PLCFRS Parsing of English Discontinuous Constituents

Kilian Evang
Humanities Computing
University of Groningen
k.evang@rug.nl

Laura Kallmeyer
Institut für Sprache und Information
University of Düsseldorf
kallmeyer@phil.uni-duesseldorf.de

Abstract

This paper proposes a direct parsing of non-local dependencies in English. To this end, we use probabilistic linear context-free rewriting systems for data-driven parsing, following recent work on parsing German. In order to do so, we first perform a transformation of the Penn Treebank annotation of non-local dependencies into an annotation using crossing branches. The resulting treebank can be used for PLCFRS-based parsing. Our evaluation shows that, compared to PCFG parsing with the same techniques, PLCFRS parsing yields slightly better results. In particular when evaluating only the parsing results concerning long-distance dependencies, the PLCFRS approach with discontinuous constituents is able to recognize about 88% of the dependencies of type *T* and *T*-PRN encoded in the Penn Treebank. Even the evaluation results concerning local dependencies, which can in principle be captured by a PCFG-based model, are better with our PLCFRS model. This demonstrates that by discarding information on non-local dependencies the PCFG model loses important information on syntactic dependencies in general.

1 Introduction

Discontinuous constituents as exemplified in (1) are more frequent than generally assumed, even in languages such as English that display a rather rigid word order. In (1), the NP *areas of the factory where the crocidolite was used* is separated into two non-adjacent parts. (1) is an example from the Penn Treebank (PTB). More generally, all constructions where head-argument or head-modifier dependencies are non-local, such as *wh*-movement, can be seen as instances of discontinuous constituency. Such instances appear in about 20% of the sentences in the PTB. They constitute a particular challenge for parsing.

(1) *Areas of the factory* were particularly dusty *where the crocidolite was used.*

In the past, data-driven parsing has largely been dominated by Probabilistic Context-Free Grammar (PCFG). This is partly due to the annotation formats of treebanks such as the Penn Treebank (PTB) (Marcus et al., 1994), which are used as a data source for grammar extraction. Their annotation generally relies on the use of trees without crossing branches, augmented with a mechanism that accounts for non-local dependencies. In the PTB, e.g., labeling conventions and trace nodes are used which establish additional implicit edges in the tree beyond the overt phrase structure.

However, given the expressivity restrictions of PCFG, work on data-driven parsing has mostly excluded non-local dependencies. When using treebanks with PTB-like annotation, labeling conventions and trace nodes are often discarded.

Some work has however been done towards incorporating non-local information into data-driven parsing. One general way to do this is (non-projective) dependency parsing where parsers are not grammar-based and the notion of constituents or phrases is not employed, see e.g. McDonald et al. (2005) or Nivre (2009). Within the domain of grammar-based constituent parsing, we can distinguish three approaches (Nivre, 2006): 1. Non-local information can be reconstructed in a post-processing step after PCFG parsing (Johnson, 2002; Levy and Manning, 2004; Jijkoun and de Rijke, 2004; Campbell, 2004; Gabbard et al., 2006). 2. Non-local information can be incorpo-

Proceedings of the 12th International Conference on Parsing Technologies, pages 104–116,
October 5-7, 2011, Dublin City University.

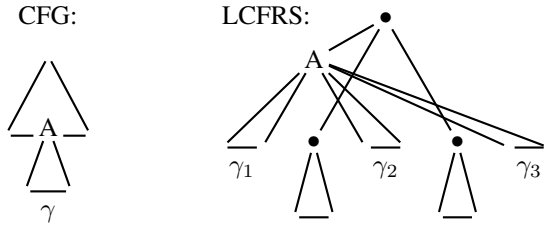


Figure 1: Different domains of locality

rated into the PCFG model (Collins, 1999) or into complex labels (Dienes and Dubey, 2003; Hockenmaier, 2003; Cahill et al., 2004). 3. A formalism can be used which accommodates the direct encoding of non-local information (Plaehn, 2004; Maier and Kallmeyer, 2010; Kallmeyer and Maier, 2010). This paper pursues the third approach.

Our work is based on recent research in using Linear Context-Free Rewriting Systems (LCFRS) (Vijay-Shanker et al., 1987) for data driven parsing. LCFRSs extend CFGs such that non-terminals can span tuples of possibly non-adjacent strings (see Fig. 1). This enables them to describe discontinuous constituents and non-projective dependencies (Kuhlmann and Satta, 2009; Maier and Lichte, 2009). Furthermore, they are able to capture synchronous derivations, something that is empirically attested in treebanks (Kallmeyer et al., 2009). In order to parse German, a language where discontinuities are particularly frequent, Kallmeyer and Maier (2010); Maier and Kallmeyer (2010) use probabilistic LCFRSs (PLCFRSs). As a data source, they use the German NEGRA and TIGER treebanks that annotate discontinuous constituents by using crossing branches.

We adapt this approach for German to English, using the PTB. For this, we first need to transform the trace-based annotation of discontinuous constituents into an annotation with crossing branches which requires a careful treatment of the different types of traces that occur in the PTB. Then we extract a PLCFRS from the resulting treebank and we use the PLCFRS parser from Kallmeyer and Maier for our parsing experiments.

The paper is structured as follows. Section 2 introduces PLCFRS and the parsing algorithm. The next section explains the transformation of the PTB into an annotation format where non-local dependencies are annotated with crossing branches. Section 4 describes further transformations we apply to the resulting treebanks, in particular binarization and category splitting. Finally, section 5 reports the results or our parsing experiments with a detailed evaluation of the way the different types of long-distance dependencies are captured. Section 6 concludes.

2 PLCFRS Parsing

2.1 PLCFRS

LCFRSs are an extension of CFG where the non-terminals can span not only single strings but, instead, tuples of strings (see Fig. 1). An LCFRS (Vijay-Shanker et al., 1987) is a tuple $\langle N, T, V, P, S\rangle$ where

a) N is a finite set of non-terminals with a function dim: $N \to \mathbb{N}$; $dim(A)$ is called the *fan-out* of A and determines the dimension of the tuples in the yield of A;

b) T and V are disjoint finite sets of terminals and variables;

c) $S \in N$ is the start symbol with $dim(S) = 1$;

d) P is a finite set of rules

$$A(\alpha_1, \ldots, \alpha_{dim(A)}) \to A_1(X_1^{(1)}, \ldots, X_{dim(A_1)}^{(1)}) \cdots A_m(X_1^{(m)}, \ldots, X_{dim(A_m)}^{(m)})$$

for $m \geq 0$ where $A, A_1, \ldots, A_m \in N$, $X_j^{(i)} \in V$ for $1 \leq i \leq m, 1 \leq j \leq dim(A_i)$ and $\alpha_i \in (T \cup V)^*$ for $1 \leq i \leq dim(A)$. For all $r \in P$, it holds that every variable X occurring in r occurs exactly once in the left-hand side (LHS) and exactly once in the right-hand side (RHS).

A rewriting rule describes how the yield of the LHS non-terminal can be computed from the yields of the RHS non-terminals. The rules $A(ab, cd) \to \varepsilon$ and $A(aXb, cYd) \to A(X, Y)$ for instance specify that 1. $\langle ab, cd\rangle$ is in the yield of A and 2. one can compute a new tuple in the yield of A from an already existing one by wrapping a and b around the first component and c and d around the second.

For every $A \in N$ in a LCFRS G, we define the yield of A, $yield(A)$ as follows:

a) For every $A(\vec{\alpha}) \to \varepsilon$, $\vec{\alpha} \in yield(A)$;

b) For every rule

$$A(\alpha_1, \ldots, \alpha_{dim(A)}) \to A_1(X_1^{(1)}, \ldots, X_{dim(A_1)}^{(1)}) \cdots A_m(X_1^{(m)}, \ldots, X_{dim(A_m)}^{(m)})$$

and all $\vec{\tau_i} \in yield(A_i)$ for $1 \leq i \leq m$, $\langle f(\alpha_1), \ldots, f(\alpha_{dim(A)})\rangle \in yield(A)$ where f is defined as follows: (i) $f(t) = t$ for all $t \in T$, (ii) $f(X_j^{(i)}) = \vec{\tau_i}(j)$ for all $1 \leq i \leq m, 1 \leq j \leq dim(A_i)$ and (iii) $f(xy) = f(x)f(y)$ for

all $x, y \in (T \cup V)^+$. f is the *composition function* of the rule.

c) Nothing else is in $yield(A)$.

The language is then $\{w \mid \langle w \rangle \in yield(S)\}$.

The *fan-out* of an LCFRS G is the maximal fan-out of all non-terminals in G. An LCFRS with a fan-out of n is called an n-LCFRS. Furthermore, the RHS length of a rewriting rules $r \in P$ is called the *rank* of r and the maximal rank of all rules in P is called the *rank* of G. We call a LCFRS *monotone* if for every $r \in P$ and every RHS non-terminal A in r and each pair X_1, X_2 of arguments of A in the RHS of r, X_1 precedes X_2 in the RHS iff X_1 precedes X_2 in the LHS.

A *probabilistic LCFRS* (PLCFRS) (Kato et al., 2006) is a tuple $\langle N, T, V, P, S, p \rangle$ such that $\langle N, T, V, P, S \rangle$ is a LCFRS and $p : P \rightarrow [0..1]$ a function such that for all $A \in N$: $\Sigma_{A(\vec{x}) \rightarrow \vec{\Phi} \in P} p(A(\vec{x}) \rightarrow \vec{\Phi}) = 1$.

2.2 CYK Parsing

We use the parser from Kallmeyer and Maier (2010); Maier (2010), Maier and Kallmeyer (2010) which is a probabilistic version of the CYK parser from Seki et al. (1991), applying techniques of weighted deductive parsing (Nederhof, 2003).

LCFRSs can be binarized (Gómez-Rodríguez et al., 2009) and ε-components in the LHS of rules can be removed (Boullier, 1998). We can therefore assume that all rules are of rank 2 (in section 4.1, we explain our binarization technique) and do not contain ε components in their LHSs. Furthermore, we assume POS tagging to be done before parsing. POS tags are non-terminals of fan-out 1. The rules are then either of the form $A(a) \rightarrow \varepsilon$ with A a POS tag and $a \in T$ or of the form $A(\vec{\alpha}) \rightarrow B(\vec{x})$ or $A(\vec{\alpha}) \rightarrow B(\vec{x})C(\vec{y})$ where $\vec{\alpha} \in (V^+)^{dim(A)}$, i.e., only the rules for POS tags contain terminals in their LHSs.

For every $w \in T^*$, we call every pair $\langle l, r \rangle$ with $0 \leq l \leq r \leq |w|$ a *range* in w. The *concatenation* of two ranges $\rho_1 = \langle l_1, r_1 \rangle, \rho_2 = \langle l_2, r_2 \rangle$ is defined as follows: if $r_1 = l_2$, then $\rho_1 \cdot \rho_2 = \langle l_1, r_2 \rangle$; otherwise $\rho_1 \cdot \rho_2$ is undefined.

For a given rule $p : A(\alpha_1, \ldots, \alpha_{dim(A)}) \rightarrow B(X_1, \ldots, X_{dim(B)})C(Y_1, \ldots, X_{dim(C)})$ we now extend the composition function f to ranges, given an input w: for all vectors of ranges $\vec{\rho_B}$ and $\vec{\rho_C}$ of dimensions $dim(B)$ and $dim(C)$ respectively, $f_r(\vec{\rho_B}, \vec{\rho_C}) = \langle g(\alpha_1), \ldots, g(\alpha_{dim(A)}) \rangle$ is defined as follows: $g(X_i) = \vec{\rho_B}(i)$ for all $1 \leq i \leq dim(B)$, $g(Y_i) = \vec{\rho_C}(i)$ for all $1 \leq i \leq dim(C)$ and $g(xy) = g(x) \cdot g(y)$ for all $x, y \in V^+$. $p : A(f_r(\vec{\rho_B}, \vec{\rho_C})) \rightarrow B(\vec{\rho_B})C(\vec{\rho_C})$ is then called an *instantiated rule*.[1]

Scan: $\dfrac{}{0 : [A, \langle \langle i, i+1 \rangle \rangle]}$ A POS tag of w_{i+1}

Unary: $\dfrac{in : [B, \vec{\rho}]}{in + |log(p)| : [A, \vec{\rho}]}$ $p : A(\vec{\alpha}) \rightarrow B(\vec{\alpha}) \in P$

Binary: $\dfrac{in_B : [B, \vec{\rho_B}], in_C : [C, \vec{\rho_C}]}{in_B + in_C + log(p) : [A, \vec{\rho_A}]}$

where $p : A(\vec{\rho_A}) \rightarrow B(\vec{\rho_B})C(\vec{\rho_C})$ is an instantiated rule.

Goal: $[S, \langle \langle 0, n \rangle \rangle]$

Figure 2: Weighted CYK deduction system

add SCAN results to $\mathcal{A}$
while $\mathcal{A} \neq \emptyset$
  remove best item $x : I$ from $\mathcal{A}$
  add $x : I$ to $\mathcal{C}$
  if I goal item
  then stop and output true
  else
    for all $y : I'$ deduced from $x : I$ and items in $\mathcal{C}$:
      if there is no z with $z : I' \in \mathcal{C} \cup \mathcal{A}$
      then add $y : I'$ to $\mathcal{A}$
      else if $z : I' \in \mathcal{A}$ for some z
        then update weight of I' in $\mathcal{A}$ to $max(y, z)$

Figure 3: Weighted deductive parsing

For a given input w, our items have the form $[A, \vec{\rho}]$ where $A \in N$, $\vec{\rho}$ a vector of ranges with $|\vec{\rho}| = dim(A)$. $\vec{\rho}$ characterizes the span of A. We specify the set of weighted parse items via the deduction rules in Fig. 2. The parser performs a weighted deductive parsing (Nederhof, 2003), based on this deduction system. It uses a chart $\mathcal{C}$ and an agenda $\mathcal{A}$, both initially empty, and proceeds as in Fig. 3.

3 Treebank Transformation

The PTB annotation guidelines (Bies et al., 1995, Section 1.1) specify a set of rules that determine where arguments and adjuncts are attached with respect to their head words. For example, subjects are attached at clause level, most other arguments and adjuncts of verbs are attached at VP level, and phrases modifying nouns such as PPs and relative clauses are adjoined at NP level. Knowing these

[1] This corresponds to the *instantiated clauses* in simple Range Concatenation Grammars (Boullier, 1998; Boullier, 2000).

rules, head-argument and head-adjunct dependencies can be read off the trees easily, e.g. for semantic interpretation.

Non-local head-argument and head-adjunct dependencies constitute exceptions to these rules. Following the rules would lead to discontinuous constituents with crossing branches, containing the head and the argument or adjunct, but not containing some intervening tokens. Examples of non-locally dependent arguments and adjuncts include *wh*-moved phrases, fronted phrases, extraposed modifiers, *it*-extraposition, and right-node-raised phrases (Fig. 4a-d). Such phrases are attached at locations in the tree that avoid discontinuity, thus the heads on which they depend cannot easily be determined from the tree structure alone. The PTB instead uses the null elements *T*, *ICH*, *EXP* and *RNR* to mark the position where the phrases would be attached according to the general rules and indices in node labels to indicate which null element stands for which phrase (shown by arcs in the tree diagrams). Null elements are embedded in "placeholder phrases" of the same category (but without WH prefixes) as the non-locally dependent phrase. This representation of non-local dependencies is not suitable for PCFG parsing since null elements pose a serious combinatorial problem and PCFG has no mechanism for dealing with indexed category labels. Null elements and indices are therefore usually removed before training PCFG parsers, resulting in parse trees that do not contain information on non-local dependencies.

We use the approach proposed and tested on the German treebanks NEGRA and TIGER in Maier and Kallmeyer (2010): permit discontinuous constituents, attach non-locally dependent arguments and adjuncts according to the general rules, resulting in a uniform representation for local and non-local dependencies, and use PLCFRS for parsing. While NEGRA and TIGER already use such a uniform representation, training and testing data for English can be obtained by removing placeholder phrases with *T*, *ICH*, *EXP* and *RNR* null elements from their locations in the PTB trees and reattaching the coindexed phrases to those locations, removing indices from node labels (Fig. 5). Other types of null elements are used to indicate control and other relations with no immediate bearing on non-local head-adjunct and head-argument dependencies. We remove these from the treebank along with corresponding indices.

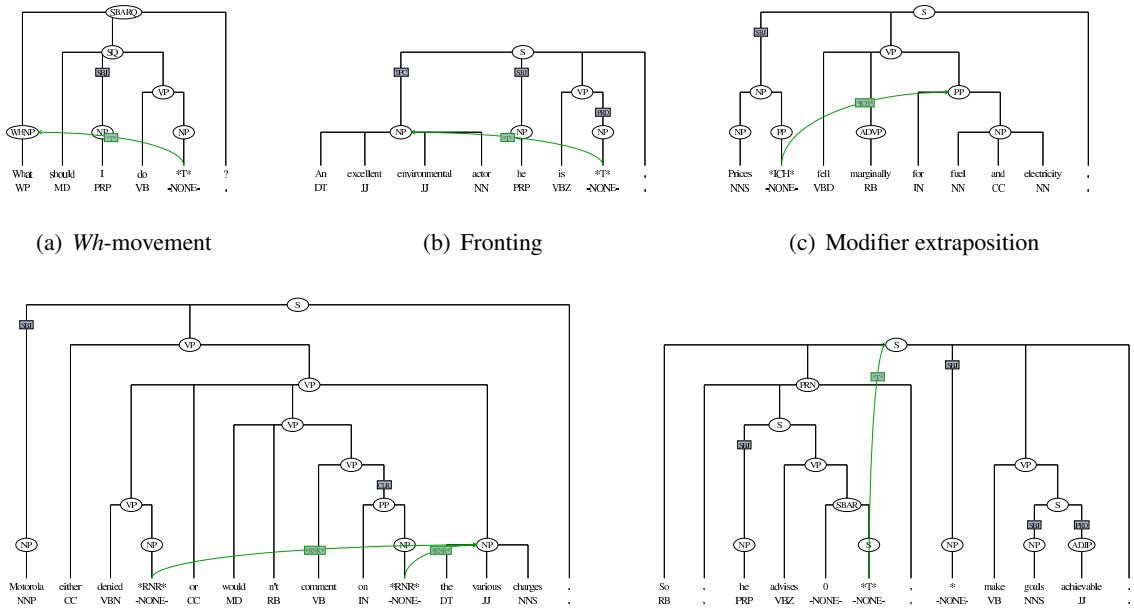
			trees with gap-degree		
type	instances	trees	0	1	2
T	18759	15452	7292	7924	236
T-PRN	843	843	0	71	772
ICH	1268	1240	7	1200	33
EXP	658	651	1	630	20
RNR	210	208	131	67	10
any reattachment	21738	17187	7397	8996	794
no reattachment	n/a	32021	32021	0	0
total	n/a	49208	39418	8996	794

Table 1: Reattachment types and gap-degrees of resulting trees

Two types of cases require special treatment. First, some arguments and adjuncts are shared between two or more heads, marked by two or more null elements with the same index (Fig. 4(d)). Since a phrase cannot be attached to more than one location in a tree even with crossing branches, the phrase must either remain in place, where no relation to any head can be immediately read off the tree, or be attached according to the general rules with respect to only one of the heads, leaving the others with no trace of the argument or adjunct. For now, we decided to put consistency in the way arguments and adjuncts are attached first and always attach phrases with multiple heads as depending on the head which is closest (Fig. 5(d)). The other special case concerns phrases, typically quotations, that surround the matrix phrase containing the head on which they depend. In the PTB annotation, the matrix phrase is embedded into such arguments under a node labeled PRN for *parenthesis* (Fig. 4(e)). To avoid cycles after the transformation, such matrix phrases are detached from within the argument and reattached to the node where the argument was originally attached, if any (Fig. 5(e)).

Table 1 gives an overview of the tendency of each type of null element[2] to introduce gaps when so transformed as indicated by *gap-degree* (Holan et al., 1998; Maier and Lichte, 2009), i.e. the maximal number of gaps in any constituent of the resulting trees. Most typically, one gap is introduced since there is a single phrase non-adjacent to the rest of the phrase to which it is attached. No gap at all is introduced by the reattachment of most *wh*-moved subjects and *EXP*-type phrases in object position. Gap degrees of 2 are almost exclusively accounted for by surrounding phrases where the

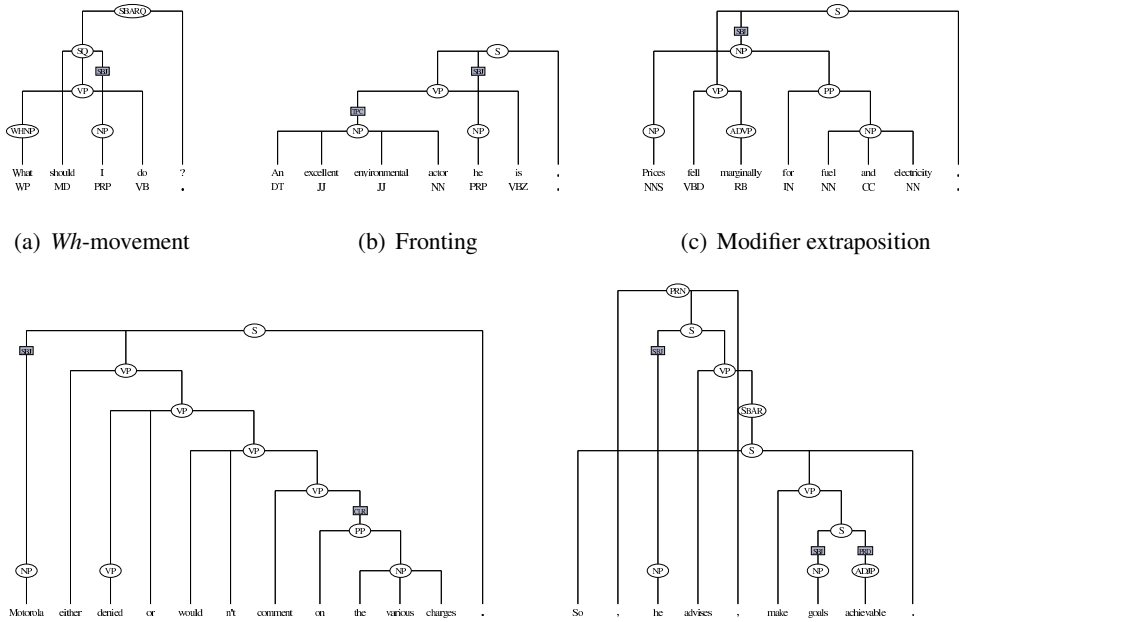
[2]Those instances of *T* reattachments where the dependent element is a surrounding phrase are given separately as *T*-PRN.



(a) *Wh*-movement (b) Fronting (c) Modifier extraposition

(d) Right-node raising (e) Surrounding argument

Figure 4: Annotation of non-local head-argument and head-adjunct dependencies in the PTB



(a) *Wh*-movement (b) Fronting (c) Modifier extraposition

(d) Right-node raising (e) Surrounding argument

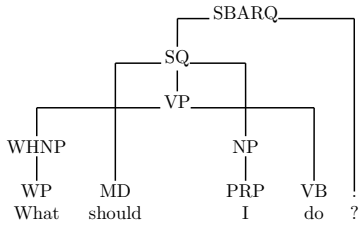
Figure 5: Transformed versions of the trees in Fig. 4

VP of the surrounded matrix clause is typically interrupted by two commas and the subject of the matrix clause. On the whole, about 20% of the trees in the transformed PTB contain discontinuities – less than the c. 30% reported by Maier and Lichte (2009) for the German treebanks NEGRA and TIGER, but still a considerable percentage.

An LCFRS is extracted from the transformed treebank using the algorithm of Maier and Søgaard (2008), simplified using the fact that leaves do not have siblings and their parents are labeled with POS tags: every leaf is represented as a variable. Every internal node n is represented as a term $A_i(X_{11} \dots X_{1j_1}, \dots, X_{i1} \dots X_{ij_i})$ where $X_{11}, \dots, X_{1j_1}, \dots, X_{i1} \dots X_{ij_1}$ represent the leaves dominated by n in order, there is an argument boundary between two variables iff the corresponding leaves are non-adjacent, A is the label of n and i is the number of arguments, used to obtain a unique non-terminal A_i with fan-out i. A rule $\alpha \to \beta_1 \dots \beta_m$ is extracted for each internal node n such that α is the term representing n and $\beta_1, \dots, \beta_m$ are the terms representing its children, conventionally ordered by leftmost dominated terminal. For parents of leaves, m is 0, and the single variable in α is replaced with the terminal labeling the corresponding leaf. For other nodes, every sequence of variables that occurs as a right-hand side argument is replaced with a single new variable on both sides. Fig. 6 shows an example. Rules are equivalent if equal up to renaming variables. The resulting LCFRS rules are ε-free and they are monotone. The latter means that the order of the arguments of a RHS element is the same as the order of these variables in the LHS. Both properties facilitate parsing.

The number of occurrences of the rules are counted and the probabilities of RHSs conditioned on LHSs are then calculated using MLE. In this way, a PLCFRS is obtained. This is a very simple probability model, much like a vanilla PCFG. In the following section, we discuss techniques we used to refine the probability model.

Non-binary tree:



Extracted LCFRS rules:

$$
\begin{aligned}
\text{SBARQ}_1(XY) &\to \text{SQ}_1(X).\text{}_1(Y) \\
\text{SQ}_1(XYZU) &\to \text{VP}_2(X,U)\text{MD}_1(Y)\text{NP}_1(Z) \\
\text{VP}_1(X,Y) &\to \text{WHNP}_1(X)\text{VB}_1(Y) \\
\text{WHNP}_1(X) &\to \text{WP}_1(X) \\
\text{NP}_1(X) &\to \text{PRP}_1(X) \\
\text{WP}_1(\text{What}) &\to \varepsilon \\
\text{MD}_1(\text{should}) &\to \varepsilon \\
\text{PRP}_1(\text{I}) &\to \varepsilon \\
\text{VB}_1(\text{do}) &\to \varepsilon \\
\text{.}_1(?) &\to \varepsilon
\end{aligned}
$$

Figure 6: LCFRS extraction from trees

4 Grammar Annotation

4.1 Binarization

Similarly to the transformation of a CFG into Chomsky Normal Form (CNF), we binarize the LCFRS extracted from the treebank. The result is an LCFRS of rank 2. A binarization technique that results in horizontal Markovization of the grammar is proposed and successfully used for parsing NEGRA and TIGER in Kallmeyer and Maier (2010). However, our experiments have shown that the beneficial effect of this horizontal Markovization technique does not carry over to parsing the PTB, presumably because compared to the two German treebanks, the PTB has a more hierarchical annotation scheme, extracted grammars have rules with shorter RHSs to begin with and can thus profit less from additional factorization; the adverse effect of wrong independence assumptions predominates. We thus use a *deterministic* binarization technique that does not change the probability model. Specifically, we introduce a *unique* new non-terminal for each right-hand side longer than 2 and split the rule into two rules, using this new intermediate non-terminal. This is repeated until all right-hand sides are of length 2. The transformation algorithm is inspired by Gómez-Rodríguez et al. (2009) and it is also specified in Kallmeyer (2010). Fig. 7 shows an example.

$$
\begin{aligned}
\text{SBARQ}_1(XY) &\to \text{SQ}_1(X).\text{}_1(Y) \\
\text{SQ}_1(XYZ) &\to \text{VP}_1(X,Z)\text{C}_1(Y) \\
\text{C}_1(XY) &\to \text{MD}_1(X)\text{NP}_1(Y) \\
\text{VP}_1(X,Y) &\to \text{WHNP}_1(X)\text{VB}_1(Y) \\
\text{WHNP}_1(X) &\to \text{WP}_1(X) \\
\text{NP}_1(X) &\to \text{PRP}_1(X)
\end{aligned}
$$

Figure 7: Binarized grammar equivalent to the grammar in Figure 6, not showing terminal rules.

Note however that the fan-out of the LCFRS can increase because of the binarization.

4.2 Category Splits

Category splitting, i.e. relabeling certain nodes in the training data depending on context, has been used to improve the performance of PCFG parsing (Klein and Manning, 2003) and also PLCFRS parsing (Kallmeyer and Maier, 2010). Our experiments have shown that a combination of three splits for the PTB annotation improved performance considerably: S nodes are relabeled to SWH if a *wh*-element is extracted from the sentence. In order to make this split more effective, SBAR nodes that have only one child after transformation to the discontinuous format are removed. VP nodes are relabeled to VPHINF if their head is labeled VB, to VPHTO if their head is labeled TO and to VPHPART if their head is labeled VBN or VBG. S nodes rooting infinitival clauses (head child labeled VPHINF or VPHTO) are relabeled to SINF.

5 Evaluation

We use the Wall Street Journal sections 1-22 of the Penn Treebank (version 2.0) as training data and sections 23-24 as test data. Due to time constraints and the complexity of PLCFRS parsing, sentences with more than 25 tokens (not counting null elements) are excluded, resulting in 25801 training sentences and 2233 test sentences. After a small number of corrections to the annotation, concerning chiefly wrong indices and missing PRN nodes, we create discontinuous versions of the training and test set by carrying out the reattachment operations described in Section 3 while also keeping context-free versions. All four sets are then preprocessed by removing all (remaining) indices, null elements and empty constituents. We call the resulting context-free training and test set Tr and Te, and the resulting discontinuous training and test set Tr' and Te'.

5.1 EVALB-Style Evaluation

Since the structure in Te' encodes local as well as non-local dependencies, it serves as our primary gold standard. In a first step, we use the standard EVALB metric, generalized to trees with discontinuous constituents as in Maier and Kallmeyer (2010), to measure how much of the structure in the gold standard is captured by different parsers. We compare Maier and Kallmeyer's parser trained on Tr' (resulting in a 3-PLCFRS) with three parsers that do *not* produce discontinuous structures: the Berkeley parser (Petrov et al., 2006; Petrov and Klein, 2007) trained on Tr using our manual category splits but no automatic splitting/merging/smoothing, the Berkeley parser trained on Tr using its default setting of six iterations of split/merge/smooth, and Maier and Kallmeyer's parser with a grammar extracted from Tr (a 1-PLCFRS, i.e. a PCFG). The upper half of Table 2 shows the results. For comparison, we also evaluated the three context-free parsers on the untransformed context-free test set Te. These figures are given in the lower half of the table. For Maier and Kallmeyer's parser, the number of rules in the grammar before and after binarizing is also given, as well as the number of items created during parsing as an indicator of parsing complexity.

Across these experiments, the most crucial factor for parsing accuracy seems to be splitting/merging/smoothing. As the comparison between the two parsing experiments with the Berkeley parser shows, this technique is key to achieving its state-of-the-art results. We plan to transfer this technique to discontinuous constituent parsing in future work. For now, we must compare discontinuous to context-free constituent parsing on a level below the state of the art. Comparison between the two experiments with Maier and Kallmeyer's parser shows that it works with about the same accuracy when trained and tested on discontinuous data as when trained and tested on context-free data, although parsing complexity is considerably higher in the discontinuous experiment as evidenced by the number of items produced. Note that scores would presumably be lower if sentences with more than 25 tokens were included.

Even when trained on the context-free data, both parsers get most of the structure in Te' right since only a relatively small fraction of constituents is discontinuous. However, for those test sentences that do contain discontinuous constituents (Te'_D), context-free parsers fare much worse than for sentences that do not (Te'_C). For Maier and Kallmeyer's parser trained on Tr' they seem to be only slightly harder to parse. Although its scores for Te'_D with discontinuous parsing are lower than for Te_D with context-free parsing, the former may be considered a better parse result than the latter since the Te'_D gold standard con-

Parser		Berkeley		Maier&Kallmeyer	
Training set		Tr	Tr	Tr	Tr'
Split/merge		6 it.	man.	man.	man.
Test set					
Te'	LP	87.29	72.86	78.13	80.36
	LR	86.89	67.88	74.60	77.61
	LF_1	87.09	70.28	76.33	78.96
	UP	90.40	77.70	82.59	83.74
	UR	89.98	72.39	78.86	80.00
	UF_1	90.19	74.95	80.68	82.29
Te'_C	LP	89.43	74.55	79.97	80.66
	LR	89.37	69.75	76.57	77.81
	LF_1	89.40	72.07	78.23	79.21
	UP	91.85	78.63	83.91	83.95
	UR	91.78	73.57	80.34	80.99
	UF_1	91.82	76.02	82.09	82.44
Te'_D	LP	77.06	64.76	69.42	78.90
	LR	75.31	59.14	65.44	76.61
	LF_1	76.17	61.82	67.37	77.74
	UP	83.48	73.23	76.31	82.77
	UR	81.58	66.88	71.93	80.37
	UF_1	82.52	69.91	74.06	81.55
Te	LP	89.82	74.88	80.37	
	LR	89.64	69.94	76.94	
	LF_1	89.73	72.32	78.61	
	UP	91.89	78.80	83.91	-
	UR	91.70	73.60	80.33	
	UF_1	91.80	76.11	82.08	
Te_C	LP	89.90	74.94	80.36	
	LR	89.85	70.12	76.95	
	LF_1	89.88	72.45	78.62	
	UP	91.90	78.63	83.92	-
	UR	91.84	73.57	80.36	
	UF_1	91.87	76.02	82.10	
Te_D	LP	89.43	74.58	80.40	
	LR	88.64	69.08	76.87	
	LF_1	89.03	71.73	78.60	
	UP	91.86	79.61	83.85	-
	UR	91.05	73.74	80.16	
	UF_1	91.46	76.56	81.96	
Rules				8892	9761
Bin. rules				27809	29218
Items				580M	1056M

Table 2: EVALB-style evaluation of parsing experiments (scores in %). Tr and Te are the context-free training and test sets, Tr' and Te' the discontinuous transformed versions. The D and C subscripts indicate the subsets of the test sets containing the sentences that actually have (D) resp. do not have (C) one or more discontinuities in Te'. For Maier and Kallmeyer's parser, the number of rules in the unbinarized and binarized grammar as well as the number of parse items produced is given.

Parser	Maier&Kallmeyer
Training set	Tr''
Split/merge	man.
Test set	
Te''	LP 80.71
	LR 77.85
	LF_1 79.26
	UP 84.07
	UR 81.09
	UF_1 82.55
Te''_C	LP 80.82
	LR 77.90
	LF_1 79.33
	UP 84.12
	UR 81.07
	UF_1 82.57
Te''_D	LP 78.87
	LR 76.38
	LF_1 77.60
	UP 82.49
	UR 79.88
	UF_1 81.16
Rules	9653
Bin. rules	29096
Items	852M

Table 3: Results of a second discontinuous parsing experiment where *ICH* and *EXP* transformations have been omitted in the transformation

tains information on non-local dependencies while Te_D does not.

5.2 Dependency Evaluation

In order to assess to what degree this is the case, we perform a *dependency evaluation* (Lin, 1995), first used for evaluating discontinuous constituent parser output in Maier (2010). This method requires a conversion of constituent trees to sets of word-word dependencies. We use Lin's dependency conversion method, where each phrase is represented by its lexical head. To determine the head of each phrase, we use the head-finding algorithm of Collins (1999), ordering the children of each node by leftmost dominated terminal.

Under this standard dependency conversion method, the transformation described in Section 3 introduces new word-word (head-argument/head-adjunct) dependencies that are relevant to semantic interpretation. Word-word dependencies lost in the transformation are not normally relevant since they result from attachment of phrases outside of the domains of their heads. We therefore choose Te' as the gold standard against which to evaluate both context-free and discontinuous parsing results. Table 4 shows that discontinuous parsing as compared to context-free parsing boosts the unlabeled attachment score (i.e. recall on word-word dependencies) slightly for local dependencies and considerably for non-local dependencies. The lat-

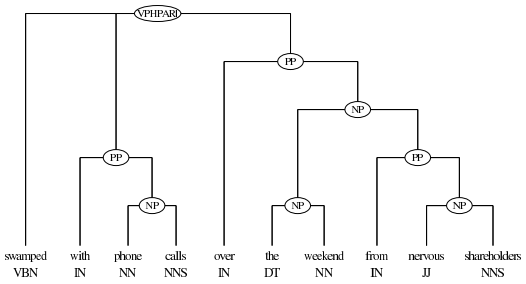


Figure 8: Failure to recognize the discontinuous NP *phone calls from nervous shareholders*

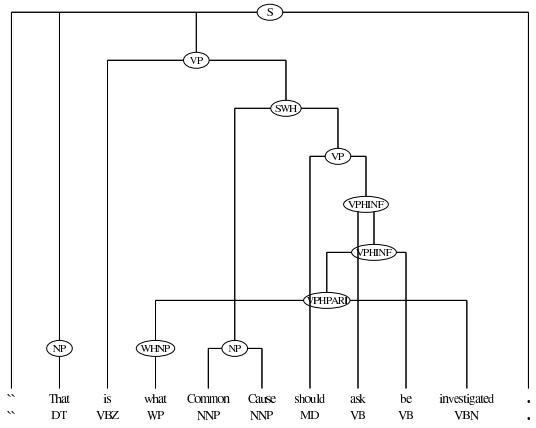


Figure 9: Correct parse of a deeply embedded moved *wh*-phrase

ter are broken down by type as in Section 3.

Dependency evaluation also allows a direct comparison with state-of-the-art dependency parsers. In Table 4 we give results for MSTParser (McDonald and Pereira, 2006) trained on two dependency versions of Tr', converted from constituents to dependencies once without dependency labels and once with dependency labels using the method of Hall and Nivre (2008). It can be seen that MSTParser recognizes a fair percentage of even the difficult *ICH* and *EXP* type dependencies (cf. Section 5.3) and that it has a considerably better overall score. We expect that this gap can be bridged by optimizing Maier and Kallmeyer's parser with techniques successfully used for context-free constituent parsing as outlined above, but this remains to be proven experimentally.

5.3 A Closer Look at the Parses

Further inspection of the parse trees produced in the discontinuous experiment confirms that Maier and Kallmeyer's parser trained on Tr' correctly analyzes the majority of the *T* and *T*-PRN type discontinuities[3]: *wh*-movement (108 of 129), fronted quotations (129 of 142) and surrounding arguments (22 of 31). Fronting apart from quotations (0 of 6) and right-node raising (3 of 4) seems too rare to allow for a meaningful assessment. *ICH* and *EXP* type discontinuities are almost never correctly parsed (2 of 30 resp. 2 of 14).

The latter suffer from massive attachment ambiguity – attaching the right part of a discontinuous constituent of this type locally as illustrated in Fig. 8 almost always leads to higher-scoring parses in the current model. Augmenting it with lexical information as is common in modern PCFG-based parsers could help to better resolve these and other attachment ambiguities. In the present model, including discontinuous annotation of *ICH* and *EXP* type dependencies adds no value to the parser output and is even detrimental since grammar rules typically found with them like $\text{NP}(X, Y) \rightarrow \text{NP}(X)\text{SINF}(Y)$ are used by the parser in many falsely detected discontinuities. We therefore conducted another experiment with Maier and Kallmeyer's parser using the training and test sets Tr'' and Te''. These are like Tr' and Te' except that *ICH* and *EXP* type dependencies were not included in the transformation. This makes EVALB scores for the continuous test trees rise slightly and parsing time drop considerably due to the smaller number of discontinuous rules (Table 3). However, a higher unlabeled attachment score is not achieved by this (cf. Table 4).

T type discontinuities are well recognized presumably due to their strong correlations with patterns like a *wh*-word followed by a sentence, a sentence followed by another, a sentence interrupted by another, and characteristic punctuation in the case of fronted and discontinuous quotations. Even correctly recognizing deeply embedded attachment of *wh*-moved phrases as illustrated in Fig. 9 poses no major problem. One of the problems that do exist is the many possible ways that a sentence can be split into two parts as a surrounding quotation, resulting in many required S rules of fan-out 2 and in sparse data. This problem could

[3]This excludes most cases of subject *wh*-movement which accounts for some of the *T* type dependencies but does not induce discontinuities.

	gold	Maier&Kallmeyer			MSTParser	
		Tr	Tr'	Tr''	Tr' unlabeled	Tr' Hall&Nivre
T	436	134	386	380	374	379
T-PRN	32	8	26	26	25	26
ICH	31	3	5	4	10	9
EXP	18	0	1	0	8	9
RNR	4	2	3	3	3	3
other	35918	29785	30252	30241	32452	32457
total	36439	29932	30673	30654	32872	32883
		82.14%	84.18%	84.12%	90.21%	90.24%

Table 4: Unlabeled attachment scores in dependency evaluation on the dependency-converted Te'

be tackled by factoring rules into an expansion part (what RHS categories) and a separation part (where the gap is), similar to the factorization proposed in Levy (2005, Section 4.8). Note also that there is nothing in the present model to prevent LCFRS rules associated with different constructions, such as *wh*-movement and fronting, from recombining, producing nonsensical parses in a few cases. Finally, it should be noted that attaching commas surrounding parentheses inside surrounding quotations rather than to the PRN node could reduce the fan-out of the grammar from 3 to 2, benefiting parsing efficiency.

6 Conclusion and Future Work

This paper pursues an approach of direct parsing of discontinuous constituents. We have applied data-driven PLCFRS parsing to English. To this end, we have first transformed the trace-based Penn Treebank annotation format into a format with crossing branches and explicit discontinuous constituents. The latter can then be used for PLCFRS parsing.

Our evaluation has shown that, compared to PCFG parsing with the same techniques, PLCFRS parsing yields slightly better results. In particular when evaluating only the parsing results concerning long-distance dependencies, the PLCFRS approach with discontinuous constituents is able to recognize about 88% of the dependencies of type *T* and *T*-PRN. Even the results concerning local dependencies, which can in principle be captured by a CFG-based model, are better with the PLCFRS model. This demonstrates that by discarding information on non-local dependencies the PCFG model loses important information on syntactic dependencies in general.

Our results show that data-driven PLCFRS parsing is a promising and feasible strategy not only for so-called free word order languages such as German but also for English where we obtain competitive parsing results.

However, our experiments also reveal some shortcomings of the chosen probabilistic model. A general problem is that some decisions, for instance on PP-attachments, cannot be taken solely based on the syntactic information we have used. This problem occurs independent from the choice of PLCFRS. A careful integration of more lexical information can help to overcome this problem. A shortcoming that is specific to LCFRS is the assumption that the expansions of the same category with different fan-outs (for instance a continuous VP and a discontinuous VP) are independent from each other. This bears two problems. Firstly, since categories of higher fan-out are rather rare, we have a sparse data problem. Secondly, the independence assumption is probably wrong. In order to tackle this problem, we plan to develop models that factor rules into an expansion part and a separating part that introduces gaps. We leave this issue for future work.

Acknowledgments

We would like to thank Wolfgang Maier for fruitful discussions and help with implementation, and the anonymous reviewers for their valuable suggestions on improving the paper.

References

Ann Bies, Mark Ferguson, Karen Katz, and Robert MacIntyre. 1995. Bracketing Guidelines for Treebank II Style Penn Treebank Project. University of Pennsylvania.

Pierre Boullier. 1998. A Proposal for a Natu-

ral Language Processing Syntactic Backbone. Technical Report 3342, INRIA.

Pierre Boullier. 2000. Range Concatenation Grammars. In *Proceedings of the Sixth International Workshop on Parsing Technologies (IWPT2000)*, pages 53–64, Trento, Italy, February.

Aoife Cahill, Michael Burke, Ruth O'Donovan, Josef Van Genabith, and Andy Way. 2004. Long-distance dependency resolution in automatically acquired wide-coverage PCFG-based LFG approximations. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 319–326, Barcelona, Spain, July.

Richard Campbell. 2004. Using linguistic principles to recover empty categories. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 645–652, Barcelona, Spain, July.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Pétr Dienes and Amit Dubey. 2003. Deep syntactic processing by combining shallow methods. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 431–438, Sapporo, Japan, July. Association for Computational Linguistics.

Ryan Gabbard, Seth Kulick, and Mitchell Marcus. 2006. Fully parsing the Penn Treebank. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 184–191, New York City, USA, June. Association for Computational Linguistics.

Carlos Gómez-Rodríguez, Marco Kuhlmann, Giorgio Satta, and David Weir. 2009. Optimal reduction of rule length in linear context-free rewriting systems. In *Proceedings of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies Conference (NAACL'09:HLT)*, pages 539–547, Boulder, Colorado.

Johan Hall and Joakim Nivre. 2008. Parsing discontinuous phrase structure with grammatical functions. In *Proceedings of the 6th International Conference on Natural Language Processing (GoTAL)*, pages 169–180.

Julia Hockenmaier. 2003. *Data and models for statistical parsing with Combinatory Categorial Grammar*. Ph.D. thesis, School of Informatics, University of Edinburgh.

Tom Holan, Vladislav Kubo, Karel Oliva, and Martin Pltek. 1998. Two useful measures of word order complexity. In *Workshop on Processing of Dependency-Based Grammars*, pages 21–29, Montreal, Canada.

Valentin Jijkoun and Maarten de Rijke. 2004. Enriching the output of a parser using memory-based learning. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 311–318, Barcelona, Spain, July.

Mark Johnson. 2002. A simple pattern-matching algorithm for recovering empty nodes and their antecedents. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 136–143, Philadelphia, Pennsylvania, USA, July. Association for Computational Linguistics.

Laura Kallmeyer and Wolfgang Maier. 2010. Data-driven parsing with probabilistic Linear Context-Free Rewriting Systems. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING 2010)*, Beijing, China.

Laura Kallmeyer, Wolfgang Maier, and Giorgio Satta. 2009. Synchronous rewriting in treebanks. In *Proceedings of IWPT 2009*.

Laura Kallmeyer. 2010. *Parsing Beyond Context-Free Grammars*. Springer, Berlin. Textbook.

Yuki Kato, Hiroyuki Seki, and Tadao Kasami. 2006. Stochastic multiple context-free grammar for RNA pseudoknot modeling. In *Proceedings of The Eighth International Workshop on Tree Adjoining Grammar and Related Formalisms (TAG+8)*, pages 57–64, Sydney, Australia, July.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 423–430,

Sapporo, Japan, July. Association for Computational Linguistics.

Marco Kuhlmann and Giorgio Satta. 2009. Treebank grammar techniques for non-projective dependency parsing. In *Proceedings of EACL*.

Roger Levy and Christopher Manning. 2004. Deep dependencies from context-free statistical parsers: Correcting the surface dependency approximation. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 327–334, Barcelona, Spain, July.

Roger Levy. 2005. *Probabilistic models of word order and syntactic discontinuity*. Ph.D. thesis, Stanford University.

Dekang Lin. 1995. A dependency-based method for evaluating broad-coverage parsers. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI 1995)*, Montreal, Quebec, Canada.

Wolfgang Maier and Laura Kallmeyer. 2010. Discontinuity and non-projectivity: Using mildly context-sensitive formalisms for data-driven parsing. In *Proceedings of the Tenth International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+10)*, New Haven.

Wolfgang Maier and Timm Lichte. 2009. Characterizing discontinuity in constituent treebanks. In *Proceedings of Formal Grammar 2009*, Bordeaux, France, July. To appear in Lecture Notes in Computer Science, Springer.

Wolfgang Maier and Anders Søgaard. 2008. Treebanks and mild context-sensitivity. In Philippe de Groote, editor, *Proceedings of the 13th Conference on Formal Grammar (FG-2008)*, pages 61–76, Hamburg, Germany. CSLI Publications.

Wolfgang Maier. 2010. Direct parsing of discontinuous constituents in German. In *Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 58–66, Los Angeles, CA, USA, June. Association for Computational Linguistics.

Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The penn treebank: annotating predicate argument structure. In *HLT '94: Proceedings of the workshop on Human Language Technology*, pages 114–119, Morristown, NJ, USA. Association for Computational Linguistics.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *11th Conference of the European Chapter of the Association for Computational Linguistics*.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 523–530, Vancouver, British Columbia, Canada, October. Association for Computational Linguistics.

Mark-Jan Nederhof. 2003. Weighted Deductive Parsing and Knuth's Algorithm. *Computational Linguistics*, 29(1):135–143.

Joakim Nivre. 2006. Constraints on non-projective dependency parsing. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 73–80, Trento, Italy. Association for Computational Linguistics.

Joakim Nivre. 2009. Non-projective dependency parsing in expected linear time. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 351–359, Suntec, Singapore, August. Association for Computational Linguistics.

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

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 433–440, Sydney, Australia, July. Association for Computational Linguistics.

Oliver Plaehn. 2004. Computing the most probable parse for a discontinuous phrase-structure grammar. In *New developments in parsing technology*. Kluwer.

Hiroyuki Seki, Takahashi Matsumura, Mamoru Fujii, and Tadao Kasami. 1991. On multiple context-free grammars. *Theoretical Computer Science*, 88(2):191–229.

K. Vijay-Shanker, David J. Weir, and Aravind K. Joshi. 1987. Characterizing structural descriptions produced by various grammatical formalisms. In *Proceedings of ACL*, Stanford.