A Conditional Random Field Framework for Thai Morphological Analysis

Canasai Kruengkrai, Virach Sornlertlamvanich, Hitoshi Isahara

Thai Computational Linguistics Laboratory
National Institute of Information and Communications Technology
112 Paholyothin Road, Klong 1, Klong Luang, Pathumthani 12120, Thailand
{canasai,virach}@tcllab.org, isahara@nict.go.jp

Abstract

This paper presents a framework for Thai morphological analysis based on the theoretical background of conditional random fields. We formulate morphological analysis of an unsegmented language as the sequential supervised learning problem. Given a sequence of characters, all possibilities of word/tag segmentation are generated, and then the optimal path is selected with some criterion. We examine two different techniques, including the Viterbi score and the confidence estimation. Preliminary results are given to show the feasibility of our proposed framework.

1. Introduction

Morphological analysis is the process of segmenting text into morphemes and performing some tasks such as word formation analysis or part-of-speech (POS) tagging. Morphological analysis is often an initial step for many kinds of text analysis of any languages. In English and other Western languages, text can be tokenized into words by whitespace or punctuation, and morphological analysis can start by considering words as primitive units. In unsegmented languages, such as Chinese, Japanese, and Thai, words are not explicitly delimited by whitespace. As a result, the problem of morphological analysis is more difficult for these languages. More specifically, in the context of Thai morphological analysis, a major challenge is how to solve ambiguities of both word boundary detection and POS tagging simultaneously.

The morphological analysis task can be thought of as the sequential supervised learning problem (Dietterich, 2002), where text is formulated as a sequence of characters. In sequential supervised learning, one of the most widely used techniques is hidden Markov models (HMMs). Based on the concept of generative models, HMMs typically define the joint probability distribution $p(\boldsymbol{y}, \boldsymbol{x})$ over an observation sequence $\boldsymbol{x}$ and a label sequence $\boldsymbol{y}$. The limitation of generatively-trained models is that they must make independent assumptions among elements of the observation sequence in order to make inference tractable. In the morphological analysis task, non-independent elements of the observation sequence, such as prefixes, suffixes, or surrounding words, are useful features for learning and predicting.

One solution to relax the independent assumptions is to formulate the model with the conditional probability distribution $p(\boldsymbol{y}|\boldsymbol{x})$. McCallum et al. (2000) proposed maximum entropy Markov models (MEMMs) that can learn an observation-dependent transition from the previous label to the current label. However, MEMMs and other discriminative models with independently trained next-state classifiers suffer from a serious problem called the label-bias. To deal with this problem, Lafferty et al. (2001) proposed conditional random fields (CRFs) that globally normalize the probability of the label sequence to avoid the label-bias problem. CRFs also provide the flexibility to use non-independent features. Recently, CRFs have been successfully applied in many tasks in natural language processing, including morphological analysis (Kudo et al., 2004), noun-phrase chunking (Sha and Pereira, 2003), and name entity recognition (McCallum and Li, 2003).

In this paper, we propose a unified framework for dealing with ambiguities in Thai morphological analysis based on the theoretical background of CRFs. As mentioned earlier, the Thai writing system has no word boundary indicators. This leads to a problem where elements of the observation sequence are inconsistent due to word boundary ambiguity. Unlike the Chinese writing system which is monosyllabic, the Thai writing system is alphabetic. Each Chinese character can function as a single morpheme, so the ambiguity of the observation sequence can be avoided by performing word segmentation and POS tagging at the character level (Peng et al., 2004). However, Thai word formation is similar to English in which it is composed of consonants and vowels but without tense and inflection. Thus, tagging at the character level for Thai is not a practical way.

Our framework is more closely related to morphological analysis for Japanese (Kudo et al., 2004). Given a sequence of characters, all possibilities of word/tag segmentation are first generated. We present the combination of the longest matching algorithm and the backtracking technique for constructing the word/tag lattice. Then, the optimal path is selected with some criterion. In our work, we examine two different techniques for selecting the most likely word/tag path, including the Viterbi score and the confidence estimation. Preliminary results on a standard benchmark for the Thai POS tagging task named the ORCHID corpus are provided. To the best of our knowledge, a study of Thai morphological analysis based on the concept of CRFs has not been reported on the literature.

The rest of the paper is organized as follows. In Section 2, we discuss related work on Thai morphological analysis, consisting of various techniques for tackling word segmentation and POS tagging. Section 3 reviews some important concepts of conditional random fields. In Section 4, we describe how to obtain all possible word/tag segmentation patterns, and how to select the optimal path with the Viterbi algorithm and the confidence estimation. Section

5 provides details of our experiments, including data sets, evaluation methods, and results. Finally, conclusion and future work are given in Section 6.

2. Related Work

In this section, we briefly review related work on Thai morphological analysis. Most of previous researches have focused on the word segmentation and POS tagging problems separately. Sornlertlamvanich (1993) introduced the maximum matching algorithm that splits a sequence of characters into all possibilities of segmentation using a word list. The word list can be derived from unique head words in a lexicon. The algorithm attempts to minimize the occurrence of unknown words in each candidate, and selects the best segmentation with the lowest number of segmented tokens. Despite of the simple idea, the maximum matching algorithm performs reasonably well for the word segmentation problem. Kawtrakul et al. (1997) proposed a language modeling technique to select the optimal segmentation rather than using heuristics. A trigram Markov model is used to estimate the probabilities of word clusters from a segmented text corpus. However, solving word boundary ambiguity often requires some higher levels of linguistic knowledge.

Meknavin et al. (1997) combined word segmentation with POS tagging based on a generatively-trained model. The optimal segmentation is selected by the highest marginal probability of word sequences. Feature-based approach is utilized to deal with segmentation ambiguity. Charoenpornsawat et at. (1999) also applied the Winnow algorithm to learn contextual features for handling the unknown word problem. Murata et al. (2002) examined other machine learning methods for solving the Thai POS tagging problem, including decision lists, maximum entropy, and support vector machines.

Another direction of researches focuses on a more fine-grained level of word formation. Morphological rules are first applied for syllable/morpheme segmentation, and then the process of word recovery is performed. Jaruskulchai (1998) used a model selection technique called minimum description length (MDL) to select the most likely morpheme combination. Aroonmanakun (2002) exploited statistics of collocation to merge syllables into a word. However, these proposed methods cannot directly integrate the POS tagging task into a single framework.

3. Conditional Random Fields

3.1. Basic Definition

We describe CRFs through the concept of graphical models. Let $\boldsymbol{y}$ be a linear-chain graph structure, consisting of nodes $y_1, \ldots, y_{|\boldsymbol{y}|}$. In the case of undirected graphical models, the probability distribution can be factorized according to the definition on cliques of the graph. Each clique $C_i \in \mathcal{C}$ is a fully connected subset of nodes y_{C_i}, which can be parameterized by using a clique potential ψ_{C_i}. Thus, we can express the probability distribution of the graph as the product of overall clique potentials:

$$p(\boldsymbol{y}) = \frac{1}{Z} \prod_{C_i \in \mathcal{C}} \psi_{C_i}(y_{C_i}) \, , \tag{1}$$

where $Z = \sum_{\boldsymbol{y}} \prod_{C_i \in \mathcal{C}} \psi_{C_i}(y_{C_i})$ is a normalization term called the partition function to ensure that $\sum_{\boldsymbol{y}} p(\boldsymbol{y}) = 1$. By applying the idea of log-linear models, the clique potential can be written in the form of feature functions:

$$\psi_{C_i}(y_{C_i}) = \prod_k \exp\{\lambda_k f_k(y_{C_i})\} = \exp\{\sum_{k=1}^{K} \lambda_k f_k(y_{C_i})\} \, , \tag{2}$$

where K is the number of all features, and $\lambda_1, \ldots, \lambda_K$ are the weight parameters corresponding to local feature functions f_k. In our context, we formulate a linear-chain CRF with the conditional probability distribution $p_{\boldsymbol{\lambda}}(\boldsymbol{y}|\boldsymbol{x})$ of the label (tag) sequence $\boldsymbol{y}$ given the observation (word) sequence $\boldsymbol{x}$.

For simplicity, we assume that both $\boldsymbol{y}$ and $\boldsymbol{x}$ have the same length T, where $\boldsymbol{y} = (y_1, \ldots, y_T)$ and $\boldsymbol{x} = (x_1, \ldots, x_T)$. We also assume the first-order Markov process on $\boldsymbol{y}$. Each local feature function f_k for a clique can be uniformly defined as a state feature $s(y_t, \boldsymbol{x}, t)$ and a transition feature $t(y_{t-1}, y_t, \boldsymbol{x}, t)$ at a position (or time) t. We compactly denote the feature function by $f_k(y_{t-1}, y_t, \boldsymbol{x}, t)$. Thus, the conditional probability distribution for a linear-chain CRF becomes:

$$p_{\boldsymbol{\lambda}}(\boldsymbol{y}|\boldsymbol{x}) = \frac{1}{Z_{\boldsymbol{\lambda}}(\boldsymbol{x})} \exp\{\sum_{t=1}^{T} \sum_{k=1}^{K} \lambda_k f_k(y_{t-1}, y_t, \boldsymbol{x}, t)\} \, , \tag{3}$$

where

$$Z_{\boldsymbol{\lambda}}(\boldsymbol{x}) = \sum_{\boldsymbol{y}} \exp\{\sum_{t=1}^{T} \sum_{k=1}^{K} \lambda_k f_k(y_{t-1}, y_t, \boldsymbol{x}, t)\} \, . \tag{4}$$

3.2. Parameter Estimation and Inference

Given a set of training data $\mathcal{D} = \{\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}\}_{i=1}^{N}$, where N is the number of all training samples, the objective is to find a set of weight parameters $\boldsymbol{\lambda} = \{\lambda_1, \ldots, \lambda_K\}$. A common method is to use maximum likelihood estimation (MLE). We can express the log likelihood as follows:

$$\begin{aligned} l(\boldsymbol{\lambda}; \mathcal{D}) &= \log p(\mathcal{D}|\boldsymbol{\lambda}) \\ &= \log \prod_{i=1}^{N} p_{\boldsymbol{\lambda}}(\boldsymbol{y}^{(i)}|\boldsymbol{x}^{(i)}) \\ &= \sum_{i=1}^{N} \log p_{\boldsymbol{\lambda}}(\boldsymbol{y}^{(i)}|\boldsymbol{x}^{(i)}) \, . \end{aligned}$$

However, MLE often overfits the the training data. Thus, we choose to maximize the penalized log-likelihood (or maximum a posteriori) instead, where $\log p(\boldsymbol{\lambda}|\mathcal{D}) = \log p(\mathcal{D}|\boldsymbol{\lambda}) + \log p(\boldsymbol{\lambda})$. The term $\log p(\boldsymbol{\lambda})$ is defined by a spherical Gaussian prior, so we obtain:

$$\log p(\boldsymbol{\lambda}|\mathcal{D}) = \sum_{i=1}^{N} \log p_{\boldsymbol{\lambda}}(\boldsymbol{y}^{(i)}|\boldsymbol{x}^{(i)}) - \sum_{k=1}^{K} \frac{\lambda_k^2}{2\sigma^2} \, . \tag{5}$$

Let $\mathbf{F}(\boldsymbol{y}, \boldsymbol{x})$ be a global feature for the label sequence. We can compactly write:

$$\exp\Big\{\sum_{t=1}^{T} \sum_{k=1}^{K} \lambda_k f_k(y_{t-1}, y_t, \boldsymbol{x}, t)\Big\} = \exp\Big\{\boldsymbol{\lambda} \cdot \mathbf{F}(\boldsymbol{y}, \boldsymbol{x})\Big\} \, . \tag{6}$$

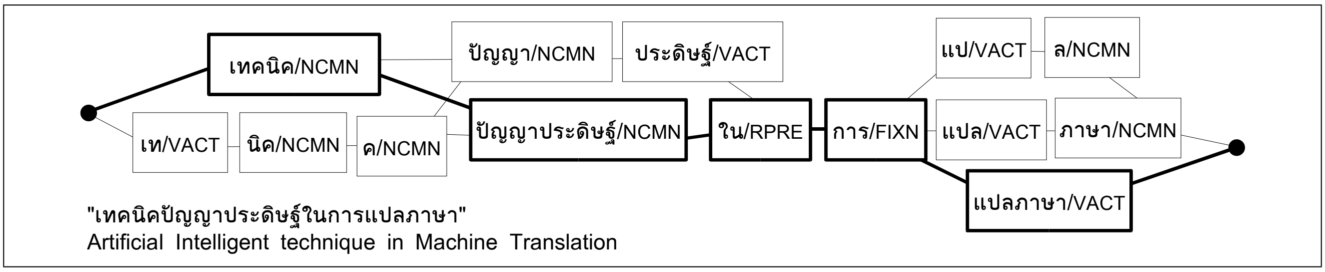

Figure 1: An example of the generated word/tag lattice from a Thai phrase

Using Eq. (3) and Eq. (6), we can rewrite Eq. (5) as:

$$\log p(\boldsymbol{\lambda}|\mathcal{D}) = \sum_{i=1}^{N} [\boldsymbol{\lambda} \cdot \mathbf{F}(\boldsymbol{y}^{(i)}, \boldsymbol{x}^{(i)}) - \log Z_{\boldsymbol{\lambda}}] - \frac{\|\boldsymbol{\lambda}\|^2}{2\sigma^2} \ . \quad (7)$$

One can apply several numerical methods to optimize Eq. (7). In our implementation, we use a limited-memory quasi-Newton method (L-BFGS) that is known to be a very efficient technique for training CRFs (Sha and Pereira, 2003). To find the most likely label sequence given an observation sequence, we can apply the Viterbi algorithm as performed in generative models.

4. Thai Morphological Analysis Framework

In this section, we describe the framework for dealing with ambiguities of Thai morphological analysis. We divide the task into two subproblems: (a) how to obtain all possible word/tag segmentation patterns, and (b) how to select the most likely path.

4.1. Constructing All Possible Paths

Given a character sequence $\boldsymbol{x}$, all possibilities of word segmentation are produced. This results a set of candidate paths $\mathcal{X}$ that can be thought of as a lattice of words. We can efficiently generate $\mathcal{X}$ by using the combination of the longest matching algorithm and the backtracking technique. Our method is reminiscent of Sornlertlamvanich's approach (1993) but attempts to build all possible word/tag paths instead of only word paths.

The process starts by constructing an initial segmentation with the longest matching algorithm. The algorithm tries to search the longest prefix occurred in a word list. If the current prefix matches any token in the word list, the algorithm inserts a boundary at the end of prefix. It continues the longest prefix search starting at the character following the match. If no match is found, the algorithm skips that character and begins the new search starting at the next character. The longest matching algorithm iterates this procedure until the input character sequence is exhausted. Consequently, we can obtain a list of segmented tokens, which is considered to be the initial path.

We then proceed by finding all possible paths using the backtracking technique. This technique can greatly reduce the amount of work in performing all exhaustive searches. Backtracking performs on each segmented token in the initial path by retracing from the left side to the right side. If the considered token cannot be segmented into smaller lexical morphemes, we keep it as it is. On the other hand, if the considered token can be segmented further, we store the first match and allow the longest matching algorithm to run forward again starting at the character following the match. However, backtracking can also produce unknown tokens composed of one or more characters.

After we obtain the candidate paths from the previous process, we then assign each word in the paths with all possible part-of-speech tags. Unfortunately, this leads to a very large number of word/tag patterns. In our current work, we limit the generated paths by constraining each candidate word path with the most likely tag path found by the Viterbi algorithm. The details are described in the following section. Figure 1 shows an example of the generated word/tag lattice.

4.2. Finding The Optimal Path

So far, the question is how to select the optimal path from the word/tag lattice. As mentioned earlier, the most likely tag path for the word path can be found through the Viterbi algorithm. Based on the learnt model, the Viterbi algorithm is capable of finding the optimal solution, reflecting the most probable label sequence for a given observation sequence. We can formally write:

$$\boldsymbol{y}^* = \text{argmax}_{\boldsymbol{y}} p_{\boldsymbol{\lambda}}(\boldsymbol{y}|\boldsymbol{x}) \ . \quad (8)$$

The Viterbi algorithm is an efficient dynamic programming technique that can avoid an exponential-time search over all possible settings of the label sequence $\boldsymbol{y}$. The idea is to store the probability (δ) of the most likely path that leads to the the considered label y_i. In the CRF framework, we can define the recursion for computing the probability of y_i at each position in the sequence by:

$$\delta_{t+1}(y_i) = \max_{y'} \left[\delta_t(y') \exp \left(\sum_k \lambda_k f_k(y', y_i, \mathbf{x}, t) \right) \right] \ . \quad (9)$$

In the termination step ($t = T$), we find the label with the highest score:

$$p^* = \text{argmax}_i \delta_T(y_i) \ , \quad (10)$$

and backtrack through the dynamic programming table to recover the most probable label sequence $\boldsymbol{y}^*$. To select the global best path on the word/tag lattice, we can exploit the probability score produced by the Viterbi search at the termination step as a criterion. Thus, we choose the global best path with the highest Viterbi score.

However, in our preliminary tests, we find that the global Viterbi score sometimes indicates incorrect word/tag paths.

# of sentences	23,125
# of tags	46
# of training/test sentences	18,500/4,625
# of training/test tokens	274,469/68,168
# of training/test ambiguous tokens	205,271/45,559
# of (state+transition) features	19,708
# of words (LEX*i*TRON)	32,363

Table 1: Statistics of ORCHID used in our experiments

We think that these situations may occur from the ambiguity in the observation sequence. Thus, we examine an alternative score to measure the confidence of candidate word/tag paths. The confidence estimation (CE) is equal to the normalized value of a constrained lattice (Culotta and McCallum, 2004):

$$\text{CE} = \frac{Z'_{\boldsymbol{\lambda}}(\boldsymbol{x})}{Z_{\boldsymbol{\lambda}}(\boldsymbol{x})} \tag{11}$$

In our context, the constraints C correspond to a set of segmented tokens with their specified tags. We estimate the confidence of the entire word/tag path. For example, in Figure 1, the constrained lattice is marked by the bold path.

In order to obtain the normalization term $Z_{\boldsymbol{\lambda}}(\boldsymbol{x})$, we have to marginalize out the label selection probabilities, which can be computed by applying the Constrained Forward algorithm. Instead of selecting the optimal label sequence as performed in the Viterbi algorithm, the Constrained Forward algorithm evaluates all possible label sequences given the observation sequence. Only the $\max$ term in Eq. (9) is replaced by the summation, so we can compute the forward value by:

$$\alpha_{t+1}(y_i) = \sum_{y'} \left[\alpha_t(y') \exp\left(\sum_k \lambda_k f_k(y', y_i, \mathbf{x}, t)\right)\right] . \tag{12}$$

In the termination step, we obtain:

$$Z_{\boldsymbol{\lambda}}(\boldsymbol{x}) = \sum_i \alpha_T(y_i) . \tag{13}$$

Let $C = \langle y_t, y_{t+1}, \ldots \rangle$ be a constrained path. The constrained forward value is defined by: $\alpha'_{t+1}(y_i) =$

$$\begin{cases} \sum_{y'} \left[\alpha'_t(y') \exp\left(\sum_k \lambda_k f_k(y', y_i, \mathbf{x}, t)\right)\right] & \text{if } y_i = y_{t+1} , \\ 0 & \text{otherwise,} \end{cases} \tag{14}$$

and we finally obtain the constrained lattice value in the termination step as follows:

$$Z'_{\boldsymbol{\lambda}}(\boldsymbol{x}) = \sum_i \alpha'_T(y_i) . \tag{15}$$

5. Preliminary Experiments

5.1. Data Sets and Evaluation Methods

We performed experiments on a standard benchmark for the Thai POS tagging task named the ORCHID corpus. The ORCHID corpus was constructed at the Linguistics and Knowledge Science Laboratory under the National Electronics and Computer Technology of Thailand. The statistics of the ORCHID corpus used in our experiments are given in Table 1.

We randomly split the corpus into 80% for training and the remaining 20% for testing. We de-segmented the test set by removing all tags from words. We then merged all the words in each sentence into a character sequence. Thus, the task is to recover the word boundaries and assign the words with the most likely tags. In our preliminary experiments, we only trained a linear-chain CRF with basic features. Transition features are based on the first-order Markov assumption, and state features are composed of tokenized words found in the training set. In particular, for the state features, we also added unknown features that are generated when the frequencies of the tokenized words are less than 4 times. The word list for the lattice construction process was taken from an online lexicon called LEX*i*TRON[1].

For the propose of comparison, we experimented with two different techniques of the decoding process (Viterbi and CE) as described in Section 4.2. We also compared our techniques with another two methods: the longest matching (LM) algorithm and the maximum matching (MM) algorithm. After obtaining the segmented tokens, both the LM and MM algorithms performed POS tagging with the unigram baseline (UB) technique. The idea of this technique is just to assign each token with the most likely tag that can be estimated from the training set (Jurafsky and Martin, 2000).

We used precision, recall, and F_1 for evaluation. For word segmentation, precision is defined as the percentage of tokens recovered by the algorithm that also occurred in the test set in the same sentence, whereas recall is defined as the percentage of tokens in the test set recovered by the algorithm. For POS tagging, a token is considered to be a correct one only if both the word boundary and its corresponding POS tag are correctly identified. In order to get a single measure of effectiveness, we employ F_1 that is a combination of precision and recall. These measures can be summarized as follows:

$$\text{Precision} = \frac{\text{\# of correct tokens}}{\text{\# of tokens recovered by the algorithm}} ,$$

$$\text{Recall} = \frac{\text{\# of correct tokens}}{\text{\# of tokens in the test set}} ,$$

$$F_1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision + Recall}} .$$

We implemented a parallel version of the CRF trainer with C language and MPI (Massive Passing Interface) library. We observed that the process of computing the log likelihood function and its gradient is inherently data-parallel, and this process runs iteratively. Thus, the loop can be parallelized by evenly distributing the training samples across processors, and computing the log likelihood function and its gradient simultaneously. At the end of each iteration,

[1] `http://lexitron.nectec.or.th.`

Method	Precision	Recall	F_1
LM	82.717%	82.059%	82.389%
MM	83.305%	82.392%	82.846%
CRF-Viterbi	74.727%	**84.147%**	79.157%
CRF-CE	**83.991%**	83.149%	**83.568%**

Table 2: Results of word segmentation on ORCHID

Method	Precision	Recall	F_1
LM-UB	75.715%	75.113%	75.413%
MM-UB	76.272%	75.436%	75.851%
CRF-Viterbi	70.955%	**79.900%**	75.162%
CRF-CE	**79.744%**	78.945%	**79.342%**

Table 3: Results of POS tagging on ORCHID

all the values are summed up and redistributed to all processors. We ran the training process on a Linux cluster, consisting of 10 nodes. Each node is an Intel® Xeon™ 3 GHz with 4 Gbytes of RAM.

5.2. Results

We now provide experimental results with some discussion. Table 2 shows the summary of the word segmentation results produced by LM, MM, CRF-Viterbi, and CRF-CE. We can see that the two baseline methods give acceptable results. However, the CRF-Viterbi yields poor results in terms of precision and F_1, while CRF-CE provides the best results on F_1 score. We check the segmentation results to see why CRF-Viterbi performs worse on the word segmentation task. We observe that CRF-Viterbi often prefers paths that have greater numbers of segmented tokens. This corresponds to the segmentation results in which CRF-Viterbi achieves the highest recall value. In contrast, CRF-CE can select better paths. It is interesting to note that the path selection of CRF-CE seems to be independent from the number of segmented tokens.

Table 3 shows the summary of the POS tagging results. The two baseline methods perform well even with the simple tagging technique. CRF-Viterbi still yields unsatisfactory results, while CRF-CE outperforms other methods for the POS tagging task.

6. Conclusion and Future Work

This paper has described our initial effort to deal with ambiguities in Thai morphological analysis based on the concept of conditional random fields. We present a unified framework for performing word segmentation and POS tagging at the same time. The word/tag lattice is efficiently constructed, and then the optimal segmentation path on the lattice is selected. Preliminary experiments find that the path selection with the Viterbi score often prefers paths that contain greater numbers of segmented tokens. This can yield unsatisfactory segmentation results. To alleviate this problem, we apply an alternative path selection criterion called the confidence estimation. This criterion is the normalized value of the constrained lattice, which can be computed by the Constrained Forward algorithm. The evaluation on the ORCHID corpus shows that selecting the optimal path with the confidence estimation is very promising.

Several issues of future work remain. In the current work, we use only basic features for learning and predicting. However, one of strengths of CRFs is to allow using arbitrary, overlapping, and non-dependent features. The feature induction technique will be explored (McCallum, 2003). The unknown word problem is also an issue in morphological analysis. We will integrate this problem into our framework. In (Peng et al., 2004), the authors show that the confidence estimation is very useful for unknown word detection.

7. References

Wirote Aroonmanakun. (2002). Collocation and thai word segmentation. In *Proc. of the 5th SNLP & 5th Oriental COCOSDA Workshop*, pages 68–75.

Paisarn Charoenpornsawat. (1999). *Feature-based Thai Word Segmentation*. Master’s Thesis, Computer Engineering, Chulalongkorn University.

Aron Culotta and Andrew McCallum. (2004). Confidence estimation for information extraction. In *Proc. of HLT-NAACL 2004 (short paper)*.

Thomas G. Dietterich. (2002). *Machine Learning for Sequential Data: A Review.* In T. Caelli (Ed.) Structural, Syntactic, and Statistical Pattern Recognition, Lecture Notes in Computer Science, Vol. 2396, Springer-Verlag.

Chuleerat Jaruskulchai. (1998). An automatic thai lexical acquisition from text. In *Proc. of PRICAI’98*, pages 289–296.

Daniel Jurafsky and James H. Martin. (2000). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice-Hall, Inc.

Asanee Kawtrakul and Chalatip Thumkanon. (1997). A statistical approach to thai morphological analyzer. In *Proc. of the 5th Workshop on Very Large Corpora*, pages 289–296.

Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. (2004). Applying conditional random fields to japanese morphological analysis. In *Proc. of EMNLP 2004*.

John Lafferty, Andrew McCallum, and Fernando Pereira. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of ICML 2001*.

Andrew McCallum and Wei Li. (2003). Early results for name entity recognition with conditional random fields, feature induction and web enhanced lexicon. In *Proc. of CoNLL 2003*.

Andrew McCallum, Dayne Freitag, and Fernando Pereira. (2000). Maximum entropy markov model for information extraction and segmentation. In *Proc. of ICML 2000*.

Andrew McCallum. (2003). Efficiently inducing features of conditional random fields. In *Proc. of the 19th Annual Conference on Uncertainty in Artificial Intelligent (UAI-03)*.

Surapant Meknavin, Paisarn Charoenpornsawat, and Boonserm Kijsirikul. (1997). Feature-based thai word segmentation. In *Proc. of NLPRS’97*, pages 289–296.

Masaki Murata, Qing Ma, and Hitoshi Isahara. (2002). Comparison of three machine-learning methods for thai part-of-speech tagging. *ACM Trans. Asian Lang. Inf. Process.*, 1(2):145–158.

Fuchun Peng, Fangfang Feng, and Andrew McCallum. (2004). Chinese segmentation and new word detection using conditional random fields. In *Proc. of COLING 2004*.

Fei Sha and Fernando Pereira. (2003). Shallow parsing with conditional random fields. In *Proc. of HLT-NAACL 2003*.

Virach Sornlertlamvanich. (1993). *Word Segmentation for Thai in Machine Translation System*. Machine Translation, National Electronics and Computer Technology Center, Bangkok.