Combining Prediction by Partial Matching and Logistic Regression for Thai Word Segmentation

Ohm Sornil
Department of Computer Science
National Institute of Development
Administration, Bangkok, Thailand
osornil@as.nida.ac.th

Paweena Chaiwanarom
National Statistical Office
Bangkok, Thailand
paweena@nso.go.th

Abstract

Word segmentation is an important part of many applications, including information retrieval, information filtering, document analysis, and text summarization. In Thai language, the process is complicated since words are written continuously, and their structures are not well-defined. A recognized effective approach to word segmentation is Longest Matching, a method based on dictionary. Nevertheless, this method suffers from character-level and syllable-level ambiguities in determining word boundaries. This paper proposes a technique to Thai word segmentation using a two-step approach. First, text is segmented, using an application of Prediction by Partial Matching, into syllables whose structures are more well-defined. This reduces the earlier type of ambiguity. Then, the syllables are combined into words by an application of a syllable-level longest matching method together with a logistic regression model which takes into account contextual information. The experimental results show the syllable segmentation accuracy of more than 96.65% and the overall word segmentation accuracy of 97%.

1 Introduction

In Thai language, characters are written without explicit word boundaries. Depending on the contexts, there can be many ways to break a string into words, for instance, "อาจอง" can be segmented as "อา*จอง" or "อาจ*อง", and "นั่งตากลม" can be segmented as "นั่ง*ตา*กลม" or "นั่ง*ตาก*ลม". This complicates the task of identifying word boundaries.

Longest matching is the most popular approach to Thai word segmentation (Pooworawan, 1986). The algorithm scans text from left to right and selects the longest match with a dictionary entry at each point, in a greedy fashion. However, longest possible words may not comply with the actual meanings. For example, "ชาวบ้านรอกราบพระ" is segmented by the longest matching as "ชาวบ้าน-รอก-ราบ-พระ" instead of the correct segmentation "ชาวบ้าน-รอ-กราบ-พระ". This type of ambiguity is referred to as character-level ambiguity. In addition, "เขารับรองเท้าจากเพื่อน" is segmented as "เขา-รับรอง-เท้า-จาก-เพื่อน" instead of the correct segmentation "เขา-รับ-รองเท้า-จาก-เพื่อน". This is referred to as syllable-level ambiguity.

The technique we propose is a two-step process to word segmentation. In the first step, text is segmented into a sequence of syllables, whose structures are more well-defined. This reduces the character-level ambiguity. The remaining syllable-level ambiguity is the task of combining those syllables into words.

2 Related Work

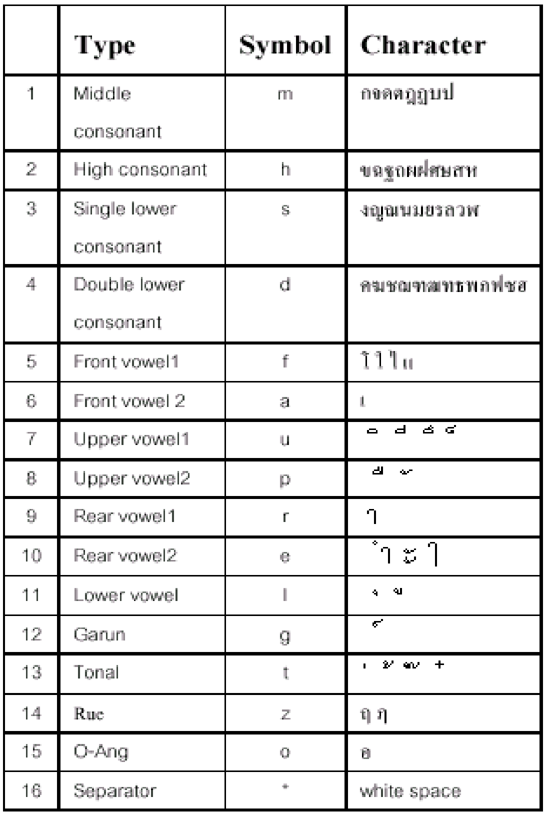
In addition to the longest matching algorithm, discussed earlier, the maximum matching algorithm (Sornlertlamvanich, 1993) was proposed to get around the greedy characteristic of the longest matching algorithm by generating all possible segmentations for a sentence and then selecting the one which contains the fewest number of entries in the dictionary.

An application of statistical techniques was proposed by (Pornprasertkul, 1994), using a Viterbi-based approach to exploit statistical information derived from grammatical tags. Later, (Kawtrakul and Chalathip, 1995) and (Meknawin *et al.*, 1997) used variants of the trigram model to compute the most likely segmentation. (Theeramunkong and Sornlertlamvanich, 2000) observed that, in Thai language, some contiguous characters tend to be inseparable units, called Thai character cluster (TCC), and proposed a set of rules to group characters into TCCs for the purpose of text retrieval.

3 Syllable Segmentation

Prediction by Partial Matching (PPM) (Bell *et al.*, 1990; Cleary and Witten, 1984), a symbolwise compression scheme, is used to build the model for Thai text. PPM generates a prediction for each input symbol based on its previous context (i.e., a few, say *k*, forecoming symbols in the text). The prediction is encoded in form of conditional probability, conditioned on the preceding context. PPM maintains predictions, computed from the training data, for the largest context (*k*) as well as all shorter contexts in tables, as shown in Table 2.

Syllable segmentation can be viewed as the problem of inserting spaces between pairs of characters in the text. Thai language consists of 66 distinct characters. Treating each character individually as in (Teahan *et al*., 2000) requires a large amount of training data in order to calculate all the probabilities in the tables, as well as a large amount of table space and time to lookup data from the tables. We reduce the amount of training data required by partitioning the characters into 16 types, as shown in Table 1. As a side effect of the character classification, the algorithm can handle syllables not present in the training data. Each character is represented by its respective type symbol. For instance "ทำ*ฤทัย*ดี*สู้*เล่ห์*เหลี่ยม*เอา*ไว้" is represented as: "*de*zdps*mu*hlt*asthg*ahsutss* aor*fst*". We then compute the predictions for each symbol as described in the previous section, and the results are shown in Table 2.



	Type	Symbol	Character
1	Middle consonant	m	กจดตฎฏบป
2	High consonant	h	ขฉฐถผฝศษสห
3	Single lower consonant	s	งญณนมยรลวฬ
4	Double lower consonant	d	คฆชฌฑฒทธพภฟซฮ
5	Front vowel1	f	โใไแ
6	Front vowel 2	a	เ
7	Upper vowel1	u	ั ิ ี ึ
8	Upper vowel2	p	ื ็
9	Rear vowel1	r	า
10	Rear vowel2	e	ำ ะ ๅ
11	Lower vowel	l	ุ ู
12	Garun	g	์
13	Tonal	t	่ ้ ๊ ๋
14	Ruc	z	ฤ ฦ
15	O-Ang	o	อ
16	Separator	*	white space

Table 1: Types of Thai Characters

We illustrate the insertion of spaces between characters using text "ช้ำในก็ขู่". In Thai, tonals are not useful for the segmentation purpose, thus are first filtered out, and the text is converted to "*de*fs*mu*hl*".

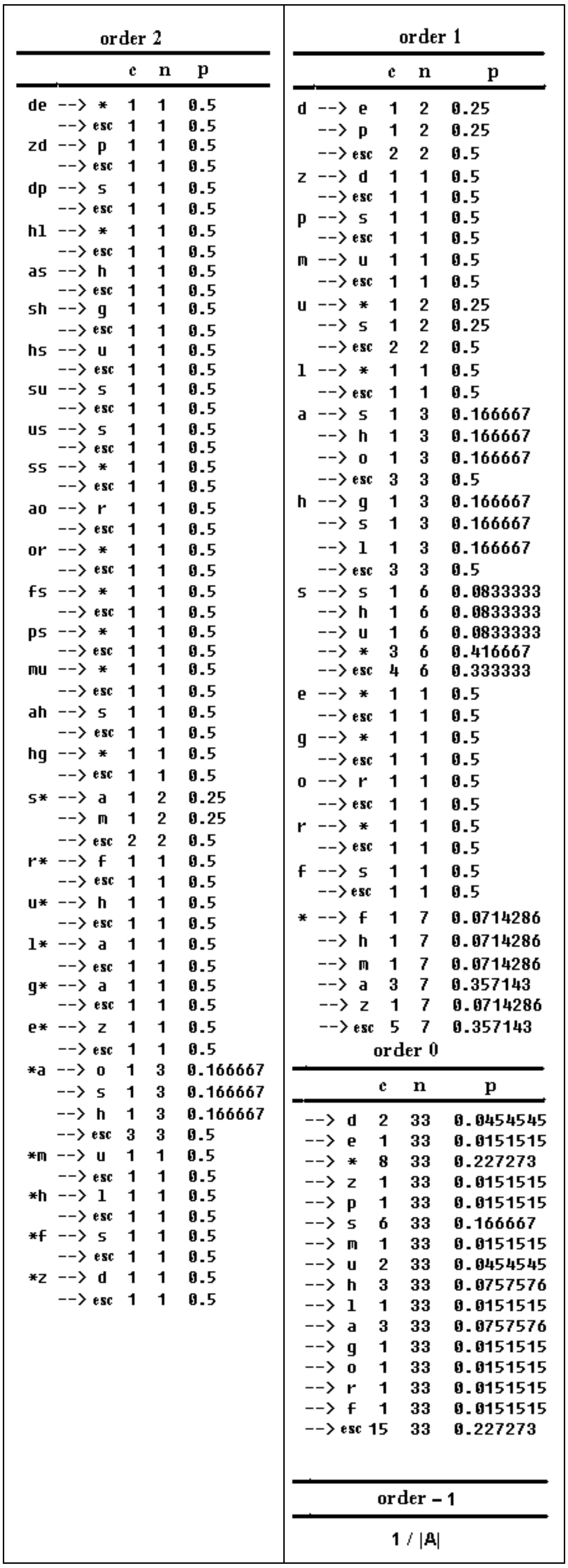
Given an order of *k*, the algorithm computes the likelihood of each possible next symbol (i.e., the next character in the text or a space) by considering a context of size *k* at a time and then proceed to the next symbol in the text. The process is repeated until the text is exhausted. From the text "*de*fs*mu*hl*", the model for space insertion becomes a tree-like structure, as shown in Figure 1.

In order to predict the next symbol, the algorithm follows the concept of PPM by attempting to find first the context of length *k* (*k* = 2 in this example) for this symbol in the context table (i.e., *e*->f*). If the context is not found, it passes the probability of the escape character at this level and goes down one level to the (*k-1*) context table to find the current context of length *k-1* (i.e., **->f*). The process is repeated until a context is found. If it continues to fail to find a context, it may go down ultimately to order (-1) corresponding to equiprobable level for which the probability of any next character is 1/|*A*|, where *A* is the number of distinct characters.

If, on the other hand, a context of length *q*, *0<=q <=k*, is found, then the probability of this next character is estimated to be the product of probabilities of escape characters at levels *k*, *k-1, ..., q+1* multiplied by the probability for the context found at the q-th level.

To handle zero frequency, we use method D (PPMD) (Witten and Bell, 1991) where the escape character gets a probability of *(d/2n)*, and the symbol gets a probability of *(2c-1)/2n* where n is the total number of symbols seen previously, d is the total number of distinct contexts, and c is the total number of contexts that appear in the string.

After the tree-like structure is created, the algorithm selects as the final result the path with the highest probability at the lowest node. This corresponds to the path that gives the best compression according to the PPM text compression method.



order 2		c	n	p
de	--> *	1	1	0.5
	--> esc	1	1	0.5
zd	--> p	1	1	0.5
	--> esc	1	1	0.5
dp	--> s	1	1	0.5
	--> esc	1	1	0.5
hl	--> *	1	1	0.5
	--> esc	1	1	0.5
as	--> h	1	1	0.5
	--> esc	1	1	0.5
sh	--> g	1	1	0.5
	--> esc	1	1	0.5
hs	--> u	1	1	0.5
	--> esc	1	1	0.5
su	--> s	1	1	0.5
	--> esc	1	1	0.5
us	--> s	1	1	0.5
	--> esc	1	1	0.5
ss	--> *	1	1	0.5
	--> esc	1	1	0.5
ao	--> r	1	1	0.5
	--> esc	1	1	0.5
or	--> *	1	1	0.5
	--> esc	1	1	0.5
fs	--> *	1	1	0.5
	--> esc	1	1	0.5
ps	--> *	1	1	0.5
	--> esc	1	1	0.5
mu	--> *	1	1	0.5
	--> esc	1	1	0.5
ah	--> s	1	1	0.5
	--> esc	1	1	0.5
hg	--> *	1	1	0.5
	--> esc	1	1	0.5
s*	--> a	1	2	0.25
	--> m	1	2	0.25
	--> esc	2	2	0.5
r*	--> f	1	1	0.5
	--> esc	1	1	0.5
u*	--> h	1	1	0.5
	--> esc	1	1	0.5
l*	--> a	1	1	0.5
	--> esc	1	1	0.5
g*	--> a	1	1	0.5
	--> esc	1	1	0.5
e*	--> z	1	1	0.5
	--> esc	1	1	0.5
*a	--> o	1	3	0.166667
	--> s	1	3	0.166667
	--> h	1	3	0.166667
	--> esc	3	3	0.5
*m	--> u	1	1	0.5
	--> esc	1	1	0.5
*h	--> l	1	1	0.5
	--> esc	1	1	0.5
*f	--> s	1	1	0.5
	--> esc	1	1	0.5
*z	--> d	1	1	0.5
	--> esc	1	1	0.5

order 1		c	n	p
d	--> e	1	2	0.25
	--> p	1	2	0.25
	--> esc	2	2	0.5
z	--> d	1	1	0.5
	--> esc	1	1	0.5
p	--> s	1	1	0.5
	--> esc	1	1	0.5
m	--> u	1	1	0.5
	--> esc	1	1	0.5
u	--> *	1	2	0.25
	--> s	1	2	0.25
	--> esc	2	2	0.5
l	--> *	1	1	0.5
	--> esc	1	1	0.5
a	--> s	1	3	0.166667
	--> h	1	3	0.166667
	--> o	1	3	0.166667
	--> esc	3	3	0.5
h	--> g	1	3	0.166667
	--> s	1	3	0.166667
	--> l	1	3	0.166667
	--> esc	3	3	0.5
s	--> s	1	6	0.0833333
	--> h	1	6	0.0833333
	--> u	1	6	0.0833333
	--> *	3	6	0.416667
	--> esc	4	6	0.333333
e	--> *	1	1	0.5
	--> esc	1	1	0.5
g	--> *	1	1	0.5
	--> esc	1	1	0.5
o	--> r	1	1	0.5
	--> esc	1	1	0.5
r	--> *	1	1	0.5
	--> esc	1	1	0.5
f	--> s	1	1	0.5
	--> esc	1	1	0.5
*	--> f	1	7	0.0714286
	--> h	1	7	0.0714286
	--> m	1	7	0.0714286
	--> a	3	7	0.357143
	--> z	1	7	0.0714286
	--> esc	5	7	0.357143

order 0	c	n	p
--> d	2	33	0.0454545
--> e	1	33	0.0151515
--> *	8	33	0.227273
--> z	1	33	0.0151515
--> p	1	33	0.0151515
--> s	6	33	0.166667
--> m	1	33	0.0151515
--> u	2	33	0.0454545
--> h	3	33	0.0757576
--> l	1	33	0.0151515
--> a	3	33	0.0757576
--> g	1	33	0.0151515
--> o	1	33	0.0151515
--> r	1	33	0.0151515
--> f	1	33	0.0151515
--> esc	15	33	0.227273

order – 1
1 / |A|

Table 2: PPM Tables (Order 2) After Processing the String

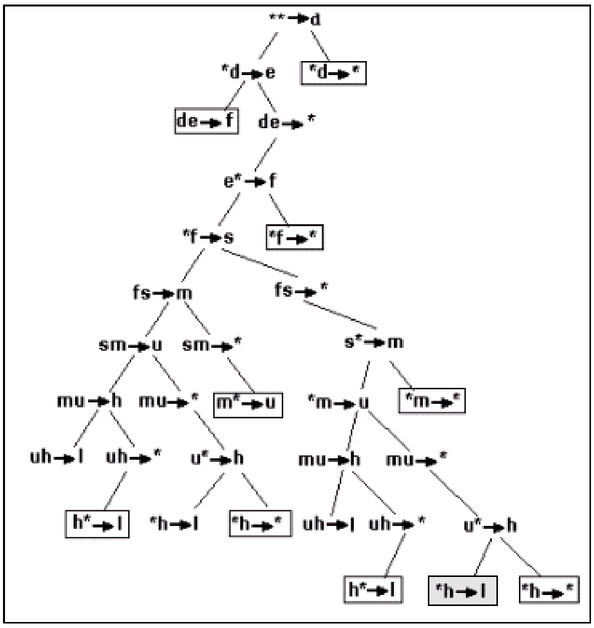


Figure 1: Space Insertion Model

To improve the efficiency of the algorithm, the structure can be pruned by the following set of rules, generated from the language analysis:

1	Rear vowel 2 must be in the last position of a syllable
2	Front vowel must not be in the last position of a syllable
3	There must only be a consonant or space before a front vowel
4	Upper vowel 2 must not be in the last position of a syllable
5	Upper vowel must not be in the first position of a syllable
6	Rear vowel must not be in the first position of a syllable
7	Lower vowel must not be in the first position of a syllable
8	Garun must not be in the first position of a syllable
9	There must not be a space before Garun

The nodes surrounded by a rectangle in Figure 1 are pruned according to the rules above. Thus, they do not generate further subtrees.

4 Combining Syllables into Words

In this section, we propose a technique to form words by combining syllables together. In order to combine syllables into words, for each sentence we first locate ambiguous sequences of syllables, i.e., syllable sequences that can be combined in many ways. The forward and backward syllable-level longest matching are performed. These algorithms are modified from the original longest matching, described in Section 1, by considering syllable as a unit, instead of character. For instance, a syllable sequence "ราย*งาน*เป็น*ต้น*ฉบับ" is processed according to the forward longest matching as "รายงาน*เป็นต้น*ฉบับ", while as "รายงาน*เป็น*ต้นฉบับ" according to the backward longest matching. The inconsistencies between the two algorithms suggest ambiguous sequences of syllables in the sentence. In this example, an ambiguous sequence of syllables is "เป็น*ต้น*ฉบับ".

After identifying ambiguous syllable sequences, we perform the following steps:

Step 1: Between the results of the forward and backward longest matching, the one with all words appearing in the dictionary is selected as the result of the ambiguous sequence. If both results satisfy this condition, go to Step 2.

Step 2: The result with the least number of words is taken as the answer. If the number of words are equal, go to Step 3.

Step 3: A logistic regression model for combining syllables is consulted. This step will be discussed in details below.

	Syllable 1	Syllable 2	Syllable 3	Syllable 4	Merge (Y/N)
1	เขา	รับ	รอง	เท้า	N
2	รับ	รอง	เท้า	จาก	Y
3	รอง	เท้า	จาก	เพื่อน	N

Table 3: Syllable Organization for the Logistic Regression Model

4.1 Logistic Regression Model for Combining Syllables

The model to combine syllables is built upon Binary Logistic Regression whose answers are either combine or not combine. The model considers four consecutive syllables at a time when modeling the decision of whether to combine the middle two syllables together. The first and the fourth syllables are considered the context of the two middle ones. Table 3 shows the organization of data for the model. In the first row, the training data specifies that syllables "รับ" and "รอง" (with the preceding contextual syllable "เขา" and the following contextual syllable "เท้า") should not be combined. The model is trained by every row of the training data. The result is a trained logistic regression model that can be used for guiding whether the middle two syllables should be combined in the context of the surrounding syllables (the first and the fourth syllables).

In the model, each syllable (in Table 3) is represented by a set of features. The syllables under consideration (the second and the third syllables) are represented by 65 features, listed in Table 4.

The contextual syllables (the first and the fourth) are represented by a fewer number of features to make it less specific to the training contexts. The variables for contextual syllables are those statistically significant to the prediction, returned with the regression. The final set consists of 35 variables, as shown in Table 5. The value of each variable is either 1 or -1 which means either the syllable contains or does not contain that particular character, respectively.

Var#	Char	Var#	Char	Var#	Char
1	ก	23	น	45	ะ
2	ข	24	บ	46	า
3	ค	25	ป	47	ิ
4	ฆ	26	ผ	48	ี
5	ง	27	ฝ	49	ึ
6	จ	28	พ	50	ื
7	ฉ	29	ฟ	51	ุ
8	ช	30	ภ	52	ู
9	ซ	31	ม	53	เ
10	ฌ	32	ย	54	แ
11	ญ	33	ร	55	โ
12	ฎ	34	ล	56	ำ
13	ฏ	35	ว	57	ใ
14	ฐ	36	ศ	58	ไ
15	ฑ	37	ษ	59	ั
16	ฒ	38	ส	60	็
17	ณ	39	ห	61	์
18	ด	40	อ	62	่
19	ต	41	ฬ	63	้
20	ถ	42	ฮ	64	๊
21	ท	43	ฤ	65	+
22	ธ	44	ฦ		

Table 4: Syllable Representation for the Second and Third Syllables

5 Experimental Evaluation

In the first experiment, we evaluate the proposed syllable segmentation method. The algorithm is trained with 2,200 syllables, manually segmented from a dictionary. The test data used is a text excerpt from a thesis written in Thai. The results in Table 6 show that the algorithm at order 4 yields the best result which is, from the 1,714 manually segmented syllables, the algorithm correctly identifies 1,694 (or 98.83%) of them correctly. Figure 2 shows an example of segmentation results.

Var#	Char	Var#	Char	Var#	Char
1	ข	13	ส	25	โ
2	ง	14	อ	26	ำ
3	จ	15	ะ	27	ใ
4	ญ	16	า	28	ไ
5	ฎ	17	ิ	29	ั
6	ธ	18	ื	30	็
7	บ	19	ึ	31	์
8	ฝ	20	ี	32	่
9	พ	21	ุ	33	้
10	ฟ	22	ู	34	๊
11	ย	23	เ	35	+
12	ว	24	แ		

Table 5: Syllable Representation for the First and Fourth Syllables

Order	Accuracy
1	77.36%
2	96.38%
3	98.54%
4	98.83%
5	98.19%

Table 6: Results of the PPM Model at Different Orders

Next, we evaluate the proposed algorithm at order 4 against five 1,000-syllable test texts which are not part of the text used in the training. The results in Table 7 show 96.65 to 98.26% segmentation accuracy.

Order	Accuracy
1	77.36%
2	96.38%
3	98.54%
4	98.83%
5	98.19%

Table 7: Results of Five 1,000-Syllable Texts

To evaluate the syllable combination technique, we create 50 ambiguous test cases. The results show that 47 cases (94%) are segmented correctly using the technique proposed, in which 13 cases are correctly segmented in Step 1; 11 cases are correctly segmented in Step 2, and 23 cases are correctly segmented in Step 3.

An evaluation of the entire process of word segmentation (i.e., from syllable segmentation to syllable combination) shows an accuracy of 97.17% by which 76.92% of those incorrect segmentation roots from incorrect syllable segmentation.

Example Text	Syllable Segmentation Result
บัญชีแยกประเภททั่วไปประกอบด้วยบัญชีประเภทต่างๆ คือสินทรัพย์ หนี้สินส่วนของเจ้าของ รายได้ และค่าใช้จ่าย ซึ่งจะนำไปทำงบทดลอง ถ้ากิจการใช้บัญชีย่อยประกอบบัญชีคุมยอด เฉพาะบัญชีคุมยอดเท่านั้นจึงจะปรากฏในบัญชีแยก ประเภททั่วไป เนื่องจากวัตถุประสงค์ขั้นสุดท้ายของการทำบัญชีแยกประเภททั่วไปก็คือ ให้ข้อมูลที่เพียงพอในการจัดทำงบการเงิน ดังนั้นในการจัดผังบัญชีของกิจการใดก็ตามควรคำนึงถึงความต้องการของฝ่ายจัดการ เกี่ยวกับข้อมูลที่จะต้องใช้เพื่อการตัดสินใจ	บัญ*ชี*แยก*ประ *เภท*ทั่ว*ไป*ประ*กอบ*ด้วย*บัญ*ชีป*ระ*เภท*ต่าง*ๆ*คือ*สิน*ทรัพย์*หนี้*สิน*ส่วน*ของ*เจ้า*ของ*ราย*ได้*และ*ค่า*ใช้*จ่าย*ซึ่ง*จะ*นำ*ไป*ทำ*งบ*ทด*ลอง*ถ้า*กิจ*การ*ใช้*บัญ*ชี*ย่อย*ประ*กอบ*บัญ*ชี*คุม*ยอด*เฉพาะ*บัญ*ชี*คุม*ยอด*เท่า*นั้น*จึง*จะ*ปรา*กฏ*ใน*บัญ*ชี*แยก*ประ*เภท*ทั่ว*ไป*เนื่อง*จา*กวัต*ถุ*ประ*สงค์*ขั้น*สุด*ท้าย*ของ*การ*ทำ*บัญ*ชี*แยก*ประ*เภท*ทั่ว*ไป*ก็*คือ*ให้*ข้อ*มูล*ที่*เพียง*พอ*ใน*การ*จัด*ทำ*งบ*การ*เงิน*ดัง*นั้น*ใน*การ*จัด*ผัง*บัญ*ชี*ของ*กิจ*การ*ใด*ก็*ตาม*ควร*คำ*นึง*ถึง*ความ*ต้อง*การ*ของ*ฝ่าย*จัด*การ*เกี่ยว*กับ*ข้อ*มูล*ที่*จะ*ต้อง*ใช้*เพื่อ*การ*ตัด*สิน*ใจ

Figure 2: An Example of Syllable Segmentation

Lastly, we use the same test data however with correctly identified syllables, the performance shows 99.35% accuracy. This emphasizes the importance of pre-segmenting syllables and at the same time indicates that the proposed syllable combining method is effective.

6 Conclusion

This paper proposes a two-step approach to Thai word segmentation. Studying the characteristics of Thai language, we find that word segmentation possesses ambiguities at both character and syllable levels. The proposed technique consists of two steps. The first step is designed to reduce the character-level ambiguity by focusing on extracting syllables whose structures are more well-defined. Then the second step combines syllables into words by using binary logistic regression model. Experimental evaluations emphasize the importance of pre-identifying syllables correctly, show the accuracy of applying PPM to syllable segmentation of 98%, and indicate the effectiveness of the proposed approach to combine syllables into words. The overall accuracy of Thai word segmentation is 97.17%.

References

W. Aroonmanakun 2002. Collocation and Thai Word Segmentation. *Proceedings of SNLP-Oriental COCOSDA.*

T. C. Bell, J. G. Cleary, and I. H. Witten 1990. *Text Compression*. Prentice Hall, NJ.

J. G. Cleary and I. H. Witten 1984. Data Compression Using Adaptive Coding and Partial String Matching. *IEEE Transactions on Communications*, 32(4):396-402.

A. Kawtrakul and T. Chalathip 1995. A Statistical Ap-proach to Thai Morphological Analyzer. *Natural Lan-guage Processing and Intelligent Information System Technology Research Laboratory.*

S. Meknavin, P. Charenpornsawat, and B. Kijsirikul 1997. Feature-based Thai Words Segmentation. *NLPRS, Incorporating SNLP-97.*

Y. Poowarawan 1986. Dictionary-based Thai Syllable Separation. *Proceedings of the Ninth Electronics Engineering Conference*.

A. Pornprasertkul 1994. *Thai Syntactic Analysis*. Ph.D. thesis, Asian Institute of Technoloty.

V. Sornlertlamvanich 1993. Word Segmentation for Thai in a Machine Translation System. *Journal of NECTEC.*

W. J. Teahan, Yingying Wen, R. McNab, and I. H. Witten 2000. A Compression-Based Algorithm for Chinese Word Segmentation. *Computational Linguistics*, 26(3), 375-393.

I. H. Witten and T. C. Bell 1991. The Zero-Frequency Problem: Estimating the Probabilities of Novel Events in Adaptive Text Compression. *IEEE Transactions on Information Theory*, 37 (4):1085-1094.

T. Theeramunkong and V. Sornlertlamvanich 2000. Character Cluster Based Thai Information Retrieval. Proceedings of the 5th International Workshop in *Information Retrieval with Asian Languages.*