# Ambiguity-aware Ensemble Training for Semi-supervised Dependency Parsing

**Zhenghua Li , Min Zhang*, Wenliang Chen**
Provincial Key Laboratory for Computer Information Processing Technology
Soochow University
{zhli13,minzhang,wlchen}@suda.edu.cn

## Abstract

This paper proposes a simple yet effective framework for semi-supervised dependency parsing at entire tree level, referred to as *ambiguity-aware ensemble training*. Instead of only using 1-best parse trees in previous work, our core idea is to utilize parse forest (*ambiguous labelings*) to combine multiple 1-best parse trees generated from diverse parsers on unlabeled data. With a conditional random field based probabilistic dependency parser, our training objective is to maximize mixed likelihood of labeled data and auto-parsed unlabeled data with ambiguous labelings. This framework offers two promising advantages. 1) ambiguity encoded in parse forests compromises noise in 1-best parse trees. During training, the parser is aware of these ambiguous structures, and has the flexibility to distribute probability mass to its preferred parse trees as long as the likelihood improves. 2) diverse syntactic structures produced by different parsers can be naturally compiled into forest, offering complementary strength to our single-view parser. Experimental results on benchmark data show that our method significantly outperforms the baseline supervised parser and other entire-tree based semi-supervised methods, such as self-training, co-training and tri-training.

## 1 Introduction

Supervised dependency parsing has made great progress during the past decade. However, it is very difficult to further improve performance of supervised parsers. For example, Koo and Collins (2010) and Zhang and McDonald (2012) show that incorporating higher-order features into a graph-based parser only leads to modest increase in parsing accuracy. In contrast, semi-supervised approaches, which can make use of large-scale unlabeled data, have attracted more and more interest. Previously, unlabeled data is explored to derive useful local-context features such as word clusters (Koo et al., 2008), subtree frequencies (Chen et al., 2009; Chen et al., 2013), and word co-occurrence counts (Zhou et al., 2011; Bansal and Klein, 2011). A few effective learning methods are also proposed for dependency parsing to implicitly utilize distributions on unlabeled data (Smith and Eisner, 2007; Wang et al., 2008; Suzuki et al., 2009). All above work leads to significant improvement on parsing accuracy.

Another line of research is to pick up some high-quality auto-parsed training instances from unlabeled data using bootstrapping methods, such as self-training (Yarowsky, 1995), co-training (Blum and Mitchell, 1998), and tri-training (Zhou and Li, 2005). However, these methods gain limited success in dependency parsing. Although working well on constituent parsing (McClosky et al., 2006; Huang and Harper, 2009), self-training is shown unsuccessful for dependency parsing (Spreyer and Kuhn, 2009). The reason may be that dependency parsing models are prone to amplify previous mistakes during training on self-parsed unlabeled data. Sagae and Tsujii (2007) apply a variant of co-training to dependency parsing and report positive results on out-of-domain text. Søgaard and Rishøj (2010) combine tri-training and parser ensemble to boost parsing accuracy. Both work employs two parsers to process the unlabeled data, and only select as extra training data sentences on which the 1-best parse trees of the two parsers are identical. In this way, the auto-parsed unlabeled data becomes more reliable.

*Correspondence author



*Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 457–467,
Baltimore, Maryland, USA, June 23-25 2014. 

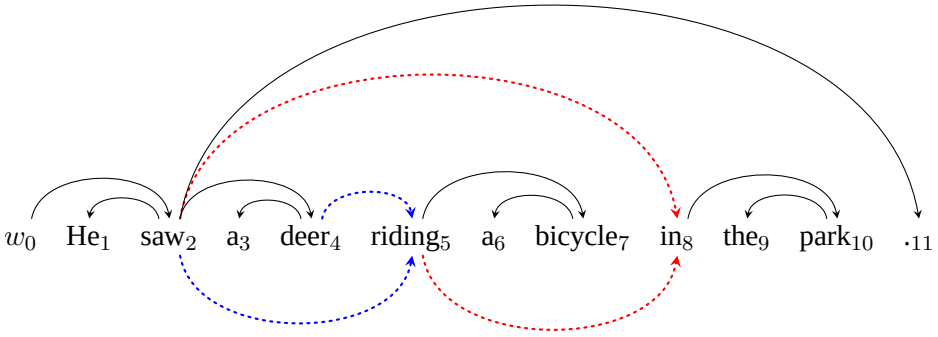


Figure 1: An example sentence with an ambiguous parse forest.

However, one obvious drawback of these methods is that they are unable to exploit unlabeled data with divergent outputs from different parsers. Our experiments show that unlabeled data with identical outputs from different parsers tends to be short (18.25 words per sentence on average), and only has a small proportion of 40% (see Table 6). More importantly, we believe that unlabeled data with divergent outputs is equally (if not more) useful. Intuitively, an unlabeled sentence with divergent outputs should contain some ambiguous syntactic structures (such as preposition phrase attachment) that are very hard to resolve and lead to the disagreement of different parsers. Such sentences can provide more discriminative instances for training which may be unavailable in labeled data.

To solve above issues, this paper proposes a more general and effective framework for semi-supervised dependency parsing, referred to as *ambiguity-aware ensemble training*. Different from traditional self/co/tri-training which only use 1-best parse trees on unlabeled data, our approach adopts ambiguous labelings, represented by parse forest, as gold-standard for unlabeled sentences. Figure 1 shows an example sentence with an ambiguous parse forest. The forest is formed by two parse trees, respectively shown at the upper and lower sides of the sentence. The differences between the two parse trees are highlighted using dashed arcs. The upper tree take *"deer"* as the subject of *"riding"*, whereas the lower one indicates that *"he"* rides the bicycle. The other difference is where the preposition phrase (PP) *"in the park"* should be attached, which is also known as the PP attachment problem, a notorious challenge for parsing. Reserving such uncertainty has three potential advantages. First, noise in unlabeled data is largely alleviated, since parse forest encodes only a few highly possible parse trees with high oracle score. Please note that the parse forest in Figure 1 contains four parse trees after combination of the two different choices. Second, the parser is able to learn useful features from the unambiguous parts of the parse forest. Finally, with sufficient unlabeled data, it is possible that the parser can learn to resolve such uncertainty by biasing to more reasonable parse trees.

To construct parse forest on unlabeled data, we employ three supervised parsers based on different paradigms, including our baseline graph-based dependency parser, a transition-based dependency parser (Zhang and Nivre, 2011), and a generative constituent parser (Petrov and Klein, 2007). The 1-best parse trees of these three parsers are aggregated in different ways. Evaluation on labeled data shows the oracle accuracy of parse forest is much higher than that of 1-best outputs of single parsers (see Table 3). Finally, using a conditional random field (CRF) based probabilistic parser, we train a better model by maximizing mixed likelihood of labeled data and auto-parsed unlabeled data with ambiguous labelings. Experimental results on both English and Chinese datasets demonstrate that the proposed ambiguity-aware ensemble training outperforms other entire-tree based methods such as self/co/tri-training. In summary, we make following contributions.

1. We propose a generalized ambiguity-aware ensemble training framework for semi-supervised dependency parsing, which can

make better use of unlabeled data, especially when parsers from different views produce divergent syntactic structures.

2. We first employ a generative constituent parser for semi-supervised dependency parsing. Experiments show that the constituent parser is very helpful since it produces more divergent structures for our semi-supervised parser than discriminative dependency parsers.

3. We build the first state-of-the-art CRF-based dependency parser. Using the probabilistic parser, we benchmark and conduct systematic comparisons among ours and all previous bootstrapping methods, including self/co/tri-training.

## 2 Supervised Dependency Parsing

Given an input sentence $\mathbf{x} = w_0 w_1 ... w_n$, the goal of dependency parsing is to build a dependency tree as depicted in Figure 1, denoted by $\mathbf{d} = \{(h, m) : 0 \leq h \leq n, 0 < m \leq n\}$, where $(h, m)$ indicates a directed arc from the *head* word $w_h$ to the *modifier* $w_m$, and $w_0$ is an artificial node linking to the root of the sentence.

In parsing community, two mainstream methods tackle the dependency parsing problem from different perspectives but achieve comparable accuracy on a variety of languages. The graph-based method views the problem as finding an optimal tree from a fully-connected directed graph (McDonald et al., 2005; McDonald and Pereira, 2006; Carreras, 2007; Koo and Collins, 2010), while the transition-based method tries to find a highest-scoring transition sequence that leads to a legal dependency tree (Yamada and Matsumoto, 2003; Nivre, 2003; Zhang and Nivre, 2011).

### 2.1 Graph-based Dependency Parser (GParser)

In this work, we adopt the graph-based paradigm because it allows us to naturally derive conditional probability of a dependency tree $\mathbf{d}$ given a sentence $\mathbf{x}$, which is required to compute likelihood of both labeled and unlabeled data. Under the graph-based model, the score of a dependency tree is factored into the scores of small subtrees $\mathbf{p}$.

$$\begin{aligned} Score(\mathbf{x}, \mathbf{d}; \mathbf{w}) &= \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, \mathbf{d}) \\ &= \sum_{\mathbf{p} \subseteq \mathbf{d}} Score(\mathbf{x}, \mathbf{p}; \mathbf{w}) \end{aligned}$$

(a) *single dependency* (b) *adjacent sibling*

Figure 2: Two types of scoring subtrees in our second-order graph-based parsers.

| |
|---|
| Dependency features $\mathbf{f}_{dep}(\mathbf{x}, h, m)$: |
| $w_h$, $w_m$, $t_h$, $t_m$, $t_{h\pm1}$, $t_{m\pm1}$, $t_b$, $dir(h,m)$, $dist(h,m)$ |
| Sibling features $\mathbf{f}_{sib}(\mathbf{x}, h, m, s)$: |
| $w_h$, $w_s$, $w_m$, $t_h$, $t_m$, $t_s$, $t_{h\pm1}$, $t_{m\pm1}$, $t_{s\pm1}$ |
| $dir(h,m)$, $dist(h,m)$ |

Table 1: Brief illustration of the syntactic features. $t_i$ denotes the POS tag of $w_i$. $b$ is an index between $h$ and $m$. $dir(i,j)$ and $dist(i,j)$ denote the direction and distance of the dependency $(i,j)$.

We adopt the second-order graph-based dependency parsing model of McDonald and Pereira (2006) as our core parser, which incorporates features from the two kinds of subtrees in Fig. 2.[1] Then the score of a dependency tree is:

$$\begin{aligned} Score(\mathbf{x}, \mathbf{d}; \mathbf{w}) = &\sum_{\{(h,m)\} \subseteq \mathbf{d}} \mathbf{w}_{dep} \cdot \mathbf{f}_{dep}(\mathbf{x}, h, m) \\ &+ \sum_{\{(h,s),(h,m)\} \subseteq \mathbf{d}} \mathbf{w}_{sib} \cdot \mathbf{f}_{sib}(\mathbf{x}, h, s, m) \end{aligned}$$

where $\mathbf{f}_{dep}(\mathbf{x}, h, m)$ and $\mathbf{f}_{sib}(\mathbf{x}, h, s, m)$ are the feature vectors of the two subtree in Fig. 2; $\mathbf{w}_{dep/sib}$ are feature weight vectors; the dot product gives scores contributed by corresponding subtrees.

For syntactic features, we adopt those of Bohnet (2010) which include two categories corresponding to the two types of scoring subtrees in Fig. 2. We summarize the atomic features used in each feature category in Table 1. These atomic features are concatenated in different combinations to compose rich feature sets. Please refer to Table 4 of Bohnet (2010) for the complete feature list.

### 2.2 CRF-based GParser

Previous work on graph-based dependency parsing mostly adopts linear models and perceptron based training procedures, which lack probabilistic explanations of dependency trees and do not need to compute likelihood of labeled training

[1] Higher-order models of Carreras (2007) and Koo and Collins (2010) can achieve higher accuracy, but has much higher time cost ($O(n^4)$). Our approach is applicable to these higher-order models, which we leave for future work.

data. Instead, we build a log-linear CRF-based dependency parser, which is similar to the CRF-based constituent parser of Finkel et al. (2008). Assuming the feature weights $\mathbf{w}$ are known, the probability of a dependency tree $\mathbf{d}$ given an input sentence $\mathbf{x}$ is defined as:

$$p(\mathbf{d}|\mathbf{x};\mathbf{w}) = \frac{exp\{Score(\mathbf{x},\mathbf{d};\mathbf{w})\}}{Z(\mathbf{x};\mathbf{w})}$$
$$Z(\mathbf{x};\mathbf{w}) = \sum_{\mathbf{d}' \in \mathcal{Y}(\mathbf{x})} exp\{Score(\mathbf{x},\mathbf{d}';\mathbf{w})\} \quad (1)$$

where $Z(\mathbf{x})$ is the normalization factor and $\mathcal{Y}(\mathbf{x})$ is the set of all legal dependency trees for $\mathbf{x}$.

Suppose the labeled training data is $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{d}_i)\}_{i=1}^{N}$. Then the log likelihood of $\mathcal{D}$ is:

$$\mathcal{L}(\mathcal{D};\mathbf{w}) = \sum_{i=1}^{N} \log p(\mathbf{d}_i|\mathbf{x}_i;\mathbf{w})$$

The training objective is to maximize the log likelihood of the training data $\mathcal{L}(\mathcal{D})$. The partial derivative with respect to the feature weights $\mathbf{w}$ is:

$$\frac{\partial \mathcal{L}(\mathcal{D};\mathbf{w})}{\partial \mathbf{w}} = \sum_{i=1}^{N} \left( \mathbf{f}(\mathbf{x}_i,\mathbf{d}_i) - \sum_{\mathbf{d}' \in \mathcal{Y}(\mathbf{x}_i)} p(\mathbf{d}'|\mathbf{x}_i;\mathbf{w}) \mathbf{f}(\mathbf{x}_i,\mathbf{d}') \right) \quad (2)$$

where the first term is the empirical counts and the second term is the model expectations. Since $\mathcal{Y}(\mathbf{x}_i)$ contains exponentially many dependency trees, direct calculation of the second term is prohibitive. Instead, we can use the classic inside-outside algorithm to efficiently compute the model expectations within $O(n^3)$ time complexity, where $n$ is the input sentence length.

## 3 Ambiguity-aware Ensemble Training

In standard entire-tree based semi-supervised methods such as self/co/tri-training, automatically parsed unlabeled sentences are used as additional training data, and noisy 1-best parse trees are considered as gold-standard. To alleviate the noise, the tri-training method only uses unlabeled data on which multiple parsers from different views produce identical parse trees. However, unlabeled data with divergent syntactic structures should be more useful. Intuitively, if several parsers disagree on an unlabeled sentence, it implies that the unlabeled sentence contains some difficult syntactic phenomena which are not sufficiently covered in manually labeled data. Therefore, exploiting such unlabeled data may introduce more discriminative syntactic knowledge, largely compensating labeled training data.

To address above issues, we propose *ambiguity-aware ensemble training*, which can be interpreted as a *generalized tri-training* framework. The key idea is the use of *ambiguous labelings* for the purpose of aggregating multiple 1-best parse trees produced by several diverse parsers. Here, "ambiguous labelings" mean an unlabeled sentence may have multiple parse trees as gold-standard reference, represented by parse forest (see Figure 1). The training procedure aims to maximize mixed likelihood of both manually labeled and auto-parsed unlabeled data with ambiguous labelings. For an unlabeled instance, the model is updated to maximize the probability of its parse forest, instead of a single parse tree in traditional tri-training. In other words, the model is free to distribute probability mass among the trees in the parse forest to its liking, as long as the likelihood improves (Täckström et al., 2013).

### 3.1 Likelihood of the Unlabeled Data

The auto-parsed unlabeled data with ambiguous labelings is denoted as $\mathcal{D}' = \{(\mathbf{u}_i, \mathcal{V}_i)\}_{i=1}^{M}$, where $\mathbf{u}_i$ is an unlabeled sentence, and $\mathcal{V}_i$ is the corresponding parse forest. Then the log likelihood of $\mathcal{D}'$ is:

$$\mathcal{L}(\mathcal{D}';\mathbf{w}) = \sum_{i=1}^{M} \log \left( \sum_{\mathbf{d}' \in \mathcal{V}_i} p(\mathbf{d}'|\mathbf{u}_i;\mathbf{w}) \right)$$

where $p(\mathbf{d}'|\mathbf{u}_i;\mathbf{w})$ is the conditional probability of $\mathbf{d}'$ given $\mathbf{u}_i$, as defined in Eq. (1). For an unlabeled sentence $\mathbf{u}_i$, the probability of its parse forest $\mathcal{V}_i$ is the summation of the probabilities of all the parse trees contained in the forest.

Then we can derive the partial derivative of the log likelihood with respect to $\mathbf{w}$:

$$\frac{\partial \mathcal{L}(\mathcal{D}';\mathbf{w})}{\partial \mathbf{w}} = \sum_{i=1}^{M} \left( \sum_{\mathbf{d}' \in \mathcal{V}_i} \tilde{p}(\mathbf{d}'|\mathbf{u}_i,\mathcal{V}_i;\mathbf{w}) \mathbf{f}(\mathbf{u}_i,\mathbf{d}') - \sum_{\mathbf{d}' \in \mathcal{Y}(\mathbf{u}_i)} p(\mathbf{d}'|\mathbf{u}_i;\mathbf{w}) \mathbf{f}(\mathbf{u}_i,\mathbf{d}') \right) \quad (3)$$

where $\tilde{p}(\mathbf{d}'|\mathbf{u}_i,\mathcal{V}_i;\mathbf{w})$ is the probability of $\mathbf{d}'$ un-

der the space constrained by the parse forest $\mathcal{V}_i$.

$$\tilde{p}(\mathbf{d}'|\mathbf{u}_i, \mathcal{V}_i; \mathbf{w}) = \frac{\exp\{Score(\mathbf{u}_i, \mathbf{d}'; \mathbf{w})\}}{Z(\mathbf{u}_i, \mathcal{V}_i; \mathbf{w})}$$

$$Z(\mathbf{u}_i, \mathcal{V}_i; \mathbf{w}) = \sum_{\mathbf{d}' \in \mathcal{V}_i} \exp\{Score(\mathbf{u}_i, \mathbf{d}'; \mathbf{w})\}$$

The second term in Eq. (3) is the same with the second term in Eq. (2). The first term in Eq. (3) can be efficiently computed by running the inside-outside algorithm in the constrained search space $\mathcal{V}_i$.

### 3.2 Stochastic Gradient Descent (SGD) Training

We apply L2-norm regularized SGD training to iteratively learn feature weights $\mathbf{w}$ for our CRF-based baseline and semi-supervised parsers. We follow the implementation in CRFsuite.[2] At each step, the algorithm approximates a gradient with a small subset of the training examples, and then updates the feature weights. Finkel et al. (2008) show that SGD achieves optimal test performance with far fewer iterations than other optimization routines such as L-BFGS. Moreover, it is very convenient to parallel SGD since computations among examples in the same batch is mutually independent.

Training with the combined labeled and unlabeled data, the objective is to maximize the mixed likelihood:

$$\mathcal{L}(\mathcal{D}; \mathcal{D}') = \mathcal{L}(\mathcal{D}) + \mathcal{L}(\mathcal{D}')$$

Since $\mathcal{D}'$ contains much more instances than $\mathcal{D}$ (1.7M vs. 40K for English, and 4M vs. 16K for Chinese), it is likely that the unlabeled data may overwhelm the labeled data during SGD training. Therefore, we propose a simple corpus-weighting strategy, as shown in Algorithm 1, where $\mathcal{D}^b_{i,k}$ is the subset of training data used in $k^{th}$ update and $b$ is the batch size; $\eta_k$ is the update step, which is adjusted following the simulated annealing procedure (Finkel et al., 2008). The idea is to use a fraction of training data ($\mathcal{D}_i$) at each iteration, and do corpus weighting by randomly sampling labeled and unlabeled instances in a certain proportion ($N_1$ vs. $M_1$).

Once the feature weights $\mathbf{w}$ are learnt, we can parse the test data to find the optimal parse tree.

$$\mathbf{d}^* = \arg\max_{\mathbf{d}' \in \mathcal{Y}(\mathbf{x})} p(\mathbf{d}'|\mathbf{x}; \mathbf{w}) = \arg\max_{\mathbf{d}' \in \mathcal{Y}(\mathbf{x})} Score(\mathbf{x}, \mathbf{d}'; \mathbf{w})$$

This can be done with the Viterbi decoding algorithm described in McDonald and Pereira (2006) in $O(n^3)$ parsing time.

**Algorithm 1** SGD training with mixed labeled and unlabeled data.

**Input:** Labeled data $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{d}_i)\}_{i=1}^N$, and unlabeled data $\mathcal{D}' = \{(\mathbf{u}_i, \mathcal{V}_i)\}_{j=1}^M$; Parameters: $I$, $N_1$, $M_1$, $b$
**Output:** $\mathbf{w}$
**Initialization:** $\mathbf{w}^{(0)} = \mathbf{0}$, $k = 0$;
**for** $i = 1$ **to** $I$ **do** {*iterations*}
  Randomly select $N_1$ instances from $\mathcal{D}$ and $M_1$ instances from $\mathcal{D}'$ to compose a new dataset $\mathcal{D}_i$, and shuffle it.
  Traverse $\mathcal{D}_i$: a small batch $\mathcal{D}^b_{i,k} \subseteq \mathcal{D}_i$ at one step.
    $\mathbf{w}_{k+1} = \mathbf{w}_k + \eta_k \frac{1}{b} \nabla \mathcal{L}(\mathcal{D}^b_{i,k}; \mathbf{w}_k)$
    $k = k + 1$
**end for**

### 3.3 Forest Construction with Diverse Parsers

To construct parse forests for unlabeled data, we employ three diverse parsers, i.e., our baseline GParser, a transition-based parser (ZPar[3]) (Zhang and Nivre, 2011), and a generative constituent parser (Berkeley Parser[4]) (Petrov and Klein, 2007). These three parsers are trained on labeled data and then used to parse each unlabeled sentence. We aggregate the three parsers' outputs on unlabeled data in different ways and evaluate the effectiveness through experiments.

## 4 Experiments and Analysis

To verify the effectiveness of our proposed approach, we conduct experiments on Penn Treebank (PTB) and Penn Chinese Treebank 5.1 (CTB5). For English, we follow the popular practice to split data into training (sections 2-21), development (section 22), and test (section 23). For CTB5, we adopt the data split of (Duan et al., 2007). We convert original bracketed structures into dependency structures using Penn2Malt with its default head-finding rules.

For unlabeled data, we follow Chen et al. (2013) and use the BLLIP WSJ corpus (Charniak et al., 2000) for English and Xinhua portion of Chinese

[2] http://www.chokkan.org/software/crfsuite/

[3] http://people.sutd.edu.sg/~yue_zhang/doc/

[4] https://code.google.com/p/berkeleyparser/

| | Train | Dev | Test | Unlabeled |
|---|---|---|---|---|
| PTB | 39,832 | 1,700 | 2,416 | 1.7M |
| CTB5 | 16,091 | 803 | 1,910 | 4M |

Table 2: Data sets (in sentence number).

Gigaword Version 2.0 (LDC2009T14) (Huang, 2009) for Chinese. We build a CRF-based bigram part-of-speech (POS) tagger with the features described in (Li et al., 2012), and produce POS tags for all train/development/test/unlabeled sets (10-way jackknifing for training sets). The tagging accuracy on test sets is $97.3\%$ on English and $94.0\%$ on Chinese. Table 2 shows the data statistics.

We measure parsing performance using the standard unlabeled attachment score (UAS), excluding punctuation marks. For significance test, we adopt Dan Bikel's randomized parsing evaluation comparator (Noreen, 1989).[5]

### 4.1 Parameter Setting

When training our CRF-based parsers with SGD, we use the batch size $b = 100$ for all experiments. We run SGD for $I = 100$ iterations and choose the model that performs best on development data. For the semi-supervised parsers trained with Algorithm 1, we use $N_1 = 20$K and $M_1 = 50$K for English, and $N_1 = 15$K and $M_1 = 50$K for Chinese, based on a few preliminary experiments. To accelerate the training, we adopt parallelized implementation of SGD and employ 20 threads for each run. For semi-supervised cases, one iteration takes about 2 hours on an IBM server having 2.0 GHz Intel Xeon CPUs and 72G memory.

Default parameter settings are used for training ZPar and Berkeley Parser. We run ZPar for 50 iterations, and choose the model that achieves highest accuracy on the development data. For Berkeley Parser, we use the model after 5 split-merge iterations to avoid over-fitting the training data according to the manual. The phrase-structure outputs of Berkeley Parser are converted into dependency structures using the same head-finding rules.

### 4.2 Methodology Study on Development Data

Using three supervised parsers, we have many options to construct parse forest on unlabeled data. To examine the effect of different ways for forest construction, we conduct extensive methodology study on development data. Table 3 presents the results. We divide the systems into three types: 1) supervised single parsers; 2) CRF-based GParser with conventional self/co/tri-training; 3) CRF-based GParser with our approach. For the latter two cases, we also present the oracle accuracy and averaged head number per word ("Head/Word") of parse forest when applying different ways to construct forests on development datasets.

**The first major row** presents performance of the three supervised parsers. We can see that the three parsers achieve comparable performance on English, but the performance of ZPar is largely inferior on Chinese.

**The second major row** shows the results when we use single 1-best parse trees on unlabeled data. When using the outputs of GParser itself ("Unlabeled ← G"), the experiment reproduces traditional self-training. The results on both English and Chinese re-confirm that *self-training may not work for dependency parsing*, which is consistent with previous studies (Spreyer and Kuhn, 2009). The reason may be that dependency parsers are prone to amplify previous mistakes on unlabeled data during training.

The next two experiments in the second major row reimplement *co-training*, where another parser's 1-best results are projected into unlabeled data to help the core parser. Using unlabeled data with the results of ZPar ("Unlabeled ← Z") significantly outperforms the baseline GParser by 0.30% (93.15-82.85) on English. However, the improvement on Chinese is not significant. Using unlabeled data with the results of Berkeley Parser ("Unlabeled ← B") significantly improves parsing accuracy by 0.55% (93.40-92.85) on English and 1.06% (83.34-82.28) on Chinese. We believe the reason is that being a generative model designed for constituent parsing, Berkeley Parser is more different from discriminative dependency parsers, and therefore can provide more divergent syntactic structures. This kind of syntactic divergence is helpful because it can provide complementary knowledge from a different perspective. Surdeanu and Manning (2010) also show that the diversity of parsers is important for performance improvement when integrating different parsers in the supervised track. Therefore, we can conclude that *co-training helps dependency parsing, especially when using a more divergent parser*.

The last experiment in the second major row is known as *tri-training*, which only uses unla-

[5] http://www.cis.upenn.edu/~dbikel/software.html

| | | English | | | Chinese | | |
|---|---|---|---|---|---|---|---|
| | | UAS | Oracle | Head/Word | UAS | Oracle | Head/Word |
| Supervised | GParser | **92.85** | — | — | 82.28 | — | — |
| | ZPar | 92.50 | | | 81.04 | | |
| | Berkeley | 92.70 | | | **82.46** | | |
| Semi-supervised GParser with Single 1-best Trees | Unlabeled ← G (self-train) | 92.88 | 92.85 | 1.000 | 82.14 | 82.28 | 1.000 |
| | Unlabeled ← Z (co-train) | 93.15 † | 92.50 | | 82.54 | 81.04 | |
| | Unlabeled ← B (co-train) | 93.40 † | 92.70 | | **83.34** † | 82.46 | |
| | Unlabeled ← B=Z (tri-train) | **93.50** † | 97.52 | | 83.10 † | 95.05 | |
| Semi-supervised GParser Ambiguity-aware Ensemble | Unlabeled ← Z+G | 93.18 † | 94.97 | 1.053 | 82.78 | 86.66 | 1.136 |
| | Unlabeled ← B+G | 93.35 † | 96.37 | 1.080 | 83.24 † | 89.72 | 1.188 |
| | Unlabeled ← B+Z | **93.78** †‡ | 96.18 | 1.082 | 83.86 †‡ | 89.54 | 1.199 |
| | Unlabeled ← B+(Z∩G) | 93.77 †‡ | 95.60 | 1.050 | **84.26** †‡ | 87.76 | 1.106 |
| | Unlabeled ← B+Z+G | 93.50 † | 96.95 | 1.112 | 83.30 † | 91.50 | 1.281 |

Table 3: Main results on development data. G is short for GParser, Z for ZPar, and B for Berkeley Parser. † means the corresponding parser significantly outperforms supervised parsers, and ‡ means the result significantly outperforms co/tri-training at confidence level of $p < 0.01$.

beled sentences on which Berkeley Parser and ZPar produce identical outputs ("Unlabeled ← B=Z"). We can see that with the verification of two views, the oracle accuracy is much higher than using single parsers (97.52% vs. 92.85% on English, and 95.06% vs. 82.46% on Chinese). Although using less unlabeled sentences (0.7M for English and 1.2M for Chinese), *tri-training achieves comparable performance to co-training* (slightly better on English and slightly worse on Chinese).

**The third major row** shows the results of the semi-supervised GParser with our proposed approach. We experiment with different combinations of the 1-best parse trees of the three supervised parsers. The first three experiments combine 1-best outputs of two parsers to compose parse forest on unlabeled data. "Unlabeled ← B+(Z∩G)" means that the parse forest is initialized with the Berkeley parse and augmented with the intersection of dependencies of the 1-best outputs of ZPar and GParser. In the last setting, the parse forest contains all three 1-best results.

When the parse forests of the unlabeled data are the union of the outputs of GParser and ZPar, denoted as "Unlabeled ← Z+G", each word has 1.053 candidate heads on English and 1.136 on Chinese, and the oracle accuracy is higher than using 1-best outputs of single parsers (94.97% vs. 92.85% on English, 86.66% vs. 82.46% on Chinese). However, we find that although the parser significantly outperforms the supervised GParser on English, it does not gain significant improvement over co-training with ZPar ("Unlabeled ← Z") on both English and Chinese.

Combining the outputs of Berkeley Parser and GParser ("Unlabeled ← B+G"), we get higher oracle score (96.37% on English and 89.72% on Chinese) and higher syntactic divergence (1.085 candidate heads per word on English, and 1.188 on Chinese) than "Unlabeled ← Z+G", which verifies our earlier discussion that Berkeley Parser produces more different structures than ZPar. However, it leads to slightly worse accuracy than co-training with Berkeley Parser ("Unlabeled ← B"). This indicates that adding the outputs of GParser itself does not help the model.

Combining the outputs of Berkeley Parser and ZPar ("Unlabeled ← B+Z"), we get the best performance on English, which is also significantly better than both co-training ("Unlabeled ← B") and tri-training ("Unlabeled ← B=Z") on both English and Chinese. This demonstrates that *our proposed approach can better exploit unlabeled data than traditional self/co/tri-training*. More analysis and discussions are in Section 4.4.

During experimental trials, we find that "Unlabeled ← B+(Z∩G)" can further boost performance on Chinese. A possible explanation is that by using the intersection of the outputs of GParser and ZPar, the size of the parse forest is better controlled, which is helpful considering that ZPar performs worse on this data than both Berkeley Parser and GParser.

Adding the output of GParser itself ("Unlabeled ← B+Z+G") leads to accuracy drop, although the oracle score is higher (96.95% on English and 91.50% on Chinese) than "Unlabeled ← B+Z". We suspect the reason is that the model is likely to distribute the probability mass to these parse trees produced by itself instead of those by Berkeley Parser or ZPar under this setting.

| | Sup | Semi |
|---|---|---|
| McDonald and Pereira (2006) | 91.5 | — |
| Koo and Collins (2010) [higher-order] | 93.04 | |
| Zhang and McDonald (2012) [higher-order] | 93.06 | |
| Zhang and Nivre (2011) [higher-order] | 92.9 | |
| Koo et al. (2008) [higher-order] | 92.02 | 93.16 |
| Chen et al. (2009) [higher-order] | 92.40 | 93.16 |
| Suzuki et al. (2009) [higher-order,cluster] | 92.70 | 93.79 |
| Zhou et al. (2011) [higher-order] | 91.98 | 92.64 |
| Chen et al. (2013) [higher-order] | 92.76 | 93.77 |
| **This work** | 92.34 | 93.19 |

Table 4: UAS comparison on English test data.

In summary, we can conclude that *our proposed ambiguity-aware ensemble training is significantly better than both the supervised approaches and the semi-supervised approaches that use 1-best parse trees*. Appropriately composing the forest parse, our approach outperforms the best results of co-training or tri-training by 0.28% (93.78-93.50) on English and 0.92% (84.26-83.34) on Chinese.

### 4.3 Comparison with Previous Work

We adopt the best settings on development data for semi-supervised GParser with our proposed approach, and make comparison with previous results on test data. Table 4 shows the results.

The first major row lists several state-of-the-art supervised methods. McDonald and Pereira (2006) propose a second-order graph-based parser, but use a smaller feature set than our work. Koo and Collins (2010) propose a third-order graph-based parser. Zhang and McDonald (2012) explore higher-order features for graph-based dependency parsing, and adopt beam search for fast decoding. Zhang and Nivre (2011) propose a feature-rich transition-based parser. All work in the second major row adopts semi-supervised methods. The results show that our approach achieves comparable accuracy with most previous semi-supervised methods. Both Suzuki et al. (2009) and Chen et al. (2013) adopt the higher-order parsing model of Carreras (2007), and Suzuki et al. (2009) also incorporate word cluster features proposed by Koo et al. (2008) in their system. We expect our approach may achieve higher performance with such enhancements, which we leave for future work. Moreover, our method may be combined with other semi-supervised approaches, since they are orthogonal in methodology and utilize unlabeled data from different perspectives.

Table 5 make comparisons with previous results on Chinese test data. Li et al. (2012) and Bohnet and Nivre (2012) use joint models for POS tagging and dependency parsing, significantly outperforming their pipeline counterparts. Our approach can be combined with their work to utilize unlabeled data to improve both POS tagging and parsing simultaneously. Our work achieves comparable accuracy with Chen et al. (2013), although they adopt the higher-order model of Carreras (2007). Again, our method may be combined with their work to achieve higher performance.

| | | UAS |
|---|---|---|
| Supervised | Li et al. (2012) [joint] | 82.37 |
| | Bohnet and Nivre (2012) [joint] | 81.42 |
| | Chen et al. (2013) [higher-order] | 81.01 |
| | **This work** | 81.14 |
| Semi | Chen et al. (2013) [higher-order] | 83.08 |
| | **This work** | 82.89 |

Table 5: UAS comparison on Chinese test data.

| Unlabeled data | UAS | #Sent | Len | Head/Word | Oracle |
|---|---|---|---|---|---|
| NULL | 92.34 | 0 | — | — | — |
| Consistent (tri-train) | 92.94 | 0.7M | 18.25 | 1.000 | 97.65 |
| Low divergence | 92.94 | 0.5M | 28.19 | 1.062 | 96.53 |
| High divergence | 93.03 | 0.5M | 27.85 | 1.211 | 94.28 |
| ALL | 93.19 | 1.7M | 24.15 | 1.087 | 96.09 |

Table 6: Performance of our semi-supervised GParser with different sets of "Unlabeled ← B+Z" on English test set. "Len" means averaged sentence length.

### 4.4 Analysis

To better understand the effectiveness of our proposed approach, we make detailed analysis using the semi-supervised GParser with "Unlabeled ← B+Z" on English datasets.

**Contribution of unlabeled data with regard to syntactic divergence:** We divide the unlabeled data into three sets according to the divergence of the 1-best outputs of Berkeley Parser and ZPar. The first set contains those sentences that the two parsers produce identical parse trees, denoted by "consistent", which corresponds to the setting for tri-training. Other sentences are split into two sets according to averaged number of heads per word in parse forests, denoted by "low divergence" and "high divergence" respectively. Then we train semi-supervised GParser using the three sets of unlabeled data. Table 6 illustrates the results and statistics. We can see that unlabeled data with identical outputs from Berkeley Parser and ZPar tends to be short sentences (18.25 words per sen-

tence on average). Results show all the three sets of unlabeled data can help the parser. Especially, the unlabeled data with highly divergent structures leads to slightly higher improvement. This demonstrates that *our approach can better exploit unlabeled data on which parsers of different views produce divergent structures*.

**Impact of unlabeled data size:** To understand how our approach performs with regards to the unlabeled data size, we train semi-supervised GParser with different sizes of unlabeled data. Fig. 3 shows the accuracy curve on the test set. We can see that the parser consistently achieves higher accuracy with more unlabeled data, demonstrating the effectiveness of our approach. We expect that our approach has potential to achieve higher accuracy with more additional data.

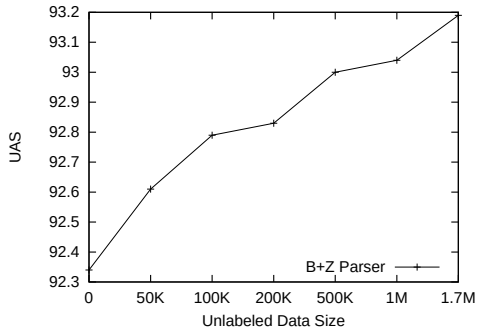


Figure 3: Performance of GParser with different sizes of "Unlabeled ← B+Z" on English test set.

## 5 Related Work

Our work is originally inspired by the work of Täckström et al. (2013). They first apply the idea of ambiguous labelings to multilingual parser transfer in the unsupervised parsing field, which aims to build a dependency parser for a resource-poor target language by making use of source-language treebanks. Different from their work, we explore the idea for semi-supervised dependency parsing where a certain amount of labeled training data is available. Moreover, we for the first time build a state-of-the-art CRF-based dependency parser and conduct in-depth comparisons with previous methods. Similar ideas of learning with ambiguous labelings are previously explored for classification (Jin and Ghahramani, 2002) and sequence labeling problems (Dredze et al., 2009).

Our work is also related with the parser ensemble approaches such as stacked learning and re-parsing in the supervised track. Stacked learning uses one parser's outputs as guide features for another parser, leading to improved performance (Nivre and McDonald, 2008; Torres Martins et al., 2008). Re-parsing merges the outputs of several parsers into a dependency graph, and then apply Viterbi decoding to find a better tree (Sagae and Lavie, 2006; Surdeanu and Manning, 2010). One possible drawback of parser ensemble is that several parsers are required to parse the same sentence during the test phase. Moreover, our approach can benefit from these methods in that we can get parse forests of higher quality on unlabeled data (Zhou, 2009).

## 6 Conclusions

This paper proposes a generalized training framework of semi-supervised dependency parsing based on ambiguous labelings. For each unlabeled sentence, we combine the 1-best parse trees of several diverse parsers to compose ambiguous labelings, represented by a parse forest. The training objective is to maximize the mixed likelihood of both the labeled data and the auto-parsed unlabeled data with ambiguous labelings. Experiments show that our framework can make better use of the unlabeled data, especially those with divergent outputs from different parsers, than traditional tri-training. Detailed analysis demonstrates the effectiveness of our approach. Specifically, we find that our approach is very effective when using divergent parsers such as the generative parser, and it is also helpful to properly balance the size and oracle accuracy of the parse forest of the unlabeled data.

For future work, among other possible extensions, we would like to see how our approach performs when employing more diverse parsers to compose the parse forest of higher quality for the unlabeled data, such as the easy-first non-directional dependency parser (Goldberg and Elhadad, 2010) and other constituent parsers (Collins and Koo, 2005; Charniak and Johnson, 2005; Finkel et al., 2008).

## Acknowledgments

The authors would like to thank the critical and insightful comments from our anonymous reviewers. This work was supported by National Natural Science Foundation of China (Grant No. 61373095, 61333018).

## References


Mohit Bansal and Dan Klein. 2011. Web-scale features for full-scale parsing. In *Proceedings of ACL*, pages 693–702.

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the 11th Annual Conference on Computational Learning Theory*, pages 92–100.

Bernd Bohnet and Joakim Nivre. 2012. A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *Proceedings of EMNLP 2012*, pages 1455–1465.

Bernd Bohnet. 2010. Top accuracy and fast dependency parsing is not a contradiction. In *Proceedings of COLING*, pages 89–97.

Xavier Carreras. 2007. Experiments with a higher-order projective dependency parser. In *Proceedings of EMNLP/CoNLL*, pages 141–150.

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

Eugene Charniak, Don Blaheta, Niyu Ge, Keith Hall, John Hale, and Mark Johnson. 2000. BLLIP 1987-89 WSJ Corpus Release 1, LDC2000T43. *Linguistic Data Consortium*.

Wenliang Chen, Jun'ichi Kazama, Kiyotaka Uchimoto, and Kentaro Torisawa. 2009. Improving dependency parsing with subtrees from auto-parsed data. In *Proceedings of EMNLP*, pages 570–579.

Wenliang Chen, Min Zhang, and Yue Zhang. 2013. Semi-supervised feature transformation for dependency parsing. In *Proceedings of EMNLP*, pages 1303–1313.

Michael J. Collins and Terry Koo. 2005. Discriminative reranking for natural language parsing. *Computational Linguistics*, pages 25–70.

Mark Dredze, Partha Pratim Talukdar, and Koby Crammer. 2009. Sequence learning from data with multiple labels. In *ECML/PKDD Workshop on Learning from Multi-Label Data*.

Xiangyu Duan, Jun Zhao, and Bo Xu. 2007. Probabilistic models for action-based Chinese dependency parsing. In *Proceedings of ECML/ECPPKDD*, pages 559–566.

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

Yoav Goldberg and Michael Elhadad. 2010. An efficient algorithm for easy-first non-directional dependency parsing. In *Proceedings of NAACL*.

Zhongqiang Huang and Mary Harper. 2009. Self-training PCFG grammars with latent annotations across languages. In *Proceedings of EMNLP 2009*, pages 832–841.

Chu-Ren Huang. 2009. Tagged Chinese Gigaword Version 2.0, LDC2009T14. *Linguistic Data Consortium*.

Rong Jin and Zoubin Ghahramani. 2002. Learning with multiple labels. In *Proceedings of NIPS*.

Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *ACL*, pages 1–11.

Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *Proceedings of ACL*, pages 595–603.

Zhenghua Li, Min Zhang, Wanxiang Che, and Ting Liu. 2012. A separately passive-aggressive training algorithm for joint POS tagging and dependency parsing. In *COLING 2012*, pages 1681–1698.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings of the Human Language Technology Conference of the NAACL*, pages 152–159.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of EACL*, pages 81–88.

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

Joakim Nivre and Ryan McDonald. 2008. Integrating graph-based and transition-based dependency parsers. In *Proceedings of ACL*, pages 950–958.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of IWPT*, pages 149–160.

Eric W. Noreen. 1989. *Computer-intensive methods for testing hypotheses: An introduction*. John Wiley & Sons, Inc., New York.

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

Kenji Sagae and Alon Lavie. 2006. Parser combination by reparsing. In *Proceedings of NAACL*, pages 129–132.

Kenji Sagae and Jun'ichi Tsujii. 2007. Dependency parsing and domain adaptation with LR models and parser ensembles. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL*, pages 1044–1050.

David A. Smith and Jason Eisner. 2007. Bootstrapping feature-rich dependency parsers with entropic priors. In *Proceedings of EMNLP*, pages 667–677.

Anders Søgaard and Christian Rishøj. 2010. Semi-supervised dependency parsing using generalized tri-training. In *Proceedings of ACL*, pages 1065–1073.

Kathrin Spreyer and Jonas Kuhn. 2009. Data-driven dependency parsing of new languages using incomplete and noisy training data. In *CoNLL*, pages 12–20.

Mihai Surdeanu and Christopher D. Manning. 2010. Ensemble models for dependency parsing: Cheap and good? In *Proceedings of NAACL*, pages 649–652.

Jun Suzuki, Hideki Isozaki, Xavier Carreras, and Michael Collins. 2009. An empirical study of semi-supervised structured conditional models for dependency parsing. In *Proceedings of EMNLP*, pages 551–560.

Oscar Täckström, Ryan McDonald, and Joakim Nivre. 2013. Target language adaptation of discriminative transfer parsers. In *Proceedings of NAACL*, pages 1061–1071.

André Filipe Torres Martins, Dipanjan Das, Noah A. Smith, and Eric P. Xing. 2008. Stacking dependency parsers. In *Proceedings of EMNLP*, pages 157–166.

Qin Iris Wang, Dale Schuurmans, and Dekang Lin. 2008. Semi-supervised convex training for dependency parsing. In *Proceedings of ACL*, pages 532–540.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of IWPT*, pages 195–206.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of ACL*, pages 189–196.

Hao Zhang and Ryan McDonald. 2012. Generalized higher-order dependency parsing with cube pruning. In *Proceedings of EMNLP-CoNLL*, pages 320–331.

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proceedings of ACL*, pages 188–193.

Zhi-Hua Zhou and Ming Li. 2005. Tri-training: Exploiting unlabeled data using three classifiers. In *IEEE Transactions on Knowledge and Data Engineering*, pages 1529–1541.

Guangyou Zhou, Jun Zhao, Kang Liu, and Li Cai. 2011. Exploiting web-derived selectional preference to improve statistical dependency parsing. In *Proceedings of ACL*, pages 1556–1565.

Zhi-Hua Zhou. 2009. When semi-supervised learning meets ensemble learning. In *MCS*.