Direct Parsing of Discontinuous Constituents in German

Wolfgang Maier
SFB 833, University of Tübingen
Nauklerstr. 35
72074 Tübingen, Germany
wmaier@sfs.uni-tuebingen.de

Abstract

Discontinuities occur especially frequently in languages with a relatively free word order, such as German. Generally, due to the long-distance dependencies they induce, they lie beyond the expressivity of Probabilistic CFG, i.e., they cannot be directly reconstructed by a PCFG parser. In this paper, we use a parser for Probabilistic Linear Context-Free Rewriting Systems (PLCFRS), a formalism with high expressivity, to directly parse the German NeGra and TIGER treebanks. In both treebanks, discontinuities are annotated with crossing branches. Based on an evaluation using different metrics, we show that an output quality can be achieved which is comparable to the output quality of PCFG-based systems.

1 Introduction

Languages with a rather free word order, like German, display discontinuous constituents particularly frequently. In (1), the discontinuity is caused by an extraposed relative clause.

(1) wieder treffen alle Attribute zu, die auch
again match all attributes VPART which also
sonst immer passen
otherwise always fit
'Again, the same attributes as always apply.'

Another language with a rather free word order is Bulgarian. In (2), the discontinuity is caused by topicalization.

(2) Химикали$_1$ аз купувам само евтини t$_1$
Pens$_1$ I buy only expensive t$_1$
'As for pens, I only buy expensive ones.'

In most constituency treebanks, sentence annotation is restricted to having the shape of trees without crossing branches, and the non-local dependencies induced by the discontinuities are modeled by an additional mechanism. In the Penn Treebank (PTB) (Marcus et al., 1994), e.g., this mechanism is a combination of special labels and empty nodes, establishing implicit additional edges. In the German TüBa-D/Z (Telljohann et al., 2006), additional edges are established by a combination of topological field annotation and special edge labels. As an example, Fig. 1 shows a tree from TüBa-D/Z with the annotation of (1). Note here the edge label ON-MOD on the relative clause which indicates that the subject of the sentence (*alle Attribute*) is modified.

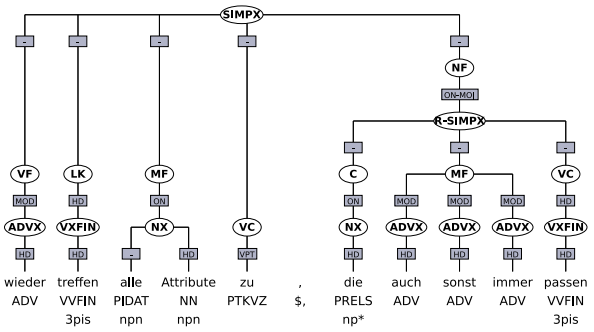

Figure 1: A tree from TüBa-D/Z

However, in a few other treebanks, such as the German NeGra and TIGER treebanks (Skut et al., 1997; Brants et al., 2002), crossing branches are allowed. This way, all dependents of a long-distance dependency can be grouped under a single node.

Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages, pages 58–66,
Los Angeles, California, June 2010.

Fig. 2 shows a tree from NeGra with the annotation of (3).

(3) Noch nie habe ich so viel gewählt
Yet never have I so much chosen
'Never have I had that much choice.'

Note the direct annotation of the discontinuous VP.

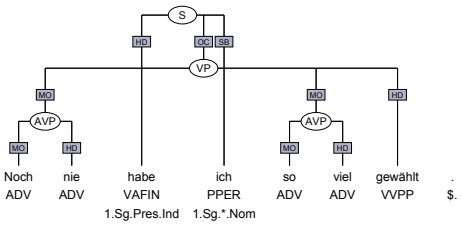

Figure 2: A tree from NeGra

Since in general, the annotation mechanisms for non-local dependencies lie beyond the expressivity of Context-Free Grammar, non-local information is inaccessible for PCFG parsing and therefore generally discarded. In NeGra/TIGER annotation, e.g., tree transformation algorithms are applied before parsing in order to resolve the crossing branches. See, e.g., Kübler et al. (2008) and Boyd (2007) for details. If one wants to avoid the loss of annotation information which is implied with such transformations, one possibility is to use a probabilistic parser for a formalism which is more expressive than CFG.

In this paper, we tackle the question if qualitatively good results can be achieved when parsing German with such a parser. Concretely, we use a parser for Probabilistic Linear Context-Free Rewriting Systems (PLCFRS) (Kallmeyer and Maier, 2010). LCFRS (Vijay-Shanker et al., 1987) are a natural extension of CFG in which a single non-terminal node can dominate more than one continuous span of terminals. We can directly interpret NeGra-style trees as its derivation structures, i.e., we can extract grammars without making further linguistic assumptions (Maier and Lichte, 2009) (see Sect. 2.3), as it is necessary for other formalisms such as Probabilistic Tree Adjoining Grammars (Chiang, 2003). Since the non-local dependencies are immediately accessible in NeGra and TIGER, we choose these treebanks as our data source. In order to judge parser output quality, we use four different evaluation types. We use an EVALB-style measure, adapted for LCFRS, in order to compare our parser to previous work on parsing German treebanks. In order to address the known shortcomings of EVALB, we perform an additional evaluation using the tree distance metric of Zhang and Shasha (1989), which works independently of the fact if there are crossing branches in the trees or not, and a dependency evaluation (Lin, 1995), which has also be applied before in the context of parsing German (Kübler et al., 2008). Last, we evaluate certain difficult phenomena by hand on TePaCoC (Kübler et al., 2008), a set of sentences hand-picked from TIGER. The evaluations show that with a PLCFRS parser, competitive results can be achieved.

The remainder of the article is structured as follows. In Sect. 2, we present the formalism, the parser, and how we obtain our grammars. In Sect. 3, we discuss the evaluation methods we employ. Sect. 4 contains our experimental results. Sect. 5 is dedicated to related work. Sect. 6 contains the conclusion and presents some possible future work.

2 A Parser for PLCFRS

2.1 Probabilistic Linear Context-Free Rewriting Systems

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 $G = (N, T, V, P, S)$ 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

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

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).

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. An LCFRS is called **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.

Borrowed from SRCG, we specify the language of an LCFRS based on the notion of ranges. For some input word $w = w_1 \cdots w_n$, a range is a pair $\langle i, j \rangle$ of integers with $0 \leq i \leq n$ denoting the substring $w_{i+1} \cdots w_j$. Note that a range denotes ε iff $i = j$. Only consecutive ranges can be concatenated into new ranges. We can replace the variables and terminals of the rewriting rules with ranges. E.g., $A(\langle g, h \rangle) \rightarrow B(\langle g+1, h-1 \rangle)$ is a replacement of the clause $A(aX_1b) \rightarrow B(X_1)$ if the input word w is such that $w_{g+1} = a$ and $w_h = b$. A rewriting rule in which all elements of all arguments have been consistently replaced by ranges is called an **instantiated rule**. A derivation is built by successively rewriting the LHSs of instantiated rules with its RHSs. The language $L(G)$ of some LCFRS G consists of all words $w = w_1 \cdots w_n$ for which it holds that there is a rule with the start symbol on the LHS which can be instantiated to $\langle 0, n \rangle$ and rewritten to ε.

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 $p : P \rightarrow [0..1]$ a function such that for all $A \in N$: $\Sigma_{A(\vec{x}) \rightarrow \vec{\Phi} \in P} p(A(\vec{x}) \rightarrow \vec{\Phi}) = 1$. There are possibly other ways to extend LCFRS with probabilities. This definition is supported by the fact that probabilistic MCFGs[1] have been defined in the same way (Kato et al., 2006).

[1]MCFGs are equivalent to LCFRSs and SRCGs (Boullier, 1998).

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

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

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 3: Weighted CYK deduction system

2.2 A CYK Parser for PLCFRS

We use the parser of Kallmeyer and Maier (2010). It is a probabilistic CYK parser (Seki et al., 1991), using the technique of weighted deductive parsing (Nederhof, 2003). While for symbolic parsing, other elaborate algorithms exist (Kallmeyer and Maier, 2009), for probabilistic parsing, CYK is a natural choice.

It is assumed for the parser that our LCFRSs are of rank 2 and do not contain rules where some of the LHS components are ε. Both assumptions can be made without loss of generality since every LCFRS can be binarized (Gómez-Rodríguez et al., 2009) and ε-components on LHS of rules can be removed (Boullier, 1998). We make the assumption that POS tagging is done before parsing. The POS tags are special non-terminals of fan-out 1. Consequently, the rules are either of the form $A(a) \rightarrow \varepsilon$ where A is 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.

The parser items have the form $[A, \vec{\rho}]$, with $A \in N$ and $\vec{\rho}$ a 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. 3.

Parsing time can be reduced by reordering the agenda during parsing such that those items are processed first which lead to a complete parse more quickly than others (Klein and Manning, 2003a). The parser uses for this purpose an admissible, but not monotonic estimate called **LR estimate**. It gives (relative to a sentence length) an estimate of the outside probability of some non-terminal A with a span of a certain length (the sum of the lengths of all the

components of the span), a certain number of terminals to the left of the first and to the right of the last component and a certain number of terminals $gaps$ in between the components of the A span, i.e., filling the gaps. A discussion of other estimates is presented at length in Kallmeyer and Maier (2010).

2.3 LCFRS for Modeling Discontinuities

We use the algorithm from Maier and Søgaard (2008) to extract LCFRS rules from our data sets. For all nonterminals A_0 with the children $A_1 \cdots A_m$ (i.e., for all non-terminals which are not preterminals), we create a clause $\psi_0 \rightarrow \psi_1 \cdots \psi_m$ with ψ_i, $0 \leq i \leq m$, labeled A_i. The arguments of each ψ_i, $1 \leq i \leq m$, are single variables, one for each of the continuous yield part dominated by the node A_i. The arguments of ψ_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. For all preterminals A dominating some terminal a, we extract a production $A(a) \rightarrow \varepsilon$. Since by definition, a label is associated with a certain fan-out, we distinguish the labels by corresponding subscripts. Note that this extraction algorithm yields only ordered LCFRS. Furthermore, note that for trees without crossing branches, this algorithm yields a PLCFRS with fan-out 1, i.e., a PCFG.

As mentioned before, the advantage of using LCFRS is that grammar extraction is straightforward and that no separate assumptions must be made. Note that unlike, e.g., Range Concatenation Grammar (RCG) (Boullier, 1998), LCFRS cannot model re-entrancies, i.e., nodes with more than one incoming edge. While those do not occur in NeGra-style annotation, some of the annotation in the PTB, e.g., the annotation for right node raising, can be interpreted as re-entrancies. This topic is left for future work. See Maier and Lichte (2009) for further details, especially on how treebank properties relate to properties of extracted grammars.

Before parsing, we binarize our grammar. We first mark the head daughters of all non-terminal nodes using Collins-style head rules based on the NeGra rules of the Stanford Parser (Klein and Manning, 2003b) and the reorder the RHSs of all LCFRS rules such that sequence of elements to the right of the head daughter is reversed and moved to the beginning of the RHS. From this point, the binarization works like the transformation into Chomsky Normal Form for CFGs. For each rule with an RHS of length ≥ 3, we introduce a new non-terminal which covers the RHS without the first element and continue successively from left to right. The rightmost new rule, which covers the head daughter, is binarized to unary.

We markovize the grammar as in the CFG case. To the new symbols introduced during the binarization, a variable number of symbols from the vertical and horizontal context of the original rule is added. Following the literature, we call the respective quantities v and h. As an example, Fig. 4 shows the output for the production for the VP in the left tree in Fig. 2.

After extraction and head marking:
VP2(X_1,X_2X_3) → AVP1(X_1) AVP1(X_2) VVPP1'(X_3)

After binarization and markovization with $v = 1, h = 2$:
VP2(X_1,X_2) → AVP1(X_1) @-VP2^v-AVP1^h-VVPP1^h(X_2)
@-VP2^v-AVP1^h-VVPP1^h(X_1X_2)
→ AVP1(X_1) @-VP2^v-VVPP1^h(X_2)
@-VP2^v-VVPP1^h(X_1) → VVPP1(X_1)
After binarization and markovization with $v = 2, h = 1$:
VP2(X_1,X_2) → AVP1(X_1) @-VP2^v-S2^v-AVP1^h(X_2)
@-VP2^v-S2^v-AVP1^h(X_1X_2)
→ AVP1(X_1) @-VP2^v-S2^v-VVPP1^h(X_2)
@-VP2^v-S2^v-VVPP1^h(X_1) → VVPP1(X_1)

Figure 4: Grammar extraction and binarization example

The probabilities are then computed based on the number of occurrences of rules in the transformed treebank, using a Maximum Likelihood estimator.

3 Evaluation methods

We assess the quality of our parser output using different methods.

The first is an **EVALB-style metric** (henceforth **EVALB**), i.e., we compare phrase boundaries. In spite of its shortcomings (Rehbein and van Genabith, 2007), it allows us to compare to previous work on parsing NeGra. In the context of LCFRS, we compare sets of tuples of the form $[A, (i_l^1, i_r^1), \ldots, (i_l^k, i_r^k)]$, where A is a non-terminal in some derivation tree with $dim(A) = k$ and each (i_l^m, i_r^m), $1 \leq m \leq k$, is a tuple of indices denoting a continuous sequence of terminals dominated by A. One set is obtained from the parser output, and

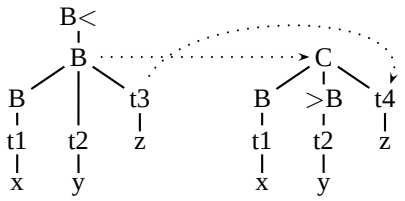

Figure 5: TDIST example

one from the corresponding treebank trees. Using these tuple sets, we compute labeled and unlabeled recall (LR/UR), precision (LP/UP), and the F_1 measure (LF_1/UF_1) in the usual way. Note that if $k = 1$, our metric is identical to its PCFG version.

EVALB does not necessarily reflect parser output quality (Rehbein and van Genabith, 2007; Emms, 2008; Kübler et al., 2008). One of its major problems is that attachment errors are penalized too hard. As the second evaluation method, we therefore choose the **tree-distance measure** (henceforth **TDIST**) (Zhang and Shasha, 1989), which levitates this problem. It has been proposed for parser evaluation by Emms (2008). TDIST is an ideal candidate for evaluation of the output of a PLCFRS, since it the fact if trees have crossing branches or not is not relevant to it. Two trees τ_k and τ_A are compared on the basis of T-**mappings** from τ_k to τ_A. A T-mapping is a partial mapping σ of nodes of τ_k to nodes of τ_A where all node mappings preserve left-to-right order and ancestry. Within the mappings, node insertion, node deletion, and label swap operations are identified, represented resp. by the sets $\mathcal{I}$, $\mathcal{D}$ and $\mathcal{S}$. Furthermore, we consider the set $\mathcal{M}$ representing the matched (i.e., unchanged) nodes. The cost of a T-mapping is the total number of operations, i.e. $|\mathcal{I}|+|\mathcal{D}|+|\mathcal{S}|$. The **tree distance** between two trees τ_K and τ_A is the cost of the cheapest T-mapping. Fig. 5, borrowed from Emms, shows an example for a T-mapping. Inserted nodes are prefixed with $>$, deleted nodes are suffixed with $<$, and nodes with swapped labels are linked with arrows. Since in total, four operations are involved, to this T-mapping, a cost of 4 is assigned. For more details, especially on algorithms which compute TDIST, refer to Bille (2005). In order to convert the tree distance measure into a similarity measure like EVALB, we use the macro-averaged Dice and Jaccard normalizations as defined by Emms. Let τ_K and τ_A be two trees with $|\tau_K|$ and $|\tau_A|$ nodes, respectively. For a T-mapping σ from τ_K to τ_A with the sets $\mathcal{D}$, $\mathcal{I}$, $\mathcal{S}$ and $\mathcal{M}$, we compute them as follows.

$$dice(\sigma) = 1 - \frac{|\mathcal{D}| + |\mathcal{I}| + |\mathcal{S}|}{|\tau_K| + |\tau_A|}$$

$$jaccard(\sigma) = 1 - \frac{|\mathcal{D}| + |\mathcal{I}| + |\mathcal{S}|}{|\mathcal{D}| + |\mathcal{I}| + |\mathcal{S}| + |\mathcal{M}|}$$

where, in order to achieve macro-averaging, we sum the numerators and denominators over all tree pairs before dividing. See Emms (2008) for further details.

The third method is **dependency evaluation** (henceforth **DEP**), as described by Lin (1995). It consists of comparing dependency graphs extracted from the gold data and from the parser output. The dependency extraction algorithm as given by Lin does also not rely on trees to be free of crossing branches. It only relies on a method to identify the head of each phrase. We use our own implementation of the algorithm which is described in Sect. 4 of Lin (1995), combined with the head finding algorithm of the parser. Dependency evaluation abstracts away from another bias of EVALB. Concretely, it does not prefer trees with a high node/token ratio, since two dependency graphs to be compared necessarily have the same number of (terminal) nodes. In the context of parsing German, this evaluation has been employed previously by Kübler et al. (2008).

Last, we evaluate on **TePaCoC** (**T**esting **P**arser **P**erformance **o**n **Co**mplex Grammatical **C**onstructions), a set of particularly difficult sentences hand-picked from TIGER (Kübler et al., 2008).

4 Experiments

Our data sources are the German NeGra (Skut et al., 1997) and TIGER (Brants et al., 2002) treebanks. In a preprocessing step, following common practice, we attach all punctuation to nodes within the tree, since it is not included in the NeGra annotation. In a first pass, using heuristics, we attach all nodes to the in each case highest available phrasal node such that ideally, we do not introduce new crossing branches. In a second pass, parentheses and quotation marks are preferably attached to the same node. Grammatical function labels are

discarded. After this preprocessing step, we create a separate version of the data set, in which we resolve the crossing branches in the trees, using the common approach of re-attaching nodes to higher constituents. We use the first 90% of our data sets for training and the remaining 10% for testing. Due to memory limitations, we restrict ourselves to sentences of a maximal length of 30 words. Our TIGER data sets (TIGER and T-CF) have 31,568 sentences of an average length of 14.81, splitted into 31,568 sentences for training and 3,508 sentences for testing. Our NeGra data sets (NeGra and N-CF) have 18,335 sentences, splitted into 16,501 sentences for training and 1,834 sentences for testing.

We parse the data sets described above with activated LR estimate. For all our experiments, we use the markovization settings $v = 2$ and $h = 1$, which have proven to be successful in previous work on parsing NeGra (Rafferty and Manning, 2008). We provide the parser with the gold tagging. Fig. 6 shows the average parsing times for all data sets on an AMD Opteron node with 8GB of RAM (pure Java implementation), Tab. 1 shows the percentage of parsed sentences.

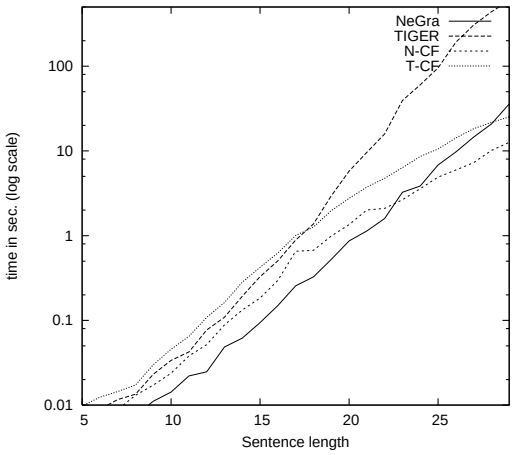

Figure 6: Parsing times

	NeGra	TIGER	N-CF	T-CF
total	1834	3508	1834	3508
parsed	1779	3462	1804	3462
	(97.0%)	(98.7%)	(98.4%)	(98.7%)

Table 1: Parsed sentences

4.1 Evaluation Using EVALB

Tab. 2 shows the evaluation of the parser output using EVALB, as described in the previous section. We report labeled and unlabeled precision, recall and F_1 measure.

	LP	LR	LF_1	UP	UR	UF_1
NeGra	72.39	70.68	71.52	76.01	74.22	75.10
TIGER	74.97	71.95	73.43	78.58	75.42	76.97
N-CF	74.85	73.26	74.04	78.11	76.45	77.28
T-CF	77.51	73.73	75.57	80.59	76.66	78.57

Table 2: EVALB results

Not surprisingly, reconstructing discontinuities is hard. Therefore, when parsing without crossing branches, the results are slightly better. In order to see the influence of discontinuous structures during parsing on the underlying phrase structure, we resolve the crossing branches in the parser output of NeGra and TIGER and compare it to the respective gold test data of N-CF and T-CF. Tab. 3 shows the results.

	LP	LR	LF_1	UP	UR	UF_1
NeGra	72.75	71.04	71.89	76.38	74.58	75.47
TIGER	75.28	72.25	73.74	78.81	75.64	77.20

Table 3: EVALB results (resolved crossing branches)

The results deteriorate slightly in comparison with N-CF and T-CF, however, they are slightly higher than for than for NeGra and TIGER. This is due to the fact that during the transformation, some errors in the LCFRS parses get "corrected": Wrongly attached phrasal nodes are re-attached to unique higher positions in the trees.

In order to give a point of comparison with previous work on parsing TIGER and NeGra, in Tab. 4, we report some of the results from the literature. All of them were obtained using PCFG parsers: Kübler (2005) (Tab. 1, plain PCFG for NeGra), Kübler et al. (2008) (Tab. 3, plain PCFG and Stanford parser with markovization $v = 2$ and $h = 1$ for TIGER), and Petrov and Klein (2007) (Tab. 1, Berkeley parser, latent variables). We include the results for N-CF and T-CF.

Our results are slightly better than for the plain PCFG models. We would expect the result for T-CF to be closer to the corresponding result for the Stanford parser, since we are using a comparable

	plain	**this work**	markov.	latent
NeGra	69.94	**74.04**	–	80.1
TIGER	74.00	**75.57**	77.30	–

Table 4: PCFG parsing of NeGra, Labeled F_1

model. This difference is mostly likely due to losses induced by the LR estimate. All items to which the estimate assigns an outside log probability estimate of $-\infty$ get blocked and are not put on the agenda. This blocking has an extremely beneficial effect on parser speed. However, it is paid by a worse recall, as experiments with smaller data sets have shown. A complete discussion of the effects of estimates, as well as a discussion of other possible optimizations, is presented in Kallmeyer and Maier (2010).

Recall finally that LCFRS parses are more informative than PCFG parses – a lower score for LCFRS EVALB than for PCFG EVALB does not necessarily mean that the PCFG parse is "better".

4.2 Evaluation Using Tree Distance

Tab. 5 shows the results of evaluating with TDIST, excluding unparsed sentences. We report the $dice$ and $jaccard$ normalizations, as well as a summary of the distribution of the tree distances between gold trees and trees from the parser output (see Sect. 3).

			tree distance distrib.		
	dice	jaccard	0	≤ 3	≥ 10
NeGra	88.86	79.79	31.65	53.77	15.08
TIGER	89.47	80.84	29.87	56.78	18.18
N-CF	92.50	85.99	33.43	61.92	6.71
T-CF	92.70	86.46	31.80	63.81	4.56

Table 5: Tree distance evaluation

Again, we can observe that parsing LCFRS is harder than parsing PCFG. As for EVALB, the results for TIGER are slightly higher than the ones for NeGra. The distribution of the tree distances shows that about a third of all sentences receive a completely correct parse. More than a half, resp. a third of all parser output trees require ≤ 3 operations to be mapped to the corresponding gold tree, and a only a small percentage requires ≥ 10 operations.

To our knowledge, TDIST has not been used to evaluate parser output for NeGra and TIGER. However, Emms (2008) reports results for the PTB using different parsers. Collins' Model 1 (Collins, 1999), e.g., lies at 93.62 (Dice) and 87.87 (Jaccard). For the Berkeley Parser (Petrov and Klein, 2007), 94.72 and 89.87 is reported. We see that our results lie in them same range. However, Jaccard scores are lower since this normalization punishes a higher number of edit operations more severely than Dice. In order to meaningfully interpret which treebank properties are responsible for the fact that between the gold trees and the trees from the parser, the German data requires more tree edit operations than the English data, a TDIST evaluation of the output of an off-the-shelf PCFG parser would be necessary. This is left for future work.

4.3 Dependency Evaluation

For the dependency evaluation, we extract dependency graphs from both the gold data and the test data and compare the unlabeled accuracy. Tab. 6 shows the results. We report unlabeled attachment score (UAS).

	UAS
NeGra	76.50
TIGER	77.84
N-CF	77.52
T-CF	78.67

Table 6: Dependency evaluation

The dependency results are consistent with the previous results in as much as the scores for PCFG parsing are again higher. The dependency results reported in Kübler et al. (2008) however are much higher (85.6 UAS for the markovized Stanford parser). While a part of the losses can again be attributed to the LR estimate, another reason lies undoubtedly in the different dependency conversion method which we employ, and in further treebank transformations which Kübler et al. perform. In order to get a more fine grained result, in future work, we will consider graph modifications as proposed by Lin (1995) as well as including annotation-specific information from NeGra/TIGER in our conversion procedure.

4.4 TePaCoC

The TePaCoC data set (Kübler et al., 2008) provides 100 hand-picked sentences from TIGER which contain constructions that are especially difficult to

parse. Out of these 100 sentences, we only consider 69. The remaining 31 sentences are either longer than 30 words or not included in the TIGER 2003 release (Kübler et al. use the 2005 release). The data is partitioned in groups of sentences with extraposed relative clauses (ERC), forward conjunction reduction (FCR), noun PP attachment (PPN), verb PP attachment (PPV), subject gap with finite/fronted verbs (SGF) and coordination of unlike constituents (CUC). Tab. 7 shows the EVALB results for the (discontinuous) TePaCoC. We parse these sentences using the same training set as before with all TePaCoC sentences removed.

	LP	LR	LF_1	UP	UR	UF_1
ERC	59.34	61.36	60.34	64.84	67.05	65.92
FCR	78.03	76.70	77.36	82.66	81.25	81.95
PPN	72.15	72.15	72.15	75.95	75.95	75.95
PPV	73.33	73.33	73.33	76.66	76.66	76.66
CUC	58.76	57.58	58.16	69.07	67.68	68.37
SGF	82.67	81.05	81.85	85.33	83.66	84.49
all	72.27	71.83	72.05	77.26	76.78	77.02

Table 7: EVALB scores for TePaCoC

While we cannot compare our results directly with the PCFG results (using grammatical function labels) of Kübler et al., their results nevertheless give an orientation.

We take a closer look at all sentence groups. Our result for ERC is more than 15 points worse than the result of Kübler et al. The relative clause itself is mostly recognized as a sentence (though not explicitly marked as a relative clause, since we do not consider grammatical functions). However, it is almost consistently attached too high (on the VP or on clause level). While this is correct for Kübler et al., with crossing branches, it treated as an error and punished especially hard by EVALB. FCR is parsed mostly well and with comparable results to Kübler et al. There are too few sentences to make a strong claim about PP attachment. However, in both PPN and PPV flat phrases seem to be preferred, which has as a consequence that in PPN, PPs are attached too high and in PPV too low. Our output confirms the claim of Kübler et al.'s that unlike coordinations is the most difficult of all TePaCoC phenomena. The conjuncts themselves are correctly identified in most cases, however then coordinated at the wrong level. SGF is parsed best. Kübler et al. report for this group only 78.6 labeled F_1 for the Stanford Parser. Our overall results are slightly worse than the results of Kübler et al., but show less variance.

To sum up, not surprisingly, getting the right attachment positions seems to be hard for LCFRS, too. Additionally, with crossing branches, the output is rated worse, since some attachments are not present anymore without crossing branches. Since especially for the relative clauses, attachment positions are in fact a matter of discussion from a syntactic point of view, we will consider in future studies to selectively resolve some of the crossing branches, e.g., by attaching relative clauses to higher positions.

5 Related Work

The use of formalisms with a high expressivity has been explored before (Plaehn, 2004; Levy, 2005). To our knowledge, Plaehn is the only one to report evaluation results. He uses the formalism of Discontinuous Phrase Structure Grammar (DPSG). Limiting the sentence length to 15, he obtains 73.16 labeled F_1 on NeGra. Evaluating all sentences of our NeGra data with a length of up to 15 words results, however, in 81.27 labeled F_1. For a comparison between DPSG and LCFRS, refer to Maier and Søgaard (2008).

6 Conclusion and Future Work

We have investigated the possibility of using Probabilistic Linear Context-Free Rewriting Systems for direct parsing of discontinuous constituents. Consequently, we have applied a PLCFRS parser on the German NeGra and TIGER treebanks. Our evaluation, which used different metrics, showed that a PLCFRS parser can achieve competitive results.

In future work, all of the presented evaluation methods will be investigated to greater detail. In order to do this, we will parse our data sets with current state-of-the-art systems. Especially a more elaborate dependency conversion should enable a more informative comparison between the output of PCFG parsers and the output of the PLCFRS parser. Last, since an algorithm is available which extracts LCFRSs from dependency structures (Kuhlmann and Satta, 2009), the parser is instantly ready for parsing them. We are currently performing the corresponding experiments.

References

Philip Bille. 2005. A survey on tree edit distance and related problems. *Theoretical Computer Science*, 337.

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

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

Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2002. The TIGER Treebank. In *Proceedings of Treebanks and Linguistic Theories*.

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

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

Martin Emms. 2008. Tree Distance and some other variants of Evalb. In *Proceedings of LREC 08*.

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*.

Laura Kallmeyer and Wolfgang Maier. 2009. An incremental earley parser for simple range concatenation grammar. In *Proceedings of IWPT 09*.

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

Yuki Kato, Hiroyuki Seki, and Tadao Kasami. 2006. RNA pseudoknotted structure prediction using stochastic multiple context-free grammar. *IPSJ Digital Courier*, 2.

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

Dan Klein 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)*. MIT Press.

Sandra Kübler, Wolfgang Maier, Ines Rehbein, and Yannick Versley. 2008. How to compare treebanks. In *Proceedings of LREC 08*.

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. 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 IJCAI 95*.

Wolfgang Maier and Timm Lichte. 2009. Characterizing discontinuity in constituent treebanks. In *Proceedings of Formal Grammar 2009*.

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

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 *Proceedings of HLT*.

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

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 *Proceedings of the Workshop on Parsing German at ACL 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*.

Heike Telljohann, Erhard Hinrichs, Sandra Kübler, and Heike Zinsmeister. 2006. Stylebook for the Tübingen Treebank of Written German (TüBa-D/Z). Technischer Bericht, Universität Tübingen.

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

Kaizhong Zhang and Dennis Shasha. 1989. Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal of Computing*, 18.