Data-Driven Parsing with Probabilistic Linear Context-Free Rewriting Systems

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

Abstract

This paper presents a first efficient implementation of a weighted deductive CYK parser for Probabilistic Linear Context-Free Rewriting Systems (PLCFRS), together with context-summary estimates for parse items used to speed up parsing. LCFRS, an extension of CFG, can describe discontinuities both in constituency and dependency structures in a straightforward way and is therefore a natural candidate to be used for data-driven parsing. We evaluate our parser with a grammar extracted from the German NeGra treebank. Our experiments show that data-driven LCFRS parsing is feasible with a reasonable speed and yields output of competitive quality.

1 Introduction

Data-driven parsing has largely been dominated by Probabilistic Context-Free Grammar (PCFG). The use of PCFG is tied to the annotation principles of popular 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. In contrast, some other treebanks, such as the German NeGra and TIGER treebanks allow annotation with crossing branches (Skut et al., 1997). Non-local dependencies can then be expressed directly by grouping all dependent elements under a single node.

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, while in NeGra, resp. TIGER, tree transformations are applied which resolve the crossing branches (Kübler, 2005; Boyd, 2007, e.g.). Especially for these treebanks, such a transformation is questionable, since it is non-reversible and implies information loss.

Some research has gone into incorporating non-local information into data-driven parsing. Levy and Manning (2004) distinguish three approaches: 1. Non-local information can be incorporated directly into the PCFG model (Collins, 1999), or can be reconstructed in a post-processing step after PCFG parsing (Johnson, 2002; Levy and Manning, 2004). 2. Non-local information can be incorporated into complex labels (Hockenmaier, 2003). 3. A formalism can be used which accommodates the direct encoding of non-local information (Plaehn, 2004). This paper pursues the third approach.

Our work is motivated by the following recent developments: Linear Context-Free Rewriting Systems (LCFRS) (Vijay-Shanker et al., 1987) have been established as a candidate for modeling both discontinuous constituents and non-projective dependency trees as they occur in treebanks (Kuhlmann and Satta, 2009; Maier and Lichte, 2009). LCFRS extend CFG such that non-terminals can span tuples of possibly non-

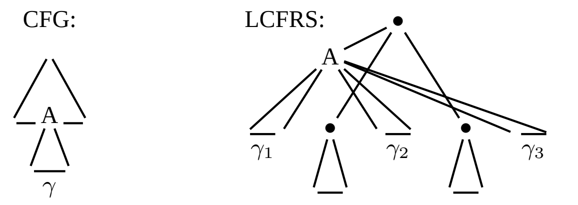

Figure 1: Different domains of locality

adjacent strings (see Fig. 1). PCFG techniques, such as Best-First Parsing (Charniak and Caraballo, 1998), Weighted Deductive Parsing (Nederhof, 2003) and A* parsing (Klein and Manning, 2003a), can be transferred to LCFRS. Finally, German has attracted the interest of the parsing community due to the challenges arising from its frequent discontinuous constituents (Kübler and Penn, 2008).

We bring together these developments by presenting a parser for probabilistic LCFRS. While parsers for subclasses of PLCFRS have been presented before (Kato et al., 2006), to our knowledge, our parser is the first for the entire class of PLCFRS. We have already presented an application of the parser on constituency and dependency treebanks together with an extensive evaluation (Maier, 2010; Maier and Kallmeyer, 2010). This article is mainly dedicated to the presentation of several methods for context summary estimation of parse items, and to an experimental evaluation of their usefulness. The estimates either act as figures-of-merit in a best-first parsing context or as estimates for A* parsing. Our evaluation shows that while our parser achieves a reasonable speed already without estimates, the estimates lead to a great reduction of the number of produced items, all while preserving the output quality.

Sect. 2 and 3 of the paper introduce probabilistic LCFRS and the parsing algorithm. Sect. 4 presents different context summary estimates. In Sect. 5, the implementation and evaluation of the work is discussed.

2 Probabilistic LCFRS

LCFRS are an extension of CFG where the non-terminals can span not only single strings but, instead, tuples of strings. We will notate LCFRS with the syntax of *simple Range Concatenation Grammars* (SRCG) (Boullier, 1998), a formalism that is equivalent to LCFRS.

A 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: $N \to \mathbb{N}$ 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 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 $r \in P$, 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
$$A(\alpha_1, \dots, \alpha_{dim(A)}) \to A_1(X_1^{(1)}, \dots, X_{dim(A_1)}^{(1)}) \cdots A_m(X_1^{(m)}, \dots, X_{dim(A_m)}^{(m)})$$
and all $\vec{\tau_i} \in yield(A_i)$ for $1 \leq i \leq m$, $\langle f(\alpha_1), \dots, f(\alpha_{dim(A)})\rangle \in yield(A)$ 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. 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 *ordered* 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 $p : P \to [0..1]$ a function such that for all $A \in N$: $\Sigma_{A(\vec{x}) \to \vec{\Phi} \in P} p(A(\vec{x}) \to \vec{\Phi}) = 1$.

3 The CYK Parser

We use a probabilistic version of the CYK parser from (Seki et al., 1991), applying techniques of weighted deductive parsing (Nederhof, 2003).

LCFRS 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 and do not contain ε components in their LHS. 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 $A(a) \to \varepsilon$ with A a POS tag and $a \in T$ or of the form $A(\vec{\alpha}) \to B(\vec{x})$ or $A(\vec{\alpha}) \to B(\vec{x})C(\vec{y})$ where $\vec{\alpha} \in (V^+)^{dim(A)}$, i.e., only the rules for POS tags contain terminals in their LHSs.

For every $w \in T^*$, where $w = w_1 \dots w_n$ with $w_i \in T$ for $1 \leq i \leq n$, we define: $Pos(w) := \{0, \dots, n\}$. A pair $\langle l, r\rangle \in Pos(w) \times Pos(w)$ with $l \leq r$ is a *range* in w. Its *yield* $\langle l, r\rangle(w)$ is the string $w_{l+1} \dots w_r$. The yield $\vec{\rho}(w)$ of a vector of ranges $\vec{\rho}$ is the vector of the yields of the single ranges. For two ranges $\rho_1 = \langle l_1, r_1\rangle, \rho_2 = \langle l_2, r_2\rangle$: 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 $p : A(\alpha_1, \dots, \alpha_{dim(A)}) \to B(X_1, \dots, X_{dim(B)})C(Y_1, \dots, X_{dim(C)})$ we now extend the composition function f to ranges, given an input w: for all range vectors $\vec{\rho_B}$ and $\vec{\rho_C}$ of dimensions $dim(B)$ and $dim(C)$ respectively, $f_r(\vec{\rho_B}, \vec{\rho_C}) = \langle g(\alpha_1), \dots, g(\alpha_{dim(A)})\rangle$ 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 $x, y \in V^+$. $p : A(f_r(\vec{\rho_B}, \vec{\rho_C})) \to B(\vec{\rho_B})C(\vec{\rho_C})$ is then called an *instantiated rule*.

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 $\vec{\rho}$ characterizes the span of A. We specify the set of weighted parse items via the deduction rules in Fig. 2. Our parser performs a weighted deductive parsing (Nederhof, 2003), 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.

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 $\mathcal{A} \neq \emptyset$
  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

4 Outside Estimates

In order to speed up parsing, we add an estimate of the log of the outside probabilities of the items to their weights in the agenda. All our outside estimates are *admissible* (Klein and Manning, 2003a) which means that they never underestimate the actual outside probability of an item. However, most of them are not monotonic. In other words, it can happen that we deduce an item I_2 from an item I_1 where the weight of I_2 is greater than the weight of I_1. The parser can therefore end up in a local maximum that is not the global maximum we are searching for. In other words, our outside weights are only *figures of merit* (FOM). Only for the full SX estimate, the monotonicity is guaranteed and we can do true A* parsing as described in (Klein and Manning, 2003a) that always finds the best parse.

All outside estimates are computed for a certain maximal sentence length len_{max}.

POS tags: $\dfrac{}{0 : [A, \langle 1 \rangle]}$ A a POS tag

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

Binary: $\dfrac{in_B : [B, \vec{l}_B], in_C : [C, \vec{l}_C]}{in_B + in_C + log(p) : [A, \vec{l}_A]}$

where $p : A(\vec{\alpha_A}) \rightarrow B(\vec{\alpha_B})C(\vec{\alpha_C}) \in P$ and the following holds: we define $\mathcal{B}(i)$ as $\{1 \leq j \leq dim(B) \,|\, \vec{\alpha_B}(j)$ occurs in $\vec{\alpha_A}(i)\}$ and $\mathcal{C}(i)$ as $\{1 \leq j \leq dim(C) \,|\, \vec{\alpha_C}(j)$ occurs in $\vec{\alpha_A}(i)\}$. Then for all i, $1 \leq i \leq dim(A)$: $\vec{l}_A(i) = \Sigma_{j \in \mathcal{B}(i)} \vec{l}_B(j) + \Sigma_{j \in \mathcal{C}(i)} \vec{l}_C(j)$.

Figure 4: Inside estimate

Axiom : $\dfrac{}{0 : [S, \langle 0, len, 0 \rangle]}$ $1 \leq len \leq len_{max}$

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

Binary-right:

$$\frac{w : [X, \vec{l}_X]}{w + in(A, \vec{l'}_A) + log(p) : [B, \vec{l}_B]}$$

Binary-left:

$$\frac{w : [X, \vec{l}_X]}{w + in(B, \vec{l'}_B) + log(p) : [A, \vec{l}_A]}$$

where, for both rules, there is an instantiated rule $p : X(\vec{\rho}) \rightarrow A(\vec{\rho_A})B(\vec{\rho_B})$ such that $\vec{l}_X = l_{out}(\rho)$, $\vec{l}_A = l_{out}(\rho_A), \vec{l'}_A = l_{in}(\rho_A), \vec{l}_B = l_{out}(\rho_B, \vec{l}_B = l_{in}(\rho_B$.

Figure 5: Full SX estimate top-down

4.1 Full SX estimate

The full SX estimate, for a given sentence length n, is supposed to give the minimal costs (maximal probability) of completing a category X with a span ρ into an S with span $\langle\langle 0, n \rangle\rangle$.

For the computation, we need an estimate of the inside probability of a category C with a span ρ, regardless of the actual terminals in our input. This inside estimate is computed as shown in Fig. 4. Here, we do not need to consider the number of terminals outside the span of C (to the left or right or in the gaps), they are not relevant for the inside probability. Therefore the items have the form $[A, \langle l_1, \dots, l_{dim(A)} \rangle]$, where A is a non-terminal and l_i gives the length of its ith component. It holds that $\Sigma_{1 \leq i \leq dim(A)} l_i \leq len_{max} - dim(A) + 1$.

A straight-forward extension of the CFG algorithm from (Klein and Manning, 2003a) for computing the SX estimate is given in Fig. 5. For a given range vector $\rho = \langle\langle l_1, r_1 \rangle, \dots, \langle l_k, r_k \rangle\rangle$ and a sentence length n, we define its *inside length vector* $l_{in}(\rho)$ as $\langle r_1 - l_1, \dots, r_k - l_k \rangle$ and its *outside length vector* $l_{out}(\rho)$ as $\langle l_1, r_1 - l_1, l_2 - r_1, \dots, l_k - r_{k-1}, r_k - l_k, n - r_k \rangle$.

This algorithm has two major problems: Since it proceeds top-down, in the *Binary* rules, we must compute all splits of the antecedent X span into the spans of A and B which is very expensive. Furthermore, for a category A with a certain number of terminals in the components and the gaps, we compute the lower part of the outside estimate several times, namely for every combination of number of terminals to the left and to the right (first and last element in the outside length vector). In order to avoid these problems, we now abstract away from the lengths of the part to the left and the right, modifying our items such as to allow a bottom-up strategy.

The idea is to compute the weights of items representing the derivations from a certain lower C up to some A (C is a kind of "gap" in the yield of A) while summing up the inside costs of off-spine nodes and the log of the probabilities of the corresponding rules. We use items $[A, C, \rho_A, \rho_C, shift]$ where $A, C \in N$ and ρ_A, ρ_C are range vectors, both with a first component starting at position 0. The integer $shift \leq len_{max}$ tells us how many positions to the right the C span is shifted, compared to the starting position of the A. ρ_A and ρ_C represent the spans of C and A while disregarding the number of terminals to the left the right. I.e., only the lengths of the components and of the gaps are encoded. This means in particular that the length n of the sentence does not play a role here. The right boundary of the last range in the vectors is limited to len_{max}. For any $i, 0 \leq i \leq len_{max}$, and any range vector ρ, we define $shift(\rho, i)$ as the range vector one obtains from adding i to all range boundaries in ρ and $shift(\rho, -i)$ as the range vector one obtains from subtracting i from all boundaries in ρ.

The weight of $[A, C, \rho_A, \rho_C, i]$ estimates the costs for completing a C tree with yield ρ_C into an A tree with yield ρ_A such that, if the span of A starts at position j, the span of C starts at position $i + j$. Fig. 6 gives the computation. The value of $in(A, \vec{l})$ is the inside estimate of $[A, \vec{l}]$.

The SX-estimate for some predicate C with

POS tags: $\dfrac{}{0 : [C, C, \langle 0,1\rangle, \langle 0,1\rangle, 0]}$ C a POS tag

Unary: $\dfrac{0 : [B, B, \rho_B, \rho_B, 0]}{log(p) : [A, B, \rho_B, \rho_B, 0]}$ $p : A(\vec{\alpha}) \rightarrow B(\vec{\alpha}) \in P$

Binary-right:

$$\dfrac{0 : [A, A, \rho_A, \rho_A, 0], 0 : [B, B, \rho_B, \rho_B, 0]}{in(A, l(\rho_A)) + log(p) : [X, B, \rho_X, \rho_B, i]}$$

Binary-left:

$$\dfrac{0 : [A, A, \rho_A, \rho_A, 0], 0 : [B, B, \rho_B, \rho_B, 0]}{in(B, l(\rho_B)) + log(p) : [X, A, \rho_X, \rho_A, 0]}$$

where i is such that for $shift(\rho_B, i) = \rho'_B$ $p : X(\rho_X) \rightarrow A(\rho_A)B(\rho'_B)$ is an instantiated rule.

Starting sub-trees with larger gaps:

$$\dfrac{w : [B, C, \rho_B, \rho_C, i]}{0 : [B, B, \rho_B, \rho_B, 0]}$$

Transitive closure of sub-tree combination:

$$\dfrac{w_1 : [A, B, \rho_A, \rho_B, i], w_2 : [B, C, \rho_B, \rho_C, j]}{w_1 + w_2 : [A, C, \rho_A, \rho_C, i + j]}$$

Figure 6: Full SX estimate bottom-up

span ρ where i is the left boundary of the first component of ρ and with sentence length n is then given by the maximal weight of $[S, C, \langle 0, n\rangle, shift(-i, \rho), i]$. Among our estimates, the full SX estimate is the only one that is monotonic and that allows for true A* parsing.

4.2 SX with Left, Gaps, Right, Length

A problem of the previous estimate is that with a large number of non-terminals the computation of the estimate requires too much space. Our experiments have shown that for treebank parsing where we have, after binarization and markovization, appr. 12,000 non-terminals, its computation is not feasible. We therefore turn to simpler estimates with only a single non-terminal per item. We now estimate the outside probability of a non-terminal A with a span of a length *length* (the sum of the lengths of all the components of the span), with *left* terminals to the left of the first component, *right* terminals to the right of the last component and *gaps* terminals in between the components of the A span, i.e., filling the gaps. Our items have the form $[X, len, left, right, gaps]$ with $X \in N$, $len + left + right + gaps \leq len_{max}$, $len \geq dim(X)$, $gaps \geq dim(X) - 1$.

Let us assume that, in the rule $X(\vec{\alpha}) \rightarrow A(\vec{\alpha_A})B(\vec{\alpha_B})$, when looking at the vector $\vec{\alpha}$, we have $left_A$ variables for A-components preceding the first variable of a B component, $right_A$ variables for A-components following the last variable of a B component and $right_B$ variables for B-components following the last variable of a A component. (In our grammars, the first LHS argument always starts with the first variable from A.) Furthermore, $gaps_A = dim(A) - left_A - right_A$, $gaps_B = dim(B) - right_B$.

Axiom : $\dfrac{}{0 : [S, len, 0, 0, 0]}$ $1 \leq len \leq len_{max}$

Unary: $\dfrac{w : [X, len, l, r, g]}{w + log(p) : [A, len, l, r, g]}$

where $p : X(\vec{\alpha}) \rightarrow A(\vec{\alpha}) \in P$.

Binary-right:

$$\dfrac{w : [X, len, l, r, g]}{w + in(A, len - len_B) + log(p) : [B, len_B, l_B, r_B, g_B]}$$

Binary-left:

$$\dfrac{w : [X, len, l, r, g]}{w + in(B, len - len_A) + log(p) : [A, len_A, l_A, r_A, g_A]}$$

where, for both rules, $p : X(\vec{\alpha}) \rightarrow A(\vec{\alpha_A})B(\vec{\alpha_B}) \in P$.

Figure 7: SX with length, left, right, gaps

POS tags: $\dfrac{}{0 : [A, 1]}$ A a POS tag

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

Binary: $\dfrac{in_B : [B, l_B], in_C : [C, l_C]}{in_B + in_C + log(p) : [A, l_B + l_C]}$

where either $p : A(\vec{\alpha_A}) \rightarrow B(\vec{\alpha_B})C(\vec{\alpha_C}) \in P$ or $p : A(\vec{\alpha_A}) \rightarrow C(\vec{\alpha_C})B(\vec{\alpha_B}) \in P$.

Figure 8: Inside estimate with total span length

Fig. 7 gives the computation of the estimate. The following side conditions must hold: For *Binary-right* to apply, the following constraints must be satisfied: a) $len + l + r + g = len_B + l_B + r_B + g_B$, b) $l_B \geq l + left_A$, c) if $right_A > 0$, then $r_B \geq r + right_A$, else ($right_A = 0$), $r_B = r$, d) $g_B \geq gaps_A$. Similarly, for *Binary-left* to apply, the following constraints must be satisfied: a) $len + l + r + g = len_A + l_A + r_A + g_A$, b) $l_A = l$, c) if $right_B > 0$, then $r_A \geq r + right_B$, else ($right_B = 0$), $r_A = r$ d) $g_A \geq gaps_B$.

The value $in(X, l)$ for a non-terminal X and a length l, $0 \leq l \leq len_{max}$ is an estimate of the probability of an X category with a span of length l. Its computation is specified in Fig. 8.

The SX-estimate for a sentence length n and for some predicate C with a range characterized by $\vec{\rho} = \langle\langle l_1, r_1\rangle, \ldots, \langle l_{dim(C)}, r_{dim(C)}\rangle\rangle$ where $len = \Sigma_{i=1}^{dim(C)}(r_i - l_i)$ and $r = n - r_{dim(C)}$ is then given by the maximal weight of the item $[C, len, l_1, r, n - len - l_1 - r]$.

Axiom : $\frac{}{0 : [S, len, 0, 0]}\ 1 \leq len \leq len_{max}$

Unary: $\frac{w : [X, len, lr, g]}{w + log(p) : [A, len, lr, g]}$

where $p : X(\vec{\alpha}) \to A(\vec{\alpha}) \in P$.

Binary-right:

$$\frac{w : [X, len, lr, g]}{w + in(A, len - len_B) + log(p) : [B, len_B, lr_B, g_B]}$$

Binary-left:

$$\frac{w : [X, len, lr, g]}{w + in(B, len - len_A) + log(p) : [A, len_A, lr_A, g_A]}$$

where, for both rules, $p : X(\vec{\alpha}) \to A(\vec{\alpha_A})B(\vec{\alpha_B}) \in P$.

Figure 9: SX estimate with length, LR, gaps

4.3 SX with LR, Gaps, Length

In order to further decrease the space complexity, we can simplify the previous estimate by subsuming the two lengths *left* and *right* in a single length lr. I.e., the items now have the form $[X, len, lr, gaps]$ with $X \in N$, $len + lr + gaps \leq len_{max}$, $len \geq dim(X)$, $gaps \geq dim(X) - 1$.

The computation is given in Fig. 9. Again, we define $left_A$, $gaps_A$, $right_A$ and $gaps_B$, $right_B$ for a rule $X(\vec{\alpha}) \to A(\vec{\alpha_A})B(\vec{\alpha_B})$ as above. The side conditions are as follows: For *Binary-right* to apply, the following constraints must be satisfied: a) $len + lr + g = len_B + lr_B + g_B$, b) $lr < lr_B$, and c) $g_B \geq gaps_A$. For *Binary-left* to apply, the following must hold: a) $len + lr + g = len_A + lr_A + g_A$, b) if $right_B = 0$ then $lr = lr_A$, else $lr < lr_A$ and c) $g_A \geq gaps_B$.

The SX-estimate for a sentence length n and for some predicate C with a span $\vec{\rho} = \langle\langle l_1, r_1\rangle, \ldots, \langle l_{dim(C)}, r_{dim(C)}\rangle\rangle$ where $len = \Sigma_{i=1}^{dim(C)}(r_i - l_i)$ and $r = n - r_{dim(C)}$ is then the maximal weight of $[C, len, l_1 + r, n - len - l_1 - r]$.

5 Evaluation

The goal of our evaluation of our parser is to show that, firstly, reasonable parser speed can be achieved and, secondly, the parser output is of promising quality.

5.1 Data

Our data source is the German NeGra treebank (Skut et al., 1997). In a preprocessing step, following common practice (Kübler and Penn, 2008), we attach punctuation (not included in the NeGra annotation) as follows: In a first pass, using heuristics, we attach punctuation as high as possible while avoiding to introduce new crossing branches. In a second pass, parentheses and quotation marks preferably attach to the same node. Grammatical function labels on the edges are discarded.

We create data sets of different sizes in order to see how the size of the training set relates to the gain using context summary estimates and to the output quality of the parser. The first set uses the first 4000 sentences and the second one all sentences of NeGra. Due to memory limitations, in both sets, we limit ourselves to sentences of a maximal length of 25 words. We use the first 90% of both sets as training set and the remaining 10% as test set. Tab. 1 shows the resulting sizes.

	NeGra-small		NeGra	
	training	test	training	test
size	2839	316	14858	1651

Table 1: Test and training sets

5.2 Treebank Grammar Extraction

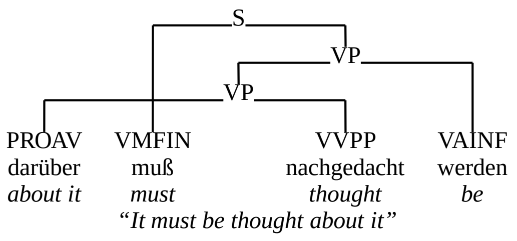

Figure 10: A sample tree from NeGra

As already mentioned, in NeGra, discontinuous phrases are annotated with crossing branches (see Fig. 10 for an example with two discontinuous VPs). Such discontinuities can be straightforwardly modelled with LCFRS. We use the algorithm from Maier and Søgaard (2008) to extract LCFRS rules from NeGra and TIGER. It first creates rules of the form $P(a) \to \varepsilon$ for each preterminal P dominating some terminal a. Then for all other nonterminals A_0 with the children $A_1 \cdots A_m$, a clause $A_0 \to A_1 \cdots A_m$ is created. The arguments of the $A_1 \cdots A_m$ are single variables where the number of arguments is the number of discontinuous parts in the yield of a predicate. The arguments of A_0 are concatenations of these variables that describe how the discontinuous parts of the yield of A_0 are obtained from the yields of its daughters. Different occurrences of the same non-terminal, only with different fan-outs, are distinguished by corresponding subscripts. Note that this extraction algorithm yields only *monotone* LCFRS (equivalent to ordered simple RCG). See Maier and Søgaard (2008) for further details. For Fig. 10, we obtain for instance the rules in Fig. 11.

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

Figure 11: LCFRS rules for the tree in Fig. 10

5.3 Binarization and Markovization

Before parsing, we binarize the extracted LCFRS. For this we first apply Collins-style head rules, based on the rules the Stanford parser (Klein and Manning, 2003b) uses for NeGra, to mark the resp. head daughters of all non-terminal nodes. Then, we reorder the RHSs such that the sequence γ of elements to the right of the head daughter is reversed and moved to the beginning of the RHS. We then perform a binarization that proceeds from left to right. The binarization works like the transformation into Chomsky Normal Form for CFGs in the sense that for RHSs longer than 2, we introduce a new non-terminal that covers the RHS without the first element. The rightmost new rule, which covers the head daughter, is binarized to unary. We do not use a unique new non-terminal for every new rule. Instead, to the new symbols introduced during the binarization (VP_{bin} in the example), a variable number of symbols from the vertical and horizontal context of the original rule is added in order to achieve *markovization*. Following the literature, we call the respective quantities v and h. For reasons of space we restrict ourselves here to the example in Fig. 12. Refer to Maier and Kallmeyer (2010) for a detailed presentation of the binarization and markovization.

The probabilities are then computed based on the rule frequencies in the transformed treebank, using a Maximum Likelihood estimator.

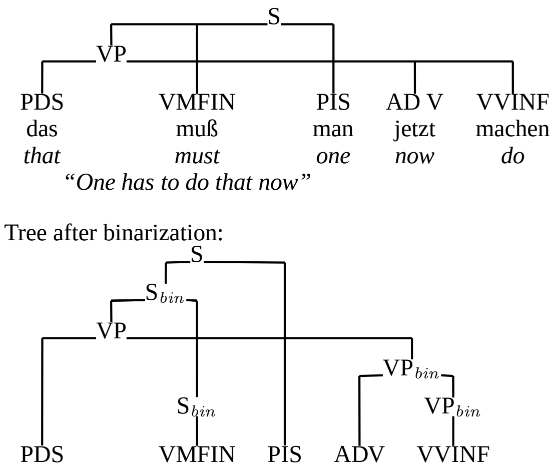

Figure 12: Sample binarization

5.4 Evaluation of Parsing Results

In order to assess the quality of the output of our parser, we choose an EVALB-style metric, i.e., we compare phrase boundaries. In the context of LCFRS, we compare sets of items $[A, \vec{\rho}]$ that characterize the span of a non-terminal A in a derivation tree. One set is obtained from the parser output, and one from the corresponding treebank trees. Using these item sets, we compute labeled and unlabeled recall (LR/UR), precision (LP/UP), and the F_1 measure (LF_1/UF_1). Note that if $k = 1$, our metric is identical to its PCFG equivalent.We are aware of the recent discussion about the shortcomings of EVALB. A discussion of this issue is presented in Maier (2010).

5.5 Experiments

In all experiments, we provide the parser with gold part-of-speech tags. For the experiments with *NeGra-small*, the parser is given the markovization settings $v = 1$ and $h = 1$. We compare the parser performance without estimates (OFF) with its performance with the estimates described in 4.2 (SIMPLE) and 4.3 (LR). Tab. 2 shows the results. Fig. 13 shows the number of items produced by the parser, indicating that the estimates have the desired effect of preventing unnecessary items from being produced. Note that it is even the case that the parser produces less items for the big set with LR than for the small set without estimate.

We can see that the estimates lead to a slightly

	OFF	SIMPLE	LR
UP/UR	72.29/72.40	70.49/71.81	72.10/72.60
UF_1	72.35	71.14	72.35
LP/LR	68.31/68.41	64.93/66.14	67.35/66.14
LF_1	68.36	65.53	65.53
Parsed	313 (99.05%)	313 (99.05%)	313 (99.05%)

Table 2: Experiments with *NeGra-small*

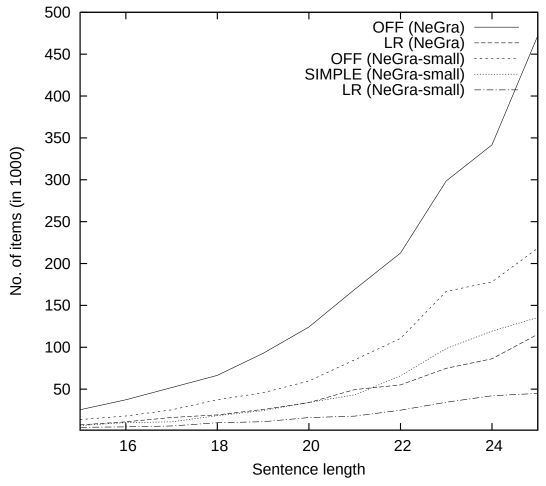

Figure 13: Items produced by the parser

lower F-score. However, while the losses in terms of F_1 are small, the gains in parsing time are substantial, as Fig. 13 shows.

Tab. 3 shows the results of experiments with *NeGra*, with the markovization settings $v = 2$ and $h = 1$ which have proven to be successful for PCFG parsing of NeGra (Rafferty and Manning, 2008). Unfortunately, due to memory restrictions, we were not able to compute SIMPLE for the large data set.[1] Resp. LR, the findings are comparable to the ones for *NeGra-short*. The speedup is paid with a lower F_1.

	OFF	LR
UP/UR	76.89/77.35	75.22/75.99
UF_1	77.12	75.60
LP/LR	73.03/73.46	70.98/71.70
LF_1	73.25	71.33
Parsed	1642 (99.45%)	1642 (99.45%)

Table 3: Experiments with *NeGra*

Our results are not directly comparable with PCFG parsing results, since LCFRS parsing is a harder task. However, since the EVALB metric coincides for constituents without crossing branches, in order to place our results in the context of previous work on parsing NeGra, we cite some of the results from the literature which were obtained using PCFG parsers[2]: Kübler (2005) (Tab. 1, plain PCFG) obtains 69.4, Dubey and Keller (2003) (Tab. 5, sister-head PCFG model) 71.12, Rafferty and Manning (2008) (Tab. 2, Stanford parser with markovization $v = 2$ and $h = 1$) 77.2, and Petrov and Klein (2007) (Tab. 1, Berkeley parser) 80.1. Plaehn (2004) obtains 73.16 Labeled F_1 using Probabilistic Discontinuous Phrase Structure Grammar (DPSG), albeit only on sentences with a length of up to 15 words. On those sentences, we obtain 81.27.

The comparison shows that our system delivers competitive results. Additionally, when comparing this to PCFG parsing results, one has to keep in mind that LCFRS parse trees contain non-context-free information about discontinuities. Therefore, a correct parse with our grammar is actually better than a correct CFG parse, evaluated with respect to a transformation of NeGra into a context-free treebank where precisely this information gets lost.

6 Conclusion

We have presented the first parser for unrestricted Probabilistic Linear Context-Free Rewriting Systems (PLCFRS), implemented as a CYK parser with weighted deductive parsing. To speed up parsing, we use context summary estimates for parse items. An evaluation on the NeGra treebank, both in terms of output quality and speed, shows that data-driven parsing using PLCFRS is feasible. Already in this first attempt with a straightforward binarization, we obtain results that are comparable to state-of-the-art PCFG results in terms of F_1, while yielding parse trees that are richer than context-free trees since they describe discontinuities. Therefore, our approach demonstrates convincingly that PLCFRS is a natural and tractable alternative for data-driven parsing which takes non-local dependencies into consideration.

[1]SIMPLE also proved to be infeasible to compute for the small set for the markovization settings $v = 2$ and $h = 1$ due to the greatly increased label set with this settings.

[2]Note that these results were obtained on sentences with a length of ≤ 40 words and that those parser possibly would deliver better results if tested on our test set.

References

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

Boyd, Adriane. 2007. Discontinuity revisited: An improved conversion to context-free representations. In *The Linguistic Annotation Workshop at ACL 2007*.

Charniak, Eugene and Sharon A. Caraballo. 1998. New figures of merit for best-first probabilistic chart parsing. *Computational Linguistics*, 24.

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

Dubey, Amit and Frank Keller. 2003. Probabilistic parsing for German using sisterhead dependencies. In *Proceedings of ACL*.

Gómez-Rodríguez, Carlos, Marco Kuhlmann, Giorgio Satta, and David Weir. 2009. Optimal reduction of rule length in linear context-free rewriting systems. In *Proceedings of NAACL-HLT*.

Hockenmaier, Julia. 2003. *Data and models for Statistical Parsing with Combinatory Categorial Grammar*. Ph.D. thesis, University of Edinburgh.

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

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

Klein, Dan and Christopher D. Manning. 2003a. A* Parsing: Fast Exact Viterbi Parse Selection. In *Proceedings of NAACL-HLT*.

Klein, Dan and Christopher D. Manning. 2003b. Fast exact inference with a factored model for natural language parsing. In *In Advances in Neural Information Processing Systems 15 (NIPS)*.

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

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

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

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

Maier, Wolfgang and Laura Kallmeyer. 2010. Discontinuity and non-projectivity: Using mildly context-sensitive formalisms for data-driven parsing. In *Proceedings of TAG+10*.

Maier, Wolfgang and Timm Lichte. 2009. Characterizing Discontinuity in Constituent Treebanks. In *Proceedings of Formal Grammar 2009*.

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

Maier, Wolfgang. 2010. Direct parsing of discontinuous constituents in german. In *Proceedings of the SPMRL workshop at NAACL HLT 2010*.

Marcus, Mitchell, 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 *Proceedings of HLT*.

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

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

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

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

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

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

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