Fourth-Order Dependency Parsing[*]

Xuezhe Ma[1,2] *Hai Zhao*[1,2†]
(1) Center for Brain-Like Computing and Machine Intelligence
Department of Computer Science and Engineering, Shanghai Jiao Tong University
(2) MOE-Microsoft Key Laboratory for Intelligent Computing and Intelligent Systems
Shanghai Jiao Tong University
800 Dong Chuan Rd., Shanghai 200240, China
`xuezhe.ma@gmail.com, zhaohai@cs.sjtu.edu.cn`

ABSTRACT

We present and implement a fourth-order projective dependency parsing algorithm that effectively utilizes both “grand-sibling” style and “tri-sibling” style interactions of third-order and “grand-tri-sibling” style interactions of forth-order factored parts for performance enhancement. This algorithm requires $O(n^5)$ time and $O(n^4)$ space. We implement and evaluate the parser on two languages—English and Chinese, both achieving state-of-the-art accuracy. This results show that a higher-order (≥4) dependency parser gives performance improvement over all previous lower-order parsers.

KEYWORDS: Dependency Parsing, Fourth-order.

[*] This work was partially supported by the National Natural Science Foundation of China (Grant No. 60903119 and Grant No. 61170114), the National Research Foundation for the Doctoral Program of Higher Education of China under Grant No.20110073120022, the National Basic Research Program of China (Grant No. 2009CB320901), and the European Union Seventh Framework Program (Grant No. 247619).

[†] Corresponding author

1 Introduction

In recent years, dependency parsing has gained universal interest due to its usefulness in a wide range of applications such as synonym generation (Shinyama et al., 2002), relation extraction (Nguyen et al., 2009) and machine translation (Katz-Brown et al., 2011; Xie et al., 2011).

CoNLL-X shared task on dependency parsing (Buchholz and Marsi, 2006; Nivre et al., 2007) made a comparison of many algorithms, and graph-based parsing models have achieved state-of-the-art accuracy for a wide range of languages. Graph-based dependency parsing algorithms usually use the factored representations of dependency trees: a set of small *parts* with special structures. The types of features that the model can exploit depend on the information included in the factorizations. Several previous works have shown that higher-order parsers utilizing richer contextual information achieve higher accuracy than lower-order ones— Chen et al. (2010) illustrated that a wide range of decision history can lead to significant improvements in accuracy for graph-based dependency parsing models. Meanwhile, several previous works (Carreras, 2007; Koo and Collins, 2010) have shown that grandchild interactions provide important information for dependency parsing. However, the computational cost of the parsing algorithm increases with the need for more expressive factorizations. Consequently, the existing most powerful parser (Koo and Collins, 2010) is limited to third-order parts, which requires $O(n^4)$ time and $O(n^3)$ space.

In this paper, we further present a fourth-order parsing algorithm that can utilize more richer information by enclosing grand-sibling and tri-sibling parts into a grand-tri-sibling part. Koo and Collins (2010) discussed the possibility that the third-order parsers are extended to fourth-order by increasing vertical context (e.g. from grand-siblings to “great-grand-siblings”) or horizontal context (e.g. from grand-siblings to “grand-tri-siblings”), and Koo (2010) first described this algorithm. In this work, we show that grand-tri-siblings can effectively work. The computational requirements of this algorithm are $O(n^5)$ time and $O(n^4)$ space. To achieve empirical evaluations of our parser, we implement and evaluate the proposed parsing algorithm on the Penn WSJ Treebank (Marcus et al., 1993) for English, and Penn Chinese Treebank (Xue et al., 2005) for Chinese, both achieving state-of-the-art accuracy. A free distribution of our implementation in C++ has been put on the Internet.[1]

2 Related Work

There have been several existing graph-based dependency parsing algorithms, which are the backbones of the new fourth-order dependency parser. In this section, we mainly describe four graph-based dependency parsers with different types of factorization.

The first-order parser (McDonald et al., 2005) decomposes a dependency tree into its individual edges. Eisner (2000) introduced a widely-used dynamic programming algorithm for first-order parsing, which is to parse the left and right dependents of a word independently, and combine them at a later stage. This algorithm introduces two types of dynamic programming structures: *complete* spans, and *incomplete* spans (McDonald, 2006). Larger spans are created from two smaller, adjacent spans by recursive combination in a bottom-up procedure.

McDonald and Pereira (2006) defined a second-order sibling dependency parser in which interactions between adjacent siblings are allowed. Koo and Collins (2010) proposed an algorithm

[1] http://sourceforge.net/projects/maxparser/

that factors each dependency tree into a set of *grandchild* parts. Formally, a grandchild part is a triple of indices (g, s, t) where g is the head of s and s is the head of t. In order to parse this factorization, it is necessary to augment both complete and incomplete spans with grandparent indices. Following Koo and Collins (2010), we refer to these augmented structures as *g-spans*. The second-order parser proposed in Carreras (2007) is capable of scoring both sibling and grandchild parts with complexities of $O(n^4)$ time and $O(n^3)$ space. However, the parser suffers a crucial limitation that it can only evaluate events of grandchild parts for outermost grandchildren.

Koo and Collins (2010) proposed a third-order grand-sibling parser that decomposes each tree into set of *grand-sibling* parts—parts combined with sibling parts and grandchild parts. This factorization defines all grandchild and sibling parts and still requires $O(n^4)$ time and $O(n^3)$ space. Koo and Collins (2010) also discussed the possibility that the third-order parsers are extended to fourth-order by increasing vertical context or horizontal context and Koo (2010) first described this algorithm.

Zhang and McDonald (2012) generalized the Eisner (1996) algorithm to handle arbitrary features over higher-order dependencies. However, their generalizing algorithm suffers quite high complexities of time and space – for instance, the parsing complexity of time is $O(n^5)$ for a third-order factored model. In order to achieve asymptotic efficiency of cost, cube pruning for decoding is utilized (Chiang, 2007).

Another dominant category of data-driven dependency parsing systems is local-and-greedy transition-based parsing (Yamada and Matsumoto, 2003; Nivre and Scholz, 2004; Attardi, 2006; McDonald and Nivre, 2007) which parameterizes models over transitions from state to another in an abstract state-machine. In these models, dependency trees are constructed by making a series of incremental decisions. Parameters in these models are typically learned using standard classification techniques.

3 Fourth-Order Parsing Algorithm

In this section, we propose our fourth-order dependency parsing algorithm, which factors each dependency tree into a set of *grand-tri-sibling* parts. Specifically, a grand-tri-sibling is a 5-tuple of indices (g, s, r, m, t) where (s, r, m, t) is a tri-sibling part and (g, s, r, m) and (g, s, m, t) are grand-sibling parts.

The algorithm is characterized by introducing a new type incomplete g-spans structure: grand-sibling-spans or *gs-spans*, by augmenting incomplete g-spans with a sibling index. Formally, we denote gs-spans as $[g, s, m, t]$ where $[g, s, t]$ is a normal incomplete g-span and m is an index lying in the strict interior of the range $[s, t]$, such that (s, m, t) forms a valid sibling part.

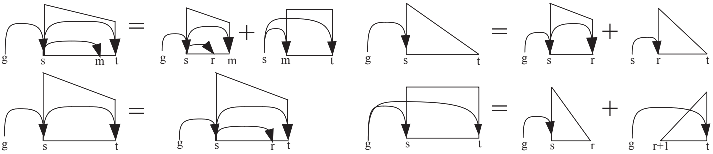

Figure 1: The dynamic-programming structures and derivation of fourth-order grand-tri-sibling parser. Symmetric right-headed versions are elided for brevity.

```
Initialization: $C[g][s][s][d][c] = 0.0 \quad \forall g, s, d, c$
for $k : 1..n$
  for $s : 0..n-k$
    $t = s + k$
    for $g < s$ or $g > t$
      for $m > s$ and $m < t$
        $D[g][s][m][t][\rightarrow] = C[s][s+1][m][\leftarrow][1] + S(g,s,-,-,m) + C[s][m][t][-][2] + S(g,s,-,m,t)$
        $D[g][s][m][t][\leftarrow] = C[t][m][t-1][\rightarrow][1] + S(g,t,-,-,m) + C[t][s][m][-][2] + S(g,t,-,m,s)$

        $D[g][s][m][t][\rightarrow] = \max\{D[g][s][m][t][\rightarrow], \max_{s<r<m}\{D[g][s][r][m][\rightarrow] + C[s][m][t][-][2] + S(g,s,r,m,t)\}\}$
        $D[g][s][m][t][\leftarrow] = \max\{D[g][s][m][t][\leftarrow], \max_{m<r<t}\{D[g][m][r][t][\leftarrow] + C[t][s][m][-][2] + S(g,t,r,m,s)\}\}$
      end for
      $C[g][s][t][-][2] = \max_{s \leq r < t}\{C[g][s][r][\rightarrow][1] + C[g][r+1][t][\leftarrow][1]\}$

      $C[g][s][t][\rightarrow][0] = C[g][s][s][\leftarrow][1] + C[s][s+1][t] \leftarrow][1] + S(g,s,-,-,t)$
      $C[g][s][t][\leftarrow][0] = C[g][t][t][\rightarrow][1] + C[t][s][t-1][\rightarrow][1] + S(g,t,-,-,s)$

      $C[g][s][t][\rightarrow][0] = \max\{C[g][s][t][\rightarrow][0], \max_{s<r<t}\{D[g][s][r][t][\rightarrow]\}\}$
      $C[g][s][t][\leftarrow][0] = \max\{C[g][s][t][\leftarrow][0], \max_{s<r<t}\{D[g][s][r][t][\leftarrow]\}\}$

      $C[g][s][t][\rightarrow][1] = \max_{s<r \leq t}\{C[g][s][r][\rightarrow][0] + C[s][r][t][\rightarrow][1]\}$
      $C[g][s][t][\leftarrow][1] = \max_{s \leq r<t}\{C[g][r][t][\leftarrow][0] + C[t][s][r][\leftarrow][1]\}$
    end for
  end for
end for
```

Figure 2: Pseudo-code of bottom-up chart parser for fourth-order grand-tri-sibling parsing algorithm

Figure 1 provides a graphical specification of the fourth-order grand-tri-sibling parsing algorithm. An incomplete gs-span is constructed by combining a smaller incomplete gs-span, representing the next-innermost pair of modifiers, with a sibling g-span. The algorithm resembles the third-order grand-sibling parser except that the incomplete g-spans are constructed by an incomplete gs-span with the same region.

We will now describe the fourth-order grand-tri-sibling parsing algorithm in more detail. Like factored parsing algorithms presented in the previous section, this parsing algorithm can be parsed via adaptations of standard chart-parsing techniques. Following McDonald (2006), let $C[g][s][t][d][c]$ be a dynamic programming table that stores the score of the best subtree from position s to position t, $s < t$, with grandparent position g, direction d and complete value c. The variable $d \in \{\leftarrow, \rightarrow\}$ indicates the direction of the subtree (gathering left or right dependents). The variable $c \in \{0, 1, 2\}$ indicates if a subtree is complete ($c = 1$), incomplete ($c = 0$) or represents sibling subtrees ($c = 2$). Sibling types have no inherent direction, so it will be always able to assume that when $c = 2$ then $d = null(-)$. We introduce another dynamic programming table $D[g][s][m][t][d]$ to store the score of the best gs-span from position s to position t, $s < t$, with grandparent position g, sibling position m and direction d. Since gs-spans are all incomplete ($c = 1$), the complete value can be omitted. Pseudo code for filling up the dynamic programming tables is in Figure 2. Since the introduction of gs-span, this parsing algorithm requires $O(n^5)$ time and $O(n^4)$ space.

grand-sibling features for part (g,s,m,t)	
4-gram features	**context features**
L(g)·P(s)·P(m)·P(t)	P(g)·P(s)·P(m)·P(t)·P(g+1)·P(s+1)·P(t+1)
P(g)·L(s)·P(m)·P(t)	P(g)·P(s)·P(m)·P(t)·P(g-1)·P(s-1)·P(t-1)
P(g)·P(s)·L(m)·P(t)	P(g)·P(s)·P(m)·P(t)·P(g+1)·P(s+1)
P(g)·P(s)·P(m)·L(t)	P(g)·P(s)·P(m)·P(t)·P(g-1)·P(s-1)
L(g)·L(s)·P(m)·P(t)	P(g)·P(m)·P(t)·P(g+1)·P(m+1)·P(t+1)
L(g)·P(s)·L(m)·P(t)	P(g)·P(m)·P(t)·P(g+1)·P(m-1)·P(t-1)
L(g)·P(s)·P(m)·L(t)	P(g)·P(m)·P(g+1)·P(m+1)
P(g)·L(s)·L(m)·P(t)	P(g)·P(m)·P(g-1)·P(m-1)
L(g)·L(s)·P(m)·L(t)	P(g)·P(t)·P(g+1)·P(t+1)
P(g)·P(s)·L(m)·L(t)	P(g)·P(t)·P(g-1)·P(t-1)
P(g)·P(s)·P(m)·P(t)	P(m)·P(t)·P(m+1)·P(t+1)
	P(m)·P(t)·P(m-1)·P(t-1)

coordination features			**backed-off features**	
L(g)·L(s)·L(t)	L(g)·P(s)	P(s)·P(t)	L(g)·P(m)·P(t)	L(g)·P(m)·L(t)
L(g)·P(s)·P(t)	P(g)·L(s)	P(g)·P(s)	P(g)·L(m)·P(t)	P(g)·L(m)·L(t)
P(g)·L(s)·P(t)	L(g)·P(t)	P(g)·L(t)	P(g)·P(m)·L(t)	P(g)·P(m)·P(t)
P(g)·P(s)·L(t)	P(g)·P(t)	L(s)·P(t)	L(g)·L(m)·P(t)	
L(g)·L(s)·P(t)	P(g)·P(s)·P(t)	P(s)·L(t)		
L(g)·P(s)·L(t)	P(g)·L(s)·L(t)			

tri-sibling features for part (s,r,m,t)			
4-gram features		**backed-off features**	
L(s)·P(r)·P(m)·P(t)	L(s)·P(r)·P(m)·L(t)	L(r)·P(m)·P(t)	P(r)·P(m)·P(t)
P(s)·L(r)·P(m)·P(t)	P(s)·L(r)·L(m)·P(t)	P(r)·L(m)·P(t)	L(r)·L(t)
P(s)·P(r)·L(m)·P(t)	P(s)·L(r)·P(m)·L(t)	P(r)·P(m)·L(t)	L(r)·P(t)
P(s)·P(r)·P(m)·L(t)	P(s)·P(r)·L(m)·L(t)	L(r)·L(m)·P(t)	P(r)·L(t)
L(s)·L(r)·P(m)·P(t)	P(s)·P(r)·P(m)·P(t)	L(r)·P(m)·L(t)	P(r)·P(t)
L(s)·P(r)·L(m)·P(t)		P(r)·L(m)·L(t)	

context features	
P(r)·P(m)·P(t)·P(r+1)·P(m+1)·P(t+1)	P(r)·P(m)·P(r+1)·P(m+1)
P(r)·P(m)·P(t)·P(r-1)·P(m-1)·P(t-1)	P(r)·P(m)·P(r-1)·P(m-1)
P(s)·P(r)·P(m)·P(t)·P(s+1)·P(r+1)	P(r)·P(t)·P(r+1)·P(t+1)
P(s)·P(r)·P(m)·P(t)·P(s-1)·P(r-1)	P(r)·P(t)·P(r-1)·P(t-1)
P(s)·P(r)·P(m)·P(t)·P(r+1)·P(t+1)	P(m)·P(t)·P(m+1)·P(t+1)
P(s)·P(r)·P(m)·P(t)·P(r-1)·P(t-1)	P(m)·P(t)·P(m-1)·P(t-1)
P(s)·P(r)·P(m)·P(t)·P(s+1)·P(r+1)·P(t+1)	P(s)·P(r)·P(m)·P(t)·P(s+1)·P(t+1)
P(s)·P(r)·P(m)·P(t)·P(s-1)·P(r-1)·P(t-1)	P(s)·P(r)·P(m)·P(t)·P(s-1)·P(t-1)

grand-tri-sibling features for part (g,s,r,m,t)			
5-gram features		**4-gram backed-off features**	
L(g)·P(s)·P(r)·P(m)·P(t)	P(g)·L(s)·P(r)·P(m)·P(t)	L(g)·P(r)·P(m)·P(t)	P(g)·L(r)·P(m)·P(t)
P(g)·P(s)·L(r)·P(m)·P(t)	P(g)·P(s)·P(r)·L(m)·P(t)	P(g)·P(r)·L(m)·P(t)	P(g)·P(r)·P(m)·L(t)
P(g)·P(s)·P(r)·P(m)·L(t)	P(g)·P(s)·P(r)·P(m)·P(t)	P(g)·P(r)·P(m)·P(t)	

Table 1: All feature templates used by the fourth-order grand-tri-sibling parser. L(·) and P(·) are the lexicon and POS tag of each token.

4 Feature Space

Following previous works (McDonald and Pereira, 2006; Koo and Collins, 2010), the fourth-order parser captures not only features associated with corresponding fourth-order grand-tri-sibling parts, but also the features of relevant lower-order parts that are enclosed in its factorization.

The lower-order features (first-order features of dependency parts and second-order features of grandchild and sibling parts) are based on feature sets from previous work (McDonald et al., 2005; McDonald and Pereira, 2006; Carreras, 2007). We added lexicalized versions of several features. For example, second-order grandchild feature set defines lexical trigram features, while previous work only used POS trigram features.

Table 1 outlines all feature templates of third-order grand-sibling, third-order tri-sibling, and fourth-order grand-tri-sibling parts. The fourth-order feature set consists of two sets of features. The first set of features is defined to be *5-gram features* that is a 5-tuple consisting of five relevant indices using words and POS tags. The second set of features is defined as *backed-off features* (Koo and Collins, 2010) for grand-tri-sibling part (g, s, r, m, t)—the 4-gram (g, r, m, t), which never exist in any lower-order part. The determination of this feature set is based on on experiments on the development data for both English and Chinese. In section 5.1 we examine the impact of these new features on parsing performance.

According to Table 1, several features in our parser depend on part-of-speech (POS) tags of input sentences. For English, POS tags are automatically assigned by the SVMTool tagger (Gimenez and Marquez, 2004). The accuracy of the SVMTool tagger on PTB is 97.3%; For Chinese, we used gold-standard POS tags in CTB. Following Koo and Collins (2010), two versions of POS tags are used for any features involve POS: one using is normal POS tags and another is a coarsened version of the POS tags. [2]

5 Experiments

The proposed fourth-order dependency parsing algorithm is evaluated on the Penn English Treebank (PTB 3.0) (Marcus et al., 1993) and the Penn Chinese Treebank (CTB 5.0).

For English, the PTB data is prepared by using the standard split: sections 2-21 are used for training, section 22 is for development, and section 23 for test. For Chinese, we adopt the identical training/validation/testing data split and experimental set-up as Zhang and Clark (2009). Dependencies are extracted by using Penn2Malt[3] tool.

Parsing accuracy is measured with unlabeled attachment score (UAS): the percentage of words with the correct head, and the percentage of complete matches (CM).[4]

The k-best version of the Margin Infused Relaxed Algorithm (MIRA) (Crammer and Singer, 2003; Crammer et al., 2006; McDonald, 2006) for the max-margin models (Taskar et al., 2003) is chosen for parameter estimation of our parsing model, In practice, we set $k = 10$ and exclude the sentences containing more than 100 words in both the training data sets of English and Chinese in all experiments.[5]

[2] For English, we used first two characters of the tag, except PRP$; For Chinese, we dropped the last character, except PU and CD

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

[4] As in previous work, English evaluation ignores any token whose gold-standard POS tag is one of { “ ” : , .}, and Chinese evaluation ignores any token whose tag is "PU".

[5] The number of sentences with more than 100 words is 3 for PTB and 67 for CTB.

	Eng		Chn	
Feature	**UAS**	**CM**	**UAS**	**CM**
baseline	93.45	49.06	87.38	38.85
+tri-sibling	93.62	49.42	87.60	38.94
+grand-tri-sibling	93.70	49.76	87.69	39.12
+4-gram backed-off	93.77	50.82	87.74	39.23

Table 2: The effect of different types of features on the development sets for English and Chinese.

5.1 Development Experiments

In this section, we dissect the contributions of each type of features. Table 2 shows the effect of different types of features on the development data sets for English and Chinese. Each row in Table 2 uses a super set of features than the previous one. Third-order grand-sibling parser is used as the baseline, and third-order tri-sibling, 5-gram grand-tri-sibling and 4-gram backed-off feature templates in Table 1 are incrementally added. All systems use our proposed fourth-order parsing algorithm. Since the only difference between systems is the set of features used, we can analyze the improvement from additional features.

From Table 2, we can see that each of the following parser capturing a group of new feature templates makes improvement on parsing performance over the previous one. Thus, we can conclude that the improvements come from the factorization's ability of capturing richer features which contains more context information.The parser with all these features achieves UAS of 93.77% and CM of 50.82% on PTB and UAS of 87.74%, CM of 39.23% on CTB.

5.2 Results and Analysis

Our parser obtains UAS of 93.4% and CM 50.3% of on PTB, and UAS of 87.4%, CM of 36.8% on CTB. Both of the results are state-of-the-art performance on these two treebanks.

Table 3 illustrates the UAS and CM of the fourth-order parser on PTB, together with some relevant results from related work. We compare our method to first-order and second-order sibling dependency parsers (McDonald and Pereira, 2006), and two third-order graph-based parsers (Koo and Collins, 2010). Additionally, we compare to a state-of-the-art graph-based parser (Zhang and McDonald, 2012) as well as a state-of-the-art transition-based parser (Zhang and Nivre, 2011).

Our experimental results show an improvement in performance over the results in Zhang and Nivre (2011), which are based on a transition-based dependency parser with rich non-local features. Our results are also better than the results of the two third-order graph-based dependency parsing models in Koo and Collins (2010). Moreover, our algorithm achieves better parsing performance than the generalized higher-order parser with cube-pruning (Zhang and McDonald, 2012), which is the state-of-the-art graph-based dependency parser so far. The models marked † or ‡ are not directly comparable to our work. The models marked † use semi-supervised methods with large amount of unlabeled data, and those marked ‡ utilize phrase-structure annotations, whiling our parser obtains results competitive with these works. All three models marked † or ‡ are based on the Carreras (2007) parser, which might be replaced by our fourth-order parser to get an even better performance.

Parser	UAS	CM
McDonald and Pereira (2006), 1^{st}order	90.9	36.7
McDonald and Pereira (2006), 2^{nd}order	91.5	42.1
Zhang and Clark (2008)	92.1	45.4
Zhang and Nivre (2011)	92.9	48.0
Koo and Collins (2010), model 2	92.9	–
Koo and Collins (2010), model 1	93.0	–
Zhang and McDonald (2012)	93.1	–
this paper	**93.4**	**50.3**
Koo et al. (2008)†	93.2	–
Carreras et al. (2008)‡	93.5	–
Suzuki et al. (2009)†	93.8	–

Table 3: UAS and CM of different parsers on PTB 3.0

Parser	UAS	CM
Huang and Sagae (2010)	85.2	33.7
Zhang and Clark (2008)	85.7	34.4
Zhang and Nivre (2011)	86.0	36.9
3^{rd}order grand-sibling	86.8	35.5
Zhang and McDonald (2012)	86.9	–
this paper	**87.4**	**36.8**
Zhang and Clark (2009)‡	86.6	36.1

Table 4: UAS and CM of different parsers on CTB 5.0

Next, we turn to the impact of our fourth-order parser on Chinese. Table 4 shows the comparative results for Chinese. Here we compare our method to an implement of the third-order grand-sibling parser — whose parsing performance on CTB is not reported in Koo and Collins (2010), and the dynamic programming transition-based parser of Huang and Sagae (2010). Additionally, we compare to the state-of-the-art graph-based dependency parser (Zhang and McDonald, 2012) as well as a state-of-the-art transition-based parser (Zhang and Nivre, 2011). The results indicates that our parser achieved significant improvement of the previous systems on this data set. The parsing model of Zhang and Clark (2009), which is marked ‡, also depends on phrase-structure annotations. So it cannot compare with ours directly, even through our results are better.

6 Conclusion

We have presented an even higher-order projective dependency parsing algorithm that can evaluate the fourth-order sub-structures of grand-tri-siblings. This algorithm achieves stage-of-the-art performance on both PTB and CTB, which demonstrates that the fourth-order grand-tri-sibling features have important contribution to dependency parsing.

A wide range of further research involving the fourth-order parsing algorithm is available. One idea would be to identify the highest *n* for which the information of *n*th-order part still improves parsing performance. Moreover, as the fourth-order parser has achieved state-of-the-art accuracy on standard parsing benchmarks, many NLP tasks may benefit from it.

References

Attardi, G. (2006). Experiments with a multilanguage non-projective dependency parser. In *Proceedings of the Tenth Conference on Natural Language Learning (CoNLL-2006)*, pages 166–170, New York, USA.

Buchholz, S. and Marsi, E. (2006). CoNLL-X shared task on multilingual dependency parsing. In *Proceeding of the 10th Conference on Computational Natural Language Learning (CoNLL'06)*, pages 149–164, New York, NY.

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

Carreras, X., Collins, M., and Koo, T. (2008). Tag, dynamic programming, and the perceptron for efficient, feature-rich parsing. In *Proceedings of CoNLL*, pages 9–16, Manchester, England.

Chen, W., Kazama, J., Tsuruoka, Y., and Torisawa, K. (2010). Improving graph-based dependency parsing with decision history. In *Proceeding of the 23rd International Conference on Computional Linguistics (COLING'10)*, pages 126–134, BeiJing, China.

Chiang, D. (2007). Hierarchical phrase-based translation. *Computational linguistics*, 33(2).

Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., and Singer, Y. (2006). Online passive-aggressive algorithms. *Jornal of Machine Learning Research*, 7:551–585.

Crammer, K. and Singer, Y. (2003). Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learining Research*, 3:951–991.

Eisner, J. (1996). Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 9th International Conference on Computational Linguistics (COLING'06)*, pages 340–345.

Eisner, J. (2000). Bilexical grammars and their cubic-time parsing algorithms. In Bunt, H. and Nijholt, A., editors, *Advances in Probabilistic and Other Parsing Technologies*, pages 29–62. Kluwer Academic Publishers.

Gimenez and Marquez (2004). Svmtool: A general pos tagger generator based on support vector machines. In *Proceedings of the 4th International Conference of Language Resources and Evaluation (IREC'04)*, Lisbon, Portugal.

Huang, L. and Sagae, K. (2010). Dynamic programming for linear-time incremental parsing. In *Proceeding of ACL 2010*, pages 1077–1086, Uppsala, Sweden.

Katz-Brown, J., Petrov, S., McDonald, R., Och, F., Talbot, D., Ichikawa, H., Seno, M., and Kazawa, H. (2011). Training a parser for machine translation reordering. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 183–192, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Koo, T. (2010). *Advances in Discriminative Dependency Parsing*. PhD thesis, Massachusetts Institute of Technology.

Koo, T., Carreras, X., and Collins, M. (2008). Simple semi-supervised dependency parsing. In *Proceedings of ACL-08: HLT*, pages 595–603, Columbus, Ohio, USA.

Koo, T. and Collins, M. (2010). Efficient third-order dependency parsers. In *Proceedings of 48th Meeting of the Association for Computional Linguistics (ACL'10)*, pages 1–11, Uppsala, Sweden.

Marcus, M., Santorini, B., and Marcinkiewicz, M. A. (1993). Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

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

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

McDonald, R. and Nivre, J. (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 2007)*, pages 122–131, Prague, Czech.

McDonald, R. and Pereira, F. (2006). Online learning of approximate dependency parsing algorithms. In *European Association for Computational Linguistics (EACL-2006)*, pages 81–88, Trento, Italy.

Nguyen, T.-V. T., Moschitti, A., and Riccardi, G. (2009). Convolution kernels on constituent, dependency and sequential structures for relation extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1378–1387, Singapore. Association for Computational Linguistics.

Nivre, J., Hall, J., Kübler, S., Mcdonald, R., Nilsson, J., Riedel, S., and Yuret, D. (2007). The CoNLL 2007 shared task on dependency parsing. In *Proceeding of EMNLP-CoNLL 2007*, pages 915–932, Prague, Czech.

Nivre, J. and Scholz, M. (2004). Deterministic dependency parsing of english text. In *Proceedings of the 20th international conference on Computational Linguistics (COLING'04)*, pages 64–70, Geneva, Switzerland.

Shinyama, Y., Sekine, S., and Sudo, K. (2002). Automatic paraphrase acquisition from news articles. In *Proceeding of the 2nd International Conference on Human Language Technology Research (HLT'02)*, pages 313–318.

Suzuki, J., Isozaki, H., Carreras, X., and Collins, M. (2009). An empirial study of semi-supervised structured conditional models for dependency parsing. In *Proceedings of EMNLP*, pages 551–560.

Taskar, B., Guestrin, C., and Koller, D. (2003). Max margin markov networks. In Sebastian Thrun, L. K. S. and Scholköopf, B., editors, *NIPS*. MIT Press.

Xie, J., Mi, H., and Liu, Q. (2011). A novel dependency-to-string model for statistical machine translation. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 216–226, Edinburgh, Scotland, UK. Association for Computational Linguistics.

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

Yamada, H. and Matsumoto, Y. (2003). Statistical dependency analysis with support vector machines. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT-2003)*, pages 195–206, Nancy, France.

Zhang, H. and McDonald, R. (2012). Generalized higher-order dependency parsing with cube pruning. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 320–331, Jeju Island, Korea. Association for Computational Linguistics.

Zhang, Y. and Clark, S. (2008). A tale of two parsers: investigating and combining graph-based and transition-based dependency parsing using beam search. In *Proceedings of EMNLP*, pages 562–571.

Zhang, Y. and Clark, S. (2009). Transition-based parsing of Chinese treebank using a global discriminative model. In *Proceedings of the 11th International Conference on Parsing Technologies (IWPT'09)*, pages 162–171, Paris, France.

Zhang, Y. and Nivre, J. (2011). Transition-based dependency parsing with rich non-local features. In *Proceeding of ACL 2011*, pages 188–193, Portland, Oregon.