Stacking Heterogeneous Joint Models of Chinese POS Tagging and Dependency Parsing

Meishan Zhang Wanxiang Che Ting Liu[*] *Zhenghua Li*
School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China
{mszhang, car, tliu, lzh}@ir.hit.edu.cn

ABSTRACT

Previous joint models of Chinese part-of-speech (POS) tagging and dependency parsing are extended from either graph- or transition-based dependency models. Our analysis shows that the two models have different error distributions. In addition, integration of graph- and transition-based dependency parsers by *stacked learning* (stacking) has achieved significant improvements. These motivate us to study the problem of stacking graph- and transition-based joint models. We conduct experiments on Chinese Penn Treebank 5.1 (CTB5.1). The results demonstrate that the guided transition-based joint model obtains better performance than the guided graph-based joint model. Further, we introduce a constituent-based joint model which derives the POS tag sequence and dependency tree from the output of PCFG parsers, and then integrate it into the guided transition-based joint model. Finally, we achieve the best performance on CTB5.1, 94.95% in tagging accuracy and 83.98% in parsing accuracy respectively.

TITLE AND ABSTRACT IN CHINESE

采用堆方法融合异种的中文词性和依存句法联合模型

过去的中文词性和依存句法联合模型基本上都根据基于图的依存句法分析模型或者基于转移的依存句法分析模型进行拓展而形成的。我们的分析结果表明这两种不同的模型错误分布并不一样，而且在依存句法中，将基于图的模型和基于转移的模型使用堆方法融合之后，能够显著的提升依存句法的性能，这些促使我们进一步研究采用堆方法去融合基于图的和基于转移的词性依存句法联合模型。我们在中文宾州树库5.1版本（CTB5.1）上进行试验，实验结果表明，相比使用基于图的联合模型为被指导模型，采用转移的联合模型为被指导模型能取得较好的性能。更进一步，我们介绍了基于短语句法结构的联合模型，它从一个句子的概率短语文法分析器输出结果中提取句子的词性序列以及依存树结果，然后我们采用基于短语句法结构的联合模型更进一步指导基于转移的联合模型，最终我们在CTB5.1的数据上取得了最好结果，词性标注准确率达到94.95%，同时，依存句法准确率达到83.98%。

KEYWORDS: Chinese POS Tagging, Dependency Parsing, Joint Model, Stacked Learning.

KEYWORDS IN CHINESE: 中文词性标注,依存分析,联合模型,堆方法融合学习.

[*]Corresponding author

Proceedings of COLING 2012: Technical Papers, pages 3071–3088,
COLING 2012, Mumbai, December 2012.

1 Introduction

Part-of-speech (POS) tagging and dependency parsing are two fundamental natural language processing (NLP) tasks. Typically, POS tagging is a preprocessing step for dependency parsing, especially in a pipeline architecture. There are two main problems in a pipeline system: (1) Dependency parsing suffers the problem of error propagation; (2) POS tagging cannot exploit useful, important syntactic information for disambiguation.

For Chinese POS tagging and dependency parsing, a pipeline system seriously suffers these two problems. The study presented in (Li et al., 2011b) demonstrates the error propagation factor. The authors develop a graph-based joint model for Chinese POS tagging and dependency parsing. The most interesting thing they found is that even with lower tagging accuracy, a joint model could achieve higher parsing accuracy. The work presented in (Hatori et al., 2011) also demonstrates a joint model can largely improve the performance of dependency parsing further. They propose a transition-based joint model for Chinese POS tagging and dependency parsing.

Recently, ensemble models have been gained a lot of interests in NLP community. Stacked learning (stacking) (Wolpert, 1992; Breiman, 1996), which is a typical method for ensemble models, has been applied to a number of NLP tasks for its elegance and conciseness, such as Chinese Word segmentation (Sun, 2011), POS tagging (Li et al., 2011a), named entity recognition (Dekai Wu and Carpuat, 2003) and dependency parsing (McDonald, 2006; Nivre and McDonald, 2008; Martins et al., 2008; Søgaard and Rishøj, 2010; McDonald and Nivre, 2011). Especially, (Nivre and McDonald, 2008) demonstrate that the performance of dependency parsing can be largely improved by stacking a graph-based dependency parser and a transition-based dependency parser. Thus it is interesting to investigate the effect of stacked learning when it is applied to joint models.

Graph- and transition-based joint models are extended from graph- and transition-based models of dependency parsing respectively. They are the two mainstream approaches for dependency parsing. It is noteworthy that, the probabilistic context-free grammar (PCFG) parsers, such as Brown parser (Charniak and Johnson, 2005) and Berkeley parser (Petrov et al., 2006; Petrov and Klein, 2007), which are traditionally used for constituent parsing, have also been suggested for dependency parsing (Yamada and Matsumoto, 2003; McDonald, 2006; Sun, 2012; Che et al., 2012). We denote these methods by constituent-based models. The precondition of constituent-based models is that the output constituent structure of the PCFG parsers can be transformed into dependency structure by rules adequately. This is satisfied for Chinese. Moreover, the PCFG models can process POS tagging simultaneously in constituent parsing. They treat POS tagging as a submodule of constituent parsing. Thus we can also adopt a PCFG model for joint Chinese POS tagging and dependency parsing. We denote it by constituent-based joint model. (Sun, 2012) proposed to improve the Chinese parsing accuracy by a PCFG parser. Similarly, (Sun and Uszkoreit, 2012) exploited a PCFG parser to enhance Chinese POS tagging. Thus it is reasonable to investigate the performance of constituent-based joint models and to improve the performance of joint Chinese POS tagging and dependency parsing by a constituent-based joint model.

In this paper, first we study the integration of a graph-based joint model (JGraph) and a transition-based joint model (JTrans) by stacked learning. The stacked learning is implemented using a two-level architecture, where the level-0 consists of one or more predictors of which the results are exploited as input to enhance the level-1 predictor. Thus either JGraph or JTrans can be chosen as the level-1 model. We call the stacking model using JGraph as level-1 model by the guided graph-based joint model and the stacking model using JTrans as level-1 model by the guided transition-based joint model. Further we introduce a constituent-based joint model (JConst) and then integrate this model into the previous better stacking model by further stacking.

We conduct the experiments on Chinese Penn Treebank 5.1 (CTB5.1) data set (Xue et al., 2005). First, we evaluate the performance of stacking models. The guided transition-based joint model gets better performance than the guided graph-based joint model, achieving 94.76% in tagging accuracy and 82.22% in parsing accuracy. Then we evaluate the performance of our constituent-based joint model. The reported accuracies are 93.45% in POS tagging and 81.03% in dependency parsing. Finally, we integrate the constituent-based joint model into the guided transition-based joint model by a further stacking. Our final results are 94.95% in tagging accuracy and 83.98% in parsing accuracy, resulting in further improvements of 0.19% in tagging accuracy and 1.76% in parsing accuracy.

Finally, detailed error analysis is carried out by two aspects: (1) the different error distributions of JGraph, JTrans and JConst are shown in detail to interpret the improvements in stacking and (2) the comparisons between joint models and pipeline approaches are conducted to understand the interaction between Chinese POS tagging and dependency parsing. Through the analysis, we can find several interesting phenomenons. For example, JConst can do better for long distance dependencies, the tagging accuracies of joint models are more fragile facing to wrong dependencies, dependencies with the head on the right are more easily recognized however the corresponding POS tags of the modifiers in these dependencies are more difficult to be handled.

The rest of the paper is organized as follows. Section 2 reviews the related works of our paper. Section 3 describes the graph- and transition-based joint models . Section 4 describes the stacking models including the guided graph-based joint model and the guided transition-based joint model. Section 5 describes our constituent-based model and the further stacking model. Section 6 reports the experimental results. Section 7 gives the systematic analysis of the joint models. Finally, in Section 8 we conclude this paper and point out our future works.

2 Related Works

Related Works on Joint Models of Chinese POS Tagging and Dependency Parsing (Li et al., 2011b) present the first joint model for Chinese POS tagging and dependency parsing. They extend models of graph-based dependency parsing (McDonald et al., 2005; McDonald and Pereira, 2006; Carreras, 2007; Koo and Collins, 2010), making them enable to handle POS tagging simultaneously. They conclude that joint models can achieve better performance in dependency parsing and can do much better some certain POS tagging error patterns.

Secondly, (Hatori et al., 2011) and (Bohnet and Nivre, 2012) propose joint models based on transition-based dependency parsing (Nivre, 2008; Huang and Sagae, 2010; Zhang and Nivre, 2011). (Hatori et al., 2011) also examine the results of their joint model carefully, demonstrating similar conclusions to that of (Li et al., 2011b). However, their certain error patterns are slightly different with that of (Li et al., 2011b). The differences may be induced by the different manners of modeling the joint task.

Thirdly, the constituent-based joint model processes joint Chinese POS tagging and dependency parsing in an indirectly way. It performs the POS tagging and dependency parsing by a conversion from the initial output of a PCFG parser. In Chinese POS tagging, (Sun and Uszkoreit, 2012) suggest to enhance Chinese POS tagging guided by the output POS tags of

Berkeley parser (Petrov et al., 2006; Petrov and Klein, 2007). In Chinese dependency parsing, (Sun, 2012) has proposed to improve the performance of dependency parsing by Berkeley parser. (Che et al., 2012) have compared the performance of several PCFG parsers on Stanford dependencies. This method has also been employed for English dependency parsing (Yamada and Matsumoto, 2003; McDonald, 2006).

In this paper, we study the integration of the three joint models by stacking. Then we discuss these different joint models including the stacking models comprehensively, aiming for two purposes: (1) figure out the benefits from stacking and (2) understand the interaction between Chinese POS tagging and dependency parsing.

Related Works on Stacked Learning *Stacked generalization* is a meta-learning algorithm that is first proposed by (Wolpert, 1992) and (Breiman, 1996). It has been exploited in a number of NLP tasks for integration. We mainly concern the works of stacked learning applied on POS tagging and dependency parsing. The work of (Li et al., 2011a) presented a mostly recent work for stacking POS taggers. They exploit the output of a CRF POS tagger to help a perceptron-based POS tagger with syntactic features. (McDonald, 2006) proposed the first stacking work of dependency parsing. The author incorporated parse decisions of two constituent-based parsers, Collins parser (Collins, 1999; Bikel, 2004) and Charniak parser (Charniak, 2000), into the second-order MST parser. Then (Nivre and McDonald, 2008) suggested integrating graph- and transition-based models by stacking, and more detailed analysis was given in (McDonald and Nivre, 2011). (Martins et al., 2008) also demonstrated that stacking transition- and graph-based parsers can improve parsing performance significantly and meanwhile offer theoretical interpretations for stacking. In our paper, stacked leaning is applied on the joint tasks of Chinese POS tagging and dependency parsing.

3 Two Models for Joint Chinese POS Tagging and Dependency parsing

A dependency tree for an input sentence $\mathbf{x} = w_0 w_1 \cdots w_n$ (where $w_0 = \text{ROOT}$) can be denoted by $\mathbf{d} = \{(h, m) \mid 0 \leq h \leq n, 0 < m \leq n\}$, where (h, m) represents a dependency $w_h \rightarrow w_m$ whose *head* word (or father) is w_h and *modifier* word (or child) is w_m. The task of dependency parsing is to find an optimum dependency tree $\mathbf{d}$ for the input sentence $\mathbf{x}$. Generally, the POS tag sequence of the sentence $\mathbf{t} = t_1 \cdots t_n$ (where $t_i \in T, 1 \leq i \leq n$, T is the POS tag set) is taken as an input for dependency parsing, which is determined by the task of POS tagging, thus forming a pipeline model of the two tasks. POS tagging is a typical sequence labeling problems which can be resolved by algorithms such as maximum-entropy (Ratnaparkhi, 1996), conditional random fields (CRF) (Lafferty et al., 2001) and averaged perceptron (Collins, 2002). The goal of joint models of the two tasks is to find an optimum dependency tree and an optimum POS tag sequence $(\hat{\mathbf{t}}, \hat{\mathbf{d}})$ for $\mathbf{x}$ concurrently.

3.1 Graph-based Joint Model

The graph-based joint model is first proposed by (Li et al., 2011b). Such a model is extended from a graph-based model for dependency parsing (McDonald et al., 2005; McDonald and Pereira, 2006; Carreras, 2007; Koo and Collins, 2010). In the model, the score of a dependency tree along with POS tags on each node is factored into scores of small parts.

(Li et al., 2011b) have introduced several different graph-based joint models in term of small parts employed in the models. According to the results they reported, we employ the model Li-11(v1,2nd) in terms of both accuracies and decoding speed. The scoring parts of the model include dependencies, siblings and grandchilds, as is shown in Figure 1. The score function of Li-11(v1,2nd) can be represented by Equation 1,

$$\begin{aligned}\text{Score}_{\text{joint}}(\mathbf{x},\mathbf{t},\mathbf{d}) = &\sum_{\{(h,m)\}\subseteq\mathbf{d}} \mathbf{w}_{\text{pos}}\cdot\mathbf{f}_{\text{pos}}(\mathbf{x},\mathbf{t},m)+\mathbf{w}_{\text{dep}}\cdot\mathbf{f}_{\text{dep}}(\mathbf{x},\mathbf{t},h,m)+\sum_{\{(h,s)(h,m)\}\subseteq\mathbf{d}}\mathbf{w}_{\text{sib}}\cdot\mathbf{f}_{\text{sib}}(\mathbf{x},\mathbf{t},h,s,m)\\ &+\sum_{\{(g,h)(h,m)\}\subseteq\mathbf{d}}\mathbf{w}_{\text{grd}}\cdot\mathbf{f}_{\text{grd}}(\mathbf{x},\mathbf{t},g,h,m)\end{aligned} \tag{1}$$

where $\mathbf{w}$ denotes the model parameters and $\mathbf{f}_{\text{pos}}(\cdot),\mathbf{f}_{\text{dep}}(\cdot),\mathbf{f}_{\text{sib}}(\cdot),\mathbf{f}_{\text{grd}}(\cdot)$ denote the features of POS tagging, dependency part, sibling part and grandchild part.

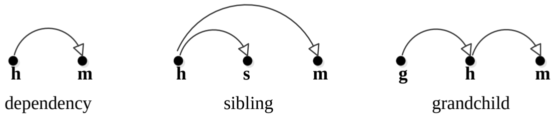


Figure 1: The scoring parts used in our graph-based joint model. Each node in the scoring parts has been tagged with POS already.

For more detailed description, we refer to the original paper. We call this graph-based joint model as JGraph for brevity.

3.2 Transition-based Joint Model

(Hatori et al., 2011) propose the first joint model of Chinese POS tagging and transition-based dependency parsing (Nivre, 2008; Huang and Sagae, 2010; Zhang and Nivre, 2011). In a transition-based system, we learn a joint model for scoring the transition A_i from one state ST_i to the next ST_j. As shown in Figure 2, the state ST of the transition system is composed by a stack S and a queue Q, where $S = (\cdots, s_1, s_0)$ is a stack of dependency trees along with POS tags and $Q = (q_0, q_1, \cdots, q_{n-j}) = (w_j, w_{j+1}, \cdots, w_n)$ is the remaining words which have not been processed at the current state.

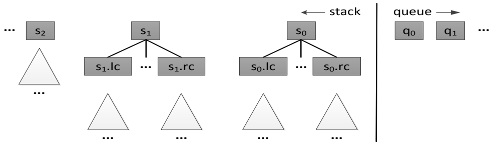


Figure 2: A state in the transition-based joint model. The figure is borrowed from (Huang and Sagae, 2010).

The candidate transition action A at each step is defined as follows:

- SHIFT(t) (SH(t)): move the head word w_j of queue Q into the stack S, assigning the word with the POS tag t.
- REDUCE-RIGHT (RR): merge the top two trees s_0, s_1 into a new subtree $s_0 \curvearrowleft s_1$.
- REDUCE-LEFT (RL): merge the top two trees s_0, s_1 into a new subtree $s_0 \curvearrowright s_1$.

Equation 2 describes the score function of the transition-based joint model.

$$\mathrm{Score}_{\mathrm{joint}}(\mathbf{x},\mathbf{t},\mathbf{d}) = \sum_{A_i=\mathrm{SHIFT(t)}} \mathbf{w}_{\mathrm{pos}} \cdot \mathbf{f}_{\mathrm{pos}}(\mathrm{ST}_i, A_i, t) + \sum \mathbf{w}_{\mathrm{syn}} \cdot \mathbf{f}_{\mathrm{syn}}(\mathrm{ST}_i, A_i) \tag{2}$$

where $\mathbf{f}_{\mathrm{pos}}(\cdot)$ refers to the general features exploited in POS tagging and $\mathbf{f}_{\mathrm{syn}}(\cdot)$ is all other features which are syntax related.

In this work, we employ the joint model of Joint-ZN^{-} in (Hatori et al., 2011) after considering the performance. We refer to their paper for more detailed descriptions. We thank the authors for sharing their code with us. We make some changes to make the basic POS tagging features the same with that in JGraph. We call this transition-based joint model as JTrans for brevity.

4 Stacking Joint Models

Stacked learning is a typical approach for model integration. The idea of stacked learning is to include two "level" of predictors : the level-0 includes one or more predictors $g_1, \cdots g_K (K \geq 1) : R^d \rightarrow R$ and the level-1 consists of one single predictor $h : R^{d+k} \rightarrow R$. Each predictor g_k of level 0 receives input $\mathbf{x} \in R^d$ and outputs a prediction$g_k(\mathbf{x})$. The level-1 predictor takes as input $\langle \mathbf{x}, g_1(\mathbf{x}), \cdots, g_K(\mathbf{x}) \rangle$ and outputs a final prediction $h(\mathbf{x}, g_1(\mathbf{x}), \cdots, g_K(\mathbf{x}))$.

In our work, we have two strategies for stacking JGraph and JTrans. The guided graph-based joint model exploits JGraph as the level-1 model and the guided transition-based joint model exploits JTrans as the level-1 model. We will describe them in the following respectively.

4.1 The Guided Graph-based Joint Model

The guided graph-based joint model, which we call JGraph(JTrans), exploits JGraph as the level-1 model and JTrans as level-0 model. As shown in Equation 1, the graph-based joint model computes a dependency tree along with POS tags by factored it into small parts including dependencies, siblings and grandchilds. Correspondingly, assuming the output of JTrans is $(\hat{\mathbf{t}}^{\mathrm{JTrans}} = t_1^{\mathrm{JTrans}} \cdots t_n^{\mathrm{JTrans}}, \hat{\mathbf{d}}^{\mathrm{JTrans}})$, the score function of JGraph(JTrans) becomes:

$$\begin{aligned}\mathrm{Score}_{\mathrm{joint}}(\mathbf{x},\mathbf{t},\mathbf{d}) = &\sum_{\{(h,m)\}\subseteq\mathbf{d}} \mathbf{w}_{\mathrm{pos}} \cdot \mathbf{f}_{\mathrm{pos}}(\mathbf{x},\mathbf{t},\hat{\mathbf{t}}^{\mathrm{JTrans}}, m) + \mathbf{w}_{\mathrm{dep}} \cdot \mathbf{f}_{\mathrm{dep}}(\mathbf{x},\mathbf{t},h,m,\hat{\mathbf{d}}^{\mathrm{JTrans}}) \\ &+ \sum_{\{(h,s)(h,m)\}\subseteq\mathbf{d}} \mathbf{w}_{\mathrm{sib}} \cdot \mathbf{f}_{\mathrm{sib}}(\mathbf{x},\mathbf{t},h,s,m,\hat{\mathbf{d}}^{\mathrm{JTrans}}) + \sum_{\{(g,h)(h,m)\}\subseteq\mathbf{d}} \mathbf{w}_{\mathrm{grd}} \cdot \mathbf{f}_{\mathrm{grd}}(\mathbf{x},\mathbf{t},g,h,m,\hat{\mathbf{d}}^{\mathrm{JTrans}})\end{aligned} \tag{3}$$

We can see that the feature functions are modified to include additional arguments $\hat{\mathbf{t}}^{\mathrm{JTrans}}$ and $\hat{\mathbf{d}}^{\mathrm{JTrans}}$. Thus new features related with the additional arguments will be produced. These new features account for the guided features over the output of JTrans. The specific features used by JGraph(JTrans) are given in Table 1.

4.2 The Guided Transition-based Joint Model

The guided transition-based joint model, which we call JTrans(JGraph), exploits JTrans as the level-1 model and JGraph as level-0 model. As shown in Equation 1, the transition-based joint model computes a dependency tree along with POS tags by the sequence of transition actions of shaping it. Correspondingly, assuming the output of JTrans is $(\hat{\mathbf{t}}^{\mathrm{JGraph}} = t_1^{\mathrm{JGraph}} \cdots t_n^{\mathrm{JoinG}}, \hat{\mathbf{d}}^{\mathrm{JGraph}})$, the score function of JTrans(JGraph) becomes:

$$\mathrm{Score}_{\mathrm{joint}}(\mathbf{x},\mathbf{t},\mathbf{d}) = \sum_{A_i=\mathrm{SHIFT(t)}} \mathbf{w}_{\mathrm{pos}} \cdot \mathbf{f}_{\mathrm{pos}}(\mathrm{ST}_i, A_i, t, \hat{\mathbf{t}}^{\mathrm{JGraph}}) + \sum \mathbf{w}_{\mathrm{syn}} \cdot \mathbf{f}_{\mathrm{syn}}(\mathrm{ST}_i, A_i, \hat{\mathbf{d}}^{\mathrm{JGraph}}) \tag{4}$$

The Guided Graph-based Joint Model: JGraph(JTrans)	
pos	$\{\hat{t}_m^{\text{JTrans}}, \hat{t}_m^{\text{JTrans}} \circ \hat{t}_{m-1}^{\text{JTrans}}, \hat{t}_m^{\text{JTrans}} \circ \hat{t}_{m+1}^{\text{JTrans}}, \hat{t}_{m-1}^{\text{JTrans}} \circ \hat{t}_{m+1}^{\text{JTrans}}\} \otimes \{t_m, w_m \circ t_m\}$
dep	{Whether $h\frown m$ is in $\hat{\mathbf{d}}^{\text{JTrans}}$?} $\otimes\{t_h, t_m, t_h \circ t_m\}$
sib	{Whether $h\frown m$ and $h\frown s$ are in $\hat{\mathbf{d}}^{\text{JTrans}}$?}$\otimes\{t_h, t_m, t_h \circ t_m\}$
grd	{Whether $g\frown h\frown m$ is in $\hat{\mathbf{d}}^{\text{JTrans}}$?}$\otimes\{t_h, t_m, t_h \circ t_m\}$
The Guided Transition-based Joint Model: JTrans(JGraph)	
pos	$\{t_m, w_m \circ t_m\} \otimes \{\hat{t}_m^{\text{JGraph}}, \hat{t}_m^{\text{JGraph}} \circ \hat{t}_{m-1}^{\text{JGraph}}, \hat{t}_m^{\text{JGraph}} \circ \hat{t}_{m+1}^{\text{JGraph}}, \hat{t}_{m-1}^{\text{JGraph}} \circ \hat{t}_{m+1}^{\text{JGraph}}\}$
syn	{Whether $s_0\frown s_1$ is in $\hat{\mathbf{d}}^{\text{JGraph}}$?, Whether $s_0\frown s_1$ is in $\hat{\mathbf{d}}^{\text{JGraph}}$?} $\otimes\{s_0.t, s_1.t, s_0.t \circ s_1.t\}$, {Whether $s_0\frown s_1$ is in $\hat{\mathbf{d}}^{\text{JGraph}}$?, Whether $s_0\frown s_1$ is in $\hat{\mathbf{d}}^{\text{JGraph}}$?} $\otimes$ {Whether $s_0\frown(s_0.\text{lc})$ is in $\hat{\mathbf{d}}^{\text{JGraph}}$?, Whether $s_0\frown(s_0.\text{rc})$ is in $\hat{\mathbf{d}}^{\text{JGraph}}$?, Whether $s_1\frown(s_1.\text{lc})$ is in $\hat{\mathbf{d}}^{\text{JGraph}}$?, Whether $s_1\frown(s_1.\text{rc})$ is in $\hat{\mathbf{d}}^{\text{JGraph}}$?} $\otimes\{s_0.t, s_1.t, s_0.t \circ s_1.t\}$

Table 1: Guided features for JGraph(JTrans) and JTrans(JGraph). These features are defined by referring to (Li et al., 2011a) for POS related features and (Nivre and McDonald, 2008) for syntax related features. The symbol $\otimes$ denotes a cross join operation, and the symbol $\circ$ denotes a conjoin operation. In features of JTrans(JGraph), the index of m refers to the involved word for tagging, $x.t$ denotes the POS tag of word x, and x.lc and x.rc denote x's leftmost and rightmost child.

We can see that the feature functions are modified to include additional arguments $\hat{\mathbf{t}}^{\text{JGraph}}$ and $\hat{\mathbf{d}}^{\text{JGraph}}$. Thus new features related with the additional arguments will be produced. These new features account for the guided features over the output of JGraph. The specific features used by JTrans(JGraph) are given in Table 1.

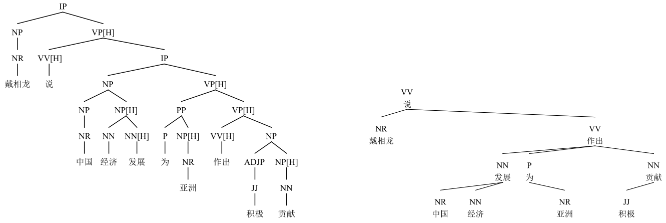
5 Constituent-based Joint Model and Further Stacking

5.1 Constituent-based Joint Model

Constituent-based joint models rely on certain PCFG parsers which exploit context-free grammar (CFG) to shape the search space for possible syntactic analysis. POS tagging and syntax analysis are usually processed simultaneously in PCFG parsers. These models have significant differences with the above two models. They model the joint POS tagging and dependency parsing in an indirect way. These models resolve the joint task by a two-step process: (1) constituent parsing and (2) rule-based transformation from constituent structures (CS) to dependency structures (DS). One advantage of constituent-based joint models is that all well-studied PCFG parsers can be used for the joint task. Figure 3 shows an example for this method.

In this work, we choose Berkeley parser[1] (Petrov et al., 2006; Petrov and Klein, 2007) to perform constituent parsing for its high performance in Chinese. We call this constituent-based joint model as JConst for brevity. Berkeley parser is an unlexicalized PCFG parser with latent variables. The observed constituent trees are automatically modeled with fined-grained unobserved constituent trees using latent variables. In Chinese POS tagging, (Sun and Uszkoreit, 2012) have proposed to use the output of Berkeley parser to enhance Chinese POS tagging. In Chinese dependency parsing, (Sun, 2012) has suggested to use Berkeley parser to improve the

[1] http://code.google.com/p/berkeleyparser



(a) Constituent structure with head nodes marked already.

(b) Dependency structure.

Figure 3: An example of constituent-based joint model: 戴相龙说中国经济发展为亚洲作出积极贡献(Dai Xianglong said that Chinese economic development made positive contributions for Asia). The left constituent structure can be converted to the right dependency structure, and the head nodes (marked by [H]) are specified by CS-to-DS rules.

performance of parsing accuracy, and (Che et al., 2012) have demonstrate that Berkeley parser outperforms several other PCFG parsers for Chinese Stanford dependencies.

5.2 Stacking Constituent-based Joint Model as the Second Level-0 Model

As mentioned in Section 4, we have introduced the integration of JGraph and JTrans by stacked learning. Here we concern a more complex case to integrate JConst into the ensemble model of JGraph and JTrans. Actually, it can be carried out very simply by adding the features of JConst similar to the other level-0 model. For example, if we exploit the guided graph-based joint model to integrate JConst, the guided features related with JConst are similar to that related with JTrans, which are produced by replacing the output of JTrans with JConst.

By adding JConst for further stacking, there are two level-0 models in stacking. Besides the independent guided features of the two level-0 models, we suppose that the consistency of the two level-0 models is also an effective indicator. For example, in dependency parsing, if $w_h \frown w_m$ exists in both level-0 models, then the probability of adding such a dependency should be much higher, and if $w_h \frown w_m$ exists in only one of the level-0 models, the probability should be lower but also a positive contribution, and further if $w_h \frown w_m$ doesn't exist in anyone of the level-0 models, the probability should be much lower with a negative impact. We name the features related with the consistency of the two level-0 models as *guided consistent features*. Table 2 lists the guided consistent features used in this work. Both the guided graph-based joint model and the guided transition-based joint model are considered.

6 Experiments

6.1 Experimental Settings

We use CTB5.1 to conduct our experiments. Following the works of (Li et al., 2011b) and (Hatori et al., 2011), we use the standard split of CTB5.1 described in (Duan et al., 2007) and the conversion rules of CS-to-DS in (Zhang and Clark, 2008).

We use the standard tagging accuracy to evaluate POS tagging. For dependency parsing, we use word accuracy (also known as dependency accuracy or UAS), root accuracy and complete

The Guided Graph-based Joint Model: JGraph(JTrans, JConst)	
pos	{Whether $\hat{t}_m^{\text{JTrans}}$ is identical to $\hat{t}_m^{\text{JConst}}$?} $\otimes\{\hat{t}_m^{\text{JTrans}} \circ t_m, \hat{t}_m^{\text{JTrans}} \circ w_m \circ t_m\}$
dep	{Whether the heads of m are identical in $\hat{\mathbf{d}}^{\text{JTrans}}$ and $\hat{\mathbf{d}}^{\text{JConst}}$?} $\otimes$ {Whether $h \frown m$ is in $\hat{\mathbf{d}}^{\text{JTrans}}$?} $\otimes\{t_h, t_m, t_h \circ t_m\}$
The Guided Transition-based Joint Model: JTrans(JGraph, JConst)	
pos	{Whether $\hat{t}_m^{\text{JGraph}}$ is identical to $\hat{t}_m^{\text{JConst}}$?} $\otimes\{\hat{t}_m^{\text{JGraph}} \circ t_m, \hat{t}_m^{\text{JGraph}} \circ w_m \circ t_m\}$
syn	{Whether the heads of s_0 are identical in $\hat{\mathbf{d}}^{\text{JGraph}}$ and $\hat{\mathbf{d}}^{\text{JConst}}$?, Whether the heads of s_1 are identical in $\hat{\mathbf{d}}^{\text{JGraph}}$ and $\hat{\mathbf{d}}^{\text{JConst}}$?} $\otimes$ {Whether $s_0 \frown s_1$ is in $\hat{\mathbf{d}}^{\text{JGraph}}$?, Whether $s_0 \frown s_1$ is in $\hat{\mathbf{d}}^{\text{JGraph}}$?} $\otimes\{s_0.t, s_1.t, s_0.t \circ s_1.t\}$

Table 2: Guided consistent features.

match rate (all excluding punctuation) to evaluate the performance.

The models are trained iteratively and the best one is chosen for final evaluation for each joint model in terms of tagging accuracy and dependency accuracy on development set. The iterative number for graph-based joint models and transition-based joint models are 15 and 50 respectively. The beam size of our transition-based joint models is set to 64. In stacked learning, we split the train data into five folds to get augmented train data for level-1 model.

6.2 Stacking Graph- and Transition-based Joint Models

6.2.1 Baseline Performance

At first, we evaluate the performance of our baseline joint models: JGraph and JTrans. The performance of the pipeline models of JGraph and JTrans are also reported. Table 3 shows the results. The POS tagger of pipeline models is trained using averaged perceptron (Collins, 2002), exploiting the features which are only related with POS tagging. Our POS tagger achieves a tagging accuracy of 94.34% on development set, and 94.11% on test set, which is higher than the pipeline models of (Li et al., 2011b) and (Hatori et al., 2011). PGraph and PTrans denote the pipeline models of JGraph and JTrans respectively.

As shown in Table 3, both JGraph and JTrans achieve higher parsing accuracies than the pipeline models, where JGraph achieves increases of 1.36% and JTrans achieves increases of 1.61%. In tagging accuracy, JGraph achieve increases of 0.4% compared to PGraph, whereas JTrans suffers very little loss compared to PTrans. JGraph has a significant improvement in POS tagging compared to the reported results in their paper as they have enhanced the model before sharing the code for us. We use the code shared by the authors of (Hatori et al., 2011) to train JTrans, meanwhile some modifications have been done to make the basic POS tagging features identical with JGraph. However our parsing accuracy is slightly lower than the reported results in their paper. The work of (Bohnet and Nivre, 2012) is the only one other related work for joint Chinese POS tagging and dependency parsing. They also reported their results on CTB5.1 data set. However we didn't make a comparison as they employed a different conversion method for CS-to-DS.

		Syntactic Metrics			Tagging
		word	root	compl.	Accuracy
Stacking	JGraph(JTrans)	82.04	**78.17**	30.21	94.52
	JTrans(JGraph)	**82.22**	78.03	**30.58**	**94.76**
Graph	JGraph	**80.88**	**75.55**	**28.83**	**94.51**
	PGraph	79.52	75.34	26.70	94.11
	(Li et al., 2011b) (v1,2^{nd})	80.74	75.80	28.24	93.08
Transition	JTrans	**80.91**	**76.91**	**29.32**	94.07
	PTrans	79.30	75.73	27.80	**94.11**
	(Hatori et al., 2011)(ZN)	81.33	77.93	29.90	93.94

Table 3: The results of baseline models on test corpus. Results of significant test show that the p-value of tagging accuracy is lower than 10^{-3} and the p-value of parsing accuracy is lower than 10^{-5} for both JGraph(JTrans) and JTrans(JGraph).

6.2.2 Stacking Results

Table 3 shows the results of integration JGraph and JConst by stacked learning. We can see both the guided graph-based joint model and the guided transition-based achieve significantly improvements compared to the baseline models. The guided transition-based joint model obtains slightly better performance than the guided graph-based joint model, achieving the tagging accuracy of 94.76% and the parsing accuracy of 82.22%. (Nivre and McDonald, 2008) have done a similar work on dependency parsing. Their result demonstrate that the the guided graph-based model is better than the guided transition-based model in Chinese. Our results are different. This is perhaps caused by our baseline models are more complex, especially the transition-based joint model.

6.3 Stacking Constituent-based Joint Model as the Second Level-0 Model

We have shown that the guided transition-based joint model achieves better performance. Thus we integrate the constituent-based joint model into this guided model in our work.

6.3.1 Performance of the Constituent-based Joint Model

Table 5 shows the performance of JConst. The parsing accuracy of JConst is slightly higher than JGraph and JTrans. This demonstrates that JConst is also an effective method for joint Chinese POS tagging and dependency parsing. However, the tagging accuracy is not good. This may be caused by that Berkeley parser is a generative model, and thus relatively poor POS related features can be used in it.

6.3.2 Effectiveness of Guided Consistent Features

For further stacking JConst, we have suggested the guided consistent features. To test the effect of guided consistent features, we conduct feature ablation experiments. Table 4 shows the results on development set, where the mark {***-GC***} denotes the model without guided consistent features. The ablation of the guided consistent features resulted in 0.2% decreases in tagging accuracy and 0.28% decreases of parsing accuracy for JTrans(JGraph, JConst), showing the effectiveness of these features.

	Syntactic Metrics			Tagging
	word	root	compl.	Accuracy
JTrans(JGraph, JConst)	**84.68**	**80.38**	**34.37**	**95.31**
JTrans(JGraph, JConst){**-GC**}	84.40	79.82	34.12	95.11

Table 4: Feature ablation for guided consistent features.

6.3.3 Final Results

Table 5 shows the final results of stacking JGraph, JTrans and JConst together. The final stacking model JTrans(JGraph, JConst) achieves a tagging accuracy of 94.95% and a dependency accuracy of 83.98%, obtaining further increases of 0.19% in tagging accuracy and 1.76% in parsing accuracy compared to JTrans(JGraph). The improvements of JTrans(JGraph, JConst) compared to JTrans(JGraph) are larger than that of JTrans(JGraph) compared to JGraph or JTrans. In both tagging and parsing performance, JTrans(JGraph, JConst) is better than JTrans(JGraph), and meanwhile JTrans(JGraph) is better than anyone of {JGraph, JTrans, JConst} . It demonstrates that each individual joint model has a positive impact in the stacking.

	Syntactic Metrics			Tagging
	word	root	compl.	Accuracy
JTrans(JGraph, JConst)	**83.98**	**81.29**	**32.15**	**94.95**
JConst	81.03	78.12	28.01	93.45

Table 5: Final results of further stacking for constituent-based joint model.

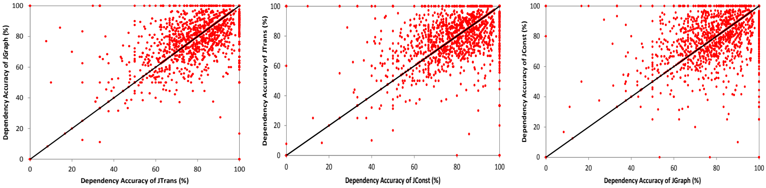
7 Analysis

The analysis is conducted on test corpus of CTB5.1. It includes two aspects. First we carefully examine the error distributions of the joint models, which can help us to figure out how stacked learning helps Chinese POS tagging and dependency parsing. Then we compare the joint models to pipeline models, which can help us to understand the interaction between Chinese POS tagging and dependency parsing.

7.1 Comparisons between Heterogeneous Joint Models

In this section, we mainly concern the performance of dependency parsing. Generally, stacking can perform effectively when the differences between baseline models are very large. For the final stacking model JTrans(JGraph, JConst), the baseline models include JGraph, JTrans, JConst.

First, we display the scatter plots of the parsing accuracies of JGraph against JTrans, JTrans against JConst and JConst against JGraph respectively. Each point (x, y) in the scatter plots denotes a sentence's parsing accuracy in the two joint models. If the point is upper the line $y = x$, it denotes that the joint model represented by vertical axis achieve higher dependency accuracy for the sentence. As is shown in Figure 4, the points seem to be random distributed in the three plots, and the number of points divided by line $y = x$ seems equal. These demonstrates that the error distributions of the three baseline models are rather different. Each model can process better on some sentences, and also may perform worse on some other sentences. By stacking, we can take advantage of the strengths of each model, thus resulting in a better performance.



(a) x:JTrans,y:JGraph (b) x:JConst,y:JTrans (c) x:JGraph,y:JConst

Figure 4: Scatter plots of dependency accuracies of every two baseline models.

Figure 5 shows the performance of dependency parsing in term of dependency length. Dependency length denotes the distance between the head and modifier in a dependency. Generally, dependency accuracies decrease when the distance between the head and modifier become longer. JConst can do better than the other two baseline models. JGraph and JTrans model dependency trees directly, and dependency length is a very important factor when building a dependency to shape the tree, thus JGraph and JTrans can be influenced much by it. However, JConst builds a dependency tree indirectly, the syntax analysis is done under the grammar of CFG, which needn't consider the attributes of a dependency tree. Thus dependency length has smaller influence to JConst. During the stacking, the model can learn that JConst is good at the dependencies with long dependency length. Thus the final stacking model can perform well for long dependency length, which is shown in Figure 5.

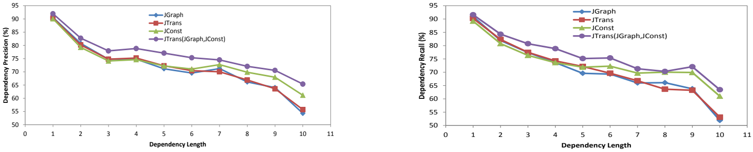


Figure 5: Dependency arc precision/recall relative to predicted/gold dependency length.

7.2 Comparisons between Joint Models and Pipeline Models

Impact on POS tagging : Intuitively, the tagging accuracy should be much higher if the dependency structures are correctly predicted. Table 6 shows the tagging accuracies of the different models with correct dependency heads and wrong heads respectively. The accuracies of words with correct heads are 10% higher than that with wrong heads on average. Joint models are more easily affected by the correctness of dependency head compared to PGraph[2]. This is caused by that we have exploited the syntactic features for POS tagging in joint models.

Next, we investigate which POS tagging error patterns can be influenced greatly in joint models. Figure 6 shows the related high frequency error patterns. The error patterns with rectangles upper the horizontal line are positive error patterns which joint models can do better, and the error patterns with rectangles below are negative error patterns which joint models can

[2] We only choose PGraph for comparison since it achieves better performance than PTrans.

	PGraph	JGraph	JTrans	JConst	JTrans(JGraph, JConst)
Correctly Headed	96.23	96.92$\uparrow$	96.64$\uparrow$	96.36$\uparrow$	97.03$\uparrow$
Wrongly Headed	86.65	85.34$\downarrow$	84.16$\downarrow$	81.79$\downarrow$	84.7$\downarrow$

Table 6: POS tagging accuracies with correct dependency heads and wrong dependency heads.

do worse. We can see that the joint models can do significantly better on such error patterns like NN → VV and DEC → DEG, and in contrast joint models can do worse on error patterns for example NR → NN and NN → JJ. Actually, the positive error patterns are important for dependency parsing and the positive error patterns usually needn't to be distinguished in dependency parsing, we will demonstrate it later.

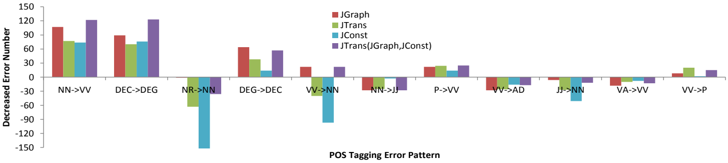


Figure 6: POS tagging error patterns influenced significantly by joint models.

Finally, the arc direction of dependency is also an important factor to influence the POS tagging accuracy. Table 7 shows the POS tagging accuracy of modifier words in term of gold arc direction $w_i \frown w_j$, $w_i \frown w_j$ $(i < j)$[3]. In Chinese, the POS tags such as NN, JJ or AD, which are difficult to be distinguished by dependency parsing, are usually tagged for modifier words in $w_i \frown w_j$, thus we can see that joint models gain little improvements for $w_i \frown w_j$.

	PGraph	JGraph	JTrans	JConst	JTrans(JGraph, JConst)
$w_i \frown w_j$	94.12	94.24	93.78	92.86	94.51
$w_i \frown w_j$	93.95	94.92	94.54	94.54	95.62

Table 7: POS tagging accuracies of modifier words in term of the gold arc direction.

Impact on dependency parsing : We mainly concern how POS tagging errors influence the performance of dependency parsing. In the previous discussion, we have demonstrated that dependency structures can influence some certain POS tagging error patterns significantly. We expect that the parsing accuracies should decrease sharply for the POS error patterns who demonstrate positive influences, because we suppose that these error patterns are harmful for dependency parsing, thus they can be distinguished by dependency parsing. Figure 7 shows the dependency accuracies of these POS tagging error patterns. We can see that the parsing accuracies of joint models for positive error patterns such as NN → VV, DEC → DEG and DEG → DEC drop drastically compared to PGraph. This conforms to our expectation. It demonstrates that joint models can do much better on certain POS tagging errors which are harmful for dependency parsing.

We also look into the impact of dependency arc direction on dependency parsing. Table 8 shows the dependency accuracies in term of gold arc direction. Generally, joint models do much better than PGraph in both $w_i \frown w_j$ and $w_i \frown w_j$. The accuracy of $w_i \frown w_j$ is significantly higher than $w_i \frown w_j$. One interpretation is that the words with POS tags such as NN, JJ or AD are usually

[3] We neglect the ROOT when considering the arc direction.

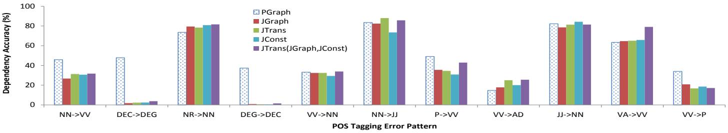


Figure 7: Dependency accuracies of different POS tagging error patterns.

easier to find their dependency head, and the head of such words is always on the right. Further, the accuracy gap of JConst between $w_i \curvearrowleft w_j$ and $w_i \curvearrowright w_j$ is apparently smaller than the JGraph and JConst, which is similar to the phenomenon that dependency length has smaller influence to JConst, as dependency length and arc direction are important factors in JTrans and JConst which process dependency parsing directly.

	PGraph	**JGraph**	**JTrans**	**JConst**	**JTrans(JGraph, JConst)**
$w_i \curvearrowleft w_j$	81.03	82.01	82.44	81.75	84.81
$w_i \curvearrowright w_j$	76.45	78.92	77.73	79.7	82.35

Table 8: Dependency accuracies in term of the gold arc direction.

Conclusions

In this paper, we present a study to investigate the integration of joint models of Chinese POS tagging and dependency parsing by stacked learning. First we integrate two joint models: JGraph and JTrans. We find that the stacking can improve the performance of joint models significantly. Meanwhile our experimental results demonstrate that the guided transition-based joint model can do better than the guided graph-based joint model. Next, we introduce a constituent-based joint model JConst, which employ a PCFG parser to get the result of constituent structure and then extract the results of POS tagging and dependency parsing from it. To further improve the performance of joint models, we integrate the JConst into the guided transition-based joint model by further stacking. The final stacking joint model achieves a POS tagging accuracy of 94.95% and a parsing accuracy of 83.98% in Chinese, resulting in total error reductions of 8% and 16% respectively compared to the best model of JGraph, JTrans and JConst.

Further, we conduct a detailed analysis aiming to figure out how stacked learning helps dependency parsing and POS tagging, and meanwhile aiming to understand the relationship between Chinese POS tagging and dependency parsing. We can find that JGraph, JTrans and JConst are very different in error distribution, joint models can do much better on certain POS tagging errors which are harmful for dependency parsing.

Acknowledgments

We especially thank Weiwei Sun for her suggestion of stacking constituent-based models in this work. This work was supported by National Natural Science Foundation of China (NSFC) via grant 61133012, the National "863" Major Projects via grant 2011AA01A207, and the National "863" Leading Technology Research Project via grant 2012AA011102.

References

Bikel, D. M. (2004). Intricacies of collins parsing model. *Computational Linguistics*, 30(4):479–511.

Bohnet, B. and Nivre, J. (2012). A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1455–1465, Jeju Island, Korea. Association for Computational Linguistics.

Breiman, L. (1996). Stacked regressions. *Machine Learning*, 24:49–64.

Carreras, X. (2007). Experiments with a higher-order projective dependency parser. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL*, pages 957–961.

Charniak, E. (2000). A maximum-entropy-inspired parser. In *Proceedings of the Second Meeting of North American Chapter of Association for Computational Linguistics (NAACL2000)*.

Charniak, E. and Johnson, M. (2005). Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 173–180, Ann Arbor, Michigan. Association for Computational Linguistics.

Che, W., Spitkovsky, V., and Liu, T. (2012). A comparison of chinese parsers for stanford dependencies. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 11–16, Jeju Island, Korea. Association for Computational Linguistics.

Collins, M. (1999). *Head-Driven Statistical Models for Natural Language Parsing*. PhD thesis, Pennsylvania University.

Collins, M. (2002). Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pages 1–8. Association for Computational Linguistics.

Dekai Wu, G. N. and Carpuat, M. (2003). A stacked, voted, stacked model for named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*.

Duan, X., Zhao, J., and Xu, B. (2007). Probabilistic models for action-based chinese dependency parsing. In Kok, J. N., Koronacki, J., de Mántaras, R. L., Matwin, S., Mladenic, D., and Skowron, A., editors, *Proceedings of ECML/ECPPKDD*, volume 4701 of *Lecture Notes in Computer Science*, pages 559–566. Springer.

Hatori, J., Matsuzaki, T., Miyao, Y., and Tsujii, J. (2011). Incremental joint pos tagging and dependency parsing in chinese. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 1216–1224, Chiang Mai, Thailand. Asian Federation of Natural Language Processing.

Huang, L. and Sagae, K. (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. Association for Computational Linguistics.

Koo, T. and Collins, M. (2010). Efficient third-order dependency parsers. In *Proceedings of the 48th Annual Meeting of the ACL*, number July, pages 1–11.

Lafferty, J. D., McCallum, A., and Pereira, F. C. N. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, pages 282–289.

Li, Z., Che, W., and Liu, T. (2011a). Improving chinese pos tagging with dependency parsing. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 1447–1451, Chiang Mai, Thailand. Asian Federation of Natural Language Processing.

Li, Z., Zhang, M., Che, W., Liu, T., Chen, W., and Li, H. (2011b). Joint models for chinese pos tagging and dependency parsing. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1180–1191, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Martins, A. F. T., Das, D., Smith, N. A., and Xing, E. P. (2008). Stacking dependency parsers. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 157–166, Honolulu, Hawaii. Association for Computational Linguistics.

McDonald, R. (2006). *Discriminative learning and spanning tree algorithms for dependency parsing*. PhD thesis, University of Pennsylvania.

McDonald, R., Crammer, K., and Pereira, F. (2005). Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics - ACL '05*, number June, pages 91–98, Morristown, NJ, USA.

McDonald, R. and Nivre, J. (2011). Analyzing and integrating dependency parsers. *Computational Linguistics*, 37(1):197–230.

McDonald, R. and Pereira, F. (2006). Online learning of approximate dependency parsing algorithms. In *Proceedings of EACL*.

Nivre, J. (2008). Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34(4):513–553.

Nivre, J. and McDonald, R. (2008). Integrating graph-based and transition-based dependency parsers. In *Proceedings of ACL-08: HLT*, pages 950–958, Columbus, Ohio. Association for Computational Linguistics.

Petrov, S., Barrett, L., Thibaux, R., and Klein, D. (2006). Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 433–440, Sydney, Australia. Association for Computational Linguistics.

Petrov, S. and Klein, D. (2007). Improved inference for unlexicalized parsing. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 404–411, Rochester, New York. Association for Computational Linguistics.

Ratnaparkhi, A. (1996). A maximum entropy part-of-speech tagger. In *Proceedings of the Empirical Methods in Natural Language Processing*.

Søgaard, A. and Rishøj, C. (2010). Semi-supervised dependency parsing using generalized tri-training. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 1065–1073, Beijing, China. Coling 2010 Organizing Committee.

Sun, W. (2011). A stacked sub-word model for joint chinese word segmentation and part-of-speech tagging. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1385–1394, Portland, Oregon, USA. Association for Computational Linguistics.

Sun, W. (2012). *Learning Chinese Language Structures with Multiple Views*. PhD thesis, Saarland University.

Sun, W. and Uszkoreit, H. (2012). Capturing paradigmatic and syntagmatic lexical relations: Towards accurate chinese part-of-speech tagging. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 242–252, Jeju Island, Korea. Association for Computational Linguistics.

Wolpert, D. H. (1992). Stacked generalization. *Neural Networks*, 5:241–259.

Xue, N., Xia, F., Chiou, F.-D., and Palmer, M. (2005). The penn chinese treebank: Phrase structure annotation of a large corpus. *Natural Language Engineering*, 11(2):207–238.

Yamada, H. and Matsumoto, Y. (2003). Statistical dependency analysis with support vector machines. In *Proceedings of 8th International Workshop on Parsing Technologies (IWPT2003)*.

Zhang, Y. and Clark, S. (2008). 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. Association for Computational Linguistics.

Zhang, Y. and Nivre, J. (2011). Transition-based dependency parsing with rich non-local features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 188–193, Portland, Oregon, USA. Association for Computational Linguistics.