An Error-Driven Word-Character Hybrid Model for Joint Chinese Word Segmentation and POS Tagging

Canasai Kruengkrai†‡ Kiyotaka Uchimoto†‡ Jun’ichi Kazama†‡
Yiou Wang‡ Kentaro Torisawa‡ Hitoshi Isahara†‡
†Graduate School of Engineering, Kobe University
I-1 Rokkodai-cho, Nada-ku, Kobe 657-8501 Japan
‡National Institute of Information and Communications Technology
3-5 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-0289 Japan
{canasai,uchimoto,kazama,wangyiou,torisawa,isahara}@nict.go.jp

Abstract

In this paper, we present a discriminative word-character hybrid model for joint Chinese word segmentation and POS tagging. Our word-character hybrid model offers high performance since it can handle both known and unknown words. We describe our strategies that yield good balance for learning the characteristics of known and unknown words and propose an error-driven policy that delivers such balance by acquiring examples of unknown words from particular errors in a training corpus. We describe an efficient framework for training our model based on the Margin Infused Relaxed Algorithm (MIRA), evaluate our approach on the Penn Chinese Treebank, and show that it achieves superior performance compared to the state-of-the-art approaches reported in the literature.

1 Introduction

In Chinese, word segmentation and part-of-speech (POS) tagging are indispensable steps for higher-level NLP tasks. Word segmentation and POS tagging results are required as inputs to other NLP tasks, such as phrase chunking, dependency parsing, and machine translation. Word segmentation and POS tagging in a joint process have received much attention in recent research and have shown improvements over a pipelined fashion (Ng and Low, 2004; Nakagawa and Uchimoto, 2007; Zhang and Clark, 2008; Jiang et al., 2008a; Jiang et al., 2008b).

In joint word segmentation and the POS tagging process, one serious problem is caused by unknown words, which are defined as words that are not found in a training corpus or in a system’s word dictionary¹. The word boundaries and the POS tags of unknown words, which are very difficult to identify, cause numerous errors. The word-character hybrid model proposed by Nakagawa and Uchimoto (Nakagawa, 2004; Nakagawa and Uchimoto, 2007) shows promising properties for solving this problem. However, it suffers from structural complexity. Nakagawa (2004) described a training method based on a word-based Markov model and a character-based maximum entropy model that can be completed in a reasonable time. However, this training method is limited by the generatively-trained Markov model in which informative features are hard to exploit.

In this paper, we overcome such limitations concerning both efficiency and effectiveness. We propose a new framework for training the word-character hybrid model based on the Margin Infused Relaxed Algorithm (MIRA) (Crammer, 2004; Crammer et al., 2005; McDonald, 2006). We describe k-best decoding for our hybrid model and design its loss function and the features appropriate for our task.

In our word-character hybrid model, allowing the model to learn the characteristics of both known and unknown words is crucial to achieve optimal performance. Here, we describe our strategies that yield good balance for learning these two characteristics. We propose an error-driven policy that delivers this balance by acquiring examples of unknown words from particular errors in a training corpus. We conducted our experiments on Penn Chinese Treebank (Xia et al., 2000) and compared our approach with the best previous approaches reported in the literature. Experimental results indicate that our approach can achieve state-of-the-art performance.

¹A system’s word dictionary usually consists of a word list, and each word in the list has its own POS category. In this paper, we constructed the system’s word dictionary from a training corpus.
Figure 1: Lattice used in word-character hybrid model.

| Tag | Description                        |
|-----|------------------------------------|
| B   | Beginning character in a multi-character word |
| I   | Intermediate character in a multi-character word |
| E   | End character in a multi-character word |
| S   | Single-character word |

Table 1: Position-of-character (POC) tags.

The paper proceeds as follows: Section 2 gives background on the word-character hybrid model, Section 3 describes our policies for correct path selection, Section 4 presents our training method based on MIRA, Section 5 shows our experimental results, Section 6 discusses related work, and Section 7 concludes the paper.

2 Background

2.1 Problem formation

In joint word segmentation and the POS tagging process, the task is to predict a path of word hypotheses \( y = (y_1, \ldots, y_{\#y}) = ((w_1, p_1), \ldots, (w_{\#y}, p_{\#y})) \) for a given character sequence \( x = (c_1, \ldots, c_{\#x}) \), where \( w \) is a word, \( p \) is its POS tag, and a “#” symbol denotes the number of elements in each variable. The goal of our learning algorithm is to learn a mapping from inputs (unsegmented sentences) \( x \in X \) to outputs (segmented paths) \( y \in Y \) based on training samples of input-output pairs \( S = \{(x_t, y_t)\}_{t=1}^T \).

2.2 Search space representation

We represent the search space with a lattice based on the word-character hybrid model (Nakagawa and Uchimoto, 2007). In the hybrid model, given an input sentence, a lattice that consists of word-level and character-level nodes is constructed. Word-level nodes, which correspond to words found in the system’s word dictionary, have regular POS tags. Character-level nodes have special tags where position-of-character (POC) and POS tags are combined (Asahara, 2003; Nakagawa, 2004). POC tags indicate the word-internal positions of the characters, as described in Table 1.

Figure 1 shows an example of a lattice for a Chinese sentence: “Chongming is China’s third largest island.” Note that some nodes and state transitions are not allowed. For example, \( I \) and \( E \) nodes cannot occur at the beginning of the lattice (marked with dashed boxes), and the transitions from \( I \) to \( B \) nodes are also forbidden. These nodes and transitions are ignored during the lattice construction processing.

In the training phase, since several paths (marked in bold) can correspond to the correct analysis in the annotated corpus, we need to select one correct path \( y_t \) as a reference for training.\(^2\) The next section describes our strategies for dealing with this issue.

With this search space representation, we can consistently handle unknown words with character-level nodes. In other words, we use word-level nodes to identify known words and character-level nodes to identify unknown words. In the testing phase, we can use a dynamic programming algorithm to search for the most likely path out of all candidate paths.

\(^2\)A machine learning problem exists called structured multi-label classification that allows training from multiple correct paths. However, in this paper we limit our consideration to structured single-label classification, which is simple yet provides great performance.
3 Policies for correct path selection

In this section, we describe our strategies for selecting the correct path \( y_t \) in the training phase. As shown in Figure 1, the paths marked in bold can represent the correct annotation of the segmented sentence. Ideally, we need to build a word-character hybrid model that effectively learns the characteristics of unknown words (with character-level nodes) as well as those of known words (with word-level nodes).

We can directly estimate the statistics of known words from an annotated corpus where a sentence is already segmented into words and assigned POS tags. If we select the correct path \( y_t \) that corresponds to the annotated sentence, it will only consist of word-level nodes that do not allow learning for unknown words. We therefore need to choose character-level nodes as correct nodes instead of word-level nodes for some words. We expect that those words could reflect unknown words in the future.

Baayen and Sproat (1996) proposed that the characteristics of infrequent words in a training corpus resemble those of unknown words. Their idea has proven effective for estimating the statistics of unknown words in previous studies (Ratnaparkhi, 1996; Nagata, 1999; Nakagawa, 2004).

We adopt Baayen and Sproat’s approach as the baseline policy in our word-character hybrid model. In the baseline policy, we first count the frequencies of words\(^3\) in the training corpus. We then collect infrequent words that appear less than or equal to \( r \) times.\(^4\) If these infrequent words are in the correct path, we use character-level nodes to represent them, and hence the characteristics of unknown words can be learned. For example, in Figure 1 we select the character-level nodes of the word “崇明” (Chongming) as the correct nodes. As a result, the correct path \( y_t \) can contain both word-level and character-level nodes (marked with asterisks (*)).

To discover more statistics of unknown words, one might consider just increasing the threshold value \( r \) to obtain more artificial unknown words. However, our experimental results indicate that our word-character hybrid model requires an appropriate balance between known and artificial un-

\[^3\]We consider a word and its POS tag a single entry.

\[^4\]In our experiments, the optimal threshold value \( r \) is selected by evaluating the performance of joint word segmentation and POS tagging on the development set.

known words to achieve optimal performance.

We now describe our new approach to leverage additional examples of unknown words. Intuition suggests that even though the system can handle some unknown words, many unidentified unknown words remain that cannot be recovered by the system; we wish to learn the characteristics of such unidentified unknown words. We propose the following simple scheme:

- Divide the training corpus into ten equal sets and perform 10-fold cross validation to find the errors.
- For each trial, train the word-character hybrid model with the baseline policy \((r = 1)\) using nine sets and estimate errors using the remaining validation set.
- Collect unidentified unknown words from each validation set.

Several types of errors are produced by the baseline model, but we only focus on those caused by unidentified unknown words, which can be easily collected in the evaluation process. As described later in Section 5.2, we measure the recall on out-of-vocabulary (OOV) words. Here, we define unidentified unknown words as OOV words in each validation set that cannot be recovered by the system. After ten cross validation runs, we get a list of the unidentified unknown words derived from the whole training corpus. Note that the unidentified unknown words in the cross validation are not necessary to be infrequent words, but some overlap may exist. Finally, we obtain the artificial unknown words that combine the unidentified unknown words in cross validation and infrequent words for learning unknown words. We refer to this approach as the error-driven policy.

4 Training method

4.1 Discriminative online learning

Let \( \mathcal{Y}_t = \{y^1_t, \ldots, y^K_t\} \) be a lattice consisting of candidate paths for a given sentence \( x_t \). In the word-character hybrid model, the lattice \( \mathcal{Y}_t \) can contain more than 1000 nodes, depending on the length of the sentence \( x_t \) and the number of POS tags in the corpus. Therefore, we require a learning algorithm that can efficiently handle large and complex lattice structures.

Online learning is an attractive method for the hybrid model since it quickly converges
Algorithm 1: Generic Online Learning Algorithm

Input: Training set $S = \{(x_t, y_t)\}_{t=1}^T$

Output: Model weight vector $w$

1: $w^{(0)} = 0; v = 0; i = 0$
2: for iter $= 1$ to $N$ do
3: for $t = 1$ to $T$ do
4: $w^{(i+1)} = \text{update } w^{(i)}$ according to $(x_t, y_t)$
5: $v = v + w^{(i+1)}$
6: $i = i + 1$
7: end for
8: end for
9: $w = v/(N \times T)$

within a few iterations (McDonald, 2006). Algorithm 1 outlines the generic online learning algorithm (McDonald, 2006) used in our framework.

4.2 $k$-best MIRA

We focus on an online learning algorithm called MIRA (Crammer, 2004), which has the desired accuracy and scalability properties. MIRA combines the advantages of margin-based and perceptron-style learning with an optimization scheme. In particular, we use a generalized version of MIRA (Crammer et al., 2005; McDonald, 2006) that can incorporate $k$-best decoding in the update procedure. To understand the concept of $k$-best MIRA, we begin with a linear score function:

$$s(x, y; w) = \langle w, f(x, y) \rangle , \quad (1)$$

where $w$ is a weight vector and $f$ is a feature representation of an input $x$ and an output $y$.

Learning a mapping between an input-output pair corresponds to finding a weight vector $w$ such that the best scoring path of a given sentence is the same as (or close to) the correct path. Given a training example $(x_t, y_t)$, MIRA tries to establish a margin between the score of the correct path $s(x_t, y_t; w)$ and the score of the best candidate path $s(x_t, \hat{y}_t; w)$ based on the current weight vector $w$ that is proportional to a loss function $L(y_t, \hat{y}_t)$.

In each iteration, MIRA updates the weight vector $w$ by keeping the norm of the change in the weight vector as small as possible. With this framework, we can formulate the optimization problem as follows (McDonald, 2006):

$$w^{(i+1)} = \arg\min_w \|w - w^{(i)}\| \quad (2)$$

s.t. $\forall \hat{y} \in \text{best}_k(x_t; w^{(i)}):$

$$s(x_t, y_t; w) - s(x_t, \hat{y}_t; w) \geq L(y_t, \hat{y}_t) ,$$

where $\text{best}_k(x_t; w^{(i)}) \in \hat{Y}_t$ represents a set of top $k$-best paths given the weight vector $w^{(i)}$. The above quadratic programming (QP) problem can be solved using Hildreth’s algorithm (Yair Censor, 1997). Replacing Eq. (2) into line 4 of Algorithm 1, we obtain $k$-best MIRA.

The next question is how to efficiently generate $\text{best}_k(x_t; w^{(i)})$. In this paper, we apply a dynamic programming search (Nagata, 1994) to $k$-best MIRA. The algorithm has two main search steps: forward and backward. For the forward search, we use Viterbi-style decoding to find the best partial path and its score up to each node in the lattice. For the backward search, we use $A^*$-style decoding to generate the top $k$-best paths. A complete path is found when the backward search reaches the beginning node of the lattice, and the algorithm terminates when the number of generated paths equals $k$.

In summary, we use $k$-best MIRA to iteratively update $w^{(i)}$. The final weight vector $w$ is the average of the weight vectors after each iteration. As reported in (Collins, 2002; McDonald et al., 2005), parameter averaging can effectively avoid overfitting. For inference, we can use Viterbi-style decoding to search for the most likely path $y^*$ for a given sentence $x$ where:

$$y^* = \arg\max_{y \in \hat{Y}} s(x, y; w) . \quad (3)$$

4.3 Loss function

In conventional sequence labeling where the observation sequence (word) boundaries are fixed, one can use the 0/1 loss to measure the errors of a predicted path with respect to the correct path. However, in our model, word boundaries vary based on the considered path, resulting in a different numbers of output tokens. As a result, we cannot directly use the 0/1 loss.

We instead compute the loss function through false positives ($FP$) and false negatives ($FN$). Here, $FP$ means the number of output nodes that are not in the correct path, and $FN$ means the number of nodes in the correct path that cannot be recognized by the system. We define the loss function by:

$$L(y_t, \hat{y}_t) = FP + FN . \quad (4)$$

This loss function can reflect how bad the predicted path $\hat{y}$ is compared to the correct path $y_t$. A weighted loss function based on $FP$ and $FN$ can be found in (Ganchev et al., 2007).

516
4.4 Features

This section discusses the structure of $f(x, y)$. We broadly classify features into two categories: unigram and bigram features. We design our feature templates to capture various levels of information about words and POS tags. Let us introduce some notation. We write $w_{-1}$ and $w_0$ for the surface forms of words, where subscripts $-1$ and $0$ indicate the previous and current positions, respectively. POS tags $p_{-1}$ and $p_0$ can be interpreted in the same way. We denote the characters by $c_j$.

**Unigram features:** Table 2 shows our unigram features. Templates W0–W3 are basic word-level unigram features, where $Length(w_0)$ denotes the length of the word $w_0$. Using just the surface forms can overfit the training data and lead to poor predictions on the test data. To alleviate this problem, we use two generalized features of the surface forms. The first is the beginning and end characters of the surface (A0–A7). For example, $(A_0(w_0))$ denotes the starting character of the current word $w_0$, and $(A_2(w_0), A_2(w_0))$ denotes the beginning and end characters in the word. The second is the types of beginning and end characters of the surface (T0–T7). We define a set of general character types, as shown in Table 4.

Templates C0–C6 are basic character-level unigram features taken from (Nakagawa, 2004). These templates operate over a window of ±2 characters. The features include characters (C0), pairs of characters (C1–C2), character types (C3), and pairs of character types (C4–C5). In addition, we add pairs of characters and character types (C6).

**Bigram features:** Table 3 shows our bigram features. Templates B0–B9 are basic word-level bigram features. These features aim to capture the most possible combinations of word and POS bigrams. Templates TB0–TB6 are the types of characters for bigrams. For example, $(T_1(w_{-1}), T_3(w_0))$ captures the change of character types from the end character in the previous word to the beginning character in the current word.

Note that if one of the adjacent nodes is a character-level node, we use the template CB0 that represents POS bigrams. In our preliminary experiments, we found that if we add more features to non-word-level bigrams, the number of features grows rapidly due to the dense connections between non-word-level nodes. However, these features only slightly improve performance over using simple POS bigrams.

### Table 2: Unigram features.

| ID | Template | Condition |
|----|----------|-----------|
| W0 | $(w_0)$  | for word-level nodes |
| W1 | $(p_0)$  |           |
| W2 | $(w_0, p_0)$ |           |
| W3 | $(Length(w_0), p_0)$ |           |
| A0 | $(A_0(w_0))$ | if $w_0$ is a single-character word |
| A1 | $(A_0(w_0), p_0)$ |           |
| A2 | $(A_2(w_0))$ | for word-level nodes |
| A3 | $(A_2(w_0), p_0)$ |           |
| A4 | $(A_2(w_0))$ |           |
| A5 | $(A_2(w_0), p_0)$ |           |
| A6 | $(A_2(w_0), A_2(w_0))$ |           |
| A7 | $(A_2(w_0), A_2(w_0))$ |           |
| T0 | $(T_0(w_0))$ | if $w_0$ is a single-character word |
| T1 | $(T_0(w_0), p_0)$ |           |
| T2 | $(T_0(w_0))$ |           |
| T3 | $(T_0(w_0), p_0)$ |           |
| T4 | $(T_0(w_0))$ |           |
| T5 | $(T_0(w_0))$ |           |
| T6 | $(T_0(w_0), T_0(w_0))$ |           |
| T7 | $(T_0(w_0), T_0(w_0), p_0)$ |           |
| C0 | $(c_j, j \in [-2, 2] \times p_0)$ | for character-level nodes |
| C1 | $(c_j, c_{j+1}), j \in [-2, 1] \times p_0$ |           |
| C2 | $(c_{-1}, c_1) \times p_0$ |           |
| C3 | $(T(c_j)), j \in [-2, 2] \times p_0$ |           |
| C4 | $(T(c_j), T(c_{j+1})), j \in [-2, 1] \times p_0$ |           |
| C5 | $(T(c_{-1}), T(c_1)) \times p_0$ |           |
| C6 | $(c_0, T(c_0)) \times p_0$ |           |

### Table 3: Bigram features.

| ID | Template | Condition |
|----|----------|-----------|
| B0 | $(w_{-1}, w_0)$ | if $w_{-1}$ and $w_0$ are word-level nodes |
| B1 | $(p_{-1}, p_0)$ |           |
| B2 | $(w_{-1}, p_0)$ |           |
| B3 | $(p_{-1}, w_0)$ |           |
| B4 | $(w_{-1}, w_0, p_0)$ |           |
| B5 | $(p_{-1}, w_0, p_0)$ |           |
| B6 | $(w_{-1}, p_{-1}, w_0)$ |           |
| B7 | $(w_{-1}, p_{-1}, p_0)$ |           |
| B8 | $(w_{-1}, p_{-1}, w_0, p_0)$ |           |
| B9 | $(Length(w_{-1}), p_0)$ |           |
| TB0 | $(T_1(w_{-1}), p_0)$ |           |
| TB1 | $(T_1(w_{-1}), T_3(w_0))$ |           |
| TB2 | $(T_1(w_{-1}), p_{-1}, p_0)$ |           |
| TB3 | $(T_1(w_{-1}), T_3(w_0))$ |           |
| TB4 | $(T_1(w_{-1}), p_{-1}, T_3(w_0))$ |           |
| TB5 | $(T_1(w_{-1}), p_{-1}, T_3(w_0))$ |           |
| TB6 | $(T_1(w_{-1}), p_{-1}, T_3(w_0), p_0)$ |           |
| CB0 | $(p_{-1}, p_0)$ | otherwise |

### Table 4: Character types.

| Character type | Description |
|----------------|-------------|
| Space          | Space       |
| Numeral        | Arabic and Chinese numerals |
| Symbol         | Symbols     |
| Alphabet       | Alphabets   |
| Chinese        | Chinese characters |
| Other          | Others      |
5 Experiments

5.1 Data sets

Previous studies on joint Chinese word segmentation and POS tagging have used Penn Chinese Treebank (CTB) (Xia et al., 2000) in experiments. However, versions of CTB and experimental settings vary across different studies.

In this paper, we used CTB 5.0 (LDC2005T01) as our main corpus, defined the training, development and test sets according to (Jiang et al., 2008a; Jiang et al., 2008b), and designed our experiments to explore the impact of the training corpus size on our approach. Table 5 provides the statistics of our experimental settings on the small and large training data. The out-of-vocabulary (OOV) is defined as tokens in the test set that are not in the training set (Sproat and Emerson, 2003). Note that the development set was only used for evaluating the trained model to obtain the optimal values of tunable parameters.

5.2 Evaluation

We evaluated both word segmentation (Seg) and joint word segmentation and POS tagging (Seg & Tag). We used recall ($R$), precision ($P$), and $F_1$ as evaluation metrics. Following (Sproat and Emerson, 2003), we also measured the recall on OOV ($R_{OOV}$) tokens and in-vocabulary ($R_{IV}$) tokens. These performance measures can be calculated as follows:

\[
Recall (R) = \frac{\# \text{ of correct tokens}}{\# \text{ of tokens in test data}}
\]

\[
Precision (P) = \frac{\# \text{ of correct tokens}}{\# \text{ of tokens in system output}}
\]

\[
F_1 = \frac{2 \cdot R \cdot P}{R + P}
\]

\[
R_{OOV} = \frac{\# \text{ of correct OOV tokens}}{\# \text{ of OOV tokens in test data}}
\]

\[
R_{IV} = \frac{\# \text{ of correct IV tokens}}{\# \text{ of IV tokens in test data}}
\]

For Seg, a token is considered to be a correct one if the word boundary is correctly identified. For Seg & Tag, both the word boundary and its POS tag have to be correctly identified to be counted as a correct token.

5.3 Parameter estimation

Our model has three tunable parameters: the number of training iterations $N$; the number of top $k$-best paths; and the threshold $r$ for infrequent words. Since we were interested in finding an optimal combination of word-level and character-level nodes for training, we focused on tuning $r$. We fixed $N = 10$ and $k = 5$ for all experiments. For the baseline policy, we varied $r$ in the range of $[1, 5]$ and found that setting $r = 3$ yielded the best performance on the development set for both the small and large training corpus experiments. For the error-driven policy, we collected unidentified unknown words using 10-fold cross validation on the training set, as previously described in Section 3.

5.4 Impact of policies for correct path selection

Table 6 shows the results of our word-character hybrid model using the error-driven and baseline policies. The third and fourth columns indicate the numbers of known and artificial unknown words in the training phase. The total number of words is the same, but the different policies yield different balances between the known and artificial unknown words for learning the hybrid model. Optimal balances were selected using the development set. The error-driven policy provides additional artificial unknown words in the training set. The error-driven policy can improve $R_{OOV}$ as well as maintain good $R_{IV}$, resulting in overall $F_1$ improvements.

Table 5: Training, development, and test data statistics on CTB 5.0 used in our experiments.
5.5 Comparison with best prior approaches

In this section, we attempt to make meaningful comparison with the best prior approaches reported in the literature. Although most previous studies used CTB, their versions of CTB and experimental settings are different, which complicates comparison.

Ng and Low (2004) (N&L04) used CTB 3.0. However, they just showed POS tagging results on a per character basis, not on a per word basis. Zhang and Clark (2008) (Z&C08) generated CTB 3.0 from CTB 4.0. Jiang et al. (2008a; 2008b) (Jiang08a, Jiang08b) used CTB 5.0. Shi and Wang (2007) used CTB that was distributed in the SIGHAN Bakeoff. Besides CTB, they also used HowNet (Dong and Dong, 2006) to obtain semantic class features. Zhang and Clark (2008) indicated that their results cannot directly compare to the results of Shi and Wang (2007) due to different experimental settings.

We decided to follow the experimental settings of Jiang et al. (2008a; 2008b) on CTB 5.0 and Zhang and Clark (2008) on CTB 4.0 since they reported the best performances on joint word segmentation and POS tagging using the training materials only derived from the corpora. The performance scores of previous studies are directly taken from their papers. We also conducted experiments using the system implemented by Nakagawa and Uchimoto (2007) (N&U07) for comparison.

Following Zhang and Clark (2008), we first generated CTB 3.0 from CTB 4.0 using sentence IDs 1–10364. We then divided CTB 3.0 into ten equal sets and conducted 10-fold cross validation.
520
Eric Brill. 1995. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics*, 21(4):543–565.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of EMNLP*, pages 1–8.

Koby Crammer, Ryan McDonald, and Fernando Pereira. 2005. Scalable large-margin online learning for structured classification. In *NIPS Workshop on Learning With Structured Outputs*.

Koby Crammer. 2004. *Online Learning of Complex Categorical Problems*. Hebrew University of Jerusalem, PhD Thesis.

Zhendong Dong and Qiang Dong. 2006. *Hownet and the Computation of Meaning*. World Scientific.

Kuzman Ganchev, Koby Crammer, Fernando Pereira, Gideon Mann, Kedar Bellare, Andrew McCallum, Steven Carroll, Yang Jin, and Peter White. 2007. Penn/umass/chop biocreative ii systems. In *Proceedings of the Second BioCreative Challenge Evaluation Workshop*.

Wenbin Jiang, Liang Huang, Qun Liu, and Yajuan Lü. 2008a. A cascaded linear model for joint chinese word segmentation and part-of-speech tagging. In *Proceedings of ACL*.

Wenbin Jiang, Haitao Mi, and Qun Liu. 2008b. Word lattice reranking for chinese word segmentation and part-of-speech tagging. In *Proceedings of COLING*.

Tetsuji Nakagawa and Kiyotaka Uchimoto. 2007. A hybrid approach to word segmentation and pos tagging. In *Proceedings of ACL Demo and Poster Sessions*.

Kiyotaka Uchimoto, Satoshi Sekine, and Hitoshi Ishihara. 2001. The unknown word problem: a morphological analysis of japanese using maximum entropy aided by a dictionary. In *Proceedings of EMNLP*, pages 91–99.

Masaki Nagata. 1999. A part of speech estimation method for japanese unknown words using a statistical model of morphology and context. In *Proceedings of ACL*, pages 277–284.

Tetsuji Nakagawa. 2004. Chinese and japanese word segmentation using word-level and character-level information. In *Proceedings of COLING*, pages 466–472.

Hwee Tou Ng and Jin Kiat Low. 2004. Chinese part-of-speech tagging: One-at-a-time or all-at-once? word-based or character-based? In *Proceedings of EMNLP*, pages 277–284.

Richard Sproat and Thomas Emerson. 2003. The first international chinese word segmentation bakeoff. In *Proceedings of the 2nd SIGHAN Workshop on Chinese Language Processing*, pages 133–143.

Kiyotaka Uchimoto, Satoshi Sekine, and Hitoshi Ishihara. 2001. The unknown word problem: a morphological analysis of japanese using maximum entropy aided by a dictionary. In *Proceedings of EMNLP*, pages 91–99.

Stavros A. Zenios Yair Censor. 1997. *Parallel Optimization: Theory, Algorithms, and Applications*. Oxford University Press.

Yue Zhang and Stephen Clark. 2008. Joint word segmentation and pos tagging on a single perceptron. In *Proceedings of ACL*.