Combine Constituent and Dependency Parsing via Reranking

Xiaona Ren[*], **Xiao Chen**[*†], **Chunyu Kit**[*]
[*]Department of Chinese, Translation and Linguistics, City University of Hong Kong
Tat Chee Avenue, Kowloon, HKSAR, China
[†]SOGOU-INC, Tower D, Tsinghua Tongfang High-tech Plaza, Haidian District, Beijing, China
rxiaona2@cityu.edu.hk, chenxiao@sogou-inc.com, ctckit@cityu.edu.hk

Abstract

This paper presents a reranking approach to combining constituent and dependency parsing, aimed at improving parsing performance on both sides. Most previous combination methods rely on complicated joint decoding to integrate graph- and transition-based dependency models. Instead, our approach makes use of a high-performance probabilistic context free grammar (PCFG) model to output k-best candidate constituent trees, and then a dependency parsing model to rerank the trees by their scores from both models, so as to get the most probable parse. Experimental results show that this reranking approach achieves the highest accuracy of constituent and dependency parsing on Chinese treebank (CTB5.1) and a comparable performance to the state of the art on English treebank (WSJ).

1 Introduction

Parsing is to infer the grammatical structure of a natural language sentence with respect to a given grammar. For different types of grammar, parsing techniques may vary greatly. Currently, two types of grammar, namely, context-free grammar and dependency grammar, are the most dominant ones in parsing. The availability of large-scale treebanks, in particular, the Penn Treebank (PTB) [Marcus *et al.*, 1993] and Prague Dependency Treebank [Hajic *et al.*, 2001], allows a wide variety of machine learning methods to facilitate both types of parsing.

These two grammars are so closely related to each other that a constituent tree can be converted into a dependency tree using a set of heuristic head rules [Yamada and Matsumoto, 2003]. The word-to-word selectional affinities in a dependency tree can be helpful to syntactic disambiguation for its constituent correspondence. There are successful lexicalized PCFG-based models [Collins, 1997; Charniak, 2000; Bikel, 2004] that include the lexical head of each constituent for the derivation of a constituent tree, at the cost of increasing the complexity of parsing. To reduce such complexity, Klein and Manning [2002] propose a factorization method to split the joint parsing into two generative sub-models. The constituent and dependency structure of a sentence are evaluated independently in order to select the best combination of them.

In this paper, we present an alternative approach to combining constituent and dependency parsing via a factored model for performance enhancement on both sides. Different from the previous works, our approach has the following advantages.

- Both constituent and dependency parsing are formulated under the discriminative framework, which is more capable of incorporating a large variety of features than a generative model. The two types of parsing thus have a better chance to benefit from the complementarity of feature selection.
- Instead of relying on complex joint inference, we opt to rerank the k-best outputs from a constituent parser with a dependency model. This design is highly modularized so that any constituent model (e.g., the Berkeley parser) can be used as long as it outputs k-best constituent trees. The dependency model also has its sub-models that can be switched on or off conveniently. Also, we do not need to retrain the constituent and dependency (sub-) models.
- This approach is computationally efficient, in that it gets around the complexity of joint inference. The efficiency of reranking allows our dependency model to use a rich set of high-order features.

Our experiments show that this approach significantly improves the performance of both sub-models involved. For constituent parsing, our factored parsing model achieves the F1 scores of 91.97% and 85.79% on the WSJ and CTB5.1 datasets, respectively. For dependency parsing, it also obtains the unlabeled attachment score (UAS) of 93.71% and 87.38% on these two datasets. All these results are comparative or even superior to those by the state-of-the-art parsers reported so far.

The remainder of this paper is organized as follows. Section 2 describes the factored parsing model, including its constituent and dependency sub-model. Section 3 presents our reranking method. Experimental results are reported in Section 4, and related works discussed in Section 5. Section 6 concludes our work and looks into some future directions of research.

2 Parsing Model

Figure 1 illustrates a constituent parse tree and its corresponding dependency tree. Combining these two trees will give a lexicalized parse tree as used in a lexicalized constituent parsing model. Let T be a lexicalized parse tree for an input sentence $s = w_1 w_2 ... w_n$, which can be projected to a constituent tree t and a dependency tree d. Our factored parsing model is formulated under the product-of-experts framework [Hinton, 2002] as

$$P(T|s) = \frac{P(t|s)P(d|s)}{Z}, \tag{1}$$

where $P(t|s)$ and $P(d|s)$ are the constituent and the dependency parsing model, respectively, and Z is a normalization factor, a constant for s that does not need to be calculated in our model. Each $T = \langle t, d \rangle$ is evaluated with the two sub-models, and the best T is two latent structures that receive the highest joint score by both sub-models.

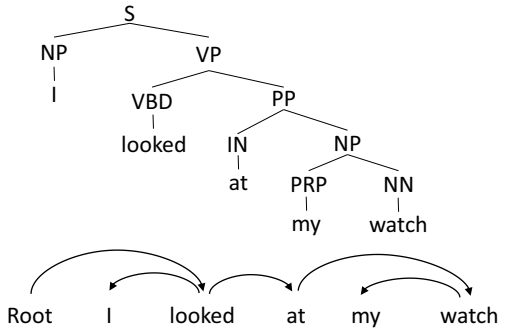

Figure 1: Correspondences between a constituent and a dependency tree.

2.1 Constituent Parsing

The constituent parser presented in [Chen and Kit, 2012] and [Chen, 2012] is used in this work. Its training follows the idea of forest re-scoring [Huang, 2008]. However, we enforce a factorization assumption for our constituent model to ensure its tractability. It is that all features are restricted to depend on the structures within a local context for each instance of grammar rule in a parse tree, and the local context consists of only immediate neighbors of the instance.

Following Collins [2002], our model $g(t, s)$ is formulated in the linear form as

$$g(t, s) = \theta \cdot \Psi(t, s), \tag{2}$$

where $\Psi(t, s)$ is a feature vector and θ the vector of associated weights. The factorization assumption induces the decomposition of a feature vector as

$$\Psi(t, s) = \sum_{u \in U} \Phi(Q(u), s), \tag{3}$$

where U is the collection of all instances of grammar rule in t, $Q(u)$ a *part* of t centered around u, and $\Phi(Q(u), s)$ the feature vector for $Q(u)$. Following Collins [2002] and Huang [2008], we opt for the averaged perceptron algorithm to estimate θ from a training treebank.

In order to combine this model with a dependency model, the score of the above model needs to be normalized, i.e.,

$$P(t|s) = \frac{\exp g(t, s)}{\sum_{t' \in \mathbb{T}(s)} \exp g(t', s)}, \tag{4}$$

where $\mathbb{T}(s)$ is the collection of all possible constituent trees for s.

Features. Similar to the previous forest re-scoring methods, our approach also adopts a PCFG-based model to generate a score for each instance of grammar rule to describe its likelihood of covering a particular span over an input sentence, and uses it as a basic feature for our constituent model. The unlexicalized PCFG-LA is used to calculate this feature:

$$\begin{aligned} \phi_0(\mathcal{Q}(u), s) = \frac{1}{I(\mathcal{S}, 0, n)} \sum_x \sum_y \sum_z O(\mathcal{A}_x, i, j) \\ \cdot \mathcal{P}(\mathcal{A}_x \rightarrow \mathcal{B}_y \, \mathcal{C}_z) \cdot I(\mathcal{B}_y, i, k) I(\mathcal{C}_z, k, j) \end{aligned} \tag{5}$$

where u is an instance of grammar rule $A \rightarrow B\ C$ covering word span $[i, j)$ and splitting at k, x, y and z are latent variables in the PCFG-LA, and $I(\cdot)$ and $O(\cdot)$ are the inside and outside probabilities, respectively.

All other features $\phi_k(Q(u), s)$ are binary functions indicating whether a configuration is contained in $Q(u)$ and s. Two types of feature, namely, lexical and structural, are used in the constituent model. Lexical features bind words with the CFG rules that cover them, which can be viewed as a way of lexicalization of the base PCFG-LA. Structural features capture the dependencies between grammar rule instances. The feature templates[1] used in our constituent model are listed in Table 1. Note that the lexical features in this model do not involve any lexical head or bilexical dependency. Such information is utilized in the dependency parsing model.

Lexical		Structural
Inner pair	N-gram on the inner edge	Parent
Outer pair	N-gram on the outer edge	Children
Split pair	Bigram on the edges	Sibling
Rule bigram		

Table 1: Feature templates used in the constituent model.

2.2 Dependency Parsing

In our dependency parsing model, the score of a dependency tree is determined by two factors, namely, the individual edges and the relations between edges in the tree. Accordingly, the model is formulated as

$$g(d, s) = g_e(d, s) + g_r(d, s) \tag{6}$$

to evaluate the event that d is the dependency tree of s, through its two sub-models: the edge model $g_e(d, s)$ to measure the goodness of all dependency edges in d and the edge-pair model $g_r(d, s)$ to measure the likelihood of two edges to form a relation of certain type.

[1] Its source code is released to the public at http://code.google.com/p/gazaparser/.

Like the graph-based method [McDonald *et al.*, 2005a; 2005b], our edge model defines the score of a dependency tree as the sum of scores of all its edges, and the edge-pair model describes the structure of a dependency tree in terms of edge-pair relations. Possible relations between two edges include parent-child, sibling, adjacent, etc. The edge-pair model calculates a score for d by summing up the scores of all types of relations for each edge-pair in d. That is,

$$\begin{aligned} g_e(d,s) &= \sum_{e \in d} h(e), \\ g_r(d,s) &= \sum_{r \in \mathbb{R}} \sum_{\substack{e_i, e_j \in d \\ i \neq j}} h_r(e_i, e_j), \end{aligned} \tag{7}$$

where $h(e)$ is the score of an edge e, $\mathbb{R}$ the set of relations defined by the model, and $h_r(e_i, e_j)$ the score for the edge-pair (e_i, e_j) in relation r.

The scores $h(e)$ and $h_r(e_i, e_j)$ can be defined in a linear form as in our constituent model and many other dependency parsing models [McDonald *et al.*, 2005a; 2005b; Koo and Collins, 2010]. They can also be calculated using a general-purpose machine learning model, as in the previous work of Jiang and Liu [2010]. In this work, we use the maximum entropy (ME) classifier for this purpose. The score of an edge is computed as

$$h(e) = \frac{\exp\left(\omega \cdot \Phi(e, +)\right)}{\sum_{c \in \{+,-\}} \exp\left(\omega \cdot \Phi(e, c)\right)}, \tag{8}$$

where $\Phi(e, c)$ is a feature vector for e being classified as c, and ω its corresponding weight vector. The value of $c \in \{+, -\}$ indicates whether the word pair in e forms a dependency edge.

For the edge-pair model, we adopt two most profitable relations between two edges: parent-child and adjacent. The score of an edge-pair (e_i, e_j) in a relation is calculated in a similar way as the above, i.e.,

$$h_p(e_i, e_j) = \frac{\exp\left(\omega_p \cdot \Phi_p(e_i, e_j, +)\right)}{\sum_{c \in \{+,-\}} \exp\left(\omega_p \cdot \Phi_p(e_i, e_j, c)\right)}. \tag{9}$$

For the sake of combination, this dependency model also needs normalization:

$$P(d|s) = \frac{\exp g(d, s)}{\sum_{d' \in \mathbb{D}(s)} \exp g(d', s)}, \tag{10}$$

where $\mathbb{D}(s)$ is the set of all possible dependency trees for s.

Three ME classifiers, one for edges, and the other two for parent-child and adjacent edge pairs, are trained on three sets of classification instances extracted from a training treebank. A dependency edge e is taken, together with its classification results, as a training instance for the edge classifier. So is an edge-pair (e_i, e_j) for the edge-pair classifier. For a dependency tree with n words, there are n positive and $n(n-2)$ negative training instances for the edge classifier, e.g., $(e_i, +)$ for $0 \leq i < n$ and $(e_j, -)$, and there are $n-1$ positive and $n(n-1)^2$ negative training instances for the edge-pair classifier, e.g., $(e_i, e_j, +)$ for $0 \leq i \neq j < n$ and $(e_i, e_j, -)$.

The number of negative training instances m is usually set proportional to the positive ones: $m = \gamma n$, with γ to be tuned using a development set. Our experiments to find the optional γ show that the best performance of our dependency parser is achieved by extracting all positive and negative instances for training the classifiers.

Features. Our dependency model adopts a similar set of features as in McDonald et al. [2005a] and Carreras [2007] that are shown to be effective. Let e be a dependency edge from word w_i to w_j. Its features include the words and POS tags surrounding the child w_j and/or its parent w_i, and the distance of the two words $d_{i,j} = i - j$. The feature templates for the edge model are combinations of the basic elements listed in Table 2, where p_x is the part-of-speech of w_x.

Some features are conjoined with the distance feature between the two words in question. This significantly improves the performance of our edge model. Moreover, we opt for a similar set of feature templates for the edge-pair model. Each of its feature instances contains two edges, i.e., grandparent and parent, parent and child. The edge-pair feature are extracted from a dependency tree according to the basic elements in Table 2, where d_{e_i,e_j} represents the relationship of two edges, and 0 and 1 represent the adjacent and parent-child relationship of e_i and e_j, respectively. The features in the edge-pair model are two times as many as those in the edge model.

Basic elements in feature templates	
Edge model	Edge-pair model
w_i, w_j, p_i, p_j, $d_{i,j}$	e_i: w_i, w_m, p_i, p_m, $d_{i,m}$
$w_{i\pm1}$, $w_{i\pm2}$, $p_{i\pm1}$, $p_{i\pm2}$	e_j: w_j, w_n, p_j, p_n, $d_{i,n}$
$w_{j\pm1}$, $w_{j\pm2}$, $p_{j\pm1}$, $p_{j\pm2}$	d_{e_i,e_j}: 0, 1

Table 2: Basic elements in feature templates for the edge and the edge-pair model.

3 Reranking

With the factored parsing model, finding the best lexicalized parse tree $\widehat{T}$ for an input sentence involves a joint inference with the two sub-models, i.e.,

$$\widehat{T} = \text{argmax}_T P(T|s) = \text{argmax}_{\langle t,d \rangle} P(t|s) P(d|s) \tag{11}$$

Since both sub-models are very complex, the joint inference is not computationally efficient. Therefore, we opt for a shortcut that uses the dependency model to rerank the k-best outputs from the constituent model. The reranking involves the following steps.

- Generate a list of k-best constituent trees for an input sentence using the constituent model, with a probability $P(t|s)$ assigned to each tree.
- Convert the k-best constituent trees into dependency trees.
- Calculate a probability $P(d|s)$ for each dependency tree using the dependency model.
- Sort the constituent-dependency tree pairs in terms of the product of two probabilities and take the one with the highest product as the final output.

4 Experiments

This section reports on our experiments to examine the effectiveness of the reranking approach, in comparison with the reported performance of other state-of-the-art parsing approaches. Two treebanks, namely, the Wall Street Journal (WSJ) portion of the Penn Treebank and the Chinese Treebank (CTB) 5.1, are used in the experiments. They have been widely used as evaluation data in the field.

For dependency parsing, all constituent trees in the two treebanks are converted to dependency trees using the `Penn2Malt`[2] conversion program. Each treebank is split into three partitions for training, development and testing, respectively, following its conventional split in most previous works in the field, as in Table 4.

Dataset	WSJ		CTB5.1	
	Sections	Sentences	Sections	Sentences
Train	2-21	39,832	others	18,104
Dev	22	1,700	271-300	350
Test	23	2,416	301-325	348

Table 4: The split of WSJ and CTB 5.1 for experiments

The constituent parsing is evaluated using PARSEVAL, with performance reported in F1 score. The accuracy of dependency parsing is measured by unlabeled attachment score (UAS), i.e., the percentage of tokens with correct heads. All models are evaluated on the same training and test sets.

4.1 Effect of Reranking

The performance of the reranking is highly dependent on the size of the k-best list. A larger k certainly has a higher probability of including the best parsing tree. Our experiments show that the parsing performance increases along with k in general. However, the performance gain starts to decrease when k is over 50, because there are more and more common parts in the constituent trees when k gets larger. The performance reaches its peak around $k = 100$, which is therefore opted as the setting for our experiments.

The performance of this reranking approach is presented in Table 5 and 6, in comparison with other constituent and dependency parsers well known for their high performance on the WSJ and CTB 5.1 datasets. The performance scores marked with * are obtained from our experiments for comparison. On the WSJ test set, the reranking approach achieves the F1 score 91.97% and the UAS 93.71% for constituent and dependency parsing, respectively. This performance is highly comparable to those of the best systems on these two types of parsing, i.e., the combination model of Fossum and Knight [2009] and the semi-supervised structured conditional model of Suzuki et al. [2009]. Our approach also renews the performance records on the CTB 5.1 test set to the F1 score 85.79% and UAS 87.38%. All these confirm its effectiveness.

[2]http://w3.msi.vxu.se/~nivre/research/Penn2Malt.html.

[3]The version released in September, 2009.

[4]The option "- Chinese" is not used for parsing Chinese, for it does not give the best parsing performance.

Dataset	System	F1 (%)
WSJ	Single	
	Berkeley parser[3]	89.87
	Charniak [2000]	89.70
	Henderson [2004]	90.10
	Bod [2003]	90.70
	Carreras et al. [2008]	91.10
	Re-scoring	
	Collins [2000]	89.70
	Charniak and Johnson [2005]	91.40
	Huang [2008]	91.69
	Combination	
	Fossum and Knight [2009]	*92.40*
	Zhang et al. [2009]	92.30
	Petrov [2010]	91.85
	Self-training	
	Huang et al. [2010], Single	91.59
	Our constituent parser	**91.86**
	Our reranking result	**91.97**
CTB5.1	Single	
	Charniak [2000]	80.85*
	Berkeley parser[4]	83.22*
	Burkett and Klein [2008]	84.24
	Combination	
	Zhang et al. [2009]	85.45
	Our constituent parser	**85.56**
	Our reranking result	**85.79**

Table 5: Performance of constituent parsing and reranking in comparison with other approaches.

These results show that both types of parsing can be further enhanced by reranking. It is simple in nature and significant in improving parsing performance. Its success seems attributable to the followings.

- The features of long-distance dependency relation are useful to facilitate syntactic disambiguation of constituents, and hence can improve the constituent parsing performance.
- The constituent parser first outputs the k-best candidate constituent trees, filtering out a large number of candidate trees that are difficult for a dependency parser to analyze while keeping the most likely ones for the dependency parser to pick the best. This strategy not only helps improve dependency parsing performance but also gets around the computational complexity of joint decoding.

4.2 Effect of POS Tagging

Apart from the size of k-best candidate trees, another factor of critical impact on parsing performance is the accuracy of POS tagging, which affects the rescoring accuracy of constituent parsing. However, the POS tagging in our datasets is not completely correct, hindering our evaluation from fully and objectively revealing the effectiveness of this reranking. To access this, another experiment approach is conducted on the gold standard POS tags, denoted as g-reranking. A similar

Dataset	System	UAS (%)
WSJ	Dependency parser	
	McDonald et al. [2005a]	90.90
	McDonald and Pereira [2006]	91.50
	Koo et al. [2008], Standard	92.02
	Koo et al. [2008]	93.16
	Zhang and Clark [2008]	92.10
	Carreras et al. [2008]	93.50
	Suzuki et al. [2009]	*93.79*
	Koo and Collins [2010]	93.04
	Constituent parser	
	Petrov and Klein [2007]	92.40
	Charniak and Johnson [2005],	93.20
	Our constituent parser	**93.16**
	Our reranking result	**93.71**
CTB5.1	Dependency parser	
	MSTParser, 1st-order	84.37
	MSTParser, 2nd-order	86.90
	Constituent parser	
	Petrov and Klein [2007]	85.00
	Our constituent parser	**87.13**
	Our reranking result	**87.38**

Table 6: Performance of dependency parsing and reranking in comparison with other approaches.

experiment is also carried out with Berkeley parser to examine the effect of g-reranking on another constituent parser.

The results of these experiments to compare the effect of the two types of POS tags are presented in Table 7. Using gold standard POS tags, the reranking achieves its highest F1 92.06% and 86.50% for constituent parsing and its highest UAS 94.09% and 89.29% for dependency parsing on the WSJ and CTB5.1 test sets, respectively. Interestingly, a greater improvement of performance is achieved by the reranking with Berkeley parser than with ours, suggesting that the g-reranking can be similarly significant to others. The greatest improvement is achieved on Chinese dependency parsing, by 2.28% and 2.16%, with Berkeley and our constituent parser, respectively. This result strongly indicates that the accuracy of POS tagging does significantly affect the performance of high-performance parsers. Thus, how to further improve the accuracy of POS tagging pops up again to be a key issue. Since surrounding words provide the most important information about the POS of a word in question, the effect of different features from these words certainly deserves more research effort to explore in order to further enhance POS tagging.

5 Related Works

Our constituent parsing shares a number of key features with the discriminative re-scoring methods [Collins, 2000; Charniak and Johnson, 2005; Huang, 2008]. The factorization assumption ensures the scalability and tractability of our parsing model. This also points to a new direction of parsing methodology in terms of the feature design for re-scoring model. Towards this, we employ more intuitive and rational features than those in the previous related works.

Our dependency parsing model is inspired by the word-pair classification model [Jiang and Liu, 2010] and a similar pairwise classification schemata in Japanese dependency parsing [Uchimoto *et al.*, 1999; Kudo and Matsumoto, 2000]. However, our model includes not only the first-order model used in these previous works, but also a second-order model using the same classification methodology. More importantly, our work focuses on applying the dependency parsing model to rerank constituent parsing, providing dependency relations to complement the localized features used in the constituent model.

Similar to other parser combinations approach, ours also involves two different types of model. The best system [Hall, 2007] in the CoNLL2007 shared task combines six transition-based parsers. Nivre and McDonald [2008] succeed in improving dependency parsing by integrating graph- and transition-based models in a way to have two complementary models to learn from each other. Farkas et al. [2011] extract utility features from automatic dependency parses of a sentence for a discriminative phrase-structure parser on German, and Green and Zabokrtsky [2012] create a new dependency structure through ensemble learning using a hybrid of the outputs of various parsers. Martins et al. [2008] present an integration method based on stacking. Zhang et al. [2009] propose a linear model-based general framework to combine k-best outputs from multiple parsers. Compared against these previous methods, our approach is conceptually simpler and, allows in particular, a more straightforward way of modularization in model design.

6 Conclusion and Future Work

In this paper, we have presented a reranking approach to combine constituent and dependency parsing, which reranks the k-best outputs from a constituent parser using a dependency parsing model. The constituent parsing model follows the basic idea of the forest re-scoring approach and uses two types of localized feature to remedy the inherent defects of a PCFG-based model, achieving a competitive performance at the state-of-the-art level. The dependency parsing model combines three independent classifiers, one on individual edges and two on relations between edges, integrating a wide array of lexical and dependency relation features. Our experiments confirm the validity and effectiveness of this reranking approach, in improving both constituent and dependency parsing on both English and Chinese.

The evaluation results from our experiments suggest that constituent and dependency parsing can complement each other through a straightforward strategy of combination. The latter provides long distance dependencies between lexical items to facilitate syntactic disambiguation of constituents. In return, the former eliminates a lot of spurious ambiguities in dependency parsing by shrinking the whole parse forest to a small candidate list.

Our future work includes examining the effectiveness and efficiency of the reranking in contrast to other combination methods, such as incorporating the dependency parsing model as a feature in the forest re-scoring process of the constituent model. We are also interested in exploring the opti-

Parser	WSJ							
	Constituent Parsing F1 (%)				Dependency Parsing UAS (%)			
	Single	Reranking	G-reranking	Increment	Single	Reranking	G-reranking	Increment
Berkeley	89.87	90.14	90.54	+0.40 (**0.67%**)	92.02	92.72	93.07	+0.35 (**1.05%**)
Ours	91.86	91.97	92.06	+0.09 (**0.20%**)	93.16	93.71	94.09	+0.38 (**0.93%**)
	CTB 5.1							
Berkeley	83.22	83.57	84.17	+0.60 (**0.95%**)	84.23	84.61	86.51	+1.90 (**2.28%**)
Ours	85.56	85.79	86.50	+0.71 (**0.94%**)	87.13	87.38	89.29	+1.91 (**2.16%**)

Table 7: Reranking performance with two types of POS tag.

mal size of k-best list for the best performance achievable by the reranking.

Acknowledgments

The research described in this paper was partially supported by City University of Hong Kong through the SRG grant 7002796 and by the Research Grants Council (UGC) of Hong Kong SAR, China, through the GRF grant CityU 144410. We would like to thank three anonymous reviewers for their helpful comments and thank Billy Wong for his help.

References

[Bikel, 2004] Daniel Bikel. Intricacies of Collins' parsing model. *Computational Linguistics*, 30(4):480–511, 2004.

[Bod, 2003] Rens Bod. An efficient implementation of a new DOP model. In *EACL 2003*, pages 19–26, 2003.

[Burkett and Klein, 2008] David Burkett and Dan Klein. Two languages are better than one (for syntactic parsing). In *EMNLP 2008*, pages 877–886, 2008.

[Carreras *et al.*, 2008] Xavier Carreras, Michael Collins, and Terry Koo. TAG, dynamic programming, and the perceptron for efficient, feature-rich parsing. In *CoNLL 2008*, pages 9–16, 2008.

[Carreras, 2007] Xavier Carreras. Experiments with a higher-order projective dependency parser. In *EMNLP-CoNLL 2007*, pages 957–961, 2007.

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

[Charniak, 2000] Eugene Charniak. A maximum-entropy-inspired parser. In *NAACL 2000*, pages 132–139, 2000.

[Chen and Kit, 2012] Xiao Chen and Chunyu Kit. Higher-order constituent parsing and parser combination. In *ACL 2012*, pages 1–5, 2012.

[Chen, 2012] Xiao Chen. *Discriminative Constituent Parsing with Localized Features*. PhD thesis, City University of Hong Kong, 2012.

[Collins, 1997] Michael Collins. Three generative, lexicalised models for statistical parsing. In *ACL 1997*, pages 16–23, 1997.

[Collins, 2000] Michael Collins. Discriminative reranking for natural language parsing. In *ICML 2000*, pages 175–182, 2000.

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

[Farkas *et al.*, 2011] Richard Farkas, Bernd Bohnet, and Helmut Schmid. Features for phrase-structure reranking from dependency parses. In *IWPT 2011*, pages 209–214, 2011.

[Fossum and Knight, 2009] Victoria Fossum and Kevin Knight. Combining constituent parsers. In *NAACL-HLT 2009*, pages 253–256, 2009.

[Green and Zabokrtsky, 2012] Nathan David Green and Zdenek Zabokrtsky. Hybrid combination of constituency and dependency trees into an ensemble dependency parser. In *EACL 2012*, pages 19–26, 2012.

[Hajic *et al.*, 2001] J. Hajic, B. V. Hladka, J. Panevova, Eva Hajicova, Petr Sgall, and Petr Pajas. Prague Dependency Treebank 1.0. *LDC 2001T10*, 2001.

[Hall, 2007] Keith Hall. K-best spanning tree parsing. In *ACL 2007*, pages 392–399, 2007.

[Henderson, 2004] James Henderson. Discriminative training of a neural network statistical parser. In *ACL 2004*, pages 95–102, 2004.

[Hinton, 2002] Geoffrey E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002.

[Huang *et al.*, 2010] Zhongqiang Huang, Mary Harper, and Slav Petrov. Self-training with products of latent variable grammars. In *EMNLP 2010*, pages 12–22, 2010.

[Huang, 2008] Liang Huang. Forest reranking: Discriminative parsing with non-local features. In *ACL-HLT 2008*, pages 586–594, 2008.

[Jiang and Liu, 2010] Wenbin Jiang and Qun Liu. Dependency parsing and projection based on word-pair classification. In *ACL 2010*, pages 12–20, 2010.

[Klein and Manning, 2002] Dan Klein and Christopher D. Manning. Fast exact inference with a factored model for natural language parsing. In *NIPS 2002*, pages 3–10, 2002.

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

[Koo *et al.*, 2008] Terry Koo, Xavier Carreras, and Michael Collins. Simple semi-supervised dependency parsing. In *ACL-HLT 2008*, pages 595–603, June 2008.

[Kudo and Matsumoto, 2000] Taku Kudo and Yuji Matsumoto. Japanese dependency structure analysis based on support vector machines. In *EMNLP 2000*, pages 18–25, 2000.

[Marcus *et al.*, 1993] Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics*, 19(2):313–330, 1993.

[Martins *et al.*, 2008] Andre F. T. Martins, Dipanjan Das, Noah A. Smith, and Eric P. Xing. Stacking dependency parsers. In *EMNLP 2008*, pages 157–166, 2008.

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

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

[McDonald *et al.*, 2005b] Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. Non-projective dependency parsing using spanning tree algorithms. In *HLT-EMNLP 2005*, pages 523–530, 2005.

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

[Petrov and Klein, 2007] Slav Petrov and Dan Klein. Improved inference for unlexicalized parsing. In *NAACL-HLT 2007*, pages 404–411, 2007.

[Petrov, 2010] Slav Petrov. Products of random latent variable grammars. In *NAACL-HLT 2010*, pages 19–27, 2010.

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

[Uchimoto *et al.*, 1999] Kiyotaka Uchimoto, Satoshi Sekine, and Hitoshi Isahara. Japanese dependency structure analysis based on maximum entropy models. In *EACL 1999*, pages 196–203, 1999.

[Yamada and Matsumoto, 2003] H Yamada and Y Matsumoto. Statistical dependency analysis with support vector machines. In *IWPT 2003*, 2003.

[Zhang and Clark, 2008] Yue Zhang and Stephen Clark. A tale of two parsers: Investigating and combining graph-based and transition-based dependency parsing. In *EMNLP 2008*, pages 562–571, 2008.

[Zhang *et al.*, 2009] Hui Zhang, Min Zhang, Chew Lim Tan, and Haizhou Li. K-best combination of syntactic parsers. In *EMNLP 2009*, pages 1552–1560, 2009.