merge into a word. Hard boundaries are anything that is neither a vowel, consonant, nor tonal mark, such as, white space, numbers, special symbols, English characters, etc. Consecutive occurrences of hard boundaries may be combined into one. Figure 4(a) and 4(b) show an input text before and after it has been preprocessed.

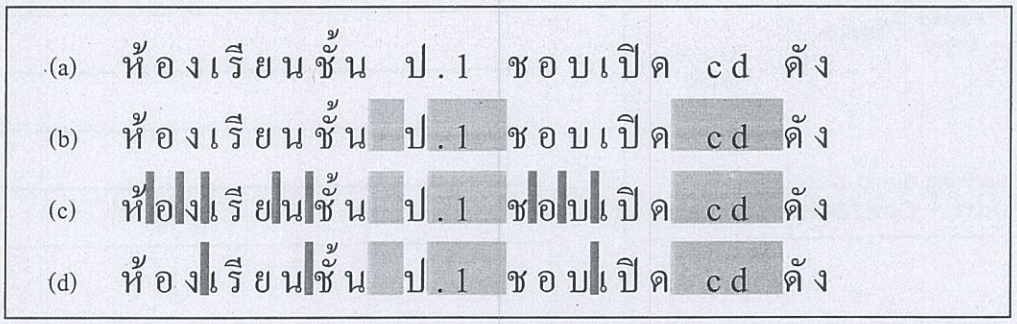

Figure 4: (a) An input text; (b) Result of preprocessing step (hard boundaries are in light grey); (c) Result of first pass (soft boundaries are in dark grey); (d) Result of second pass

First Pass: Chopping

This step is motivated by what we have mentioned: a vowel alone does not make a syllable; it needs at least one consonant. Based on this idea, we process our text by segmenting it based on occurrences of the vowels. Scanning the input text, whenever the algorithm finds a vowel character, it is extracted along with its neighboring characters that together make up as few syllables as possible. The vast majority of the result will be one-syllable instances. Few multiple-syllable instances do exist. For example, เฉพาะ (\ch&-pä\) can not be extracted as เฉ and พาะ because the -าะ is not a valid vowel combination.

When a vowel character *v* is found, determining the boundary of extraction is based on Thai spelling rules. Always extracted is the **leading consonant**(s), which is not always positioned in front of *v*. In simple cases, the leading consonant(s)' position can be determined based on whether *v* is a front, rear, upper, or lower vowel. For instance, if *v* is a front vowel, the consonant to its immediate right is the leading consonant. In other cases in which *v* is part of a multiple-character vowel, such as, those in figure 1(b) and 1(c), the algorithm determines first the vowel combination, and then the leading consonant(s) accordingly. In either case, the **trailing consonant** is extracted if and only if it is mandatory by the vowel. For instance, บิด (\bid\) is extracted as บิ and ด since the vowel -ิ does not require a trailing consonant; on the other hand, บัด (\b&d\) is extracted as one unit since the vowel -ั does require a trailing consonant.

Tonal marks, which always share a column with exactly one consonant, always accompany its consonant.

The results up to this point are clusters of characters, each having at least one vowel character, at least one consonant, and possibly a tonal mark. However, there are some characters that have not been examined, namely, those consonants not belonging to any cluster. Take each of these **loose consonants** (along with its corresponding tonal mark, if one exists) and label it a cluster as well. Let us call each of the clusters created so far **D1**, be it the earlier ones containing vowels or the later ones containing only consonants. Mark all locations between D1's as **soft boundaries**, which are locations where it is possible for items on its both sides (D1's, in this case) to merge into a word.

Figure 4(c) shows the result of this step.

Algorithm
- Preprocess: Mark hard boundaries
- First pass: Mark soft boundaries
  - For each vowel, determine its leading consonant(s), trailing consonant (if required), and tonal mark. Extract them as one unit, called D1.
  - For the rest of the characters, make D1's out of individual consonants (and their tonal marks).
  - A boundary between any two adjacent D1's is called a soft boundary.
- Second pass: Remove select soft boundaries
  - Uses learning algorithms to build a model that determines whether or not to remove a soft boundary between each pair of D1's. This model can then be applied to other input text.

Figure 5: Segmentation algorithm

Second Pass: Gluing

Given result from the previous pass, this pass uses learning algorithms to determine whether or not to remove each soft boundary. Figure 4(d) shows the result of this step. Figure 5 summarizes the algorithm.

The motivation behind this step is that, whether a soft boundary should remain or be removed, the decision is not completely random. It has some patterns.

Vowel Position: Vowel position may suggest the existence of a boundary. For instance, a front vowel usually does not take a consonant in front of it. Therefore, if there is a soft boundary in front of a front vowel, it probably should not be removed. Similarly for some rear vowels (such as, -ะ) and the following consonant.

Thai Spelling Rules: Based on Thai spelling rules, some character combination is not valid. For instance, a combination of a long vowel, a leading consonant in column 1, and a trailing consonant in column other than 5, should not be coupled with the tonal mark ้. Example: ที่|ค|รา should not be

merged as กี๋ต|รา, because กี๋ต is not a valid character combination according to the spelling rules. Instead of listing each and every rule, we take advantages of learning algorithms.

Pali Systematic Spelling: Thai language is influenced partly by Pali, in which consonants and spelling are systematic. Pali spelling utilizes the table in figure 2. An example of a spelling rule: only consonants from columns 1, 3, and 5 can be used as a trailing consonant; whenever a column-1 consonant is trailing consonant, the following consonant must be from column-1 or -2 and the same row; whenever a column-3 consonant is trailing consonant, the following consonant must be from column-3 or -4 and the same row. Pali spelling rule would suggest, for example, merging ลัท|ธิ into ลัทธิ because ท is a column-3 trailing consonant (in this context), ธ is a column-4 consonant and they both are from the same row.

Loose Consonant: Meaningful words with only one consonant and no vowel, such as, บ่ and ฉ, are exceptionally rare. Moreover, when one occurs, it is most often written with preceding and trailing spaces, which turn into hard boundaries by our algorithm. Therefore, whenever our algorithm finds a D1 that is a loose consonant, it is extremely unlikely to be a meaningful word by itself and should be merged with at least one of its neighboring D1's.

Frequency Observation: Some character combinations occur more often than others. For example, สุ|ก|รี could be merged into สุ|กรี because ก-ร combination occurs often, while สุ|ก|ดี should *not* be merged into สุ|กดี because ก-ด combination hardly ever occurs.

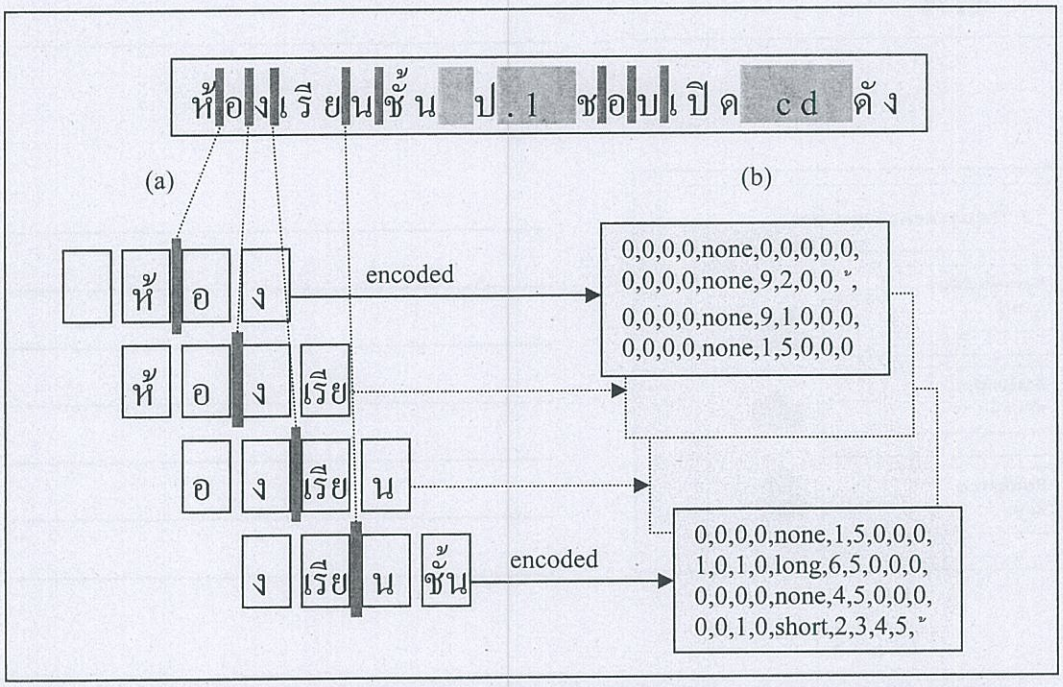

Figure 6: (a) To determine each soft boundary, inspect its left and right neighbors; (b) Each neighbor is encoded by 10 attributes, resulting in 40 attributes for each soft boundary.

Based on the ideas above, we define some features that might help in determining the existence of a boundary between a pair of D1's as follows. For each soft boundary, we look at D1's on its left and right. For each D1, we look at the following features

- Number of front vowel characters (0, 1)
- Number of rear vowel characters (0, 1, 2)
- Number of upper vowel characters or -◌๊ (0, 1, 2)
- Number of lower vowel characters (0, 1)
- Sound of vowel or -◌๊ (none, silent, short, long)
- Leading consonant (row) (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
- Leading consonant (column) (0, 1, 2, 3, 4, 5)
- Trailing consonant (row) (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
- Trailing consonant (column) (0, 1, 2, 3, 4, 5)
- Tonal mark (none, ◌่, ◌๊, ◌๋, ◌๎)

Figure 6 illustrates an example.

Learning algorithms can be used for building a model out of these features. The model can then be applied to other input text.

3. EXPERIMENTS

Dataset

Data used in our experiments were from various sources, such as, encyclopedia, newspapers, web sites, etc. Their topics span a wide range, including business, education, culture, travel, sport, history, astronomy, nature, science, mathematics, medicine, computer, religion, etc. Manually spell-checked, data were partitioned into 80% training set and 20% test set (total number of instances: 11,902).

Experiments

Three experiment sets, shown in figure 7, were carried out. Experiment set 1 was motivated by the fact that loose-consonant D1's were hardly ever a meaningful word. For experiment set 2, it was observed that a number of D1's were just meaningless syllables but not words. Experiment set 3 was based on the fact that determining a word sometimes depended upon its context.

Experiment: Input→Output	Actions	Ideal Results
Set 1: D1→W1	Each loose-consonant D1 merges with either or both of its neighbors. Other D1's do nothing.	No loose-consonant W1.
Set 2: D1→W2	Each D1 merges with neighbor(s) unless it is already a meaningful word.	Each W2 is a meaningful word.
Set 3: D1→W3	Each D1 merges with neighbor(s) unless it is already a meaningful word according to its context.	Each W3 is a meaningful word according to its context.

Figure 7: Experiment sets

In applying the algorithm, computer programs were written for the preprocessing step and the first pass. For the second pass, training data were prepared manually. Sequences of D1's, such as those in figure 4(c), were examined. Each soft boundary was manually determined whether or not to be removed. Result similar to that in figure 4(d) would be appropriate for experiment set 1. For experiment sets 2 and 3, more soft boundaries would probably have been removed. Data were then fed to learning algorithms. Each training instance corresponds to a soft boundary. A model would be trained to determine whether or not a soft boundary should be removed. For the example shown in figure 6 (and for all experiments whose results are presented in this paper), the number of adjacent D1's used for each soft boundary is 4 (2 from each left and right). We performed preliminary experiments on different numbers, namely, 2 and 6. The former yielded slightly less satisfactory results, while the latter showed no significant improvement.

Different learning algorithms, namely, C4.5 [9], ID3 [10], Naïve Bayes classifier [11], and neural networks [12], were experimented and results compared.

Results and Discussion

We performed several experiments on four different learning algorithms, each with various combinations of parameters. Our results indicated that C4.5 performed significantly better than the others. Results are shown in figure 8. In training each C4.5, 10-fold cross validation was performed.

Experiment Set	% Correct		
	Training	Cross Validation	Test
1	97.21	95.57	95.36
2	95.79	92.99	91.51
3	93.74	88.95	88.20

Figure 8: Performance of C4.5 of training and test sets

Inspecting the decision trees, we found that, out of the 40 attributes used to determine whether or not to remove a soft boundary, there was one attribute that stood out as the most discriminating one. It was the existence of the right neighbor's front vowel. If a soft boundary has a front vowel to its immediate right, the algorithm, without looking at any other attributes, will decide to leave the soft boundary untouched. This attribute was chosen as the root by all of our C4.5's.

The right neighbor's other attributes were used frequently in other nodes as well. The left neighbor's were used next often. For the second neighbors to the left and right, their attributes did not appear as a node until later in the tree, such as, at depth 4. This supported our earlier decision of not using third left/right neighbors.

Inspecting test instances, we found that mispredictions made by our model had some patterns. Our models performed below average on acronyms, names, and foreign words.

4. CONCLUSIONS AND FUTURE WORK

This work presents an algorithm for Thai text segmentation. It proposes a set of rules for breaking input text into indivisible units (D1's). It then uses learning algorithms to decide how to glue some of these D1's together to form more meaningful units, which might be a syllable, a word, etc.

This algorithm can be applied as an initial processing of text. The result is suitable for further text applications, such as, indexing.

Based on our experiments, the algorithm still has room for improvement, especially in dealing with acronyms, names, and foreign words. Future work may take into account the dependency between each soft boundary and few others in its vicinity.

REFERENCES

[1] Lorchirachoonkul, V. and Khuwinphunt, C. **1981** Thai Soundex Algorithm and Thai-Syllable Seperation Algorithm. *Research paper, National Institute of Development Administration*, Thailand.

[2] Sornlertlamvanich, V. **1993** Word Segmentation for Thai in Machine Translation System. *Machine Translation, National Electronics and Computer Technology Center*, Bangkok, pp. 50-56.

[3] Pooworawan, Y. and Imarom, V. **1986** Thai Syllable Separater by Dictionary. *Proceedings 9th National Conference on Electrical Engineering*, Khon Kaen, Thailand.

[4] Kawtrakul, A. and Thumkanon, C. **1997** A Statistical Approach to Thai Morphological Analyzer. *Proceedings 5th Workshop on Very Large Corpora*, Beijing.

[5] Meknavin, S. Charoenpornsawat, P. and Kijsirikul, B. **1997** Feature-based Thai Word Segmentation. *Proceedings Natural Language Processing Pacific Rim Symposium*, Phuket, Thailand, pp. 41-46.

[6] Theeramunkong, T. and Usanavasin, S. **2001** Non-Dictionary-Based Thai Word Segmentation Using Decision Trees. *Proceedings 1st International Conference on Human Language Technology Research*, San Diego, CA, pp. 1-5.

[7] Sojka, P. and Anto, D. **2003** Context Sensitive Pattern Based Segmentation: A Thai Challenge. *Proceedings EACL 2003 workshop Computational Linguistics for South Asian Languages -- Expanding Synergies with Europe*, Budapest.

[8] Thonglor, K. **2004** *Thai Grammar*. Bangkok, Ruamsarn.

[9] Quinlan, R. **1993** *C4.5: Programs for Machine Learning*. San Mateo, CA, Morgan Kaufmann.

[10] Quinlan, R. **1986** Induction of decision trees, *Machine Learning, 1* ().81-106.

[11] John, G.H. and Langley, P. **1995** Estimating Continuous Distributions in Bayesian Classifiers. *Proceedings 11th Conference on Uncertainty in Artificial Intelligence*, Montreal, pp. 338-345.

[12] Mitchell, T. **1997** *Machine Learning*. New York, McGraw Hill.