Discontinuity and Non-Projectivity: Using Mildly Context-Sensitive Formalisms for Data-Driven Parsing

Wolfgang Maier and **Laura Kallmeyer**
SFB 833, University of Tübingen
{wmaier,lk}@sfs.uni-tuebingen.de

Abstract

We present a parser for probabilistic Linear Context-Free Rewriting Systems and use it for constituency and dependency treebank parsing. The choice of LCFRS, a formalism with an extended domain of locality, enables us to model discontinuous constituents and non-projective dependencies in a straight-forward way. The parsing results show that, firstly, our parser is efficient enough to be used for data-driven parsing and, secondly, its result quality for constituency parsing is comparable to the output quality of other state-of-the-art results, all while yielding structures that display discontinuous dependencies.

1 Introduction

It is a well-known fact that Context-Free Grammar (CFG) does not provide enough expressivity to describe natural languages. For data-driven probabilistic CFG parsing, some of the information present in constituency treebanks, namely the annotation of non-local dependencies, cannot be captured by a CFG. It is therefore removed before learning a PCFG from the treebank and must be re-introduced in a post-processing step (Johnson, 2002; Levy and Manning, 2004). Non-projective dependencies also lie beyond the expressivity of CFG. Current dependency parsers are able to parse them (McDonald et al., 2005; Nivre et al., 2007). However, the corresponding parsing algorithms are not grammar-based.

We propose to use a grammar formalism with an extended domain of locality that is able to capture the non-local dependencies both in constituency and dependency treebanks. We chose Linear Context-Free Rewriting Systems (LCFRS), a mildly context-sensitive extension of CFG that allows non-terminals to span tuples of discontinuous strings. The reason why we think LCFRS particularly well-suited is that treebanks with a direct annotation of discontinuous constituents (with crossing branches as in the German Negra treebank) allow a straight-forward interpretation of the trees as LCFRS derivation structures, without the necessity of inducing linguistic knowledge.[1] This considerably facilitates the extraction of probabilistic LCFRSs (Maier and Søgaard, 2008). The same holds for non-projective dependency structures, which can also straight-forwardly be interpreted as LCFRS derivation structures (Kuhlmann and Satta, 2009). Previous approaches that have used non-context-free formalisms for data-driven constituency parsing (Plaehn, 2004; Chiang, 2003) are either too restricted (Kallmeyer et al., 2009) or do not allow for an immediate interpretation of the treebank trees as derivation structures. Grammar-based non-projective dependency parsing has, to our knowledge, not been attempted at all.

First results for PLCFRS constituency parsing with a detailed evaluation have been presented in Maier (2010). The contribution of this article is to present the first results for data-driven dependency parsing on the dependency version of the German NeGra treebank and on the Prague Dependency Treebank. Furthermore, we give greater detail on the parser and the experimental setup. We also additionally investigate the effect on manually introduced category splits for PLCFRS constituency parsing.

[1] Treebank trees in which non-local dependencies are annotated differently, such as with trace nodes in the Penn Treebank, could also be interpreted as LCFRS derivations given an appropriate transformation algorithm.

The remainder of this paper is structured as follows. In the following section, we present the formalism and our parser. Sect. 3 is dedicated the experimental setup, Sect. 4 contains the experimental results. In Sect. 5, we present a conclusion.

2 A Parser for Probabilistic Linear Context-Free Rewriting Systems

We notate LCFRS with the syntax of *simple Range Concatenation Grammars* (SRCG) (Boullier, 1998), a formalism that is equivalent to LCFRS.

2.1 PLCFRS

A LCFRS (Vijay-Shanker et al., 1987) is a tuple $G = (N, T, V, P, S)$ where a) N is a finite set of non-terminals with a function dim: $N \to \mathbb{N}$ that determines the *fan-out* of each $A \in N$; 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 rewriting rules

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

for $m \geq 0$ where $A, A_1, \dots, A_m \in N$, $X_j^{(i)} \in V$ 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 $r \in P$, every variable X occurring in r occurs exactly once in the left-hand side (LHS) and exactly once in the right-hand side (RHS).

For a given rule, the length of the RHS is called the *rank* of the rule. The maximal fan-out of all non-terminals in an LCFRS G is called the *fan-out* of G, and the maximal rank of all rules in an LCFRS G is called the *rank* of G.

$A(ab, cd) \to \varepsilon$ ($\langle ab, cd\rangle$ in yield of A)
$A(aXb, cYd) \to A(X, Y)$ (if $\langle X, Y\rangle$ in yield of A, then also $\langle aXb, cYd\rangle$ in yield of A)
$S(XY) \to A(X, Y)$ (if $\langle X, Y\rangle$ in yield of A, then $\langle XY\rangle$ in yield of S)

$$L = \{a^n b^n c^n d^n \mid n > 0\}$$

Figure 1: Sample LCFRS

A rewriting rule describes how to compute the yield of the LHS non-terminal from the yields of the RHS non-terminals. The yield of S is the language of the grammar. See Fig. 1 for a sample LCFRS.

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

2.2 PLCFRS Parsing

Our parser is a probabilistic CYK parser (Seki et al., 1991), using the technique of weighted deductive parsing (Nederhof, 2003). We assume without loss of generality that our LCFRSs are binary (i.e., have rank 2) (Gómez-Rodríguez et al., 2009) and do not contain rules where some of the LHS components are ε (Boullier, 1998; Seki et al., 1991). Our binarization algorithm is given in Section 3.4.

Furthermore, we make the assumption that POS tagging is done before parsing. The POS tags are special non-terminals of fan-out 1.

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

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

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 $p : A(\vec{\rho_A}) \to B(\vec{\rho_B})C(\vec{\rho_C})$ 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 $\mathcal{A} \neq \emptyset$ do
  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 $\mathcal{C}$
    do
      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 $max(y, z)$
        end if
      end if
    end for
  end if
end while
```

Figure 3: Weighted deductive parsing

For a given input w, our items have the form $[A, \vec{\rho}]$

where $A \in N$, $\vec{\rho} \in (Pos(w) \times Pos(w))^{dim(A)}$ the vector of ranges characterizing all components of the span of A. We specify the set of weighted parse items via the deduction rules in Fig. 2. An instantiated rule is a rule where variables have been replaced with corresponding vectors of ranges. Our parser performs weighted deductive parsing, based on this deduction system. We use a chart $\mathcal{C}$ and an agenda $\mathcal{A}$, both initially empty, and we proceed as in Fig. 3. For more details of the parser, see also Kallmeyer and Maier (2010).

3 Experimental Setup

3.1 Data

Our data sources are the NeGra treebank (Skut et al., 1997) and the Prague Dependency Treebank 2.0 (PDT) (Hajič et al., 2000).

We create two different data sets for constituent parsing. For the first one, we start out with the unmodified NeGra treebank. We preprocess the treebank following common practice (Kübler and Penn, 2008), attaching all nodes which are attached to the virtual root node to nodes within the tree such that ideally, no new crossing edges are created. In a second pass, we attach punctuation which comes in pairs (parentheses, quotation marks) to the same nodes. For the second data set we create a copy of the preprocessed first data set, in which we apply the usual tree transformations for NeGra PCFG parsing, i.e., moving nodes to higher positions until all crossing branches are resolved. The first 90% of both data sets are used as the training set and the remaining 10% as test set.

For dependency parsing, we also create two data sets. For the first one, we convert the NeGra constituent annotation to labeled dependencies using Lin's (1995) algorithm and Hall and Nivre's (2008) labeling scheme. For the second dependency data set, we use the training sections 1 to 5 of the PDT for training and the first 1,300 sentences for testing. Czech is a language with a rich morphology, which is reflected by a high number of POS tags with additional morphological information in the PDT. As in previous work, we use a simplified tag set in order to avoid data sparseness problems (Collins et al., 1999; McDonald et al., 2005).

We only include sentences with a maximal length of 25 words.[1] This leads to a size of 14,858, resp. 1,651 sentences for the NeGra training, resp. test sets and to 13,935, resp. 1,300 sentences for the PDT training, resp. test set.

3.2 Grammar Extraction

From all of our data sets, we extract PLCFRSs. For the constituent sets, we use the algorithm from Maier and Søgaard (2008), for the dependencies the algorithm from Kuhlmann and Satta (2009). For reasons of space, we restrict ourselves here to the examples in Fig. 4–6.

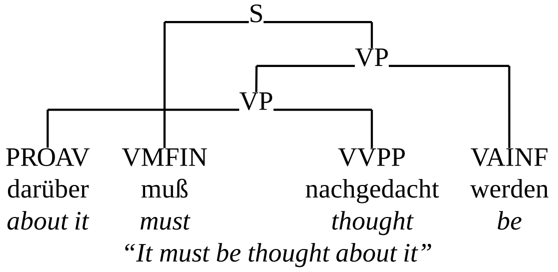


Figure 4: A sample tree from NeGra

$$\begin{aligned}
\text{PROAV(Darüber)} &\rightarrow \varepsilon\\
\text{VVPP(nachgedacht)} &\rightarrow \varepsilon\\
\text{VMFIN(muß)} &\rightarrow \varepsilon\\
\text{VAINF(werden)} &\rightarrow \varepsilon\\
\text{S}_1(X_1X_2X_3) &\rightarrow \text{VP}_2(X_1, X_3)\ \text{VMFIN}(X_2)\\
\text{VP}_2(X_1, X_2X_3) &\rightarrow \text{VP}_2(X_1, X_2)\ \text{VAINF}(X_3)\\
\text{VP}_2(X_1, X_2) &\rightarrow \text{PROAV}(X_1)\ \text{VVPP}(X_2)
\end{aligned}$$

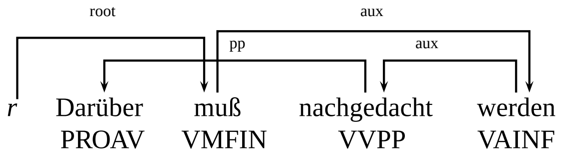
Figure 5: LCFRS rules for the tree in Fig. 4

3.3 Grammar Annotation

Grammar annotation (i.e., manual enhancement of annotation information through category splitting) has previously been successfully employed in parsing German (Versley, 2005). In order to see if such modifications can have a beneficial effect in PLCFRS parsing, we will apply the following category splits to the Negra constituency data sets with unmodified labels (inspired by Petrov and Klein (2007)): We split the category S ("sentence") into SRC ("relative clause") and S (all other categories S). Relative clauses mostly occur in a very specific

[1] This length restriction can be greatly alleviated by using an estimate of outside probabilities of parse items which speeds up parsing (Kallmeyer and Maier, 2010)

Dependency tree:



Corresponding LCFRS rules:

$$\begin{array}{rcl}
\text{PROAV(Darüber)} & \to & \varepsilon \\
\text{VVPP(nachgedacht)} & \to & \varepsilon \\
\text{VMFIN(muß)} & \to & \varepsilon \\
\text{VAINF(werden)} & \to & \varepsilon \\
\text{pp}(X) & \to & \text{PROAV}(X) \\
\text{root}(X_1X_2X_3) & \to & \text{aux}(X_1,X_3)\ \text{VMFIN}(X_2) \\
\text{aux}(X_1,X_2) & \to & \text{pp}(X_1)\ \text{VVPP}(X_2) \\
\text{aux}(X_1,X_2X_3) & \to & \text{aux}(X_1,X_2)\ \text{VAINF}(X_3) \\
\text{top}(X_1) & \to & \text{root}(X_1)
\end{array}$$

Figure 6: LCFRS rules extracted from a dependency tree[3]

```
for all rules $r = A(\vec{\alpha}) \to A_0(\vec{\alpha_0}) \dots A_m(\vec{\alpha_m})$ in $P$ with $m > 1$ do
  remove $r$ from $P$
  $R := \emptyset$
  pick new non-terminals $C_1, \dots, C_{m-1}$
  add the rule $A(\vec{\alpha}) \to A_0(\vec{\alpha_0})C_1(\vec{\gamma_1})$ to $R$ where $\vec{\gamma_1}$ is obtained by reducing $\vec{\alpha}$ with $\vec{\alpha_0}$
  for all $i$, $1 \leq i \leq m-2$ do
    add the rule $C_i(\vec{\gamma_i}) \to A_i(\vec{\alpha_i})C_{i+1}(\vec{\gamma_{i+1}})$ to $R$ where $\vec{\gamma_{i+1}}$ is obtained by reducing $\vec{\gamma_i}$ with $\vec{\alpha_i}$
  end for
  add the rule $C_{m-1}(\vec{\gamma_{m-2}}) \to A_{m-1}(\vec{\alpha_{m-1}})A_m(\vec{\alpha_m})$ to $R$
  for every rule $r' \in R$ do
    replace RHS arguments of length $> 1$ with new variables (in both sides) and add the result to $P$
  end for
end for
```

Figure 7: Algorithm for binarizing a LCFRS

context, namely as the right part of an NP or a PP. This splitting should therefore speed up parsing and increase precision.

The other category split we introduce concerns the VP category and the POS tags of verbs selecting for a VP. We distinguish between VP-PP ("VP with participle verb form"), VP-INF ("VP with infinitive without *zu*") and VP-ZU ("VP with *zu* infitive").

Apart from the *S* and *VP* splits, we also use both splits together (*S* ∘ *VP*).

3.4 Binarization

Before parsing, we binarize the LCFRS rules of the extracted grammars. The transformation is similar to the transformation of a CFG into Chomsky Normal Form (CNF). The result is an LCFRS of rank 2. As in the CFG case, in the transformation, we introduce a non-terminal for each RHS longer than 2 and split the rule into two rules, using this new intermediate non-terminal. This is repeated until all RHS are of length 2.

For the presentation of the transformation algorithm, we need the notion of a *reduction* of a vector $\vec{\alpha} \in [(T \cup V)^*]^i$ by a vector $\vec{x} \in V^j$ where all variables in $\vec{x}$ occur in $\vec{\alpha}$. A reduction is, roughly, obtained by keeping all variables in $\vec{\alpha}$ that are not in $\vec{x}$. This is defined as follows: Let $\langle N, T, V, P, S \rangle$ be an LCFRS, $\vec{\alpha} \in [(T \cup V)^*]^i$ and $\vec{x} \in V^j$ for some $i, j \in \mathcal{N}$. Let $w = \vec{\alpha}_1\$ \dots \$\vec{\alpha}_i$ be the string obtained form concatenating the components of $\vec{\alpha}$, separated by a new symbol $\$ \notin (V \cup T)$. Let w' be the image of w under a homomorphism h defined as follows: $h(a) = \$$ for all $a \in T$, $h(X) = \$$ for all $X \in \{\vec{x}_1, \dots \vec{x}_j\}$ and $h(y) = y$ in all other cases. Let $y_1, \dots y_m \in V^+$ such that $w' \in \$^* y_1 \$^+ y_2 \$^+ \dots \$^+ y_m \$^*$. Then the vector $\langle y_1, \dots y_m \rangle$ is the *reduction* of $\vec{\alpha}$ by $\vec{x}$.

For instance, $\langle aX_1, X_2, bX_3 \rangle$ reduced with $\langle X_2 \rangle$ yields $\langle X_1, X_3 \rangle$ and $\langle aX_1X_2bX_3 \rangle$ reduced with $\langle X_2 \rangle$ yields $\langle X_1, X_3 \rangle$ as well.

The binarization algorithm is given in Fig. 7. Fig. 8 shows an example. In this example, there is only one rule with a RHS longer than 2. In a first step, we introduce the new non-terminals and rules that binarize the RHS. This leads to the set R. In a second step, before adding the rules from R to the grammar, whenever a right-hand side argument contains several variables, they are collapsed into a single new variable.

The equivalence of the original LCFRS and the binarized grammar is rather straight-forward. Note however that the fan-out of the LCFRS can increase because of the binarization.

[3] An extra top rule is added in order to give the PLCFRS parser a unique start symbol in case more than one word has the root node as head, i.e., in case more than one rule with *root* as LHS label is extracted.

Original LCFRS:

$S(XYZUVW) \to A(X,U)B(Y,V)C(Z,W)$
$A(aX,aY) \to A(X,Y) \qquad A(a,a) \to \varepsilon$
$B(bX,bY) \to B(X,Y) \qquad B(b,b) \to \varepsilon$
$C(cX,cY) \to C(X,Y) \qquad C(c,c) \to \varepsilon$

Rule with right-hand side of length > 2:
$S(XYZUVW) \to A(X,U)B(Y,V)C(Z,W)$
For this rule, we obtain
$R = \{S(XYZUVW) \to A(X,U)C_1(YZ,VW),$
$\quad C_1(YZ,VW) \to B(Y,V)C(Z,W)\}$

Equivalent binarized LCFRS:

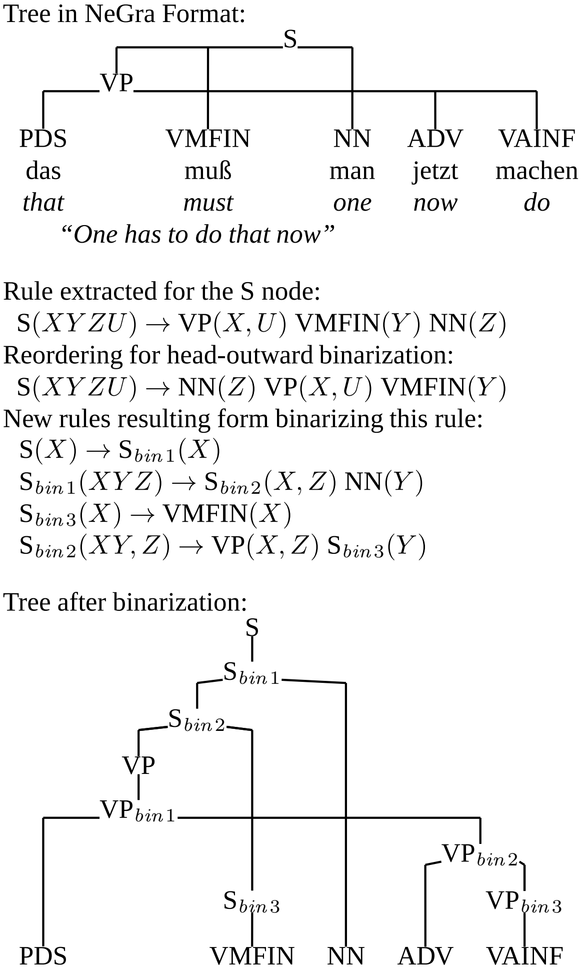
$S(XPUQ) \to A(X,U)C_1(P,Q)$
$C_1(YZ,VW) \to B(Y,V)C(Z,W)$
$A(aX,aY) \to A(X,Y) \qquad A(a,a) \to \varepsilon$
$B(bX,bY) \to B(X,Y) \qquad B(b,b) \to \varepsilon$
$C(cX,cY) \to C(X,Y) \qquad C(c,c) \to \varepsilon$

Figure 8: Sample binarization of a LCFRS

In LCFRS, in contrast to CFG, the order of the RHS elements of a rule does not matter for the result of a derivation. Therefore, we can reorder the RHS of a rule before binarizing it. In practice, we perform a head-outward binarization where the head is the lowest subtree. It is extended by adding first all sisters to its left and then all sisters to its right. Consequently, before binarizing we reorder the RHS of the rules extracted from the treebank such that first, all elements to the right of the head are listed in reverse order, then all elements to the left of the head in their original order and then the head itself.[4]

Furthermore, we add additional unary rules when introducing the highest new binarization non-terminal and when deriving the head. This allows for an additional factorization that has proved itself useful in parsing. Fig. 9 shows a sample binarization of a tree in the NeGra format.

For the LCFRSs extracted from dependency treebanks, we perform the same type of binarization. The head daughter is always the daughter with the POS tag non-terminal.


Tree in NeGra Format:

Rule extracted for the S node:
S$(XYZU) \to$ VP(X,U) VMFIN(Y) NN(Z)
Reordering for head-outward binarization:
S$(XYZU) \to$ NN(Z) VP(X,U) VMFIN(Y)
New rules resulting form binarizing this rule:
S$(X) \to$ S$_{bin1}(X)$
S$_{bin1}(XYZ) \to$ S$_{bin2}(X,Z)$ NN(Y)
S$_{bin3}(X) \to$ VMFIN(X)
S$_{bin2}(XY,Z) \to$ VP(X,Z) S$_{bin3}(Y)$

Tree after binarization:

Figure 9: Sample binarization

3.5 Markovization

Markovization (Collins, 1999) is achieved by introducing only a single new non-terminal for the new rules introduces during binarization and adding vertical and horizontal context from the original trees to each occurrence of this new non-terminal. As vertical context, we add the first v labels on the path from the root node of the tree that we want to binarize to the root of the entire treebank tree. The vertical context is actually collected during grammar extraction and then taken into account during binarization of the rules. As horizontal context, during binarization of a rule $A(\vec{\alpha}) \to A_0(\vec{\alpha_0}) \dots A_m(\vec{\alpha_m})$, for the new non-terminal that comprises the RHS elements $A_i \dots A_m$ (for some $1 \leq i \leq m$), we add the first h elements of $A_i, A_{i-1}, \dots, A_0$.

Figure 10 shows an example of a markovization of the tree from Fig. 9 with $v = 1$ and $h = 2$. Here, the superscript is the vertical context and the subscript the horizontal context of the new non-terminal X.

The probabilities are then computed based on the

[4] One could also add first the sisters to the right and then the ones to the left which is what Klein and Manning (2003) do. However, this has only a negligible effect on parsing results.

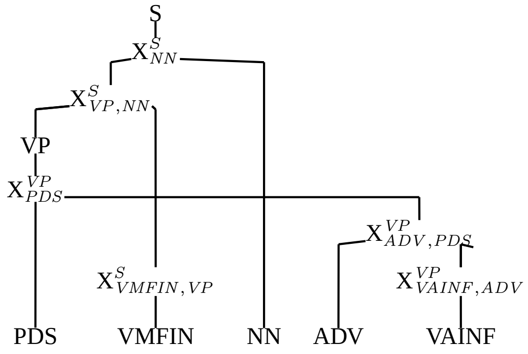


Figure 10: Sample Markovization with $v = 1, h = 2$

frequencies of rules in the treebank, using a Maximum Likelihood estimator (MLE). Such an estimation has been used before (Kato et al., 2006).

3.6 Properties of the Grammars

	unbin.	bin.	bin. lab.
NeGra LCFRS	13,858	16,904	4,142
S sp.	13,953	17,033	4,179
VP sp.	14,050	18,362	4,952
S ∘ *VP* sp.	14,144	18,503	4,995
NeGra PCFG	12,886	15,563	3,898
NeGra Dep.	18,520	68,847	49,085
PDT	12,841	38,312	24,119

Table 1: PLCFRSs extracted from training sets

Tab. 1 contains the properties of the grammars: The number of rules in the unbinarized grammar, the number of rules in the binarized and markovized grammar and the number of labels (including POS tags) in the binarized and markovized grammar.

4 Experiments

We run the parser on all data sets described above, providing gold POS tags in the input. In order to relate the costs of parsing for each of the data sets, we include Fig. 11, which shows the numbers of produced items for each data set.

4.1 Constituency Parsing

For the evaluation of the constituent parses, we use an EVALB-style metric. For a tree over a string w, a single constituent is represented by a tuple $\langle A, \vec{\rho} \rangle$ with A a node label and $\vec{\rho} \in (Pos(w) \times Pos(w))^{dim(A)}$.[5] We compute precision, recall and F_1 based on these tuples from gold and parsed test data. Despite the shortcomings of such a measure (Rehbein and van Genabith, 2007, e.g.), it still allows to some extent a comparison to previous work in PCFG parsing. For a more detailed evaluation of NeGra PLCFRS constituent parsing results, see Maier (2010). We use the previously successfully employed markovization settings $v = 2$ and $h = 1$ for all constituent experiments.

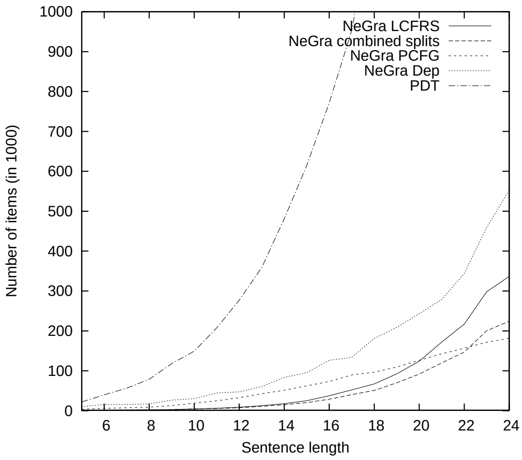


Figure 11: Number of items

	LCFRS	w/ category splits			PCFG
		VP	*S*	*S* ∘ *VP*	
LP	73.24	73.24	73.98	74.02	74.10
LR	73.56	73.91	74.17	74.45	74.83
LF_1	73.40	73.57	74.07	74.24	74.46
UP	77.12	76.98	77.47	77.39	78.08
UR	77.46	77.68	77.68	77.84	78.84
UF_1	77.29	77.33	77.58	77.62	78.46

Table 2: NeGra constituent parsing

	K 05	**here**	R&M 08	P&K 07
Labeled F_1	69.94	**74.46**	77.20	80.1

Table 3: Previous NeGra PCFG parsing

Tab. 2 presents the constituent parsing results for both data sets with (LCFRS) and without (PCFG) crossing branches. For the sake of comparison, we

[5] Note that our metric is equivalent to the corresponding PCFG metric for $dim(A) = 1$.

report PCFG parsing results from the literature[6] in Tab. 3, namely for PCFG parsing with a plain vanilla treebank grammar (Kübler, 2005), for PCFG parsing with the Stanford parser (Rafferty and Manning, 2008) (markovization as in our parser), and for the current state-of-the-art, namely PCFG parsing with a latent variable model (Petrov and Klein, 2007). We see that the LCFRS parser output (which contains more information than the output of a PCFG parser) is competitive. The PCFG (1-LCFRS) parsing results are even closer to the ones of current systems. Recall that these are just first results, much optimization potential is left.

Before we evaluate the experiments with category splits, we replace all split labels in the parser output with the corresponding original labels. The results show that the category splits are indeed beneficial, both in terms of output quality and speed (cf. the number of produced items in Fig. 11). We will continue to explore this approach using an automated split-and-merge approach in the style of Ule (2003).

Literature on parsing with discontinuous constituents is sparse. Hall and Nivre (2008) reconstruct the crossing branches of NeGra. They parse a (non-projective) dependency version of the German TIGER treebank (which follows the same annotation principles as NeGra) and convert the result back to constituents. For sentences up to length 40 and perfect tagging, they report a labeled F_1 of 70.79. While not directly comparable to our result, we still lie in the same range. Plaehn (2004) also reports results for direct parsing of the discontinuous constituents using Probabilistic Discontinuous Phrase Structure Grammar (DPSG). See Maier (2010) for details.

4.2 Dependency Parsing

In this section, we present the first grammar-based non-projective dependency parsing results. As Kuhlmann and Satta (2009) note, the principal advantage of grammar-based non-projective dependency parsing is that edge probabilities can be fine-tuned while staying polynomially parseable. This is not possible in the Maximum Spanning Tree approach (McDonald and Satta, 2007). For comparison of our dependency parser output, we report labeled and unlabeled attachment score and completely correct graphs (punctuation included). As markovization setting for the PDT set, we choose $v = 2$ and $h = \infty$.

	NeGra		PDT	
	Grammar	MST	Grammar	MST
UAS	78.98	87.96	51.44	76.01
LAS	71.84	82.62	67.09	40.54
UComp	32.65	42.16	14.92	28.92
LComp	25.03	29.56	9.46	17.23

Table 4: Dependency parsing

Tab. 4 contain the dependency parsing results for our parser and the MSTParser (McDonald et al., 2005) for NeGra and PDT. As an overall observation, the fact that our results are far off the MSTParser's results is certainly surprising. The most prominent difference between the NeGra and the PDT set is the number of edge labels. It leads to grammars with 922 non-terminals for NeGra and only 51 non-terminals for the PDT. While the MSTParser is almost not affected by this difference, the fact that our NeGra results are superior to our PDT results allows the conclusion that for grammar-based parsing, more informative edges labels are an advantage. This is also confirmed by the higher number of items for PDT (cf. Fig. 11). We expect therefore that automated category splitting will lead to a large improvement. This will be tackled in future work.

5 Conclusion

We have presented a parser for Probabilistic Linear Context-Free Rewriting Systems and have used it to parse NeGra, a German constituency treebank with directly annotated crossing branches. Furthermore, we have applied our parser to a dependency version of NeGra, and to the Prague Dependency Treebank. To our knowledge, grammar-based parsing of non-projective dependencies has not been attempted before. Experiments have shown that PLCFRS parsing is feasible and that the results for constituency parsing lie in the vicinity of the state-of-the-art.

In future work, we will concentrate particularly on the optimization potential for the parsing results. Especially dependency parsing offers many possibilities of optimization.

[6] The results from the literature were obtained on sentences longer than 25 words and would most likely be better for our sentence length.

References

Pierre Boullier. 1998. A Proposal for a Natural Language Processing Syntactic Backbone. Technical Report 3342, INRIA.

David Chiang. 2003. Statistical parsing with an automatically extracted Tree Adjoining Grammar. In *Data-Oriented Parsing*. CSLI Publications.

Michael Collins, Jan Hajič, Lance Ramshaw, and Christoph Tillmann. 1999. A statistical parser for Czech. In *Proceedings of ACL 1999*.

Michael Collins. 1999. *Head-driven statistical models for natural language parsing*. Ph.D. thesis, University of Pennsylvania.

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 NAACL-HLT*, Boulder, Colorado.

Jan Hajič, Alena Böhmová, Eva Hajičová, and Barbora Vidová-Hladká. 2000. The Prague Dependency Treebank: A Three-Level Annotation Scenario. In A. Abeillé, editor, *Treebanks: Building and Using Parsed Corpora*. Amsterdam:Kluwer.

Johan Hall and Joakim Nivre. 2008. Parsing discontinuous phrase structure with grammatical functions. In *Proceedings of GoTAL 2008*.

Mark Johnson. 2002. A simple pattern-matching algorithm for recovering empty nodes and their antecedents. In *Proceedings of ACL 2002*.

Laura Kallmeyer and Wolfgang Maier. 2010. Data-driven parsing with probabilistic linear context-free rewriting systems. Unpublished manuscript.

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

Yuki Kato, Hiroyuki Seki, and Tadao Kasami. 2006. Stochastic multiple context-free grammar for rna pseudoknot modeling. In *Proceedings of TAG+8*.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of ACL 2003*.

Sandra Kübler and Gerald Penn, editors. 2008. *Proceedings of the Workshop on Parsing German at ACL 2008*.

Sandra Kübler. 2005. How do treebank annotation schemes influence parsing results? Or how not to compare apples and oranges. In *Proceedings of RANLP 2005*.

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

Roger Levy and Christopher D. Manning. 2004. Deep dependencies from context-free statistical parsers: correcting the surface dependency approximation. In *Proceedings of ACL 2004*.

Dekang Lin. 1995. A dependency-based method for evaluating broad-coverage parsers. In *Proceedings of IJCAI 1995*.

Wolfgang Maier and Anders Søgaard. 2008. Treebanks and mild context-sensitivity. In *Proceedings of Formal Grammar 2008*.

Wolfgang Maier. 2010. Direct parsing of discontinuous constituents in German. In *Proceedings of the First Workshop on Statistical Parsing of Morphologically Rich Languages (SPMRL 2010) at NAACL 2010*.

Ryan McDonald and Giorgio Satta. 2007. On the complexity of non-projective data-driven dependency parsing. In *Proceedings of IWPT 2007*.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of HLT/EMNLP 2005*.

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

J. Nivre, J. Hall, J. Nilsson, A. Chanev, G. Eryigit, S. Kübler, S. Marinov, and E. Marsi. 2007. MaltParser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2).

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

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

Anna Rafferty and Christopher D. Manning, 2008. *Parsing Three German Treebanks: Lexicalized and Unlexicalized Baselines*. In Kübler and Penn (2008).

Ines Rehbein and Josef van Genabith. 2007. Evaluating evaluation measures. In *Proceedings of NODALIDA 2007*.

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

Wojciech Skut, Brigitte Krenn, Thorten Brants, and Hans Uszkoreit. 1997. An Annotation Scheme for Free Word Order Languages. In *Proceedings of ANLP*.

Tylman Ule. 2003. Directed treebank refinement for PCFG parsing. In *Proceedings of TLT 2003*.

Yannick Versley. 2005. Parser evaluation across text types. In *Proceedings of TLT 2005*.

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