# Data-driven, PCFG-based and Pseudo-PCFG-based Models for Chinese Dependency Parsing

**Weiwei Sun** and **Xiaojun Wan**
Institute of Computer Science and Technology, Peking University
The MOE Key Laboratory of Computational Linguistics, Peking University
{ws,wanxiaojun}@pku.edu.cn

## Abstract

We present a comparative study of transition-, graph- and PCFG-based models aimed at illuminating more precisely the likely contribution of CFGs in improving Chinese dependency parsing accuracy, especially by combining heterogeneous models. Inspired by the impact of a constituency grammar on dependency parsing, we propose several strategies to acquire pseudo CFGs only from dependency annotations. Compared to linguistic grammars learned from rich phrase-structure treebanks, well designed pseudo grammars achieve similar parsing accuracy and have equivalent contributions to parser ensemble. Moreover, pseudo grammars increase the diversity of base models; therefore, together with all other models, further improve system combination. Based on automatic POS tagging, our final model achieves a UAS of 87.23%, resulting in a significant improvement of the state of the art.

## 1 Introduction

Popular approaches to dependency parsing can be divided into two classes: grammar-free and grammar-based. Data-driven, grammar-free approaches make essential use of machine learning from linguistic annotations in order to parse new sentences. Such approaches, e.g. transition-based (Nivre, 2008) and graph-based (McDonald, 2006; Torres Martins et al., 2009) have attracted the most attention in recent years. In contrast, grammar-based approaches rely on linguistic grammars (in either dependency or constituency formalisms) to shape the search space for possible syntactic analysis. In particular, CFG-based dependency parsing exploits a mapping between dependency and constituency representations and reuses parsing algorithms developed for CFG to produce dependency structures. In previous work, data-driven, discriminative approaches have been widely discussed for Chinese dependency parsing. On the other hand, various PCFG-based constituent parsing methods have been applied to obtain phrase-structures as well. With rich linguistic rules, phrase-structures of Chinese sentences can be well transformed to their corresponding dependency structures (Xue, 2007). Therefore, PCFG parsers with such conversion rules can be taken as another type of dependency parser. We call them PCFG-based parsers, in this paper.

Explicitly defining linguistic rules to express precisely generic grammatical regularities, a constituency grammar can be applied to arrange sentences into a hierarchy of nested phrases, which determines constructions between larger phrases and their smaller component phrases. This type of information is different from, but highly related to, the information captured by a dependency representation. A constituency grammar, thus, has great possible contributions to dependency parsing. In order to pave the way for new and better methods, we study the impact of CFGs on Chinese dependency parsing. A series of empirical analysis of state-of-the-art graph-, transition-and PCFG-based parsers is presented to illuminate more precisely the properties of heterogeneous models. We show that CFGs have a great impact on dependency parsing and PCFG-based models have complementary predictive powers to data-driven models.

System ensemble is an effective and important technique to build more accurate parsers based on multiple, diverse, weaker models. Exploiting differ-



Transactions of the Association for Computational Linguistics, 1 (2013) 301–314. Action Editor: Jason Eisner.
Submitted 6/2012; Revised 10/2012; Published 7/2013. 

ent data-driven models, e.g. transition- and graph-based models, has received the most attention in dependency parser ensemble (Nivre and McDonald, 2008; Torres Martins et al., 2008; Sagae and Lavie, 2006). Only a few works investigate integrating data-driven and PCFG-based models (McDonald, 2006). We argue that grammars can significantly increase the diversity of base models, which plays a central role in parser ensemble, and therefore lead to better and more promising hybrid systems.

We introduce a general classifier enhancing technique, i.e. *bootstrap aggregating* (Bagging), to improve dependency parsing accuracy. This technique can be applied to enhance a single-view parser, or to combine multiple heterogeneous parsers. Experiments on the CoNLL 09 shared task data demonstrate its effectiveness: (1) Bagging can improve individual single-view parsers, especially the PCFG-based one; (2) Bagging is more effective than previously introduced ensemble methods to combine multi-view parsers; (3) Integrating data-driven and PCFG-based models is more useful than combining different data-driven models.

Although PCFG-based models have a big contribution to data-driven dependency parsing, they have a serious limitation: There are no corresponding constituency annotations for some dependency treebanks, e.g. Chinese Dependency Treebank (LDC2012T05). To overcome this limitation, we propose several strategies to acquire pseudo grammars only from dependency annotations. In particular, dependency trees are converted to pseudo constituency trees and PCFGs can be extracted from such trees. Another motivation of this study is to increase the diversity of candidate models for parser ensemble. Experiments show that pseudo-PCFG-based models are very competitive: (1) Pseudo grammars achieve similar or even better parsing results than linguistic grammars learned from rich constituency annotations; (2) Compared to linguistic grammars, well designed, single-view pseudo grammars have an equivalent contribution to parser ensemble; (3) Combining different pseudo grammars even work better for ensemble than linguistic grammars; (4) Pseudo-PCFG-based models increase the diversity of base models, and therefore lead to further improvements for ensemble.

Based on automatic POS tagging, our final model achieves a UAS of 87.23% on the CoNLL data and 84.65% on CTB5, which yield relative error reductions of 18-24% over the best published results in the literature.

## 2 Background and related work

### 2.1 Data-driven dependency parsing

The mainstream work on recent dependency parsing focuses on data-driven approaches that automatically learn to produce dependency graphs for sentences solely from a hand-crafted dependency treebank. The advantage of such models is that they are easily ported to any language in which labeled linguistic resources exist. Practically all statistical models that have been proposed in recent years can be mainly described as either *graph-based* or *transition-based* (McDonald and Nivre, 2007). Both models have been adopted to learn Chinese dependency structures (Zhang and Clark, 2011; Zhang and Nivre, 2011; Huang and Sagae, 2010; Hatori et al., 2011; Li et al., 2011, 2012). According to published results, graph-based and transition-based parsers achieve similar accuracy.

In the graph-based framework, informative evaluation results have been presented in (Li et al., 2011). First, second and third order projective parsing models are well evaluated. In the transition-based framework, two advanced techniques have been studied. First, developing features has been shown crucial to advancing parsing accuracy and a very rich feature set is carefully evaluated by Zhang and Nivre (2011). Second, beyond deterministic greedy search, principled dynamic programming strategies can be employed to explore more possible hypotheses (Huang and Sagae, 2010). Both techniques have been examined and shown helpful for Chinese dependency parsing. Furthermore, Hatori et al. (2011) combined both and obtained a state-of-the-art supervised parsing result.

### 2.2 PCFG-based dependency parsing

PCFG-based dependency parsing approaches are based on the finding that projective dependency trees can be transformed from constituency trees by applying rich linguistic rules. In such approaches, dependency parsing can be resolved by a two-step process: constituent parsing and rule-based extraction

of dependencies from phrase structures. The advantage of constituency-grammar-based approach is that all the well-studied parsing methods for such grammars can be used for dependency parsing as well. Two language-specific properties essentially make PCFG-based approaches easy to be applied to Chinese dependency parsing: (1) Chinese grammaticians favor using projective structures;[1] (2) Chinese phrase-structure annotations normally contain richer information and thus are reliable for tree conversion.

#### 2.2.1 Constituency parsing

Compared to many other languages, statistical constituent parsing for Chinese has reached early success, due to the fact that the language has relatively fixed word order and extremely poor inflectional morphology. Both facts allow PCFG-based statistical modeling to perform well. For the constituent parsing, the majority of the state-of-the-art parsers are based on generative PCFG learning. For example, the well-known and successful Collins and Charniak&Johnson parsers (Collins, 2003; Charniak, 2000; Charniak and Johnson, 2005) implement generative lexicalized statistical models.

Apart from lexicalized PCFG parsing, unlexicalized parsing with latent variable grammars (PCFGLA) can also produce comparable accuracy (Matsuzaki et al., 2005; Petrov et al., 2006). Latent variable grammars model an observed treebank of coarse parse trees with a model over more refined, but unobserved, derivation trees that represent much more complex syntactic processes. Rather than attempting to manually specify fine-grained categories, previous work shows that automatically inducing the sub-categories from data can work quite well. A PCFGLA parser leverages on an automatic procedure to learn refined grammars and are therefore more robust to parse non-English languages that are not well studied. For Chinese, such a parser achieves the state-of-the-art performance and defeats many other types of parsers, including Collins as well as Charniak parser (Che et al., 2012) and discriminative transition-based models (Zhang and Clark, 2009).

[1]For example, as two popular dependency treebanks, the CoNLL 2009 data and the Chinese Dependency Treebank both excluede non-projective annotations. It is worth noting that the former one is converted from a constituency treebank while the latter one is directly annotated by lingusitics.

#### 2.2.2 CS to DS conversion

In the absence of dependency and constituency structures for a particular treebank, treebank-guided parser developers normally apply rich linguistic rules to convert one representation formalism to another to get necessary data to train parsers. Xue (2007) examines the linguistic adequacy of dependency structure annotation automatically converted from phrase structure treebanks with rule-based approaches. A structural approach is introduced for the constituency structure (CS) to dependency structure (DS) conversion for the Chinese Treebank data, which is the basis of the CoNLL 2009 shared task data. By applying this conversion procedure on the outputs of an automatic phrase structure parser, we can build a PCFG-based dependency parser.

### 2.3 Parser ensemble

NLP systems built on particular single views normally capture different properties of an original problem, and therefore differ in predictive powers. As a result, NLP systems can take advantage of complementary strengths of multiple views. Combining the outputs of several systems has been shown in the past to improve parsing performance significantly, including integrating phrase-structure parsers (Henderson and Brill, 1999), dependency parsers (Nivre and McDonald, 2008), or both (McDonald, 2006). Several ensemble models have been proposed for the parsing of syntactic constituents and dependencies, including learning-based stacking (Nivre and McDonald, 2008; Torres Martins et al., 2008) and learning-free post-inference (Henderson and Brill, 1999; Sagae and Lavie, 2006). Surdeanu and Manning (2010) present a systematic analysis of these ensemble methods and find several non-obvious facts:

- the diversity of base parsers is more important than complex models for learning, and
- simplest scoring model for voting and reparsing performs essentially as well as other more complex models.

## 3 A comparative analysis of heterogeneous parsers

The information encoded in a dependency representation is different from the information captured in a constituency representation. While the dependency structure represents head-dependent relations between words, the constituency structure represents the grouping of words into phrases, classified by structural categories. These differences concern what is explicitly encoded in the respective representations, and affects data-driven and PCFG-based dependency parsing models substantially. In this section, we give a comparative analysis of transition-, graph- and PCFG-based models aimed at illuminating more precisely the likely contribution of CFGs in dependency parsing.

### 3.1 Experimental setup

Penn Chinese TreeBank (CTB) is a segmented, POS tagged, and fully bracketed corpus in the constituency formalism, and very popular to evaluate fundamental NLP tasks, including word segmentation, POS tagging, constituent parsing as well as dependency parsing. We use CTB 6 as our main corpus and define the training, development and test sets according to the CoNLL 2009 shared task. To evaluate and analyze dependency parsers, we directly use the CoNLL data. CTB's syntactic annotations also includes functional information and empty categories. Modern parsers, e.g. Collins and Berkeley parsers, ignore these types of linguistic knowledge. To train a constituent parser, we perform a heuristic procedure on the treebank data to delete function tags and empty categories as well as its associated redundant ancestors. Many papers reported parsing results of an older version CTB (namely CTB 5). To compare with systems introduced in these papers, we evaluate our final ensemble model on CTB5 in Section 5.4.

For dependency parsing, we choose a second order graph-based parser[2] (Bohnet, 2010) and a transition-based parser (Hatori et al., 2011), for experiments. For constituent parsing, we choose Berkeley parser,[3] a well known implementation of the unlexicalized PCFGLA model and Bikel parser,[4] a well known implementation of Collins' lexicalized model, for experiments. In data-driven parsing, features consisting of POS tags are very effective, so typically POS tagging is performed as a preprocessing. We use the baseline sequential tagger described in (Sun and Uszkoreit, 2012) to provide such lexical information to the graph-based parser. Note that the transition-based parser performs a joint inference to acquire POS and dependency information simultaneously, so there is no need to offer extra tagging results to it.

[2] code.google.com/p/mate-tools/
[3] code.google.com/p/berkeleyparser/
[4] cis.upenn.edu/~dbikel/software.html

### 3.2 Overall performance

Table 1 (Column 2-6) summarizes the overall accuracy of different parsers. Two transition-based parsing results are presented: The first one employ a simple feature set (Zhang and Clark, 2008) and a small beam (16); the second one employ rich features (Zhang and Nivre, 2011) and a larger beam (32). Two graph-based parsing results are reported; the difference between them is whether integrate relation labels into the parsing procedure. Roughly speaking, currently state-of-the-art data-driven models achieves slightly better precision than unlexicalized PCFG-based models with regard to unlabeled dependency prediction.

There is a big gap between lexicalized and unlexicalized parsing. The same phenomenon has been observed by (Che et al., 2012) and (Zhuang and Zong, 2010). In addition to dependency parsing, Zhuang and Zong (2010) found that Berkeley parser produce much more accurate syntactic analyses to assist a Chinese semantic role labeler than Bikel parser. Charniak and Stanford parsers are two other well-known and frequently used tools that can provide lexicalized parsing results. According to (Che et al., 2012), they perform even worse than Bikel parser, at least for Stanford dependencies. Due to the poor parsing performance, we only concentrate on the unlexicalized model in the remainder of this paper.

The performance of labeled dependency prediction of the unlexicalized PCFG-based parser is much lower. We can learn that the CS to DS conversion is not robust to assign functional categories to dependencies and simple linguistic rules are not capable to do fine-grained classification. Previous research on English indicates that the main difficulty in dependency parsing is the prediction of dependency

| Devel. | UAS | LAS | Compl. | $F_{sib}$ | $F_{grd}$ |
|---|---|---|---|---|---|
| Tran[b=16,Z08] | 82.80 | N/A | 29.00 | 66.55 | 79.74 |
| Tran[b=32,Z11] | 83.80 | N/A | 31.61 | 68.58 | 80.87 |
| Graph[-lab] | 83.66 | N/A | 29.28 | 67.96 | 80.82 |
| Graph[+lab] | 84.24 | 80.55 | 30.99 | 69.11 | 81.38 |
| Unlex | 82.86 | 67.44 | 27.98 | 69.07 | 81.22 |
| Lex | 70.38 | 58.10 | - - | - - | - - |
| Bagging(15) | | | | | |
| Tran[b=16,Z08] | 83.25 | N/A | 28.66 | 67.17 | 78.89 |
| Tran[b=32,Z11] | 84.25 | N/A | 31.21 | 69.14 | 81.49 |
| Graph[-lab] | 83.81 | N/A | 29.68 | 68.00 | 80.62 |
| Graph[+lab] | 84.50 | N/A | 31.44 | 69.48 | 81.10 |
| Unlex | 84.92 | N/A | 32.35 | 71.08 | 83.66 |
| Bagging(8) | | | | | |
| Unlex | 84.35 | N/A | 31.16 | 70.49 | 83.57 |

Table 1: Accuracy of different parsers. The first block presents baseline parsers; the last two blocks present Bagging-enhanced parsers, where $m$ is respectively set to 15 and 8. *Z08* and *Z11* distinguish different feature sets; *b=16* and *b=32* are beam sizes. *+/-lab* means whether to incorporate relation labels to a model.

structures, and an extra statistical classifier can be employed to label automatically recognized dependencies with a high accuracy. Although this issue is not well studied for Chinese dependency parsing, previous research on function tag labeling (Sun and Sui, 2009) and semantic role labeling (Sun, 2010a) gives us some clues. Their research shows that both functional and predicate-argument structural information is relatively easy to predict if high-quality syntactic parses are available. We mainly focus on the UAS metric in the following experiments.

### 3.3 Constraints

A grammar-based model utilizes an explicitly defined formal grammar to shape the search space for possible syntactic hypotheses. Parameters of a statistical grammar-based model are related to a grammar rule, and as a result specific language constructions are constrained by each other. For example, parameters are assigned to rewrite rules for a CFG-based model. Since the PCFG-based model leverages rewrite rules to locally constrain several possible dependents for one head word, it does *relatively* better for locally connected dependencies. The traditional evaluation metrics, i.e. UAS and LAS, only consider bi-lexical (first order) dependencies, which are smallest pieces of a dependency structure. Besides bi-lexical dependencies, we report the prediction accuracy of grandparent and sibling dependencies, i.e. second order dependencies. The metrics are defined as follows.

- For every word $d$ whose parent is not the root, we consider the word triple $\langle d, p, g \rangle$ among $d$ and its parent $p$ and grandparent $g$. A word triple $\langle d, p, g \rangle$ from a predicted tree is considered as correct if it also apprears in the corresponding gold tree. Based on this definition, precison, recall and f-score of grandparent dependency can be defined in a *normal* sense. All punctuations are excluded for calculation.

- For every word $h$ that governs at least two children $(d_1, ..., d_n)$, we consider every word triple $\langle h, d_i, d_{i+1} \rangle$, among $h$ and its sibling dependents $d_i$ as well as $d_{i+1}$ $(0 \leq i < n)$. Similar to the grandparent dependencies, we can define evaluation metrics for sibling dependencies.

From Table 1, we can see that the grammar-based model parses relatively better for slightly larger fragments. For example, the UAS of the graph-based model is significantly higher than the grammar-based one, but their sibling and grandparent scores are similar. In the next section, we will introduce a general parser enhancement technique and present more discussions based on enhanced parsing results (Column 7-14).

### 3.4 Endocentric and exocentric constructions

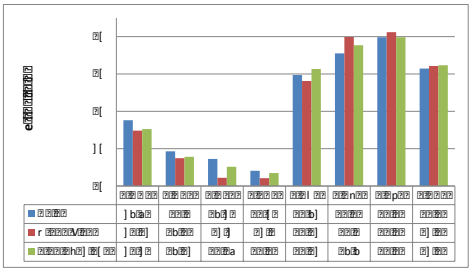

Figure 1: Nominal vs. verbal constructions.

Arguments in exocentric constructions help complete the meaning of a predicate and are taken to be obligatory and selected by their heads; adjuncts in

endocentric constructions are structurally dispensable parts that provide auxiliary information and taken to be optional and not selected by their heads. An important annotation policy of the CTB is "one grammatical relation per bracket", which means each constituent falls into one of the three primitive grammatical relations: (1) head-complementation, (2) head-adjunction and (3) coordination. Additionally, the argument is attached at a level that is "closer" to the head than the adjuncts. Due to the linguistic properties of different dependents and the annotation strategies, a grammar-based model can capture more syntactic preference properties of arguments via hard constraints, i.e. grammar rules, and are therefore more suitable to analyze exocentric constructions.

Figure 1 is the error rate of unlabeled dependencies considering different construction. A construction "$\leftarrow X \leftarrow$" is considered as correctly predicted if and only if all dependent words and head word of $X$ are completely correctly found. The error rate in terms of this metric seems rather high because the units we consider are normally much larger than word pairs. From this figure, we can clearly see that the data-driven parser does better for the prediction of nominal constructions (NN/NR/NT/PN[5]), which relate more on optional adjuncts or modifiers; the grammar-based parser performs better for the prediction of verbal constructions (VC/VE/VV), which relate more on obligatory arguments. The evaluation of the nominal and verbal constructions roughly confirms the strength of grammar-based model to predict verbal constructions.

## 4 Bagging parsers

The comparative analysis highlights the fundamental diversity between data-driven and PCFG-based models. In order to exploit the diversity gain, we address the issue of parser combination. We employ a general ensemble learning technique, i.e. Bagging, to enhance a single-view parser and to combine multi-view parsers.

[5]For the definition and illustration of these tags, please refer to the annotation guidelines (http://www.cis.upenn.edu/~chinese/posguide.3rd.ch.pdf).

### 4.1 Applying Bagging to dependency parsing

Bagging is a machine learning ensemble meta-algorithm to improve classification and regression models in terms of stability and classification accuracy (Breiman, 1996). It also reduces variance and helps to avoid overfitting. Given a training set $D$ of size $n$, Bagging generates $m$ new training sets $D_i$ of size $n' \leq n$, by sampling examples from $D$. $m$ models are separately learned on the $m$ new training sets and combined by voting (for classification) or averaging the output (for regression). Henderson and Brill (2000) successfully applied Bagging to enhance a constituent parser. Moreover, Bagging has been applied to combine multiple solutions for Chinese lexical processing (Sun, 2010b; Sun and Uszkoreit, 2012). In this paper, we apply Bagging to dependency parsing. Since training even one single parser takes hours (if not days), experiments on Bagging is time-consuming. To save time, we conduct data-driven parsing experiments based on *simple* configuration. More specifically, the beam size of the transition-based parser is set to 16, and the simple feature set is utilized; dependency relations are not incorporated for the graph-based parser.

**Bootrapping step.** In the training phase, given a training set $D$ of size $n$, our model generates $m$ new training sets $D_i$ of size $\lambda n$ by sampling uniformly without replacement. Each $D_i$ can be used to train a single-view parser or multiple parsers according to different views. Using this strategy, we can get $m$ *weak* parsers or $km$ parsers if multiple views are implemented. In the parsing phase, for each sentence, the $(k)m$ models output $(k)m$ candidate analyses that are combined in a post-inference procedure.

**Aggregating step.** Different from classification problems, simple voting scheme is not suitable for parsing, which is a typical structured prediction problem. To aggregate outputs of $(k)m$ sub-models, a structured inference procedure is needed. Sagae and Lavie (2006) present a framework for combining the output of several different parsers to produce results that are superior to each of the individual parsers. We implement their method to aggregate models. Once we have obtained multiple dependency trees respectively from base parsers, we can build a graph where each word in the sentence is a

node. We then create weighted directed edges between the nodes corresponding to words for which dependencies are obtained from each of the initial structures. The weights are the word-by-word voting results of sub-models. Based on this graph, the sentence can be reparsed by a graph-based algorithm. Taking Chinese as a projective language, we use Eisner's algorithm (Eisner, 1996) to combine multiple dependency parses. Surdeanu and Manning (2010) indicates that reparsing performs essentially as well as other simpler or more complex models.

### 4.2 Parameter tuning

We evaluate our combination model on the same data set used in the last section. The two hyper-parameters ($\lambda$ and $m$) of our Bagging model are tuned on the development (validation) set. On one hand, with the increase of the size of sub-samples, i.e. $\lambda$, the performance of sub-models is improved. However, since the sub-models overlap more, the diversity of base models for ensemble will decrease and the final prediction accuracy may go down. To evaluate the effect of $\lambda$, we separately sample 50%, 60%, 70% and 80% sentences from the original training data 5 times, train 5 sub-models for each parser, and combine them together. The beam size of the transition-based parser is set to 16. Table 2 shows the influence of the choice of $\lambda$'s. For all following experiments, we set $\lambda = 0.7$.

| $\lambda$ | 50% | 60% | **70%** | 80% |
|---|---|---|---|---|
| Tran+Graph[-lab]+Unlex | 83.50 | 85.96 | **86.15** | 85.60 |

Table 2: UAS of Bagging(5) models with different $\lambda$.

The second parameter for Bagging is the number of sub-models to be used for combination. Figure 2 summarizes the Bagging performance when different models are employed and different number (i.e. $m$) of subsamples are used. From this figure, we can learn the influence of the number of sub-models.

### 4.3 Bagging single-view parsers

#### 4.3.1 Results

Table 1 indicates that Bagging can improve individual single-view parsers, especially Berkeley parser. If we take Bagging as a general parser enhancement technique and still consider a Bagging-enhanced parser as a single view, we conclude

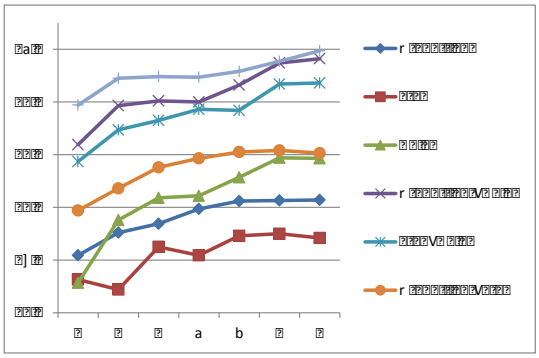


Figure 2: Averaged UAS of different Bagging models with different numbers of sampling data sets.

that Bagging-enhanced PCFG-based method works best among state-of-the-art approaches. For the transition-based parser, though the score over single words goes up, the score over sentences goes down. The main reason is that the reparsing algorithm is a graph-based one, which performs worse with regard to the prediction of a whole sentence. The improvement for the graph-based parser is very modest.

We train a Bagging(8)-enhanced Berkeley parser, which achieves equivalent overall UAS to data-driven parsers, and compare their parsing abilities of second order dependencies. Now we can more clearly see that the Bagging-enhanced PCFG-based model performs better in the prediction of second order dependencies.

#### 4.3.2 Related experiments on sequence models

Bagging has been applied to enhance discriminative sequence models for Chinese word segmentation (Sun, 2010b) and POS tagging (Sun and Uszkoreit, 2012). For word segmentation, experiments on discriminative Markov and semi-Markov tagging models are reported. Their experiments showed that Bagging can consistently enhance a semi-Markov model but not the Markov one. Experiments on POS tagging indicated that Bagging Markov models hurts tagging performance. It seems that the relationships among basic processing units affect Bagging.

PCFGLA parsers are built upon generative models with latent annotations. The use of automatically induced latent variables may also affect Bagging. Generative sequence models with latent anno-

tations can also achieve good performance for Chinese POS tagging. Huang et al. (2009) described and evaluated a bi-gram HMM tagger that utilizes latent annotations. Different from negative results of Bagging discriminative models, our auxiliary experiment shows that Bagging Huang et al.'s tagger can help Chinese POS tagging. In other words, Bagging substantially improves both HMMLA and PCFGLA models, at least for Chinese POS tagging and constituency parsing. It seems that Bagging favors the use of latent variables.

### 4.4 Bagging multi-view parsers

#### 4.4.1 Results

Figure 2 clearly shows that the Bagging model taking both data-driven and PCFG-based models as basic systems outperform the Bagging model taking either model in isolation as basic systems. The combination of a PCFG-based model and a data-driven model (either graph-based or transition-based) is more effective than the combination of two data-driven models, which has received the most attention in dependency parser ensemble. Table 3 is the performance of reparsing on the development data. From this table, we can see by utilizing more parsers, Bagging can enhance reparsing. According to Surdeanu and Manning (2010)'s findings, reparsing performs as well as other combination models. Our auxiliary experiments confirm this finding: Learning-based stacking cannot achieve better performance. Limited to the document length, we do not give descriptions of these experiments.

| Devel. | UAS |
|---|---|
| Reparsing(Tran[b=16,Z08]+Graph[-lab]+Unlex) | 85.82 |
| +Bagging(15) | 86.37 |
| bagging(reparse($g, t, c$)) | 86.09 |
| reparse(bagging($g, t, c$)) | 85.86 |

Table 3: UAS of reparsing and Bagging.

#### 4.4.2 Analysis

In our proposed model, Bagging has a two-fold effect: One is as a system combination technique and the other as a general parser enhancing technique. Two additional experiments are performed to evaluate these two effects. To illustrate the differences between these two experiments, respectively denote graph-based, transition-based and PCFG-based parsers as $g$, $t$ and $c$; denote the reparsing procedure as reparse and the Bagging procedure as bagging. The two experiments are as follows.

- **Bagging a hybrid parser.** In this experiment, for each sub-sample $D_i$, we first train three parsers: $g_i$, $t_i$ and $c_i$. Then we combine these three parsers by reparsing and construct a *hybrid* parser reparse($g_i, t_i, c_i$). Finally, all *hybrid* parsers are collected to build the final parser: bagging(reparse($g, t, c$)).

- **Combining Bagging-enhanced parsers.** In this experiment, for each model, we first train three *Bagging-enhanced* parsers: bagging($g$), bagging($t$) and bagging($c$). Then these three *Bagging-enhanced* parsers are combined by reparsing to build the final parser: reparse(bagging($g, t, c$)).

Evaluation results are presented in Table 3.

## 5 Pseudo-grammar-based models

Although the combination of data-driven and grammar-based models is very effective, it has a serious limitation: It is only applicable when constituency annotations are available to learning a grammar. However, many treebanks, e.g. Chinese Dependency Treebank (LDC2012T05), do not have such linguistically rich structures. Our experiments also suggest that a constituency grammar can significantly increase the diversity of base models for parser ensemble, which plays a major role in boosting prediction accuracy.

In order to reduce the need for phrase-structure annotations, and to increase the diversity of candidate parsers, we study learning pseudo grammars for dependency parsing. The key idea is very simple: By converting a dependency structure to a constituency one, we can reuse the PCFGLA approach to learn *pseudo* grammars for dependency parsing. Figure 3 is an example. The first tree is an original dependency parse, while the second tree is the corresponding CTB annotation. The next two trees are two automatically converted pseudo constituency trees. By applying DS to CS rules, we can acquire pseudo constituency treebanks and then learn pseudo grammars from them.

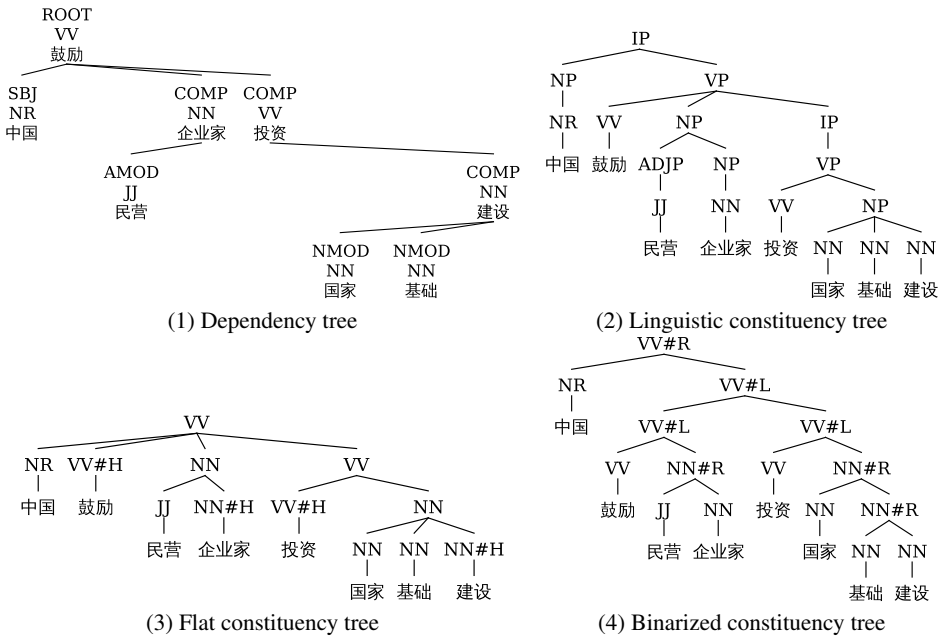


Figure 3: An example: *China encourages private entrepreneurs to invest in national infrastructure.*

The basic idea of our method is to use parsing models in one formalism for parsing in another formalism. In previous work, PCFGs are used to solve parsing problems in many other formalisms, including dependency (Collins et al., 1999), CCG (Fowler and Penn, 2010), LFG (Cahill et al., 2004) and HPSG (Zhang and Krieger, 2011) parsing.

### 5.1 Strategies for DS to CS conversion

The conversion from DS to CS is a non-trivial problem. One main issue in the conversion is the indeterminancy in the choice of a phrasal category given a dependency relation, the level and position of attachment of a dependent in the constituency structure, as dependency relations typically do not encode such information. To convert a DS to a CS, especially for dependency parsing, we should consider (1) how to transform between the topological structures, (2) how to induce a syntactic category, and (3) how to easily recover dependency trees from pseudo constituency trees. From these three aspects, we present the following strategies.

#### 5.1.1 Topological structure

The topological structures represent the boundary information of constituents in a given sentence. Dependency structures do not directly represent such boundary information. Nevertheless, a complete subtree in a projective dependency tree should be considered as a constituent. We can construct a very flat constituent tree, of which nodes are associated with complete subtrees of a dependency parse. The third tree in Figure 3 is an example of such conversion.

**Right-to-left binarization** According to the study in (Sun, 2010a), head words of most phrases in Chinese are located at the first or the last position. That means for binarizing most phrases, we only need sequentially combine the right **or** left parts together with their head phrases. Main exceptions are clauses, of which the head predicate locates inside, since Chinese is an SVO language. To deal with these exceptions, we split each phrase whose head child is inside itself into three parts: left child(ren), head and right child(ren). We first sequentially combine the head and its right child(ren) that are usually objects as intermediate phrases, then sequentially combine the left child(ren) until reach the original parent node. For example, the first rewrite rule in follows should be transferred into the second and third types of rules.

1. $\mathtt{X}_p \rightarrow \mathtt{X}_1, ..., \mathtt{X}_i, ..., \mathtt{X}_m$

2. $\mathtt{X}/_p \rightarrow \mathtt{X}_i, \mathtt{X}_{i+1}$; $\mathtt{X}_p// \rightarrow \mathtt{X}_p/, \mathtt{X}_{i+2}$; ...

3. $\mathtt{X}^*_p \rightarrow \mathtt{X}_{i-1}, \mathtt{X}_p/\cdots/$; $\mathtt{X}^{**}_p \rightarrow \mathtt{X}_{i-2}, \mathtt{X}^*_p$; ...

This right-to-left binarization strategy is consistent with most Chinese treebank annotation schemes. The fourth tree in Figure 3 is an example of binarized pseudo tree.

#### 5.1.2 Phrasal category

Projection principle is introduced by Chomsky to link together the levels of syntactic description. It connects syntactic structures with lexical entries: Lexical structure must be represented categorically at every syntactic level, and representations at each level of syntax are projected from the lexicon in that they observe the subcategorisation properties of lexical items. According to this principle, it is reasonable to use the lexical category (POS) of the head word as the phrasal category of a phrase.

#### 5.1.3 Auxiliary symbol

We can use auxiliary symbols to denote the head phrase position in a CFG rule. In other words, some categories may be splitted into subcategories according to if they are head phrases of their parent nodes or which children are their head phrases. Auxiliary symbols could be either assigned to one of the right hand side or the left hand side. The first choice is to conveniently use a *H* symbol to indicate that current phrase is the head of its parent node. The second choice is to practically use an *L* or *R* symbol to indicate the head of current node is its *left* or *right* child, in a binarized tree. The following table gives an example of different rules with auxiliary symbols.

| With head symbol | With left/right symbol |
|---|---|
| $\mathtt{X}_l \rightarrow \mathtt{X}_l\#\mathtt{H}, \mathtt{X}_r$ | $\mathtt{X}_l\#\mathtt{L} \rightarrow \mathtt{X}_l, \mathtt{X}_r$ |
| $\mathtt{X}_r \rightarrow \mathtt{X}_l, \mathtt{X}_r\#\mathtt{H}$ | $\mathtt{X}_r\#\mathtt{R} \rightarrow \mathtt{X}_l, \mathtt{X}_r$ |

### 5.2 Three conversions

Taking into account the above strategies, we propose three concrete DS to CS conversions:

**Flat conversion with *H* auxiliary symbol (FlatH).** Just as shown as the third tree in Figure 3, we can learn a grammar from very flat constituency trees where the auxiliary symbol *H* is used for extracting dependencies.

**Right-to-left binarizing with *H* auxiliary symbol (BinH).** Different from the flat conversion, we binarize a tree according to the right-to-left principle. Auxiliary symbol *H* is chosen.

**Right-to-left binarizing with *LR* auxiliary symbol (BinLR).** Different from the second type of conversion, we use auxiliary *L/R* symbols to denote head phrases. See the fourth tree in Figure 3 for instance.

Practically, every constituency parse that is produced by parsers trained with binarized trees exactly maps to one dependency tree. However, the parser trained with flat trees may produce very bad constituency results. Sometimes, one parent node may have zero child that is assigned with *H* or more than one children that are are assigned *H*. In the first case, we select the right most child as the head of such parent, while in the second case, we select the right most one from the children that are assigned *H*.

### 5.3 Evaluation

#### 5.3.1 Equivalent parsing accuracy

| Devel. | Base | Bagging(15) |
|---|---|---|
| CTB | 83.49% | 84.92% |
| FlatH | 80.15% | 83.53% |
| BinH | 81.80% | 84.64% |
| BinLR | 82.46% | 84.90% |

Table 4: UAS of pseudo-grammar-based models.

Table 4 summarizes the performance of different pseudo-grammar-based models. Compared to the linguistic grammar learned from CTB, we can see that pseudo grammars are very competitive. Not that, the *FlatH/BinH/BinLR* trees are derived from the CoNLL data, rather than the original CTB. Among different DS to CS conversion strategies, the *BinLR* conversion works best. More interestingly, when we enhance the PCFGLA method by using Bagging, the BinLR model performs as well as the *real*-grammar-based model.

#### 5.3.2 Better contribution to ensemble

The experiments above indicate that we can easily build good grammar-based dependency parser without any constituency annotations. The following experiments on parser combination show that compared to the linguistic grammar, binH and

| Devel. | UAS |
|---|---|
| Tran+Graph+CTB | 86.37% |
| Tran+Graph+FlatH | 86.14% |
| Tran+Graph+BinH | 86.29% |
| Tran+Graph+BinLR | 86.28% |
| **Tran+Graph+flat+BinH+BinLR** | **87.03%** |
| Tran+Graph+CTB+FlatH | 86.96% |
| Tran+Graph+CTB+BinH | 87.10% |
| Tran+Graph+CTB+BinLR | 87.15% |
| Tran+Graph+CTB+BinH+BinLR | 87.38% |
| Tran+Graph+CTB+FlatH+BinH+BinLR | 87.35% |

Table 5: UAS of different Bagging(15) models.

binLR grammars have equivalent contributions to parser ensemble. Table 5 presents the ensemble performance on the development data. By Bagging, the data-driven models together with either *real* grammar-based or pseudo-grammar-based model reach a similar UAS.

#### 5.3.3 Increased parser diversity

Since pseudo grammars are very different from real grammars that are induced from large-scale linguistic annotations. Pseudo-grammar-based parsing models behave very differently with grammar-based models. In other words, they increase the diversity of model candidates for parser ensemble. As a result, pseudo-grammar-based models lead to further improvements for parser combination. Table 5 shows that the combination of data-driven, PCFG-based and binarized pseudo-grammar-based models is significantly better than the combination of data-driven and PCFG-based models.

### 5.4 Comparison to the state-of-the-art

Table 6 summarizes the parsing performance on the test data set, as well as the best published result reported in Li et al. (2012). To fairly compare the performance of our parser and other systems which are built without linguistic constituency trees, we only use pseudo-PCFGs in this experiment. Based on automatic POS tagging, our final model achieves a UAS of 87.23%, which yields a relative error reduction of 24% over the best published result. Table 6 also presents the results evaluated on the CTB5 data that is more widely used for previous research. Li et al. (2011) and Hatori et al. (2011) respectively evaluated their graph-based and transition-based parsers; Zhang and Clark (2011) evaluated

| CoNLL-test | UAS |
|---|---|
| (Li et al., 2012) | 83.23% |
| Graph+Tran+FlatH+BinH+BinLR | 87.23% |
| **CTB5-test** | **UAS** |
| (Li et al., 2011) | 80.79% |
| (Hatori et al., 2011) | 81.33% |
| (Zhang and Clark, 2011) | 81.21% |
| Graph+Tran+FlatH+BinH+BinLR | 84.65% |

Table 6: UAS of different models on the test data.

a hybrid data-driven parser. Our model is significantly better than these systems: It achieves a UAS of 84.65%, which obtains an error reduction of 18% over the best system in the literature.

## 6 Conclusion and Future Work

There have been several attempts to develop high accuracy parsers in both constituency and dependency formalisms for Chinese, and many successful parsing algorithms designed for English have been applied. However, the state-of-the-art still falls far short when compared to English. This paper studies data-driven and PCFG-based models for Chinese dependency parsing. We present a comparative analysis of transition-, graph-, and PCFG-based parsers, which highlights the systematic differences between data-driven and PCFG-based models. Our analysis may benefit parser ensemble, parser co-training, active learning for treebank construction, and so on. In order to exploit the diversity gain, we address the issue of parser combination. To overcome the limitation of the lack of constituency treebanks, we study pseudo-grammar-based models. Experimental results show that combining various data-driven and PCFG-based models significantly advance the state-of-the-art, and by converting parse trees, we can still take advantages of the constituency representation even without constituency annotations.

## Acknowledgement

We would like to thank thank all anonymous reviewers whose valuable comments led to signilicant revisions. The first author would like to thank Prof. Hans Uszkoreit for discussion and feedback of an early version of this work.

The work was supported by NSFC (61170166), Beijing Nova Program (2008B03) and National High-Tech R&D Program (2012AA011101).

## References


Bernd Bohnet. 2010. Top accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 89–97. Coling 2010 Organizing Committee, Beijing, China. URL http://www.aclweb.org/anthology/C10-1011.

Leo Breiman. 1996. Bagging predictors. *Machine Learning*, 24(2):123–140.

Aoife Cahill, Michael Burke, Ruth O'Donovan, Josef Van Genabith, and Andy Way. 2004. Long-distance dependency resolution in automatically acquired wide-coverage pcfg-based lfg approximations. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 319–326. Barcelona, Spain. URL http://www.aclweb.org/anthology/P04-1041.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the first conference on North American chapter of the Association for Computational Linguistics*.

Eugene Charniak and Mark Johnson. 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. Association for Computational Linguistics, Ann Arbor, Michigan.

Wanxiang Che, Valentin Spitkovsky, and Ting Liu. 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. Association for Computational Linguistics, Jeju Island, Korea. URL http://www.aclweb.org/anthology/P12-2003.

Michael Collins. 2003. Head-driven statistical models for natural language parsing. *Computational Linguistics*, 29(4):589–637.

Michael Collins, Jan Hajic, Lance Ramshaw, and Christoph Tillmann. 1999. A statistical parser for czech. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 505–512. Association for Computational Linguistics, College Park, Maryland, USA. URL http://www.aclweb.org/anthology/P99-1065.

Jason M. Eisner. 1996. Three new probabilistic models for dependency parsing: an exploration. In *Proceedings of the 16th conference on Computational linguistics - Volume 1*, COLING '96, pages 340–345. Association for Computational Linguistics, Stroudsburg, PA, USA. URL http://dx.doi.org/10.3115/992628.992688.

Timothy A. D. Fowler and Gerald Penn. 2010. Accurate context-free parsing with combinatory categorial grammar. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 335–344. Association for Computational Linguistics, Uppsala, Sweden. URL http://www.aclweb.org/anthology/P10-1035.

Jun Hatori, Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2011. Incremental joint pos tagging and dependency parsing in chinese. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 1216–1224. Asian Federation of Natural Language Processing, Chiang Mai, Thailand. URL http://www.aclweb.org/anthology/I11-1136.

John Henderson and Eric Brill. 1999. Exploiting diversity in natural language processing: Combining parsers. In *In Proceedings of the Fourth Conference on Empirical Methods in Natural Language Processing*, pages 187–194.

John C. Henderson and Eric Brill. 2000. Bagging and boosting a treebank parser. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, NAACL 2000, pages 34–41. Association for Computational Linguistics, Stroudsburg, PA, USA. URL http://dl.acm.org/citation.cfm?id=974305.974310.

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. Association for Computational Linguistics, Uppsala, Sweden. URL http://www.aclweb.org/anthology/P10-1110.

Zhongqiang Huang, Vladimir Eidelman, and Mary Harper. 2009. Improving a simple bigram hmm part-of-speech tagger by latent annotation and self-training. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 213–216. Association for Computational Linguistics, Boulder, Colorado. URL http://www.aclweb.org/anthology/N/N09/N09-2054.

Zhenghua Li, Ting Liu, and Wanxiang Che. 2012. Exploiting multiple treebanks for parsing with quasi-synchronous grammars. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 675–684. Association for Computational Linguistics, Jeju Island, Korea. URL http://www.aclweb.org/anthology/P12-1071.

Zhenghua Li, Min Zhang, Wanxiang Che, Ting Liu, Wenliang Chen, and Haizhou Li. 2011. Joint models for Chinese pos tagging and dependency parsing. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1180–1191. Association for Computational Linguistics, Edinburgh, Scotland, UK. URL http://www.aclweb.org/anthology/D11-1109.

Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Probabilistic cfg with latent annotations. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 75–82. Association for Computational Linguistics, Stroudsburg, PA, USA. URL http://dx.doi.org/10.3115/1219840.1219850.

Ryan McDonald. 2006. *Discriminative learning and spanning tree algorithms for dependency parsing*. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA, USA. AAI3225503.

Ryan McDonald and Joakim Nivre. 2007. Characterizing the errors of data-driven dependency parsing models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 122–131. Association for Computational Linguistics, Prague, Czech Republic. URL http://www.aclweb.org/anthology/D/D07/D07-1013.

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Comput. Linguist.*, 34:513–553. URL http://dx.doi.org/10.1162/coli.07-056-R1-07-027.

Joakim Nivre and Ryan McDonald. 2008. Integrating graph-based and transition-based dependency parsers. In *Proceedings of ACL-08: HLT*, pages 950–958. Association for Computational Linguistics, Columbus, Ohio. URL http://www.aclweb.org/anthology/P/P08/P08-1108.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 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. Association for Computational Linguistics, Sydney, Australia.

Kenji Sagae and Alon Lavie. 2006. Parser combination by reparsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, NAACL-Short '06, pages 129–132. Association for Computational Linguistics, Stroudsburg, PA, USA. URL http://portal.acm.org/citation.cfm?id=1614049.1614082.

Weiwei Sun. 2010a. Improving Chinese semantic role labeling with rich syntactic features. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 168–172. Association for Computational Linguistics, Uppsala, Sweden. URL http://www.aclweb.org/anthology/P10-2031.

Weiwei Sun. 2010b. Word-based and character-based word segmentation models: Comparison and combination. In *Proceedings of the*

*23rd International Conference on Computational Linguistics (Coling 2010)*, pages 1211–1219. Coling 2010 Organizing Committee, Beijing, China. URL http://www.aclweb.org/anthology/C10-2139.

Weiwei Sun and Zhifang Sui. 2009. Chinese function tag labeling. In *Proceedings of the 23rd Pacific Asia Conference on Language, Information and Computation*. Hong Kong.

Weiwei Sun and Hans Uszkoreit. 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*. Association for Computational Linguistics.

Mihai Surdeanu and Christopher D. Manning. 2010. Ensemble models for dependency parsing: Cheap and good? In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 649–652. Association for Computational Linguistics, Los Angeles, California. URL http://www.aclweb.org/anthology/N10-1091.

Andre Torres Martins, Noah Smith, and Eric Xing. 2009. Concise integer linear programming formulations for dependency parsing. 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 342–350. Association for Computational Linguistics, Suntec, Singapore. URL http://www.aclweb.org/anthology/P/P09/P09-1039.

André Filipe Torres Martins, Dipanjan Das, Noah A. Smith, and Eric P. Xing. 2008. Stacking dependency parsers. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 157–166. Association for Computational Linguistics, Honolulu, Hawaii. URL http://www.aclweb.org/anthology/D08-1017.

Nianwen Xue. 2007. Tapping the implicit information for the PS to DS conversion of the Chinese treebank. In *Proceedings of the Sixth International Workshop on Treebanks and Linguistics Theories*.

Yi Zhang and Hans-Ulrich Krieger. 2011. Large-scale corpus-driven pcfg approximation of an hpsg. In *Proceedings of the 12th International Conference on Parsing Technologies*, pages 198–208. Association for Computational Linguistics, Dublin, Ireland. URL http://www.aclweb.org/anthology/W11-2923.

Yue Zhang and Stephen Clark. 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. Association for Computational Linguistics, Honolulu, Hawaii. URL http://www.aclweb.org/anthology/D08-1059.

Yue Zhang and Stephen Clark. 2009. Transition-based parsing of the Chinese treebank using a global discriminative model. In *Proceedings of the 11th International Conference on Parsing Technologies (IWPT'09)*, pages 162–171. Association for Computational Linguistics, Paris, France. URL http://www.aclweb.org/anthology/W09-3825.

Yue Zhang and Stephen Clark. 2011. Syntactic processing using the generalized perceptron and beam search. *Comput. Linguist.*, 37(1):105–151. URL http://dx.doi.org/10.1162/coli_a_00037.

Yue Zhang and Joakim Nivre. 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. Association for Computational Linguistics, Portland, Oregon, USA. URL http://www.aclweb.org/anthology/P11-2033.

Tao Zhuang and Chengqing Zong. 2010. A minimum error weighting combination strategy for Chinese semantic role labeling. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 1362–1370. Coling 2010 Organizing Committee, Beijing, China. URL http://www.aclweb.org/anthology/C10-1153.