Joint Models for Chinese POS Tagging and Dependency Parsing

Zhenghua Li[†], Min Zhang[‡], Wanxiang Che[†], Ting Liu[†], Wenliang Chen[‡] and Haizhou Li[‡]
[†]Research Center for Social Computing and Information Retrieval
Harbin Institute of Technology, China
{lzh,car,tliu}@ir.hit.edu.cn
[‡]Institute for Infocomm Research, Singapore
{mzhang,wechen,hli}@i2r.a-star.edu.sg

Abstract

Part-of-speech (POS) is an indispensable feature in dependency parsing. Current research usually models POS tagging and dependency parsing independently. This may suffer from error propagation problem. Our experiments show that parsing accuracy drops by about 6% when using automatic POS tags instead of gold ones. To solve this issue, this paper proposes a solution by jointly optimizing POS tagging and dependency parsing in a unique model. We design several joint models and their corresponding decoding algorithms to incorporate different feature sets. We further present an effective pruning strategy to reduce the search space of candidate POS tags, leading to significant improvement of parsing speed. Experimental results on Chinese Penn Treebank 5 show that our joint models significantly improve the state-of-the-art parsing accuracy by about 1.5%. Detailed analysis shows that the joint method is able to choose such POS tags that are more helpful and discriminative from parsing viewpoint. This is the fundamental reason of parsing accuracy improvement.

1 Introduction

In dependency parsing, features consisting of part-of-speech (POS) tags are very effective, since pure lexical features lead to severe data sparseness problem. Typically, POS tagging and dependency parsing are modeled in a pipelined way. However, the pipelined method is prone to error propagation, especially for Chinese. Due to the lack of morphological features, Chinese POS tagging is even harder than other languages such as English. The state-of-the-art accuracy of Chinese POS tagging is about 93.5%, which is much lower than that of English (about 97% (Collins, 2002)). Our experimental results show that parsing accuracy decreases by about 6% on Chinese when using automatic POS tagging results instead of gold ones (see Table 3 in Section 5). Recent research on dependency parsing usually overlooks this issue by simply adopting gold POS tags for Chinese data (Duan et al., 2007; Zhang and Clark, 2008b; Huang and Sagae, 2010). In this paper, we address this issue by jointly optimizing POS tagging and dependency parsing.

Joint modeling has been a popular and effective approach to simultaneously solve related tasks. Recently, many successful joint models have been proposed, such as joint tokenization and POS tagging (Zhang and Clark, 2008a; Jiang et al., 2008; Kruengkrai et al., 2009), joint lemmatization and POS tagging (Toutanova and Cherry, 2009), joint tokenization and parsing (Cohen and Smith, 2007; Goldberg and Tsarfaty, 2008), joint named entity recognition and parsing (Finkel and Manning, 2009), joint parsing and semantic role labeling (SRL) (Li et al., 2010), joint word sense disambiguation and SRL (Che and Liu, 2010), joint tokenization and machine translation (MT) (Dyer, 2009; Xiao et al., 2010) and joint parsing and MT (Liu and Liu, 2010). Note that the aforementioned "parsing" all refer to *constituent parsing*.

As far as we know, there are few successful models for jointly solving *dependency parsing* and other tasks. Being facilitated by Conference on Computational Natural Language Learning (CoNLL) 2008 and 2009 shared tasks, several joint models of dependency parsing and SRL have been proposed. Nevertheless, the top-ranked systems all adopt pipelined approaches (Surdeanu et al., 2008;

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1180–1191,
Edinburgh, Scotland, UK, July 27–31, 2011.

Hajič et al., 2009). Theoretically, joint modeling of POS tagging and dependency parsing should be helpful to the two individual tasks. On the one hand, syntactic information can help resolve some POS ambiguities which are difficult to handle for the sequential POS tagging models. On the other hand, more accurate POS tags should further improve dependency parsing.

For joint POS tagging and dependency parsing, the major issue is to design effective decoding algorithms to capture rich features and efficiently search out the optimal results from a huge hypothesis space.[1] In this paper, we propose several dynamic programming (DP) based decoding algorithms for our joint models by extending existing parsing algorithms. We also present effective pruning techniques to speed up our decoding algorithms. Experimental results on Chinese Penn Treebank show that our joint models can significantly improve the state-of-the-art parsing accuracy by about 1.5%.

The remainder of this paper is organized as follows. Section 2 describes the pipelined method, including the POS tagging and parsing models. Section 3 discusses the joint models and the decoding algorithms, while Section 4 presents the pruning techniques. Section 5 reports the experimental results and error analysis. We review previous work closely related to our method in Section 6, and conclude this paper in Section 7.

2 The Baseline Pipelined Method

Given an input sentence $\mathbf{x} = w_1...w_n$, we denote its *POS tag sequence* by $\mathbf{t} = t_1...t_n$, where $t_i \in \mathcal{T}$, $1 \leq i \leq n$, and $\mathcal{T}$ is the POS tag set. A *dependency tree* is denoted by $\mathbf{d} = \{(h, m) : 0 \leq h \leq n, 0 < m \leq n\}$, where (h, m) represents a dependency $w_h \rightarrow w_m$ whose *head* word (or *father*) is w_h and *modifier* (or *child*) is w_m. w_0 is an artificial root token which is used to simplify the formalization of the problem.

The pipelined method treats POS tagging and dependency parsing as two cascaded problems. First, an optimal POS tag sequence $\hat{\mathbf{t}}$ is determined.

$$\hat{\mathbf{t}} = \arg\max_{\mathbf{t}} Score_{\text{pos}}(\mathbf{x}, \mathbf{t})$$

Then, an optimal dependency tree $\hat{\mathbf{d}}$ is determined based on $\mathbf{x}$ and $\hat{\mathbf{t}}$.

$$\hat{\mathbf{d}} = \arg\max_{\mathbf{d}} Score_{\text{syn}}(\mathbf{x}, \hat{\mathbf{t}}, \mathbf{d})$$

2.1 POS Tagging

POS tagging is a typical sequence labeling problem. Many models have been successfully applied to sequence labeling problems, such as maximum-entropy (Ratnaparkhi, 1996), conditional random fields (CRF) (Lafferty et al., 2001) and perceptron (Collins, 2002). We use perceptron to build our POS tagging baseline for two reasons. Firstly, as a linear model, perceptron is simple, fast, and effective. It is competitive to CRF in tagging accuracy but requires much less training time (Shen et al., 2007). Secondly, perceptron has been successfully applied to dependency parsing as well (Koo and Collins, 2010). In this paper, perceptron is used in all models including the POS tagging model, the dependency parsing models and the joint models.

In a perceptron, the score of a tag sequence is

$$Score_{\text{pos}}(\mathbf{x}, \mathbf{t}) = \mathbf{w}_{\text{pos}} \cdot \mathbf{f}_{\text{pos}}(\mathbf{x}, \mathbf{t})$$

where $\mathbf{f}_{\text{pos}}(\mathbf{x}, \mathbf{t})$ refers to the feature vector and $\mathbf{w}_{\text{pos}}$ is the corresponding weight vector.

For POS tagging features, we follow the work of Zhang and Clark (2008a). Three feature sets are considered: *POS unigram*, *bigram* and *trigram* features. For brevity, we will refer to the three sets as $\underline{w_i\ t_i}$, $\underline{t_{i-1}\ t_i}$ and $\underline{t_{i-2}\ t_{i-1}\ t_i}$.

Given $\mathbf{w}_{\text{pos}}$, we adopt the Viterbi algorithm to get the optimal tagging sequence.

2.2 Dependency Parsing

Recently, graph-based dependency parsing has gained more and more interest due to its state-of-the-art accuracy. Graph-based dependency parsing views the problem as finding the highest scoring tree from a directed graph. Based on dynamic programming decoding, it can efficiently find an optimal tree in a huge search space. In a graph-based model, the

[1] It should be noted that it is straightforward to simultaneously do POS tagging and constituent parsing, as POS tags can be regarded as non-terminals in the constituent structure (Levy and Manning, 2003). In addition, Rush et al. (2010) describes an efficient and simple inference algorithm based on dual decomposition and linear programming relaxation to combine a lexicalized constituent parser and a trigram POS tagger.

score of a dependency tree is factored into scores of small parts (subtrees).

$$Score_{\text{syn}}(\mathbf{x}, \mathbf{t}, \mathbf{d}) = \mathbf{w}_{\text{syn}} \cdot \mathbf{f}_{\text{syn}}(\mathbf{x}, \mathbf{t}, \mathbf{d}) = \sum_{p \subseteq \mathbf{d}} Score_{\text{syn}}(\mathbf{x}, \mathbf{t}, p)$$

where p is a scoring part which contains one or more dependencies in the dependency tree $\mathbf{d}$. Figure 1 shows different types of scoring parts used in current graph-based models.

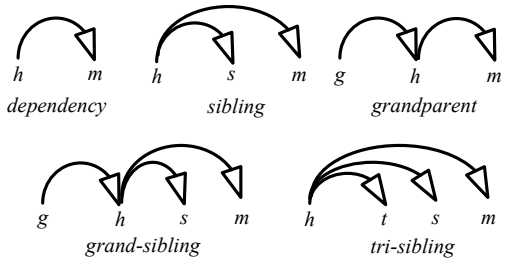

Figure 1: Different types of scoring parts used in current graph-based models (Koo and Collins, 2010).

Eisner (1996) proposes an $O(n^3)$ decoding algorithm for dependency parsing. Based on the algorithm, McDonald et al. (2005) propose the *first-order* model, in which the scoring parts only contains dependencies. The *second-order* model of McDonald and Pereira (2006) incorporates sibling parts and also needs $O(n^3)$ parsing time. The *second-order* model of Carreras (2007) incorporates both sibling and grandparent parts, and needs $O(n^4)$ parsing time. However, the grandparent parts are restricted to those composed of *outermost grandchildren*. Koo and Collins (2010) propose efficient decoding algorithms of $O(n^4)$ for *third-order* models. In their paper, they implement two versions of third-order models, Model 1 and Model 2 according to their naming. Model 1 incorporates only grand-sibling parts, while Model 2 incorporates both grand-sibling and tri-sibling parts. Their experiments on English and Czech show that Model 1 and Model 2 obtain nearly the same parsing accuracy. Therefore, we use Model 1 as our third-order model in this paper.

We use three versions of graph-based dependency parsing models.

- The first-order model (O1): the same with McDonald et al. (2005).
- The second-order model (O2): the same with Model 1 in Koo and Collins (2010), but without using grand-sibling features.[2]
- The third-order model (O3): the same with Model 1 in Koo and Collins (2010).

We adopt linear models to define the score of a dependency tree. For the third-order model, the score of a dependency tree is represented as:

$$\begin{aligned} Score_{\text{syn}}(\mathbf{x}, \mathbf{t}, \mathbf{d}) = & \sum_{\{(h,m)\} \subseteq \mathbf{d}} \mathbf{w}_{\text{dep}} \cdot \mathbf{f}_{\text{dep}}(\mathbf{x}, \mathbf{t}, h, m) \\ & + \sum_{\{(h,s)(h,m)\} \subseteq \mathbf{d}} \mathbf{w}_{\text{sib}} \cdot \mathbf{f}_{\text{sib}}(\mathbf{x}, \mathbf{t}, h, s, m) \\ & + \sum_{\{(g,h),(h,m)\} \subseteq \mathbf{d}} \mathbf{w}_{\text{grd}} \cdot \mathbf{f}_{\text{grd}}(\mathbf{x}, \mathbf{t}, g, h, m) \\ & + \sum_{\{(g,h),(h,s),(h,m)\} \subseteq \mathbf{d}} \mathbf{w}_{\text{gsib}} \cdot \mathbf{f}_{\text{gsib}}(\mathbf{x}, \mathbf{t}, g, h, s, m) \end{aligned}$$

For the first- and second-order models, the above formula is modified by deactivating extra parts.

For parsing features, we follow standard practice for graph-based dependency parsing (McDonald, 2006; Carreras, 2007; Koo and Collins, 2010). Since these features are highly related with our joint decoding algorithms, we summarize the features as follows.

- Dependency Features, $\mathbf{f}_{\text{dep}}(\mathbf{x}, \mathbf{t}, h, m)$
  - Unigram Features: $\underline{w_h t_h\ dir}$, $\underline{w_m t_m\ dir}$
  - Bigram Features: $\underline{w_h t_h\ w_m t_m\ dir\ dist}$
  - In Between Features: $\underline{t_h\ t_b\ t_m\ dir\ dist}$
  - Surrounding Features: $\underline{t_{h-1}\ t_h\ t_{h+1}\ t_{m-1}\ t_m\ t_{m+1}\ dir\ dist}$
- Sibling Features, $\mathbf{f}_{\text{sib}}(\mathbf{x}, \mathbf{t}, h, s, m)$ $\underline{w_h\ t_h\ w_s\ t_s\ w_m\ t_m\ dir}$
- Grandparent Features, $\mathbf{f}_{\text{grd}}(\mathbf{x}, \mathbf{t}, g, h, m)$ $\underline{w_g\ t_g\ w_h\ t_h\ w_m\ t_m\ dir\ gdir}$
- Grand-sibling Features, $\mathbf{f}_{\text{gsib}}(\mathbf{x}, \mathbf{t}, g, h, s, m)$ $\underline{w_g\ t_g\ w_h\ t_h\ w_s\ t_s\ w_m\ t_m\ dir\ gdir}$

[2]This second-order model incorporates grandparent features composed of all grandchildren rather than just outermost ones, and outperforms the one of Carreras (2007) according to the results in Koo and Collins (2010).

where b denotes an index between h and m; dir and $dist$ are the direction and distance of (h, m); $gdir$ is the direction of (g, h). We also use back-off features by generalizing from very specific features over word forms, POS tags, directions and distances to less sparse features over just POS tags or considering fewer nodes. To avoid producing too many sparse features, at most two word forms are used at the same time in sibling, grandparent and grand-sibling features, while POS tags are used instead for other nodes; meanwhile, at most four POS tags are considered at the same time for surrounding features.

3 Joint Models

In the joint method, we aim to simultaneously solve the two problems.

$$(\hat{\mathbf{t}}, \hat{\mathbf{d}}) = \arg\max_{\mathbf{t},\mathbf{d}} Score_{\text{joint}}(\mathbf{x}, \mathbf{t}, \mathbf{d})$$

Under the linear model, the score of a tagged dependency tree is:

$$\begin{aligned} Score_{\text{joint}}(\mathbf{x}, \mathbf{t}, \mathbf{d}) &= Score_{\text{pos}}(\mathbf{x}, \mathbf{t}) \\ &\quad + Score_{\text{syn}}(\mathbf{x}, \mathbf{t}, \mathbf{d}) \\ &= \mathbf{w}_{\text{pos}\oplus\text{syn}} \cdot \mathbf{f}_{\text{pos}\oplus\text{syn}}(\mathbf{x}, \mathbf{t}, \mathbf{d}) \end{aligned}$$

where $\mathbf{f}_{\text{pos}\oplus\text{syn}}(.)$ means the concatenation of $\mathbf{f}_{\text{pos}}(.)$ and $\mathbf{f}_{\text{syn}}(.)$. Under the joint model, the weights of POS and syntactic features, $\mathbf{w}_{\text{pos}\oplus\text{syn}}$, are simultaneously learned. We expect that POS and syntactic features can interact each other to determine an optimal joint result.

Similarly to the baseline dependency parsing models, we define the *first-*, *second-*, and *third-order* joint models according to the syntactic features contained in $\mathbf{f}_{\text{syn}}(.)$.

In the following, we propose two versions of joint models which can capture different feature sets and have different complexity.

3.1 Joint Models of Version 1

The crucial problem for the joint method is to design effective decoding algorithms to capture rich features and efficiently search out the optimal results from a huge hypothesis space. Eisner (2000) describes a preliminary idea to handle *polysemy* by extending parsing algorithms. Based on this idea, we extend decoding algorithms of McDonald et al. (2005) and Koo and Collins (2010), and propose two DP based decoding algorithms for our joint models of version 1.

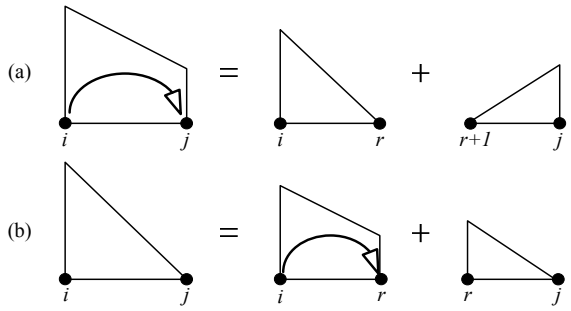

Figure 2: The DP structures and derivations of the first-order decoding algorithm of joint models of version 1. We omit symmetric right-headed versions for brevity. Trapezoids denote *incomplete spans*. Triangles denote *complete spans*. Solid circles denote POS tags of the corresponding indices.

The decoding algorithm of O1: As shown in Figure 2, the first-order joint decoding algorithm utilizes two types of dynamic programming structures. (1) ***I**ncomplete spans* consist of a dependency and the region between the head and modifier; (2) ***C**omplete spans* consist of a headword and its descendants on one side. Each span is recursively created by combining two smaller and adjacent spans in a bottom-up fashion.

The pseudo codes are given in Algorithm 1. $I_{(i,j)(t_i,t_j)}$ denotes an incomplete span from i to j whose boundary POS tags are t_i and t_j. $C_{(i,j)(t_i,t_j)}$ refers to a complete span from i to j whose boundary POS tags are t_i and t_j. Conversely, $I_{(j,i)(t_j,t_i)}$ and $C_{(j,i)(t_j,t_i)}$ represent spans of the other direction. Note that in these notations the first argument index always refers to the *head* of the span.

Line 6 corresponds to the derivation in Figure 2-(a). $Score_{\text{joint}}(\mathbf{x}, t_i, t_r, t_{r+1}, t_j, p = \{(i, j)\})$ captures the joint features invented by this combination, where $p = \{(i, j)\}$ means that the newly observed scoring part is the dependency (i, j). The syntactic features, denoted by $\mathbf{f}_{\text{syn}}(\mathbf{x}, t_i, t_j, i, j)$, can only incorporate *syntactic unigram and bigram* features. The surrounding and in between features are unavailable, because the context POS tags, such as t_b and t_{i-1}, are not contained in the DP struc-

Algorithm 1 The first-order joint decoding algorithm of version 1

1: $\forall 0 \le i \le n, t_i \in \mathcal{T}\ \ C_{(i,i)(t_i,t_i)} = 0$ ◁ initialization
2: **for** $w = 1..n$ **do** ◁ span width
3: **for** $i = 0..(n-w)$ **do** ◁ span start index
4: $j = i + w$ ◁ span end index
5: **for** $(t_i, t_j) \in \mathcal{T}^2$ **do**
6: $I_{(i,j)(t_i,t_j)} = \max_{i \le r < j} \max_{(t_r,t_{r+1}) \in \mathcal{T}^2} \{C_{(i,r)(t_i,t_r)} + C_{(j,r+1)(t_j,t_{r+1})} + Score_{\text{joint}}(\mathbf{x}, t_i, t_r, t_{r+1}, t_j, p = \{(i,j)\})\}$
7: $I_{(j,i)(t_j,t_i)} = \max_{i \le r < j} \max_{(t_r,t_{r+1}) \in \mathcal{T}^2} \{C_{(i,r)(t_i,t_r)} + C_{(j,r+1)(t_j,t_{r+1})} + Score_{\text{joint}}(\mathbf{x}, t_i, t_r, t_{r+1}, t_j, p = \{(j,i)\})\}$
8: $C_{(i,j)(t_i,t_j)} = \max_{i < r \le j} \max_{t_r \in \mathcal{T}} \{I_{(i,r)(t_i,t_r)} + C_{(r,j)(t_r,t_j)} + Score_{\text{joint}}(\mathbf{x}, t_i, t_r, t_j, p = \emptyset)\}$
9: $C_{(j,i)(t_j,t_i)} = \max_{i \le r < j} \max_{t_r \in \mathcal{T}} \{C_{(r,i)(t_r,t_i)} + I_{(j,r)(t_j,t_r)} + Score_{\text{joint}}(\mathbf{x}, t_i, t_r, t_j, p = \emptyset)\}$
10: **end for**
11: **end for**
12: **end for**

tures. Therefore, we adopt ***pseudo surrounding and in between features*** by simply fixing the context POS tags as the single most likely ones (McDonald, 2006). Taking the in between features as an example, we use $\underline{t_i\ \hat{t}_b\ t_j\ dir\ dist}$ instead, where $\hat{t}_b$ is the 1-best tag determined by the baseline POS tagger. The POS features, denoted by $\mathbf{f}_{\text{pos}}(\mathbf{x}, t_i, t_r, t_{r+1}, t_j)$, can only incorporate all *POS unigram and bigram* features.[3] Similarly, we use ***pseudo POS trigram features*** such as $\underline{\hat{t}_{r-1}\ t_r\ t_{r+1}}$.

Line 8 corresponds to the derivation in Figure 2-(b). Since this combination invents no scoring part ($p = \emptyset$), $Score_{\text{joint}}(\mathbf{x}, t_i, t_r, t_j, p = \emptyset)$ is only composed of POS features.[4]

Line 7 and Line 9 create spans in the opposite direction, which can be analogously illustrated. The space and time complexity of the algorithm are respectively $O(n^2q^2)$ and $O(n^3q^4)$, where $q = |\mathcal{T}|$.[5]

The decoding algorithm of O2 & O3: Figure 3 illustrates the second- and third-order decoding algorithm of joint models of version 1. A new kind of span, named the *sibling span*, is used to capture sibling structures. Furthermore, each span is augmented with a grandparent-index to capture both grandparent and grand-sibling structures. It is straightforward to derive the pseudo codes of the al-

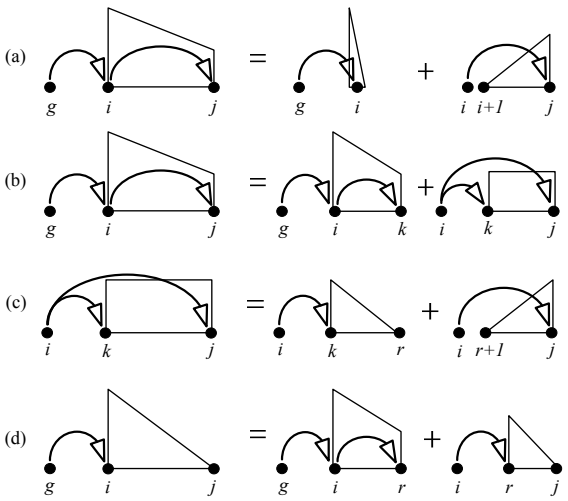

Figure 3: The DP structures and derivations of the second- and third-order joint decoding algorithm of version 1. For brevity, we elide the right-headed and right-grandparented versions. Rectangles represent sibling spans.

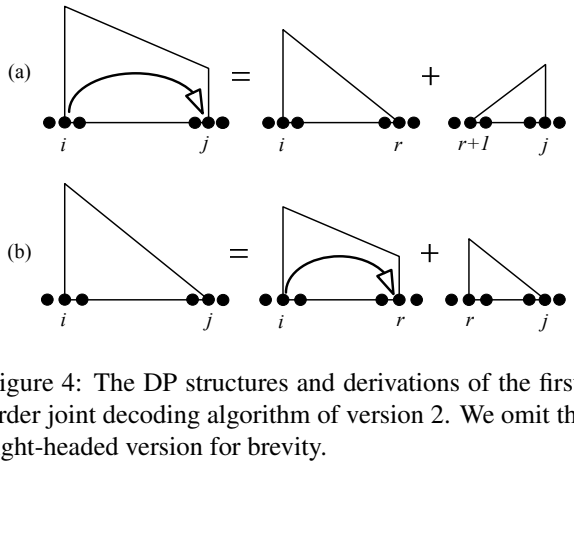

Figure 4: The DP structures and derivations of the first-order joint decoding algorithm of version 2. We omit the right-headed version for brevity.

[3] ① $\underline{w_r\ t_r}$ if $i \ne r$; ② $\underline{w_{r+1}\ t_{r+1}}$ if $r+1 \ne j$; ③ $\underline{t_r\ t_{r+1}}$ if $r \ne i$ or $r+1 \ne j$; ④ $\underline{t_i\ t_r}$ if $r-1 = i$; ⑤ $\underline{t_{r+1}\ t_j}$ if $r+2 = j$. Note that $\underline{w_i\ t_i}$, $\underline{w_j\ t_j}$ and $\underline{t_i\ t_j}$ (if $i = j-1$) are not incorporated here to avoid double counting.

[4] ① $\underline{w_r\ t_r}$ if $r \ne j$; ② $\underline{t_i\ t_r}$ if $i = r-1$; ③ $\underline{t_r\ t_j}$ if $r+1 = j$. Pseudo trigram features can be added accordingly.

[5] We can reduce the time complexity to $O(n^3q^3)$ by strictly adopting the DP structures in the parsing algorithm of Eisner (1996). However, that may make the algorithm harder to comprehend.

gorithm from Figure 3. We omit them due to space limitation. Pseudo surrounding, in between and POS trigram features are used due to the same reason as above. The space and time complexity of the algorithm are respectively $O(n^3q^3)$ and $O(n^4q^5)$.

3.2 Joint Models of Version 2

To further incorporate genuine syntactic surrounding and POS trigram features in the DP structures, we extend the algorithms of joint models of version 1, and propose our joint models of version 2.

The decoding algorithm of O1: Figure 4 illustrates the first-order joint decoding algorithm of version 2. Compared with the structures in Figure 2, each span is augmented with the POS tags surrounding the boundary indices. These context POS tags enable $Score_{\text{joint}}(.)$ in line 6-9 of Algorithm 1 to capture the ***syntactic surrounding and POS trigram features***, but also require enumeration of POS tags over more indices. For brevity, we skip the pseudo codes which can be easily derived from Algorithm 1. The space and time complexity of the algorithm are respectively $O(n^2q^6)$ and $O(n^3q^{10})$.

The decoding algorithm of O2 & O3: Using the same idea as above, the second- and third-order joint decoding algorithms of version 2 can be derived based on Figure 3. Again, we omit both its DP structures and pseudo codes for the sake of brevity. Its space and time complexity are respectively $O(n^3q^7)$ and $O(n^4q^{11})$.

In between features, which should be regarded as non-local features in the joint situation, still cannot be incorporated in our joint models of version 2. Again, we adopt the pseudo version.

3.3 Comparison

Based on the above illustration, we can see that joint models of version 1 are more efficient with regard to the number of POS tags for each word, but fail to incorporate syntactic surrounding features and POS trigram features in the DP structures. On the contrary, joint models of version 2 can incorporate both aforementioned feature sets, but have higher complexity. These two versions of models will be thoroughly compared in the experiments.

4 Pruning Techniques

In this section, we introduce two pruning strategies to constrain the search space of our models due to their high complexity.

4.1 POS Tag Pruning

The time complexity of the joint decoding algorithm is unbearably high with regard to the number of candidate POS tags for each word ($q = |\mathcal{T}|$). We find that it would be extremely time-consuming even when we only use two most likely POS tags for each word ($q = 2$) even for joint models of version 1. To deal with this problem, we propose a pruning method that can effectively reduce the POS tag space based on a probabilistic tagging model.

We adopt a conditional log-linear model (Lafferty et al., 2001), which defines a conditional distribution of a POS tag sequence $\mathbf{t}$ given $\mathbf{x}$:

$$P(\mathbf{t}|\mathbf{x}) = \frac{e^{\mathbf{w}_{\text{pos}} \cdot \mathbf{f}_{\text{pos}}(\mathbf{x},\mathbf{t})}}{\sum_{\mathbf{t}} e^{\mathbf{w}_{\text{pos}} \cdot \mathbf{f}_{\text{pos}}(\mathbf{x},\mathbf{t})}}$$

We use the same feature set $\mathbf{f}_{\text{pos}}$ defined in Section 2.1, and adopt the *exponentiated gradient* algorithm to learn the weight vector $\mathbf{w}_{\text{pos}}$ (Collins et al., 2008).

The marginal probability of tagging a word w_i as t is

$$P(t_i = t|\mathbf{x}) = \sum_{\mathbf{t}:\mathbf{t}[i]\equiv t} P(\mathbf{t}|\mathbf{x})$$

which can be efficiently computed using the *forward-backward* algorithm.

We define $\texttt{pmax}_i(\mathbf{x})$ to be the highest marginal probability of tagging the word w_i:

$$\texttt{pmax}_i(\mathbf{x}) = \max_{t \in \mathcal{T}} P(t_i = t|\mathbf{x})$$

We then define the allowable candidate POS tags of the word w_i to be

$$\mathcal{T}_i(\mathbf{x}) = \{t : t \in \mathcal{T}, P(t_i = t|\mathbf{x}) \geq \lambda_t \times \texttt{pmax}_i(\mathbf{x})\}$$

where λ_t is the pruning threshold. $\mathcal{T}_i(\mathbf{x})$ is used to constrain the POS search space by replacing $\mathcal{T}$ in Algorithm 1.

4.2 Dependency Pruning

The parsing time grows quickly for the second- and third-order models (both baseline and joint) when the input sentence gets longer ($O(n^4)$). Following Koo and Collins (2010), we eliminate unlikely dependencies using a form of coarse-to-fine pruning (Charniak and Johnson, 2005; Petrov and Klein, 2007). On the development set, 68.87% of the dependencies are pruned, while the oracle dependency accuracy is 99.77%. We use 10-fold cross validation to do pruning on the training set.

5 Experiments

We use the Penn Chinese Treebank 5.1 (CTB5) (Xue et al., 2005). Following the setup of Duan et al. (2007), Zhang and Clark (2008b) and Huang and Sagae (2010), we split CTB5 into training (secs 001-815 and 1001-1136), development (secs 886-931 and 1148-1151), and test (secs 816-885 and 1137-1147) sets. We use the head-finding rules of Zhang and Clark (2008b) to turn the bracketed sentences into dependency structures.

We use the standard *tagging **acc**uracy* to evaluate POS tagging. For dependency parsing, we use ***word** accuracy* (also known as *dependency accuracy*), ***root** accuracy* and ***compl**ete match rate* (all excluding punctuation) .

For the averaged training, we train each model for 15 iterations and select the parameters that perform best on the development set.

5.1 Results of POS Tag Pruning

Figure 5 shows the distribution of words with different number of candidate POS tags and the k-best oracle tagging accuracy under different λ_t. To avoid dealing with words that have many candidate POS tags, we further apply a hard criterion that the decoding algorithms only consider top k candidate POS tags.

To find the best λ_t, we train and evaluate the second-order joint model of version 1 on the training and development sets pruned with different λ_t (top k = 5). We adopt the second-order joint model of version 1 because of its efficiency compared with the third-order models and its capability of capturing rich features compared with the first-order models. The results are shown in Table 1. The model

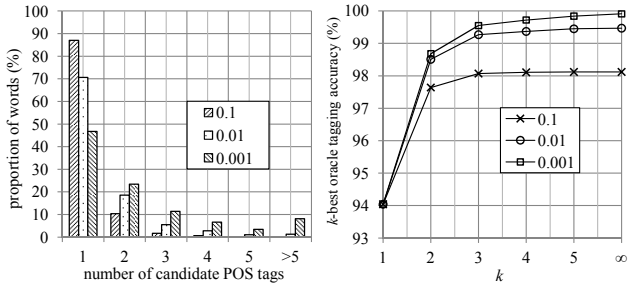

Figure 5: Results of POS tag pruning with different pruning threshold λ_t on the development set.

λ_t	word	root	compl.	acc.	speed
0.1	81.53	76.88	30.00	**94.17**	2.5
0.01	**81.83**	76.62	**30.62**	93.16	1.2
0.001	81.73	**77.38**	30.50	93.41	0.5

Table 1: Performance of the second-order joint model of version 1 with different pruning threshold λ_t (top k = 5) on the development set. "Acc." means the tagging accuracy. "Speed" refers to the parsing speed (the number of sentences processed per second).

with $\lambda_t = 0.1$ obtains the highest tagging accuracy, which is much higher than that of both $\lambda_t = 0.01$ and $\lambda_t = 0.001$. However, its parsing accuracy is inferior to the other two. $\lambda_t = 0.01$ produces slightly better parsing accuracy than $\lambda_t = 0.001$, and is twice faster. Finally, we choose $\lambda_t = 0.01$ due to the efficiency factor and our priority over the parsing accuracy.

Then we do experiments to find an optimal top k. Table 2 shows the results. We decide to choose $k = 3$ since it leads to best parsing accuracy.

From Table 1 and 2, we can have an interesting finding: it seems that the harder we filter the POS tag space, the higher tagging accuracy we get. In other words, **giving the joint model less flexibility of choosing POS tags leads to better tagging performance.**

Due to time limitation, we do not tune λ_t and k for other joint models. Instead, we simply adopt $\lambda_t = 0.01$ and top $k = 3$.

5.2 Final Results

Table 3 shows the final results on the test set. We list a few state-of-the-art results in the bottom. Duan07 refers to the results of Duan et al. (2007). They enhance the transition-based parsing model with

		Syntactic Metrics			Tagging Accuracy			Parsing Speed
		word	root	compl.	all-word	known	unknown	$Sent/Sec$
Joint Models V2	O3	**80.79**	**75.84**	**29.11**	92.80	93.88	76.80	0.3
	O2	80.49	75.49	28.24	92.68	93.77	76.27	0.6
	O1	77.37	68.64	23.09	**92.96**	94.05	76.64	2.0
Joint Models V1	O3	80.69	**75.90**	**29.06**	92.89	93.96	76.80	0.5
	O2	**80.74**	75.80	28.24	93.08	94.11	77.53	1.7
	O1	77.38	69.69	22.62	**93.20**	94.23	77.76	8.5
Auto POS	O3	**79.29**	74.65	**27.24**	93.51	94.36	80.78	2.0
	O2	79.03	**74.70**	27.19				5.8
	O1	75.68	68.06	21.10				17.4
	MSTParser2	77.95	72.04	25.50				4.1
	MSTParser1	75.84	68.55	21.36				5.2
	MaltParser	75.24	65.92	23.19				2.6
Gold POS	O3	**86.00**	77.59	**34.02**	100.0	100.0	100.0	-
	O2	**86.18**	**78.58**	**34.07**				-
	O1	82.24	70.10	26.02				-
	MSTParser2	85.24	77.41	33.19				-
	MSTParser1	83.04	71.49	27.59				-
	MaltParser	82.62	69.34	29.06				-
	H&S10	85.20	**78.32**	**33.72**				-
	Z&C08 single	84.33	76.73	32.79				-
	Z&C08 hybrid	**85.77**	76.26	**34.41**				-
	Duan07	83.88	73.70	32.70				-

Table 3: Final results on the test set. "Gold POS" means that gold POS tags are used as input by the pipelined parsing models; while "Auto POS" means that the POS tags are generated by the baseline POS tagging model.

top k	word	root	compl.	acc.	speed
2	81.46	76.12	30.50	**93.51**	2.7
3	**82.11**	**76.75**	29.75	93.31	1.7
4	81.75	76.62	30.38	93.25	1.4
5	81.83	76.62	**30.62**	93.16	1.2

Table 2: Performance of the second-order joint model of version 1 with different top k ($\lambda_t = 0.01$) on the development set.

the beam search. H&S10 refers to the results of Huang and Sagae (2010). They greatly expand the search space of the transition-based model by merging equivalent states with dynamic programming. Z&C08 refers to the results of Zhang and Clark (2008b). They use a hybrid model to combine the advantages of both graph-based and transition-based models. We also do experiments with two publicly available and widely-used parsers, MSTParser[6] and MaltParser[7]. MSTParser1 refers to the first-order graph-based model of McDonald et al. (2005), while MSTParser2 is the second-order model of McDonald and Pereira (2006). MaltParser is a transition-based parsing system. It integrates a number of classification algorithms and transition strategies. We adopt the support vector machine classifier and the arc-standard strategy (Nivre, 2008).

We can see that when using gold tags, our pipelined second- and third-order parsing models achieve best parsing accuracy, which is even higher than the hybrid model of Zhang and Clark (2008b). It is a little surprising that the second-order model slightly outperforms the third-order one. This may be possible, since Koo and Collins (2010) shows that the third-order model outperforms the second-order one by only 0.32% on English and 0.07% on Czech. In addition, we only use basic third-order features.

Both joint models of version 1 and 2 can consistently and significantly improve the parsing accuracy by about 1.5% for all first-, second- and third-order cases. Accidentally, the parsing accuracy of the second-order joint model of version 2 is lower

[6]http://sourceforge.net/projects/mstparser/

[7]http://maltparser.org/

error pattern	#	↓	error pattern	#	↑
DEC → DEG	237	114	NR → NN	184	100
NN → VV	389	73	NN → NR	106	91
DEG → DEC	170	39	NN → JJ	95	70
VV → NN	453	27	VA → VV	29	41
P → VV	52	24	JJ → NN	126	29
P → CC	39	13	VV → VA	67	10

Table 4: Error analysis of POS tagging. # means the error number of the corresponding pattern made by the baseline tagging model. ↓ and ↑ mean the error number reduced or increased by the joint model.

than that of its counterparts by about 0.3%. More experiments and further analysis may be needed to find out the reason. The two versions of joint models performs nearly the same, which indicates that using pseudo surrounding and POS trigram features may be sufficient for the joint method on this data set. In summary, we can conclude that **the joint framework is certainly helpful for dependency parsing.**

It is clearly shown in Table 3 that **the joint method surprisingly hurts the tagging accuracy**, which diverges from our discussion in Section 1. Some insights into this issue will be given in Section 5.3. Moreover, it seems that **the more syntactic features the joint method incorporates (from O1 to O3), the more the tagging accuracy drops**. We suspect that this is because the joint models are dominated by the syntactic features. Take the first-order joint model as an example. The dimension of the syntactic features $\mathbf{f}_{\text{syn}}$ is about 3.5 million, while that of $\mathbf{f}_{\text{pos}}$ is only about 0.5 million. The gap becomes much larger for the second- and third-order cases.

Comparing the parsing speed, we can find that the pruning of POS tags is very effective. The second-order joint model of version 1 can parse 1.7 sentences per second, while the pipelined second-order parsing model can parse 5.8 sentences per second, which is rather close considering that there is a factor of q^5.

5.3 Error Analysis

To find out the impact of our joint models on the individual tasks, we conduct detailed error analysis through comparing the results of the pipelined second-order parsing model and the second-order joint model of version 1.

Impact on POS tagging: Table 4 shows how the joint model changes the quantity of POS tagging error patterns compared with the pipelined model. An error pattern "X → Y" means that the focus word, whose true tag is 'X', is assigned a tag 'Y'. We choose these patterns with largest reduction or increase in the error number, and rank them in descending order of the variation.

From the left part of Table 4, we can see that the joint method is clearly better at resolving tagging ambiguities like {VV, NN} and {DEG, DEC}.[8] One common characteristic of these ambiguous pairs is that the local or even whole syntactic structure will be destructed if the wrong tag is chosen. In other words, resolving these ambiguities is critical and helpful from the parsing viewpoint. From another perspective, the joint model is capable of preferring the right tag with the help of syntactic structures, which is impossible for the baseline sequential labeling model.

In contrast, pairs like {NN, NR}, {VV, VA} and {NN, JJ} only slightly influence the syntactic structure when mis-tagged. The joint method performs worse on these ambiguous pairs, as shown in the right part of Table 4.

Impact on parsing: Table 5 studies the change of parsing error rates between the pipelined and joint model on different POS tag patterns. We present the most typical and prominent patterns in the table, and rank them in descending order of X's frequency of occurrence. We also show the change of proportion of different patterns, which is consistent with the results in Table 4.

From the table, we can see the joint model can achieve a large error reduction (0.8∼4.0%) for all the patterns "X → X". In other words, **the joint model can do better given the correct tags** than the pipelined method.

For all the patterns marked by ♢, except for the ambiguous pair {NN, JJ} (which we find is difficult to explain even after careful result analysis), the joint model also reduces the error rates (2.2∼15.4%). As

[8] DEG and DEC are the two POS tags for the frequently used auxiliary word "的" (dē, of) in Chinese. The associative "的" is tagged as DEG, such as "父亲/father 的 眼睛/eyes (eyes of the father)"; while the one in a relative clause is tagged as DEC, such as "他/he 取得/made 的 进步/progress (progress that he made)".

pattern	pipelined		joint	
	prop (%)	error (%)	prop (%)	error (%)
NN → NN	94.6	16.8	-1.1	-1.8
→ VV ♡	2.9	55.5	-0.5	+15.1
→ NR ♢	0.8	24.5	+0.7	-2.2
→ JJ ♢	0.7	17.9	+0.5	*+2.1*
VV → VV	89.6	34.2	-0.3	–4.0
→ NN ♡	6.6	66.4	-0.4	+0.7
→ VA ♢	1.0	38.8	+0.1	-15.4
NR → NR	91.7	15.4	-3.7	-0.8
→ NN ♢	5.9	21.7	+3.2	-3.7
P → P	92.8	22.6	+3.4	-3.2
→ VV ♡	3.0	50.0	-1.4	+10.7
→ CC ♡	2.3	74.4	-0.7	+21.9
JJ → JJ	80.5	11.2	-2.8	-2.0
→ NN ♢	9.8	18.3	+2.2	*+1.8*
DEG → DEG	86.5	11.1	+2.8	-3.6
→ DEC ♡	13.5	61.8	-3.1	+37.4
DEC → DEC	79.7	17.2	+12.1	-4.0
→ DEG ♡	20.2	56.5	-9.7	+40.2

Table 5: Comparison of parsing error rates on different POS tag patterns between the pipelined and joint models. Given a pattern "X → Y", "prop" means its *proportion* in all occurrence of 'X' ($\frac{Count(X \rightarrow Y)}{Count(X)}$), and "error" refers to its *parsing error rate* ($\frac{Count(\texttt{wrongly headed } X \rightarrow Y)}{Count(X \rightarrow Y)}$). The last two columns give the absolute reduction (-) or increase (+) in proportion and error rate made by the joint model. ♡ marks the patterns appearing in the left part of Table 4, while ♢ marks those in the right part of Table 4.

discussed earlier, these patterns concern ambiguous tag pairs which usually play similar roles in syntactic structures. This demonstrates that **the joint model can do better on certain tagging error patterns**.

For patterns marked by ♡, the error rate of the joint model usually increases by large margin. However, the proportion of these patterns is substantially decreased, since the joint model can better resolve these ambiguities with the help of syntactic knowledge.

In summary, we can conclude that the joint model is able to choose such POS tags that are more helpful and discriminative from parsing viewpoint. This is the fundamental reason of the parsing performance improvement.

6 Related Work

Theoretically, Eisner (2000) proposes a preliminary idea of extending the decoding algorithm for dependency parsing to handle polysemy. Here, word senses can be understood as POS-tagged words. Koo and Collins (2010) also briefly discuss that their third-order decoding algorithm can be modified to handle word senses using the idea of Eisner (2000).

In his PhD thesis, McDonald (2006) extends his second-order model with the idea of Eisner (2000) to study the impact of POS tagging errors on parsing accuracy. To make inference tractable, he uses top 2 candidate POS tags for each word based on a maximum entropy tagger, and adopts the single most likely POS tags for the surrounding and in between features. He conducts primitive experiments on English Penn Treebank, and shows that parsing accuracy can be improved from 91.5% to 91.9%. However, he finds that the model is unbearably time-consuming.

7 Conclusions

In this paper, we have systematically investigated the issue of joint POS tagging and dependency parsing. We propose and compare several joint models and their corresponding decoding algorithms which can incorporate different feature sets. We also propose an effective POS tag pruning method which can greatly improve the decoding efficiency. The experimental results show that our joint models can significantly improve the state-of-the-art parsing accuracy by more than 1.5%. Detailed error analysis shows that the fundamental reason for the parsing accuracy improvement is that the joint method is able to choose POS tags that are helpful and discriminative from parsing viewpoint.

Acknowledgments

We thank the anonymous reviewers for their helpful comments. This work was supported by National Natural Science Foundation of China (NSFC) via grant 60803093, 60975055, the Natural Scientific Research Innovation Foundation in Harbin Institute of Technology (HIT.NSRIF.2009069) and the Fundamental Research Funds for the Central Universities (HIT.KLOF.2010064).

References

Xavier Carreras. 2007. Experiments with a higher-order projective dependency parser. In *Proceedings of*

EMNLP/CoNLL, pages 141–150.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of ACL-05*, pages 173–180.

Wanxiang Che and Ting Liu. 2010. Jointly modeling wsd and srl with markov logic. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 161–169.

Shay B. Cohen and Noah A. Smith. 2007. Joint morphological and syntactic disambiguation. In *Proceedings of EMNLP-CoNLL 2007*, pages 208–217.

Michael Collins, Amir Globerson, Terry Koo, Xavier Carreras, and Peter Bartlett. 2008. Exponentiated gradient algorithms for conditional random fields and max-margin markov networks. *JMLR*, 9:1775–1822.

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

Xiangyu Duan, Jun Zhao, , and Bo Xu. 2007. Probabilistic models for action-based Chinese dependency parsing. In *Proceedings of ECML/ECPPKDD*.

Chris Dyer. 2009. Using a maximum entropy model to build segmentation lattices for mt. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 406–414.

Jason Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of COLING 1996*, pages 340–345.

Jason Eisner. 2000. Bilexical grammars and their cubic-time parsing algorithms. In *Advances in Probabilistic and Other Parsing Technologies*, pages 29–62.

Jenny Rose Finkel and Christopher D. Manning. 2009. Joint parsing and named entity recognition. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 326–334.

Yoav Goldberg and Reut Tsarfaty. 2008. A single generative model for joint morphological segmentation and syntactic parsing. In *Proceedings of ACL-08: HLT*, pages 371–379, Columbus, Ohio, June. Association for Computational Linguistics.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of CoNLL 2009*.

Liang Huang and Kenji Sagae. 2010. Dynamic programming for linear-time incremental parsing. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1077–1086, Uppsala, Sweden, July. Association for Computational Linguistics.

Wenbin Jiang, Liang Huang, Qun Liu, and Yajuan Lü. 2008. A cascaded linear model for joint chinese word segmentation and part-of-speech tagging. In *Proceedings of ACL-08: HLT*, pages 897–904.

Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1–11, Uppsala, Sweden, July. Association for Computational Linguistics.

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 the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, 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.

Roger Levy and Christopher D. Manning. 2003. Is it harder to parse chinese, or the chinese treebank? In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 439–446, Sapporo, Japan, July. Association for Computational Linguistics.

Junhui Li, Guodong Zhou, and Hwee Tou Ng. 2010. Joint syntactic and semantic parsing of chinese. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1108–1117.

Yang Liu and Qun Liu. 2010. Joint parsing and translation. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 707–715.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of EACL 2006*.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of ACL 2005*, pages 91–98.

Ryan McDonald. 2006. *Discriminative Training and Spanning Tree Algorithms for Dependency Parsing*. Ph.D. thesis, University of Pennsylvania.

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. In *Computational Linguistics*, volume 34, pages 513–553.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Proceedings of NAACL 2007*.

Adwait Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *Proceedings of EMNLP 1996*.

Alexander M Rush, David Sontag, Michael Collins, and Tommi Jaakkola. 2010. On dual decomposition and linear programming relaxations for natural language processing. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1–11.

Libin Shen, Giorgio Satta, and Aravind Joshi. 2007. Guided learning for bidirectional sequence classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 760–767, Prague, Czech Republic, June. Association for Computational Linguistics.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. In *CoNLL-2008*.

Kristina Toutanova and Colin Cherry. 2009. A global model for joint lemmatization and part-of-speech prediction. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 486–494.

Xinyan Xiao, Yang Liu, YoungSook Hwang, Qun Liu, and Shouxun Lin. 2010. Joint tokenization and translation. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 1200–1208.

Nianwen Xue, Fei Xia, Fu-Dong Chiou, and Martha Palmer. 2005. The Penn Chinese Treebank: Phrase structure annotation of a large corpus. In *Natural Language Engineering*, volume 11, pages 207–238.

Yue Zhang and Stephen Clark. 2008a. Joint word segmentation and POS tagging using a single perceptron. In *Proceedings of ACL-08: HLT*, pages 888–896.

Yue Zhang and Stephen Clark. 2008b. A tale of two parsers: Investigating and combining graph-based and transition-based dependency parsing. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 562–571, Honolulu, Hawaii, October. Association for Computational Linguistics.