Feature-based Thai Word Segmentation

Surapant Meknavin[†], **Paisarn Charoenpornsawat**[‡] and **Boonserm Kijsirikul**[‡]
[†] National Electronics and Computer Technology Center
73/1 Rama VI Road, Bangkok, Thailand
[‡] Department of Computer Engineering, Chulalongkorn University, Thailand

Abstract

Word segmentation is a problem in several Asian languages that have no explicit word boundary delimiter, e.g. Chinese, Japanese, Korean and Thai. We propose to use feature-based approaches for Thai word segmentation. A feature can be anything that tests for specific information in the context around the word in question, such as context words and collocations. To automatically extract such features from a training corpus, we employ two learning algorithms, namely RIPPER and Winnow. Experimental results show that both algorithms appear to outperform the existing Thai word segmentation methods, especially for context-dependent strings.

1 Introduction

Word segmentation is a crucial problem in natural language processing for several Asian languages that have no explicit word boundary delimiter, e.g. Chinese, Japanese, Korean and Thai. The problem can be formally defined as finding

$$\arg\max_{W_i} P(W_i|C) = \arg\max_{W_i} P(W_i)P(C|W_i)/P(C) \quad (1)$$

where $C = c_1 c_2 \ldots c_m$ is an input character string, and $W_i = w_1 w_2 \ldots w_n$ is a possible word segmentation. Since $P(C|W_i)$ is equal to 1 and $P(C)$ is a fixed constant for every candidate, Equation (1) can be simplified to

$$\arg\max_{W_i} P(W_i|C) = \arg\max_{W_i} P(W_i) \quad (2)$$

Considering Thai language, the state-of-the-art method to resolve word boundary ambiguity is based on using statistics of part-of-speech N-gram and word occurrences to find the most probable word/tag sequence of a given sentence. The method enjoys better performance than the more naive methods, e.g. maximal matching. However, since it considers only coarse information of parts of speech in a fix restricted range of context, some important information may be lost. Using word N-gram could recover more fine-grained information of specific word associations but requires an enormous training corpus to estimate all parameters accurately and consumes vast space resources to store the huge word N-gram table. In addition, another weakness of these N-gram approaches is that they do not take unordered long distance specific word collocations into account.

To overcome the disadvantages of the above mentioned methods, we propose a feature-based approach to the problem of Thai word segmentation. Although feature-based approaches have already been applied in several fields of natural language processing, they have not been considered on word segmentation. A feature can be anything that tests for specific information in the context around the target word sequence, such as context words and collocations. The idea is to learn several sources of features that characterize the contexts in which each word tends to occur. Then we can combine those features to disambiguate segmentation ambiguity by selecting the segmentation that yields the most probable sequence of words for a given context. The new problem is then how to select and combine various kinds of features. Yarowsky (Yarowsky, 1994) proposed decision lists as a way to pool several types of features, and to solve a target problem by applying the single strongest feature, whatever type it is. Golding (Golding, 1995) proposed a Bayesian hybrid method to take into account all available evidence, instead of only the strongest one. The method was applied to the task of context-sensitive spelling correction and was reported to be superior to decision lists. Later, Golding and Roth (Golding and Roth, 1996) applied Winnow algorithm in the same task and found that the algorithm performs comparably to the Bayesian hybrid method when using pruned feature sets, and is better when using unpruned sets or unfamiliar test sets.

In this paper, we investigate two machine learning algorithms: RIPPER and Winnow. We then construct our systems based on both algorithms and evaluate them by comparing with other existing approaches to Thai word segmentation.

2 Previous Approaches

2.1 Longest Matching (greedy matching)

Most of the early works in Thai word segmentation are based on longest matching method (Poowarawan, 1986; Rarunrom, 1991). The method scans an input sentence from left to right, and select the longest match with a dictionary entry at each point. In case that the selected match cannot lead the algorithm to find the rest of words in the sentence, the algorithm will backtrack to find the next longest one and continue finding the rest and so on. It is obvious that this algorithm will fail to find the correct segmentation in many cases because of its greedy characteristic. For example,

ไปหามเหสี (go to see the queen)

will be incorrectly segmented as: ไป(go) หาม (carry) เห(deviate) สี(color), while the correct one that cannot be found by the algorithm is: ไป(go) หา(see) มเหสี(queen).

2.2 Maximal Matching

The maximal matching algorithm was proposed to solve the problem of the longest matching algorithm described above (Sornlertlamvanich, 1993). This algorithm first generates all possible segmentations for a sentence and then selects the one that contains the fewest words, which can be done efficiently by using dynamic programming technique. Because the algorithm actually finds real maximal matching instead of using local greedy heuristics to guess, it always outperforms the longest matching method. Nevertheless, when the alternatives have the same number of words, the algorithm cannot determine the best candidate and some other heuristics have to be applied. The heuristic often used is again the greedy one: to prefer the longest matching at each point.

ไปหามเหสี = ไป(go) หาม (carry) เห(deviate) สี(color) |
ไป(go) หา(see) มเหสี(queen)
→ ไป(go) หา(see) มเหสี(queen)
ตากลม = ตาก(expose) ลม(wind) |
ตา(eye) กลม(round)
→ ตาก (expose) ลม(wind)
ยานอก = ยาน(vehicle) อก(breast) |
ยา(medicine) นอก(oversea)
→ ยาน(vehicle) อก(breast)

2.3 Probabilistic word segmentation

Recently, there has been increasing interest in applying statistical techniques to natural language processing. The application of the techniques to Thai word segmentation was first conducted by (Pornprasertkul, 1994), using a Viterbi-based approach to employ statistical information derived from grammatical tags. Later, (Kawtrakul et al., 1995) and (Meknavin, 1995) use variants of word trigram model with part-of-speech trigram model to compute the most likely segmentation and tag sequence at the same time. To compare probabilistic word segmentation with other approaches in our experiments, we use the following model. Let $C = c_1 c_2 \dots c_m$ be the input character string, $W_i = w_1 w_2 \dots w_n$ be a possible word segmentation, and $T_i = t_1 t_2 \dots t_n$ be a sequence of n tags. From equation (2), the problem of word segmentation is to find W that maximizes $P(W_i)$. Taking part-of-speech tags into account, we can compute $P(W_i)$ in the following fashion:

$$\begin{aligned} P(W_i) &= \sum_T P(W_i, T_i) \\ &= \sum_T \prod_i P(t_i|t_{i-1}t_{i-2}) * P(w_i|t_i) \end{aligned} \quad (3)$$

where $P(t_i|t_{i-1}t_{i-2})$ and $P(w_i|t_i)$ can be estimated by computing their relative frequencies in a corpus and smoothed by interpolated estimation. However, one may argue that when human thinks of a sentence, only one sense is communicated. In that case, the most likely segmentation should be the one with maximum joint probability of word sequence and tag sequence $P(W_i, T_i)$ instead of just $P(W_i)$. We have tried both measures and found that there is no significant difference between them.

3 Segmentation Ambiguity

Having analyzed the performance of previous approaches, we found that one of the most influential factors to their performance is context dependency of ambiguous strings. We can roughly categorize segmentation ambiguity into two groups according to their levels of context dependency as follows:

3.1 Context independent segmentation ambiguity

This kind of ambiguity can be resolved almost deterministically by the text itself; there is no need to consult the context in a wider range. This is because, though there are many possible segmentations, there is only one plausible segmentation while other alternatives are very unlikely to occur. Therefore, one can say that it is not really ambiguous. However, simple algorithms such as the maximal matching may

discriminate this kind of ambiguous text incorrectly. Probabilistic word segmentation can handle this kind of ambiguity successfully.

ขนบนอก	*ขนบ นอก / ขน บน อก
	*(tradition) (out) / (hair) (on) (breast)
พอกลางคืน	*พอก ลาง คืน / พอ กลางคืน
	*(to coat)(omen)(night)/(enough)(night)
บอกว่า	บอก ว่า / *บอ กว่า
	(to tell)(that) / *(mad)(than)
ขอมอบ	*ขอม อบ / ขอ มอบ
	*(Khmer)(to scent) / (to ask for)(to give)
ชนกลุ่ม	*ชนก ลุ่ม / ชน กลุ่ม
	*(father)(low) / (people)(group)

3.2 Context dependent segmentation ambiguity

This kind of ambiguity needs surrounding context to decide which segmentation is the most probable one because alternatives are all possible. Although it occurs less than the context independent one, it is more difficult to disambiguate and causes more errors.

ตัวอย่าง	ตัวอย่าง / ตัว อย่าง
	(example) / (body)(like)
ที่รัก	ที่รัก / ที่ รัก
	(sweetheart) / (that)(to love)
ทางการ	ทางการ / ทาง การ
	(formal) / (way)(work)
มากว่า	มาก ว่า / มา กว่า
	(many)(that) / (come)((more) than)
ตากลม	ตาก ลม / ตา กลม
	(to expose)(wind) / (eye)(round)

4 Feature-based Approaches

A number of feature-based methods have been tried for several disambiguation tasks in natural language processing, including decision lists (Yarowsky, 1994), Bayesian hybrids (Golding, 1995) and Winnow (Golding and Roth, 1996). These methods are superior to the past methods in that they can combine evidence from various sources in disambiguation. To apply the methods in our task, we treat word segmentation as a task of word sequence disambiguation. Given a sentence S, we find all of its possible segmentations $S_1, S_2, \ldots, S_n$, where $S_i = w_{i1}w_{i2}\ldots w_{im}$. The task is to decide from the context which one was actually intended. Instead of using only one type of syntactic evidence as in N-gram approaches, we can employ the synergy of several types of features. Following previous works (Golding, 1995), we have tried two types of features: *context words* and *collocations*. Context-word features is used to test for the presence of a particular word within +/- K words of the target word, and collocations test for a pattern of up to L contiguous words and/or part-of-speech tags around the target word. To automatically extract from feature space the discriminative features and to combine them in disambiguation, we have investigated two machine learning algorithms; namely RIPPER and Winnow.

4.1 Learning Algorithms

4.1.1 RIPPER

RIPPER is a propositional rule learning algorithm that constructs a ruleset which accurately classifies the training data. A rule in the constructed ruleset is represented in the form of a conjunction of conditions:

$$\text{if } T_1 \text{ and } T_2 \text{ and } \ldots T_n \text{ then class } C_x.$$

T_1 and T_2 and $\ldots T_n$ is called the body of the rule. C_x is a target class to be learned; it can be a positive or negative class in case of learning one class problem, or any class in case of learning multiple classes. A condition T_i tests for a particular value of an attribute, and it takes one of four forms: $A_n = v, A_c \geq \theta, A_c \leq \theta$ and $v \in A_s$, where A_n is a nominal attribute and v is a legal value for A_n; or A_c is a continuous variable and θ is some value for A_c that occurs in the training data; or A_s is a set-value attribute and v is a value that is an element of A_s. In fact, a condition can include negation. A set-valued attribute is an attribute whose value is a set of string. The primitive tests on a set-valued attribute A_s are of the form "$v \in A_s$".

When constructing a rule, RIPPER finds the test that maximizes information gain for a set of examples S efficiently, making only a single pass over S for each attribute. All symbols v that appear as elements of attribute A for some training examples are considered by RIPPER. One of the most powerful ability of RIPPER is to learn the set-valued attributes, and this ability is one of the main reasons for choosing RIPPER for our task. Using the set-valued attributes, examples can be represented in a more compact form. In case of one class concept, we provide training data set composed of positive and negative examples of the concept. Given the training set, RIPPER first partitions the data into two sets, a "growing set" and "pruning set", and then repeatedly constructs one rule at a time that classifies data in the growing set, until all positive examples are covered. A rule is initialized with an empty body and then built by adding conditions to the body until no negative example is covered. After a rule is constructed, the rule is immediately pruned. To prune a rule, RIPPER deletes any final sequence of conditions from the rule according to some heuristic (Cohen, 1995). The goal of pruning a rule, which may overfit the growing set when the set contains noisy data, is to improve error rates on unseen data in the pruning set. The algorithm can

also handle multiple classes with slight modification (Cohen, 1995).

4.1.2 Winnow

The Winnow algorithm used in our experiment is the algorithm described in (Blum, 1997). We will briefly explain the algorithm here. Winnow is a network that is composed of several nodes connected to a target node (Littlestone, 1988; Golding and Roth, 1996). Each node called *specialist* looks at a particular value of an attribute of the target concept, and will vote for a value of the target concept based on its specialty; i.e., based on a value of the attribute it examines. The global algorithm will then combine the votes from all specialists, and make a prediction based on weighted-majority votes. The pair of (attribute=value) that a specialist examines is a candidate of features we are trying to extract.

In our experiment, we have each specialist examine one or two attributes. For example, a specialist may predict the value of the target concept by checking for (attribute1=value1) and (attribute2=value2). A specialist only makes a prediction if its condition is true (in case of one attribute), or both of its conditions are true (in case of two attributes), and in that case it predicts the most popular outcome out of the last k times it had the chance to predict. In fact, we may have each specialist examine more than two attributes, but for the sake of simplification of preliminary experiment, let us assume that two attributes for each specialist are enough to learn the target concept.

The global algorithm updates the weight of any specialist based on the vote of that specialist. The weight of any specialist is initialized to 1. In case that the global algorithm predicts incorrectly, the weight of the specialist that predicts incorrectly is halved and the weight of the specialist that predicts correctly is multiplied by 3/2. The weight of a specialist is halved when it makes a mistake even if the global algorithm predicts correctly. This weight updating method is the same as the one used in (Blum, 1997). A specialist may choose to abstain instead of giving a prediction on any given example in case that it did not see the same value of an attribute in the example. The advantages of Winnow, which made us decide to use for our task, are that (1) it runs fast because of its incremental algorithm (2) it is not sensitive to extra irrelevant features (Littlestone, 1988).

4.2 Word Segmentation System : FEATURE-1 & FEATURE-2

4.2.1 FEATURE-1

One straightforward way of handling segmentation ambiguities is to enumerate all ambiguous strings found in the training corpus and register them as special entries in our dictionary. For each ambiguity, we find its *confusion set* by listing all of its possible segmentations. For instance, C = {มาก ว่า, มา กว่า} is the confusion set of the string "มากว่า". Then we learn the features that can discriminate each segmentation in the set by RIPPER and Winnow based on our training set. In a context, the disambiguation process will determine based on the obtained features which segmentation in the confusion set is most probable. Examples of features for the confusion set {มาก ว่า, มา กว่า} include:

(1) มากว่า number
(2) พูด within -10 words

where (1) is a collocation that tends to imply มา กว่า, and (2) is a context-word that tends to imply มาก ว่า. In general, this method is very powerful for the training set. However, problems occur when the string we want to disambiguate in the test set is unseen and cannot be found from the dictionary. In such cases, FEATURE-1 just relies on the performance of maximal matching.

4.2.2 FEATURE-2

FEATURE-1 tries to find all ambiguous word sequences and learns features that can classify only elements in confusion sets. Instead of doing that, FEATURE-2 generates all possible *prefix sets* from a word list, and then learns features that classify words in prefix sets. A prefix set is a set of words, where if a and b are any two words in the set, then either a is a prefix of b or b is a prefix of a. An instance of prefix sets is {มา, มาก, มากมาย}. Certainly {มา, มาก} is another prefix set which we must consider too. According to a prefix set such as {มา, มาก, มากมาย}, training examples are then collected from the training corpus, and used to extract features that discriminate words in the set. The extracted features are then employed to decide the way of segmenting a string; i.e., มากมาย should be segmented to มา กมาย, or มาก มาย or มากมาย.

Models learned by FEATURE-2 are more general than those of FEATURE-1 as they are able to be used for handling unseen ambiguous strings more effectively. For instance, assume that a string มากรอง, which can be segmented in 2 ways as มา กรอง and มาก รอง, does not appear in our training set. However, the model for the prefix set {มา, มาก} can be used to segment the string. FEATURE-1 is slightly better than FEATURE-2 in handling strings of confusion sets found in the training set.

4.3 Applying RIPPER and Winnow in Thai Word Segmentation

We construct our word segmentation systems, FEATURE-1 and FEATURE-2, based on RIPPER and Winnow algorithms as shown in Figure 1. Our systems are integration of maximal matching + POS tagger, RIPPER and Winnow algorithms. The train-

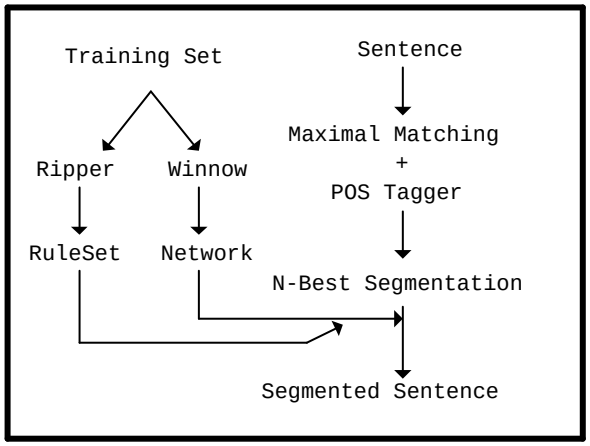

Figure 1: Our word segmentation system

ing set consists of raw sentences together with the corresponding segmented and tagged sentences. In the training mode, a training set composed of segmented and tagged sentences is passed to RIPPER and Winnow algorithms to learn a ruleset and a network, respectively by using the algorithms described above. After a ruleset or a network is learned, a test set is fed to the system for evaluating the performance.

For each sentence from the test set, N-best segmented sentences are generated. Among these sentences, the sentence that mostly matches the ruleset or the network is then selected as the answer. We will describe how to train each algorithm below.

Each training example consists of ten attributes; four attributes for the first and second words before and after the target word, four attributes for the corresponding part-of-speech tags, and two attributes for two sets of ten words before and after the target word. To train RIPPER, one might use positive and negative examples for constructing a definition for a target word. In that case, we must generate both positive and negative examples, and construct rulesets separately for all words. However, the training task will become difficult. As RIPPER can learn a ruleset from training set with multiple classes of positive examples, we then train the algorithm only with positive examples of all target words in a confusion set or a prefix set. This makes the training task easier. For one confusion set or prefix set, we will have one ruleset for classifying the words in the set.

For Winnow, we train the algorithms basically by the same examples as in RIPPER, as Winnow can also learn multiple class examples. However, an example which contains the set-value attributes cannot be directly used because Winnow does not accept set-value attributes. Therefore, we first expand each example into several examples each of which includes only one word in the set-value attributes.

5 Preliminary Results

To test the performance of the different approaches, we select a set of ambiguous strings to use in the benchmark tests. To see the effect of context dependency to the performance, we group the strings into two groups according to their context dependency.

To run experiments, each paragraph from our 25,000 sentence corpus including these ambiguous strings is separated into sentences and then into words. Each word is assigned an appropriate part of speech manually by linguists. For simplicity, we assume that words in each sentence other than the target word are unambiguous. Therefore, each sentence has only one ambiguous string to disambiguate. The resulting corpus is then separated into two parts; the first part about 80 % of the corpus is used as a training set and the rest is used as a test set. Based on the prepared corpus, experiments were conducted to compare our methods with others. The performance was measured by the percentage of the number of correctly segmented sentences to the total number of sentences. Note that this is different from other measures which use some kind of percentage of all correctly segmented words to all words. For FEATURE-1 and FEATURE-2, we measured the performance on both the training set and the test set to compare the effectiveness of each method on seen data with thai on unseen data. For the maximal matching and trigram methods, experiments were also done on both sets, but only to see the performance on different data. The results are shown in Table 1.

The experimental results reveal that, compared to itself, each algorithm gives higher accuracy on the problem of context independent ambiguity than that of context dependent one; i.e., context independent words are easier to handle than context dependent words. In general, Winnow obtains the highest accuracy, followed by RIPPER, Trigram and maximal matching. Comparing between FEATURE-1 and FEATURE-2, FEATURE-1 (both FEATURE-1-RIPPER and FEATURE-1-Winnow) disambiguates words more accurately than FEATURE-2 in almost every case. While this demonstrates that FEATURE-1 does better for a fixed set of trained ambiguous strings, we believe FEATURE-2 is better for unseen strings by virtue of its nature. To support this claim, we made another experiment on a set of a small number of unseen ambiguous strings. According to the experiment, we found that FEATURE-2 surpassed FEATURE-1 with varied margins for every string. Therefore, if we can construct the complete confusion sets for all or most ambiguous strings, FEATURE-1 may be our choice for the task. If this is not the

	Context Independent		Context Dependent	
	Training Set (%)	Test Set (%)	Training Set (%)	Test Set (%)
Maximal Matching	79.74	78.85	52.10	53.52
Trigram	99.81	99.77	73.30	73.15
FEATURE-1-RIPPER	99.94	99.74	96.98	86.60
FEATURE-1-Winnow	100.00	99.70	100.00	95.33
FEATURE-2-RIPPER	98.52	91.27	93.28	89.00
FEATURE-2-Winnow	99.97	93.82	100.00	92.10

Table 1: The result of comparing different approaches

case, however, a suitable model may be the combination in which FEATURE-1 would be used when the ambiguous strings are in some confusion sets, and FEATURE-2 would be used otherwise. Which one is better in practice is what we intend to find out in our future works.

6 Conclusion

In this paper, we have successfully applied two learning algorithms, i.e., RIPPER and Winnow, to the task of Thai word segmentation. RIPPER and Winnow have shown their ablility to construct rulesets or networks that extract the features in the data effectively. The learned features, which are context words and collocations, can capture useful information that cannot be found by traditional word segmentation model such as trigram, and make the task of word segmentation more accurate. The experimental results show that RIPPER and Winnow outperform trigram model, and that Winnow is superior to RIPPER. Our future works include in-depth investigation of feature-based approaches and application of these to the tasks of Thai part-of-speech tagging and spell checking.

References

Avrim Blum. 1997. Empirial Support for Winnow and Weighted-Majority Algorithm: Results on a Calendar Scheduling Domain. *Machine Learning*, 26.

William W. Cohen. 1995. Fast Effective Rule Induction. In *Proceedings of the Twelfth International Conference on Machine Learning*. Morgan Kaufmann.

Andrew R. Golding and Dan Roth. 1996. Applying Winnow to Context-Sensitive Spelling Correction. In *Proceedings of the Thirteenth International Conference on Machine Learning*.

Andrew R. Golding. 1995. A Bayesian Hybrid Method for Context-sensitive Spelling Correction. In *Proceedings of the Third Workshop on Very Large Corpora*.

Asanee Kawtrakul, Supapas Kumtanode, Thitima Jamjanya, and Chanvit Jewriyavech. 1995. A Lexibase Model for Writing Production Assistant System. In *Proceedings of the Symposium on Natural Language Processing in Thailand '95*.

Nick Littlestone. 1988. Learning Quickly when Irrelevant Attributes Abound: A New Linear-Threshold Algorithm. *Machine Learning*, 2.

Surapant Meknavin. 1995. Towards 99.99% Accuracy of Thai Word Segmentation. Oral Presentation at the Symposium on Natural Language Processing in Thailand '95.

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

Ampai Pornprasertkul. 1994. *Thai Syntactic Analysis*. Ph.D. thesis, Asian Institute of Technology.

Sampan Rarunrom. 1991. Dictionary-based Thai Word Separation. Senior Project Report.

Virach Sornlertlamvanich. 1993. Word Segmentation for Thai in a Machine Translation System (in Thai).

David Yarowsky. 1994. Decision Lists for Lexical Ambiguity Resolution. In *Proceedings of 32nd Annual Meeting of the Association for Computational Linguistics*.