Joint Chinese Word Segmentation, POS Tagging and Parsing

Xian Qian **Yang Liu**
Computer Science Department
The University of Texas at Dallas
qx,yangl@hlt.utdallas.edu

Abstract

In this paper, we propose a novel decoding algorithm for discriminative joint Chinese word segmentation, part-of-speech (POS) tagging, and parsing. Previous work often used a pipeline method – Chinese word segmentation followed by POS tagging and parsing, which suffers from error propagation and is unable to leverage information in later modules for earlier components. In our approach, we train the three individual models separately during training, and incorporate them together in a unified framework during decoding. We extend the CYK parsing algorithm so that it can deal with word segmentation and POS tagging features. As far as we know, this is the first work on joint Chinese word segmentation, POS tagging and parsing. Our experimental results on Chinese Tree Bank 5 corpus show that our approach outperforms the state-of-the-art pipeline system.

1 Introduction

For Asian languages such as Japanese and Chinese that do not contain explicitly marked word boundaries, word segmentation is an important first step for many subsequent language processing tasks, such as POS tagging, parsing, semantic role labeling, and various applications. Previous studies for POS tagging and syntax parsing on these languages sometimes assume that gold standard word segmentation information is provided, which is not the real scenario. In a fully automatic system, a pipeline approach is often adopted, where raw sentences are first segmented into word sequences, then POS tagging and parsing are performed. This kind of approach suffers from error propagation. For example, word segmentation errors will result in tagging and parsing errors. Additionally, early modules cannot use information from subsequent modules. Intuitively a joint model that performs the three tasks together should help the system make the best decisions.

In this paper, we propose a unified model for joint Chinese word segmentation, POS tagging, and parsing. Three sub-models are independently trained using the state-of-the-art methods. We do not use the joint inference algorithm for training because of the high complexity caused by the large amount of parameters. We use linear chain Conditional Random Fields (CRFs) (Lafferty et al., 2001) to train the word segmentation model and POS tagging model, and averaged perceptron (Collins, 2002) to learn the parsing model. During decoding, parameters of each sub-model are scaled to represent its importance in the joint model. Our decoding algorithm is an extension of CYK parsing. Initially, weights of all possible words together with their POS tags are calculated. When searching the parse tree, the word and POS tagging features are dynamically generated and the transition information of POS tagging is considered in the span merge operation.

Experiments are conducted on Chinese Tree Bank (CTB) 5 dataset, which is widely used for Chinese word segmentation, POS tagging and parsing. We compare our proposed joint model with the pipeline system, both built using the state-of-the-art sub-models. We also propose an evaluation metric to calculate the bracket scores for parsing in the face of word segmentation errors. Our experimental results show that the joint model significantly outperforms the pipeline method based on the state-of-the-art sub-models.

2 Related Work

There is very limited previous work on joint Chinese word segmentation, POS tagging, and parsing. Previous joint models mainly focus on word segmentation and POS tagging task, such as the virtual nodes method (Qian et al., 2010), cascaded linear model (Jiang et al., 2008a), perceptron (Zhang and Clark, 2008), sub-word based stacked learning (Sun, 2011), reranking (Jiang et al., 2008b). These joint models showed about $0.2 - 1\%$ F-score improvement over the pipeline method. Recently, joint tagging and dependency parsing has been studied as well (Li et al., 2011; Lee et al., 2011).

Previous research has showed that word segmentation has a great impact on parsing accuracy in the pipeline method (Harper and Huang, 2009). In (Jiang et al., 2009), additional data was used to improve Chinese word segmentation, which resulted in significant improvement on the parsing task using the pipeline framework. Joint segmentation and parsing was also investigated for Arabic (Green and Manning, 2010). A study that is closely related to ours is (Goldberg and Tsarfaty, 2008), where a single generative model was proposed for joint morphological segmentation and syntactic parsing for Hebrew. Different from that work, we use a discriminative model, which benefits from large amounts of features and is easier to deal with unknown words. Another main difference is that, besides segmentation and parsing, we also incorporate the POS tagging model into the CYK parsing framework.

3 Methods

For a given Chinese sentence, our task is to generate the word sequence, its POS tag sequence, and the parse tree (constituent parsing). A joint model is expected to make more optimal decisions than a pipeline approach; however, such a model will be too complex and it is difficult to estimate model parameters. Therefore we do not perform joint inference for training. Instead, we develop three individual models independently during training and perform joint decoding using them. In this section, we first describe the three sub-models and then the joint decoding algorithm.

3.1 Word Segmentation Model

Methods for Chinese word segmentation can be broadly categorized into character based and word based models. Previous studies showed that character-based models are more effective to detect out-of-vocabulary words while word-based models are more accurate to predict in-vocabulary words (Zhang et al., 2006). Here, we use order-0 semi-Markov model (Sarawagi and Cohen, 2004) to take advantages of both approaches.

More specifically, given a sentence $\mathbf{x} = c_1, c_2, \ldots, c_l$ (where c_i is the i^{th} Chinese character, l is the sentence length), the character-based model assigns each character with a word boundary tag. Here we use the *BCDIES* tag set, which achieved the best official performance (Zhao and Kit, 2008): *B*, *C*, *D*, *E* denote the first, second, third, and last character of a multi-character word respectively, *I* denotes the other characters, and *S* denotes the single character word. We use the same character-based feature templates as in the best official system, shown in Table 1 (1.1-1.3), including character unigram and bigram features, and transition features. Linear chain CRFs are used for training.

Feature templates in the word-based model are shown in Table 1 (1.4-1.6), including word features, sub-word features, and character bigrams within words. The word feature is activated if a predicted word w is in the vocabulary (i.e., appears in training data). Subword(w) is the longest in-vocabulary word within w. To use word features, we adopt a K-best reranking approach. The top K candidate segmentation results for each training sample are generated using the character-based model, and the gold segmentation is added if it is not in the candidate set. We use the Maximum Entropy (ME) model to learn the weights of word features such that the probability of the gold candidate is maximal.

A problem arises when combining the two models and using it in joint segmentation and parsing, since the linear chain used in the character-based model is incompatible with CYK parsing model and the word-based model due to the transition informa-

Character Level Feature Templates	
(1.1)	$c_{i-2}y_i$, $c_{i-1}y_i$, c_iy_i, $c_{i+1}y_i$, $c_{i+2}y_i$
(1.2)	$c_{i-1}c_iy_i$, $c_ic_{i+1}y_i$, $c_{i-1}c_{i+1}y_i$
(1.3)	$y_{i-1}y_i$
Word Level Feature Templates	
(1.4)	word w
(1.5)	subword(w)
(1.6)	character bigrams within w

Table 1: Feature templates for word segmentation. c_i is the i^{th} character in the sentence, y_i is its label, w is a predicted word.

tion. Thus, we slightly modify the linear chain CRF-s by fixing the weights of transition features during training and testing. That is, weights of impossible transition features (e.g., *B*→*B*) are set to $-\infty$, and weights of the other transition features (e.g., *E*→*B*) are set to 0. In this way, the transition feature could be neglected in testing for two reasons. First, all illegal label assignments are prohibited in prediction, since their weights are $-\infty$; second, because weights of legal transition features are 0, they do not affect the prediction at all. In the following, transition features are excluded.

Now we can use order-0 semi Markov model as the hybrid model. We define the score of a word as the sum of the weights of all the features within the word. Formally, the score of a multi-character word $w = c_i, \ldots, c_j$ is defined as:

$$\begin{aligned}&score_{seg}(\mathbf{x}, i, j) = \theta_{CRF} \cdot \mathbf{f}_{CRF}(\mathbf{x}, y_i = B) + \ldots \\ &+\theta_{CRF} \cdot \mathbf{f}_{CRF}(\mathbf{x}, y_j = E) + \theta_{ME} \cdot \mathbf{f}_{ME}(\mathbf{x}, i, j) \\ &\equiv \theta_{seg}\mathbf{f}_{seg}(\mathbf{x}, i, j) \qquad (1)\end{aligned}$$

where $\mathbf{f}_{CRF}$ and $\mathbf{f}_{ME}$ are the feature vectors in the character and word based models respectively, and $\theta_{CRF}, \theta_{ME}$ are their corresponding weight vectors. For simplicity, we denote $\theta_{seg} = \theta_{CRF\oplus ME}$, $\mathbf{f}_{seg} = \mathbf{f}_{CRF\oplus ME}$, where $\theta_{CRF\oplus ME}$ means the concatenation of θ_{CRF} and θ_{ME}. Scores for single character words are defined similarly. These word scores will be used in the joint segmentation and parsing task Section 3.4.

3.2 POS Tagging Model

Though syntax parsing model can directly predict the POS tag itself, we choose not to use this, but use an independent POS tagger for two reasons. First, there is a large amount of data with labeled POS tags but no syntax annotations, such as the People's Daily corpus and SIGHAN bakeoff corpora (Jin and Chen, 2008). Such data can only be used to train POS taggers, but not for training the parsing model. Often using a larger training set will result in a better POS tagger. Second, the state-of-the-art POS tagging systems are often trained by sequence labeling models, not parsing models.

(2.1)	$w_{i-2}t_i$, $w_{i-1}t_i$, w_it_i, $w_{i+1}t_i$, $w_{i+2}t_i$
(2.2)	$w_{i-2}w_{i-1}t_i$, $w_{i-1}w_it_i$, $w_iw_{i+1}t_i$, $w_{i+1}w_{i+2}t_i$ $w_{i-1}w_{i+1}t_i$
(2.3)	$c_1(w_i)t_i$, $c_2(w_i)t_i$, $c_3(w_i)t_i$, $c_{-2}(w_i)t_i$ $c_{-1}(w_i)t_i$
(2.4)	$c_1(w_i)c_2(w_i)t_i$, $c_{-2}(w_i)c_{-1}(w_i)t_i$
(2.5)	$l(w_i)t_i$
(2.5)	$t_{i-1}t_i$

Table 2: Feature templates for POS tagging. w_i is the i^{th} word in the sentence, t_i is its POS tag. For a word w, $c_j(w)$ is its j^{th} character, $c_{-j}(w)$ is the last j^{th} character, and $l(w)$ is its length.

The POS tagging problem is to assign a POS tag $t \in \mathcal{T}$ to each word in a sentence. We also use linear chain CRFs for POS tagging. Feature templates shown in Table 2 are the same as those in (Qian et al., 2010), which have been shown effective on CTB corpus. Three feature sets are considered: (i) word level features, including surrounding word unigrams, bigrams, and word length; (ii) character level features, such as the first and last characters in the words; (iii) transition features.

3.3 Parsing Model

We choose discriminative models for parsing since it is easy to handle unknown words by simply adding character level features. Online structured learning algorithms were demonstrated to be effective for training, such as stochastic optimization (Finkel et al., 2008). In this study, we use averaged perceptron algorithm for parameter estimation since it is easier to implement and has competitive performance.

A Context Free Grammar (CFG) consists of (i) a set of terminals; (ii) a set of nonterminals $\{N^k\}$; (iii) a designated start symbol ROOT; and (iv) a set of rules, $\{r = N^i \rightarrow \zeta^j\}$, where ζ^j is a sequence of terminals and nonterminals. In the parsing task, ter-

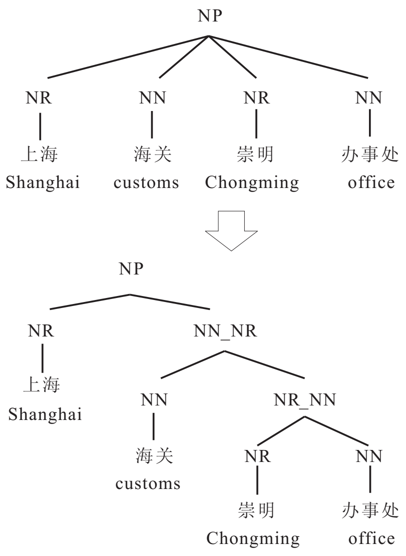

Figure 1: Parse tree binarization

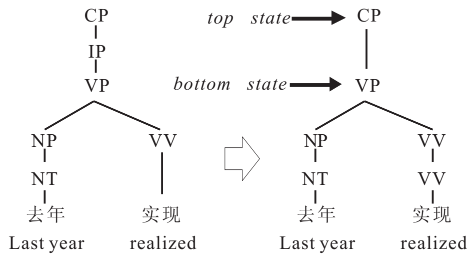

Figure 2: Unary rule normalization. Nonterminal-yield unary chains are collapsed to single unary rules. Identity unary rules are added to spans that have no unary rule.

minals are the words, and nonterminals are the POS tags and phrase types. In this paper, nonterminal is named **state** for short. A parse tree T of sentence $\mathbf{x}$ can be factorized into several one-level subtrees, each corresponding to a rule r.

In practice, binarization of rules is necessary to obtain cubic parsing time. That is, the right hand side of each rule should contain no more than 2 s-tates. We used right branching binarization, as illustrated in Figure 1. We did not use parent annotation, since we found it degraded the performance in our experiments (shown in Section 4). We used the same preprocessing step as (Harper and Huang, 2009), collapsing all the allowed nonterminal-yield unary chains to single unary rules. Therefore, all s-pans in the binarized trees contain no more than one unary rules. To facilitate decoding, we unify the form of spans so that each span contains exactly one u-nary rule. This is done by adding identity unary rules ($N \rightarrow N$) to spans that have no unary rule. These identity unary rules will be removed in evaluation. Hence, there are two states of a span: the top state $\overline{N}$ and the bottom state $\underline{N}$ that correspond to the left and right hand of the unary rule $r^{unary} = \overline{N} \rightarrow \underline{N}$ respectively, as shown in Figure 2.

Table 3 lists the feature templates we use for parsing. There are 4 feature sets: (i) bottom state features $\mathbf{f}_{bottom}(i,j,\mathbf{x},\underline{N}_{i,j})$, which depend on the bottom states; (ii) top state features $\mathbf{f}_{top}(i,j,\mathbf{x},\overline{N}_{i,j})$; (iii) unary rule features $\mathbf{f}_{unary}(i,j,\mathbf{x},r^{unary}_{i,j})$, which extract the transition information from bottom s-tates to top states; (iv) binary rule features $\mathbf{f}_{binary}(i,j,k,\mathbf{x},r^{binary}_{i,j,k} = \underline{N}_{i,j} \rightarrow \overline{N}_{i,k-1} + \overline{N}_{k,r})$, where $\overline{N}_{i,k-1}, \overline{N}_{k,r}$ are the top states of the left and right children.

The score function for a sentence $\mathbf{x}$ with parse tree T is defined as:

$$\begin{aligned} & score(\mathbf{x},T) = \\ & \sum_{\underline{N}_{i,j}\in T} \theta_{bottom}\cdot \mathbf{f}_{bottom}(i,j,\mathbf{x},\underline{N}_{i,j}) \\ & + \sum_{\overline{N}_{i,j}\in T} \theta_{top}\cdot \mathbf{f}_{top}(i,j,\mathbf{x},\overline{N}_{i,j}) \\ & + \sum_{r^{unary}_{i,j}\in T} \theta_{unary}\cdot \mathbf{f}_{unary}(i,j,\mathbf{x},r^{unary}_{i,j}) \\ & + \sum_{r^{binary}_{i,j,k}\in T} \theta_{binary}\cdot \mathbf{f}_{binary}(i,j,\mathbf{x},r^{binary}_{i,j,k}) \end{aligned}$$

where $\theta_{bottom}, \theta_{top}, \theta_{unary}, \theta_{binary}$ are the weight vectors of the four feature sets.

Given the training corpus $\{(\mathbf{x}_i, \tilde{T}_i)\}$, the learning task is to estimate the weight vectors so that for each sentence $\mathbf{x}_i$, the gold standard tree $\tilde{T}_i$ achieves the maximal score among all the possible trees. The perceptron algorithm is guaranteed to find the solution if it exists.

3.4 Joint Decoding

The three models described above are separately trained to make parameter estimation feasible as well as optimize each individual component. In test-

(3.1)	**Binary rule templates**			
	$\underline{N} \rightarrow \overline{N}_l + \overline{N}_r$			
	$\text{X}_l\ \text{X}_{m-1}\text{X}_r\ \text{len}_l\text{len}_r$		$\text{X}_l\ \text{X}_m\ \text{X}_r\ \text{len}_l\ \text{len}_r$	
	$\text{X}_l\ \text{X}_{m-1}\ \text{X}_r\ \text{word}_{m-1}$(ROOT)		X_l + $\text{X}_m\ \text{X}_r\ \text{word}_m$(ROOT)	
(3.2)	**Unary rule templates**			
	$\overline{N} \rightarrow \underline{N}$			
(3.3)	**Bottom state templates**			
	X_llen	X_rlen		
	$\text{X}_{l-2}\text{X}_{l-1}\ \text{X}_{r+1}\text{len}$		$\text{X}_{l-1}\ \text{X}_{r+1}\ \text{X}_{r+2}\text{len}$	
	$\text{wl}_l\text{wl}_r\text{X}_l\text{len}$	$\text{wl}_l\text{wl}_r\text{X}_r\text{len}$	$\text{X}_l\text{X}_r\text{wl}_l\text{len}$	$\text{X}_l\text{X}_r\text{wl}_r\text{len}$
	$\text{word}_l\text{word}_r\text{X}_l\text{X}_r\text{len}$	$\text{word}_l\text{word}_r\text{X}_l\text{X}_r$		
	X_{l-1}X_l(LEAF)	X_{l+1}X_l(LEAF)	X_lword_l(LEAF)	X_lwl_l(LEAF)
	$\text{X}_{l+a}\text{X}_{r+b}\text{len}$	$\text{word}_{l+a}\text{word}_{r+b}$	$-1 \leq a, b \leq 1$	
(3.3)	**Top state templates**			
	X_{l-1}X_l(LEAF)	X_{l+1}X_l(LEAF)	X_lword_l(LEAF)	X_lwl_l(LEAF)
	$\text{X}_{l+a}\text{X}_{r+b}\text{len}$	$\text{word}_{l+a}\text{word}_{r+b}$	$-1 \leq a, b \leq 1$	

Table 3: Feature templates for parsing, where X can be word, first and last character of word, first and last character bigram of word, POS tag. X_{l+a}/X_{r-a} denotes the first/last a^{th} X in the span, while X_{l-a}/X_{r+a} denotes the a^{th} X left/right to span. X_m is the first X of right child, and X_{m-1} is the last X of the left child. *len*, *len*$_l$, *len*$_r$ denote the length of the span, left child and right child respectively. *wl* is the length of word. ROOT/LEAF means the template can only generate the features for the root/initial span.

ing, we perform joint decoding to combine information from the three models. Parameters of word segmentation (θ_{seg}), POS tagging (θ_{pos}), and parsing models ($\theta_{parse} = \theta_{bottom \oplus top \oplus\ unary \oplus bianry}$) are scaled by three positive hyper-parameters α, β, and γ respectively, which control their contribution in the joint model. If $\alpha >> \beta >> \gamma$, then the joint model is equivalent to a pipeline model, in which there is no feedback from downstream models to upstream ones. For well tuned hyper-parameters, we expect that segmentation and POS tagging results can be improved by parsing information. The hyper-parameters are tuned on development data. In the following sections, for simplicity we drop α, β, γ, and just use $\theta_{seg}, \theta_{pos}, \theta_{parse}$ to represent the scaled parameters.

The basic idea of our decoding algorithm is to extend the CYK parsing algorithm so that it can deal with transition features in POS tagging and segmentation scores in word segmentation.

3.4.1 Algorithm

The joint decoding algorithm is shown in Algorithm 1. Given a sentence $\mathbf{x} = c_1, \ldots, c_l$, Line 0 calculates the scores of all possible words in the sentence using Eq(1). There are $l(l+1)/2$ word candidates in total.

Surrounding words are important features for POS tagging and parsing; however, they are unavailable because segmentation is incomplete before parsing. Therefore, we adopt pseudo surrounding features by simply fixing the context words as the single most likely ones. Given a word candidate $w_{i,j}$ from c_i to c_j, its previous word s' is the rightmost one in the best word sequence of $c_1, \ldots, c_{i-1}$, which can be obtained by dynamic programming. Recursively, the second word left to $w_{i,j}$ is the previous word of s'. The next word of $w_{i,j}$ is defined similarly. In Line 1, we use bidirectional Viterbi decoding to obtain all the surrounding words. In the forward direction, the algorithm starts from the first character boundary to the last, and finds the best previous word for the i^{th} character boundary b_i. In the backward direction, the algorithm starts from right to left, and finds the best next word of each b_i.

In Line 2, for each word candidate, we can calculate the score of each POS tag using state features in the POS tagging model, since the context words are available now. The score function of word $w_{i,j}$ with POS tag t is:

$$
\begin{aligned}
&score_{seg \oplus pos}(\mathbf{x}, i, j, t) = \\
&score_{seg}(\mathbf{x}, i, j) + \theta_{pos} \cdot \mathbf{f}_{pos}(\mathbf{x}, w_{i,j}, t) \quad (2)
\end{aligned}
$$

In Line 3, POS tags of surrounding words can be obtained similarly using bidirectional decoding.

Algorithm 1 Joint Word Segmentation, POS tagging, and Parsing Algorithm

Input: Sentence $\mathbf{x} = c_1, \ldots, c_l$, beam size B, scaled word segmentation model, POS tagging model and parsing model.

Output: Word sequence, POS tag sequence, and parse tree

```
 0: $\forall 0 \le i \le j \le l-1$,    calculate $score_{seg}(\mathbf{x}, i, j)$ using Equation (1)
 1: For each character boundary $b_i, 0 \le i \le l$, get the best previous and next words of $b_i$ using bidirectional
    Viterbi decoding
 2: $\forall 0 \le i \le j \le l-1, t \in \mathcal{T}$,    calculate $score_{seg \oplus pos}(\mathbf{x}, i, j, t)$ using Equation (2)
 3: $\forall b_i, 0 \le i \le l, t \in \mathcal{T}$,    get the best POS tags of words left/right to $b_i$ using bidirectional viterbi
    decoding.
 4: For each word candidate $w_{i,j}, 0 \le i \le j \le l-1$
 5:   For each bottom state $\underline{N}$, POS tag $t \in \mathcal{T}$        $\triangleleft$ step 1 (Line 5-7): get bottom states
 6:     $score_{bottom}(\mathbf{x}, i, j, w_{i,j}, t, \underline{N}) = score_{seg \oplus pos}(\mathbf{x}, i, j, t) + \theta_{bottom} \cdot \mathbf{f}_{bottom}(\mathbf{x}, i, j, w_{i,j}, t, \underline{N})$
 7:   Keep $B$ best $score_{bottom}$.
 8:   For each top state $\overline{N}$                  $\triangleleft$ step 2 (Line 8-9): get top states
 9:     $score_{top}(\mathbf{x}, i, j, w_{i,j}, t, \overline{N}) = \max_{\underline{N}} \{score_{bottom}(\mathbf{x}, i, j, w_{i,j}, t, \underline{N}) + \theta_{top} \cdot \mathbf{f}_{top}(\mathbf{x}, i, j, w_{i,j}, t, \underline{N})$
      $+\theta_{unary} \cdot \mathbf{f}_{unary}(\mathbf{x}, i, j, w_{i,j}, t, \overline{N} \to \underline{N})\}$
10: for $i = 0, \ldots, l-1$ do
11:   for $width = 1, \ldots, l-1$ do
12:   $j = i + width$
13:     for $k = i+1, \ldots, j$ do
14:     $score_{bottom}(\mathbf{x}, i, j, \mathbf{w}, \mathbf{t}, \underline{N}) = \max_{l,r} \{\underline{score_{top}}(\mathbf{x}, i, k-1, \mathbf{w}_l, \mathbf{t}_l, \overline{N}_l) + score_{top}(\mathbf{x}, k, j, \mathbf{w}_r, \mathbf{t}_r, \overline{N}_r)$
        $+\theta_{binary} \cdot \mathbf{f}_{binary}(\mathbf{x}, i, j, k, \mathbf{w}, \mathbf{t}, \underline{N} \to \overline{N}_r + \overline{N}_r) + \theta_{pos} \cdot \mathbf{f}_{pos}(t_l^{last} \to t_r^{first})$
        $+\theta_{bottom} \mathbf{f}_{bottom}(\mathbf{x}, i, j, \mathbf{w}, \mathbf{t}, \underline{N})\}$
15:     Keep $B$ best $score_{bottom}$                $\triangleleft$ step 1 (Line 14-15): get bottom states
16:     For each top state $\overline{N}$                $\triangleleft$ step 2 (Line 16-17): get top states
17:       $score_{top}(\mathbf{x}, i, j, \mathbf{w}, \mathbf{t}, \overline{N}) = \max_{\underline{N}} \{score_{bottom}(\mathbf{x}, i, j, \mathbf{w}, \mathbf{t}, \underline{N})$
        $+\theta_{unary} \cdot \mathbf{f}_{unary}(\mathbf{x}, i, j, \mathbf{w}, \mathbf{t}, \overline{N} \to \underline{N})\}$
18:     end for
19:   end for
20: end for
```

Line	0	1	2	3	6	9	14	15	Total Bound(w.r.t. l)
Complexity	l^2	l^2	$\|\mathcal{T}\|l^2$	$\|\mathcal{T}\|^2 l^2$	$\|\mathcal{T}\|Ml^2$	BMl^2	l^3MB^2	BMl^2	l^3MB^2

Table 4: Complexity Analysis of Algorithm 1.

That is, for $w_{i,j}$ with POS tag t, we use Viterbi algorithm to search the optimal POS tags of its left and right words.

In Lines 4-9, each word was initialized as a basic span. A span structure in the joint model is a 6-tuple: $S(i, j, \mathbf{w}, \mathbf{t}, \underline{N}, \overline{N})$, where i, j are the boundary indices, $\mathbf{w}, \mathbf{t}$ are the word sequence and POS sequence within the span respectively, and $\underline{N}, \overline{N}$ are the bottom and top states. There are two types of surrounding n-grams: one is inside the span, for example, the first word of a span, which can be obtained from $\mathbf{w}$; the other is outside the span, for example, the previous word of a span, which is obtained from the pseudo context information. The score of a basic s-pan depends on its corresponding word and POS pair score, and the weights of the active state and unary features.

To avoid enumerating the combination of the bottom and top states, initialization for each span is divided into 2 steps. In the first step, the score of every bottom state is calculated using bottom state features, and only the B best states are maintained (see Line 6-7). In the second step, top state features and unary rule features are used to get the score of each top state (Line 9), and only the top B states are preserved.

Similarly, there are two steps in the merge operation: $S(i, j, \mathbf{w}, \mathbf{t}, \underline{N}, \overline{N}) = S_l(i, k, \mathbf{w}_l, \mathbf{t}_l, \underline{N_l}, \overline{N_l}) + S_r(k+1, j, \mathbf{w}_r, \mathbf{t}_r, \underline{N_r}, \overline{N_r})$. The score of the bottom state $\underline{N}$ is calculated using binary features $\mathbf{f}_{binary}(\mathbf{x}, i, j, k, \mathbf{w}, \mathbf{t}, \underline{N} \rightarrow \overline{N}_r + \overline{N}_r)$, bottom state features $\mathbf{f}_{bottom}(\mathbf{x}, i, j, \mathbf{w}, \mathbf{t}, \underline{N})$, and POS tag transition features that depend on the boundary POS tags of S_l and S_r. See Line 14 of Algorithm 1, where t_l^{last} and t_r^{first} are the POS tags of the last word in the left child span and the first word in the right child span respectively.

3.4.2 Complexity analysis

Given a sentence of length l, the complexity for each line of Algorithm 1 is listed in Table 4, where $|\mathcal{T}|$ is the size of POS tag set, M is the number of states, and B is the beam size.

4 Experiments

4.1 Data

For comparison with other systems, we use the CTB5 corpus, which has been studied for Chinese word segmentation, POS tagging and parsing. We use the standard train/develop/test split of the data. Details are shown in Table 5.

	CTB files	# sent.	# words
Training	1-270 400-1151	18089	493,939
Develop	301-325	350	6,821
Test	271-300	348	8,008

Table 5: Training, development, and test data of CTB 5.

4.2 Evaluation Metric

We evaluate system performance on the individual tasks, as well as the joint tasks.[1] For word segmentation, three metrics are used for evaluation: precision (P), recall (R), and F-score (F) defined by 2PR/(P+R). Precision is the percentage of correct words in the system output. Recall is the percentage of words in gold standard annotations that are correctly predicted. For parsing, we use the standard parseval evaluation metrics: bracketing precision, recall and F-score.

[1]Note that the joint task refers to automatic segmentation and tagging/parsing. It can be achieved using a pipeline system or our joint decoding method.

For joint word segmentation and POS tagging, a word is correctly predicted if both the boundaries and the POS tag are correctly identified. For joint segmentation, POS tagging, and parsing task, when calculating the bracket scores using existing parseval tools, we need to consider possible word segmentation errors. To do this, we add the word boundary information in states – a bracket is correct only if its boundaries, label and word segmentation are all correct. One example is shown in Figure 3. Notice that identity unary rules are removed during evaluation. The basic spans are characters, not words, because the number of words in reference and prediction may be different. POS tags are removed since they do not affect the bracket scores. If the segmentation is perfect, then the bracket scores of the modified tree are exactly the same as the original tree. This is similar to evaluating parsing performance on speech transcripts with automatic sentence segmentation (Roark et al., 2006).

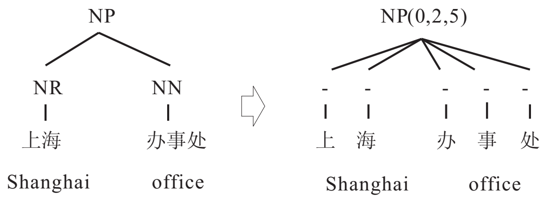

Figure 3: Boundary information is added to states to calculate the bracket scores in the face of word segmentation errors. Left: the original parse tree, Right: the converted parse tree. The numbers in the brackets are the indices of the character boundaries based on word segmentation.

4.3 Parameter Estimation

We train three submodels using the gold features, that is, POS tagger is trained using the perfect segmentation, and parser is trained using perfect segmentation and POS tags. Some studies reported that better performance may be achieved by training subsequent models using representative output of the preceding models (Che et al., 2009). Hence for comparison we trained another parser using automatically generated POS tags obtained from 10-fold cross validation, but did not find significant difference between these two parsers when testing on the perfectly segmented development dataset. Therefore we use the parser trained with perfect POS tags for the joint task.

Three hyper-parameters, α, β, and γ, are tuned on development data using a heuristic search. Parameters that achieved the best joint parsing result are selected. In the search, we fixed $\gamma = 1$ and varied α, β. First, we set $\beta = 1$, and enumerate $\alpha = \frac{1}{4}, \frac{1}{2}, 1, 2, \ldots$, and choose the best α^*. Then, we set $\alpha = \alpha^*$ and vary $\beta = \frac{1}{4}, \frac{1}{2}, 1, 2, \ldots$, and select the best β^*.

Table 6 lists the parameters we used for training the submodels, as well as the hyper-parameters for joint decoding.

Model	Parameter	Value
Character based word segmentor	Gaussian prior	0.01
	# Feature	3,875,802
Word based word segmentor	Gaussian prior	0.01
	# Feature	312,533
POS tagger	Gaussian prior	0.1
	# Feature	48,608,802
Parser	Iteration Number	10
	# Feature	49,369,843
Joint	Hyper-parameter α	4
	Hyper-parameter β	0.5
	Hyper-parameter γ	1
	Beam Size B	20

Table 6: Parameters used in our system.

4.4 Experimental Results

In this section we first show that our sub-models are better than or comparable to state-of-the-art systems, and then the joint model is superior to the pipeline approach.

4.4.1 Evaluating Sub-models

Table 7 shows word segmentation results using our word segmentation submodel, in comparison to a few state-of-the-art systems. For our segmentor, we show results for two variants: one removes transition features as described in Section 3.1, the other uses CRFs to learn the weights of transition features. We can see that our system is competitive with all the others except Sun's that used additional idiom resources. Our two word segmentors have similar performance. Since the one without transition features can be naturally integrated into the joint system, we use it in the following joint tasks.

System	P	R	F
(Jiang et al., 2008b)	-	-	97.74
(Jiang et al., 2008a)	-	-	97.85
(Kruengkrai et al., 2009)	97.46	98.29	97.87
(Zhang and Clark, 2010)	-	-	97.78
(Zhang and Clark, 2011)	-	-	97.78
(Sun, 2011)	-	-	98.17
Ours (w/o transition features)	97.45	98.24	97.85
Ours (with transition features)	97.44	98.23	97.84

Table 7: Word segmentation results.

For the POS tagging only task that takes gold standard word segmentation as input, we have two systems. One uses the linear chain CRFs as described in Section 3.2, the other is obtained using the parser described in Section 3.3 – the parser generates POS tag hypotheses when POS tag features are not used. The POS tagging accuracy is 95.53% and 95.10% using these two methods respectively. The better performance from the former system may be because the local label dependency is more helpful for POS tagging than the long distance dependencies that might be noisy. This result also confirms our choice of using an independent POS tagger for the sub-model, rather than relying on a parser for POS tagging. However, since there are no reported results for this setup, we demonstrate the competence of our POS tagger using the joint word segmentation and POS tagging task. Table 8 shows the performance of a few systems along with ours, all using the pipeline approach where automatic segmentation is followed by POS tagging. We can see that our POS tagger is comparable to the others.

System	P	R	F
(Jiang et al., 2008b)	-	-	93.37
(Jiang et al., 2008a)	-	-	93.41
(Kruengkrai et al., 2009)	93.28	94.07	93.67
(Zhang and Clark, 2010)	-	-	93.67
(Zhang and Clark, 2011)	-	-	93.67
(Sun, 2011)	-	-	94.02
Ours (pipeline)	93.10	93.96	93.53

Table 8: Results for the joint word segmentation and POS tagging task.

For parsing, Table 9 presents the parsing result on gold standard segmented sentence. Notice that the result of (Harper and Huang, 2009; Zhang and

Clark, 2011) are not directly comparable to ours, as they used a different data split. The best published system result on CTB5 is Petrov and Klein's, which used PCFG with latent Variables. Our system performs better mainly because it benefits from a large amount of features.

System	LP	LR	F
(Petrov and Klein, 2007)	84.8	81.9	83.3
(Jiang et al., 2009)	-	-	82.35
(Harper and Huang, 2009)*	83.22	82.84	83.03
(Zhang and Clark, 2011)*	78.6	78.0	78.3
Ours	84.57	83.68	**84.13**
Ours (w/ parent annotation)	83.35	82.73	83.04
Ours (no POS tag feature)	83.49	82.97	83.23

Table 9: Parsing results using gold standard word segmentation.

For our parser, besides the model described in Section 3.3, we tried two variations: one does not use the automatic POS tag features, the other one is learned on the parent annotated training data. The results in Table 9 show that there is a performance degradation when using parent annotation. This may be due to the introduction of a large number of states, resulting in sparse features. We also notice that with the help of the POS tag information, even automatically generated, the parser gained 0.9% improvement in F-score. This demonstrates the advantage of using a better independent POS tagger and incorporating it in parsing.

Finally Table 10 shows the results for the three tasks using our joint decoding method in comparison to the pipeline method. We can see that the joint model outperforms the pipeline one. This is mainly because of a better parsing module as well as joint decoding. In the table we also include results of (Jiang et al., 2009), which is the only reported joint parsing result we found using the same data split on CTB5. They achieved 80.28% parsing F-score using automatic word segmentation. Their adapted system Jiang09$^+$ leveraged additional corpus to improve Chinese word segmentation, resulting in an F-score of 81.07%. Our system has better performance than these.

System	Task	P	R	F
Jiang09	Parse	-	-	80.28
Jiang09$^+$	Parse	-	-	81.07
Ours Pipeline	Seg.	97.45	98.24	97.85
	POS	93.10	93.96	93.53
	Parse	81.87	81.65	81.76
Ours Joint	Seg.	97.56	98.36	97.96
	POS	93.43	94.20	93.81
	Parse	83.03	82.66	**82.85**

Table 10: Results for the joint segmentation, tagging, and parsing task using pipeline and joint models.

4.5 Error Analysis

We compared the results from the pipeline and our joint decoding systems in order to understand the impact of the joint model on word segmentation and POS tagging. We notice that the joint model tend to generate more words than the pipeline model. For example, "巴尔一行" is one word in the pipeline model, but correctly segmented as two words "巴尔/一行" in the joint model. This tendency of segmentation also makes it fail to recognize some long words, especially OOV words. For example, "事实上" is segmented as "事实/上". In the data set, we find that, the joint model corrected 10 missing boundaries over the pipeline method, and introduced 3 false positive segmentation errors.

For the analysis of POS tags, we only examined the words that are correctly segmented by both the pipeline and the joint models. Table 11 shows the increase and decrease of error patterns of the joint model over the pipeline POS tagger. An error pattern "X → Y" means that the word whose true tag is 'X' is assigned a tag 'Y'. All the patterns are ranked in descending order of the reduction/increase of the error number. We can see that the joint model has a clear advantage in the disambiguation of {VV, NN} and {DEG, DEC}, which results in the overall improved performance. In contrast, the joint method performs worse on ambiguous POS pairs such as $\{NN, NR\}$. This observation is similar to those reported by (Li et al., 2011; Hatori et al., 2011).

5 Conclusion

In this paper, we proposed a new algorithm for joint Chinese word segmentation, POS tagging, and parsing. Our algorithm is an extension of the CYK

error pattern	#	↓	error pattern	#	↑
NN→ VV	47	19	NN→ NR	15	12
VV→ NN	42	13	NR→ NN	7	5
DEG→ DEC	23	10	JJ→ P	1	4
NN→ JJ	29	8	NN→ DT	2	4
DEC→ DEG	11	4	P→ VV	3	2
JJ→ NN	12	4	AD→ NN	1	2

Table 11: POS tagging error patterns. # means the error number of the corresponding pattern made by the pipeline tagging model. ↓ and ↑ mean the error number reduced or increased by the joint model.

parsing method. The sub-models are independently trained for the three tasks to reduce model complexity and optimize individual sub-models. Our experiments demonstrate the advantage of the joint models. In the future work, we will compare this joint model to the pipeline approach that uses multiple candidates or soft decisions in the early modules. We will also investigate methods for joint learning as well as ways to speed up the joint decoding algorithm.

Acknowledgments The authors thank Zhongqiang Huang for his help with experiments. This work is partly supported by DARPA under Contract No. HR0011-12-C-0016. Any opinions expressed in this material are those of the authors and do not necessarily reflect the views of DARPA.

References

Wanxiang Che, Zhenghua Li, Yongqiang Li, Yuhang Guo, Bing Qin, and Ting Liu. 2009. Multilingual dependency-based syntactic and semantic parsing. In *Proceedings of CoNLL 09*, pages 49–54.

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

Jenny Rose Finkel, Alex Kleeman, and Christopher D. Manning. 2008. Efficient, feature-based, conditional random field parsing. In *Proceedings of ACL-08: HLT*, pages 959–967.

Yoav Goldberg and Reut Tsarfaty. 2008. A single generative model for joint morphological segmentation and syntactic parsing. In *Proceedings of ACL 2008: HLT*, pages 371–379.

Spence Green and Christopher D. Manning. 2010. Better arbic parsing: Baselines, evaluations, and analysis. In *Proceedings of Coling 2010*, pages 394–402.

Mary Harper and Zhongqiang Huang. 2009. Chinese statistical parsing. In *Gale Book*.

Jun Hatori, Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2011. Incremental joint pos tagging and dependency parsing in chinese. In *Proceedings of IJCNLP 2011*, pages 1216–1224.

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 2008: HLT*, pages 897–904.

Wenbin Jiang, Haitao Mi, and Qun Liu. 2008b. Word lattice reranking for chinese word segmentation and part-of-speech tagging. In *Proceedings of Coling 2008*, pages 385–392.

Wenbin Jiang, Liang Huang, and Qun Liu. 2009. Automatic adaptation of annotation standards: Chinese word segmentation and pos tagging – a case study. In *Proceedings of ACL-IJCNLP 2009*, pages 522–530.

Guangjin Jin and Xiao Chen. 2008. The fourth international chinese language processing bakeoff: Chinese word segmentation, named entity recognition and chinese pos tagging. In *Proceedings of Sixth SIGHAN Workshop on Chinese Language Processing*.

Canasai Kruengkrai, Kiyotaka Uchimoto, Jun'ichi Kazama, Yiou Wang, Kentaro Torisawa, and Hitoshi Isahara. 2009. An error-driven word-character hybrid model for joint chinese word segmentation and pos tagging. In *Proceedings of ACL 2009*, pages 513–521.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML 2001*, pages 282–289.

John Lee, Jason Naradowsky, and David A. Smith. 2011. A discriminative model for joint morphological disambiguation and dependency parsing. In *Proceedings ACL 2011: HLT*, pages 885–894.

Zhenghua Li, Min Zhang, Wanxiang Che, Ting Liu, Wenliang Chen, and Haizhou Li. 2011. Joint models for chinese pos tagging and dependency parsing. In *Proceedings of EMNLP 2011*, pages 1180–1191.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Proceedings of NAACL 2007*, pages 404–411.

Xian Qian, Qi Zhang, Yaqian Zhou, Xuanjing Huang, and Lide Wu. 2010. Joint training and decoding using virtual nodes for cascaded segmentation and tagging tasks. In *Proceedings of EMNLP 2010*, pages 187–195.

Brian Roark, Mary Harper, Eugene Charniak, Bonnie Dorr, Mark Johnson, Jeremy G. Kahn, Yang Liu, Mari Ostendorf, John Hale, Anna Krasnyanskaya, Matthew Lease, Izhak Shafran, Matthew Snover, Robin Stewart, Lisa Yung, and Lisa Yung. 2006. Sparseval: Evaluation metrics for parsing speech. In *Proceedings Language Resources and Evaluation (LREC)*.

Sunita Sarawagi and William W. Cohen. 2004. Semi-markov conditional random fields for information extraction. In *Proceedings of NIPS 2004*.

Weiwei Sun. 2011. A stacked sub-word model for joint chinese word segmentation and part-of-speech tagging. In *Proceedings of ACL 2011*, pages 1385–1394.

Yue Zhang and Stephen Clark. 2008. Joint word segmentation and POS tagging using a single perceptron. In *Proceedings of ACL 2008: HLT*, pages 888–896.

Yue Zhang and Stephen Clark. 2010. A fast decoder for joint word segmentation and POS-tagging using a single discriminative model. In *Proceedings of EMNLP 2010*, pages 843–852.

Yue Zhang and Stephen Clark. 2011. Syntactic processing using the generalized perceptron and beam search. *Comput. Linguist.*, 37(1):105–151.

Ruiqiang Zhang, Genichiro Kikui, and Eiichiro Sumita. 2006. Subword-based tagging for confidence-dependent chinese word segmentation. In *Proceedings of the COLING/ACL 2006*, pages 961–968.

Hai Zhao and Chunyu Kit. 2008. Unsupervised segmentation helps supervised learning of character tagging forword segmentation and named entity recognition. In *Proceedings of Sixth SIGHAN Workshop on Chinese Language Processing*, pages 106–111.