Efficient Parsing with Linear Context-Free Rewriting Systems

Andreas van Cranenburgh
Huygens ING & ILLC, University of Amsterdam
Royal Netherlands Academy of Arts and Sciences
Postbus 90754, 2509 LT The Hague, the Netherlands
andreas.van.cranenburgh@huygens.knaw.nl

Abstract

Previous work on treebank parsing with discontinuous constituents using Linear Context-Free Rewriting systems (LCFRS) has been limited to sentences of up to 30 words, for reasons of computational complexity. There have been some results on binarizing an LCFRS in a manner that minimizes parsing complexity, but the present work shows that parsing long sentences with such an optimally binarized grammar remains infeasible. Instead, we introduce a technique which removes this length restriction, while maintaining a respectable accuracy. The resulting parser has been applied to a discontinuous treebank with favorable results.

1 Introduction

Discontinuity in constituent structures (cf. figure 1 & 2) is important for a variety of reasons. For one, it allows a tight correspondence between syntax and semantics by letting constituent structure express argument structure (Skut et al., 1997). Other reasons are phenomena such as extraposition and word-order freedom, which arguably require discontinuous annotations to be treated systematically in phrase-structures (McCawley, 1982; Levy, 2005). Empirical investigations demonstrate that discontinuity is present in non-negligible amounts: around 30% of sentences contain discontinuity in two German treebanks (Maier and Søgaard, 2008; Maier and Lichte, 2009). Recent work on treebank parsing with discontinuous constituents (Kallmeyer and Maier, 2010; Maier, 2010; Evang and Kallmeyer, 2011; van Cranenburgh et al., 2011) shows that it is feasible to directly parse discontinuous constituency annotations, as given in the German Negra (Skut et al., 1997) and Tiger (Brants et al., 2002) corpora, or those that can be extracted from traces such as in the Penn treebank (Marcus et al., 1993) annotation. However, the computational complexity is such that until now, the length of sentences needed to be restricted. In the case of Kallmeyer and Maier (2010) and Evang and Kallmeyer (2011) the limit was 25 words. Maier (2010) and van Cranenburgh et al. (2011) manage to parse up to 30 words with heuristics and optimizations, but no further. Algorithms have been suggested to binarize the grammars in such a way as to minimize parsing complexity, but the current paper shows that these techniques are not sufficient to parse longer sentences. Instead, this work presents a novel form of coarse-to-fine parsing which does alleviate this limitation.

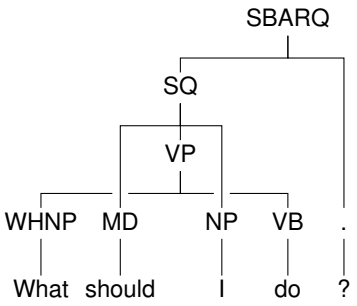

Figure 1: A tree with WH-movement from the Penn treebank, in which traces have been converted to discontinuity. Taken from Evang and Kallmeyer (2011).

The rest of this paper is structured as follows. First, we introduce linear context-free rewriting systems (LCFRS). Next, we discuss and evaluate binarization strategies for LCFRS. Third, we present a technique for approximating an LCFRS by a PCFG in a coarse-to-fine framework. Lastly, we evaluate this technique on a large corpus without the usual length restrictions.

Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics, pages 460–470,
Avignon, France, April 23 - 27 2012.

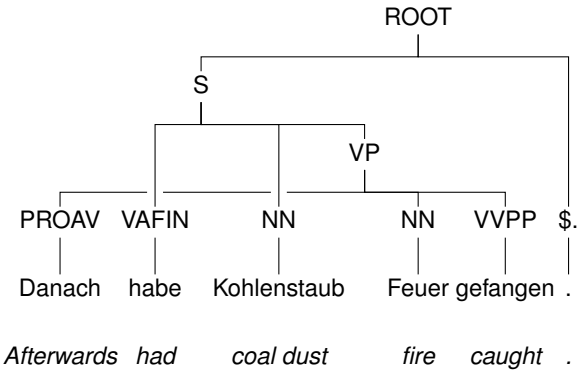

Figure 2: A discontinuous tree from the Negra corpus. Translation: *After that coal dust had caught fire.*

$\text{ROOT}(ab) \rightarrow \text{S}(a)\ \$.(b)$
$\text{S}(abcd) \rightarrow \text{VAFIN}(b)\ \text{NN}(c)\ \text{VP}_2(a, d)$
$\text{VP}_2(a, bc) \rightarrow \text{PROAV}(a)\ \text{NN}(b)\ \text{VVPP}(c)$
$\text{PROAV(Danach)} \rightarrow \epsilon$
$\text{VAFIN(habe)} \rightarrow \epsilon$
$\text{NN(Kohlenstaub)} \rightarrow \epsilon$
$\text{NN(Feuer)} \rightarrow \epsilon$
$\text{VVPP(gefangen)} \rightarrow \epsilon$
$\$.(.) \rightarrow \epsilon$

Figure 3: The productions that can be read off from the tree in figure 2. Note that lexical productions rewrite to ϵ, because they do not rewrite to any non-terminals.

2 Linear Context-Free Rewriting Systems

Linear Context-Free Rewriting Systems (LCFRS; Vijay-Shanker et al., 1987; Weir, 1988) subsume a wide variety of mildly context-sensitive formalisms, such as Tree-Adjoining Grammar (TAG), Combinatory Categorial Grammar (CCG), Minimalist Grammar, Multiple Context-Free Grammar (MCFG) and synchronous CFG (Vijay-Shanker and Weir, 1994; Kallmeyer, 2010). Furthermore, they can be used to parse dependency structures (Kuhlmann and Satta, 2009). Since LCFRS subsumes various synchronous grammars, they are also important for machine translation. This makes it possible to use LCFRS as a syntactic backbone with which various formalisms can be parsed by compiling grammars into an LCFRS, similar to the TuLiPa system (Kallmeyer et al., 2008). As all mildly context-sensitive formalisms, LCFRS are parsable in polynomial time, where the degree depends on the productions of the grammar. Intuitively, LCFRS can be seen as a generalization of context-free grammars to rewriting other objects than just continuous strings: productions are context-free, but instead of strings they can rewrite tuples, trees or graphs.

We focus on the use of LCFRS for parsing with discontinuous constituents. This follows up on recent work on parsing the discontinuous annotations in German corpora with LCFRS (Maier, 2010; van Cranenburgh et al., 2011) and work on parsing the Wall Street journal corpus in which traces have been converted to discontinuous constituents (Evang and Kallmeyer, 2011). In the case of parsing with discontinuous constituents a non-terminal may cover a tuple of discontinuous strings instead of a single, contiguous sequence of terminals. The number of components in such a tuple is called the *fan-out* of a rule, which is equal to the number of gaps plus one; the fan-out of the grammar is the maximum fan-out of its production. A context-free grammar is a LCFRS with a fan-out of 1. For convenience we will will use the rule notation of simple RCG (Boullier, 1998), which is a syntactic variant of LCFRS, with an arguably more transparent notation.

A LCFRS is a tuple $G = \langle N, T, V, P, S\rangle$. N is a finite set of non-terminals; a function *dim* : $N \rightarrow \mathbb{N}$ specifies the unique fan-out for every non-terminal symbol. T and V are disjoint finite sets of terminals and variables. S is the distinguished start symbol with $\mathit{dim}(S) = 1$. P is a finite set of rewrite rules (productions) of the form:

$$A(\alpha_1, \ldots \alpha_{\mathit{dim}(A)}) \rightarrow B_1(X_1^1, \ldots, X_{\mathit{dim}(B_1)}^1) \ldots B_m(X_1^m, \ldots, X_{\mathit{dim}(B_m)}^m)$$

for $m \geq 0$, where $A, B_1, \ldots, B_m \in N$, each $X_j^i \in V$ for $1 \leq i \leq m$, $1 \leq j \leq \mathit{dim}(A_j)$ and $\alpha_i \in (T \cup V)^*$ for $1 \leq i \leq \mathit{dim}(A_i)$.

Productions must be *linear*: if a variable occurs in a rule, it occurs exactly once on the left hand side (LHS), and exactly once on the right hand side (RHS). A rule is *ordered* if for any two variables X_1 and X_2 occurring in a non-terminal on the RHS, X_1 precedes X_2 on the LHS iff X_1 precedes X_2 on the RHS.

Every production has a fan-out determined by the fan-out of the non-terminal symbol on the left-hand side. Apart from the fan-out productions also

have a rank: the number of non-terminals on the right-hand side. These two variables determine the time complexity of parsing with a grammar. A production can be *instantiated* when its variables can be bound to non-overlapping spans such that for each component α_i of the LHS, the concatenation of its terminals and bound variables forms a contiguous span in the input, while the endpoints of each span are non-contiguous.

As in the case of a PCFG, we can read off LCFRS productions from a treebank (Maier and Søgaard, 2008), and the relative frequencies of productions form a maximum likelihood estimate, for a probabilistic LCFRS (PLCFRS), i.e., a (discontinuous) treebank grammar. As an example, figure 3 shows the productions extracted from the tree in figure 2.

3 Binarization

A probabilistic LCFRS can be parsed using a CKY-like tabular parsing algorithm (cf. Kallmeyer and Maier, 2010; van Cranenburgh et al., 2011), but this requires a binarized grammar.[1] Any LCFRS can be binarized. Crescenzi et al. (2011) state "while CFGs can always be reduced to rank two (Chomsky Normal Form), this is not the case for LCFRS with any fan-out greater than one." However, this assertion is made under the assumption of a fixed fan-out. If this assumption is relaxed then it is easy to binarize either deterministically or, as will be investigated in this work, optimally with a dynamic programming approach. Binarizing an LCFRS may increase its fan-out, which results in an increase in asymptotic complexity. Consider the following production:

$$X(pqrs) \rightarrow A(p,r)\ B(q)\ C(s) \tag{1}$$

Henceforth, we assume that non-terminals on the right-hand side are ordered by the order of their first variable on the left-hand side. There are two ways to binarize this production. The first is from left to right:

$$X(ps) \rightarrow X_{AB}(p)\ C(s) \tag{2}$$
$$X_{AB}(pqr) \rightarrow A(p,r)\ B(q) \tag{3}$$

This binarization maintains the fan-out of 1. The second way is from right to left:

$$X(pqrs) \rightarrow A(p,r)\ X_{BC}(q,s) \tag{4}$$
$$X_{BC}(q,s) \rightarrow B(q)\ C(s) \tag{5}$$

[1] Other algorithms exist which support n-ary productions, but these are less suitable for statistical treebank parsing.

This binarization introduces a production with a fan-out of 2, which could have been avoided. After binarization, an LCFRS can be parsed in $\mathcal{O}(|G| \cdot |w|^p)$ time, where $|G|$ is the size of the grammar, $|w|$ is the length of the sentence. The degree p of the polynomial is the maximum parsing complexity of a rule, defined as:

$$\text{parsing complexity} := \varphi + \varphi_1 + \varphi_2 \tag{6}$$

where φ is the fan-out of the left-hand side and φ_1 and φ_2 are the fan-outs of the right-hand side of the rule in question (Gildea, 2010). As Gildea (2010) shows, there is no one to one correspondence between fan-out and parsing complexity: it is possible that parsing complexity can be reduced by increasing the fan-out of a production. In other words, there can be a production which can be binarized with a parsing complexity that is minimal while its fan-out is sub-optimal. Therefore we focus on parsing complexity rather than fan-out in this work, since parsing complexity determines the actual time complexity of parsing with a grammar. There has been some work investigating whether the increase in complexity can be minimized effectively (Gómez-Rodríguez et al., 2009; Gildea, 2010; Crescenzi et al., 2011).

More radically, it has been suggested that the power of LCFRS should be limited to well-nested structures, which gives an asymptotic improvement in parsing time (Gómez-Rodríguez et al., 2010). However, there is linguistic evidence that not all language use can be described in well-nested structures (Chen-Main and Joshi, 2010). Therefore we will use the full power of LCFRS in this work—parsing complexity is determined by the treebank, not by a priori constraints.

3.1 Further binarization strategies

Apart from optimizing for parsing complexity, for linguistic reasons it can also be useful to parse the head of a constituent first, yielding so-called head-driven binarizations (Collins, 1999). Additionally, such a head-driven binarization can be 'Markovized'–i.e., the resulting production can be constrained to apply to a limited amount of horizontal context as opposed to the full context in the original constituent (e.g., Klein and Manning, 2003), which can have a beneficial effect on accuracy. In the notation of Klein and Manning (2003) there are two Markovization parameters: h and v. The first parameter describes the amount of

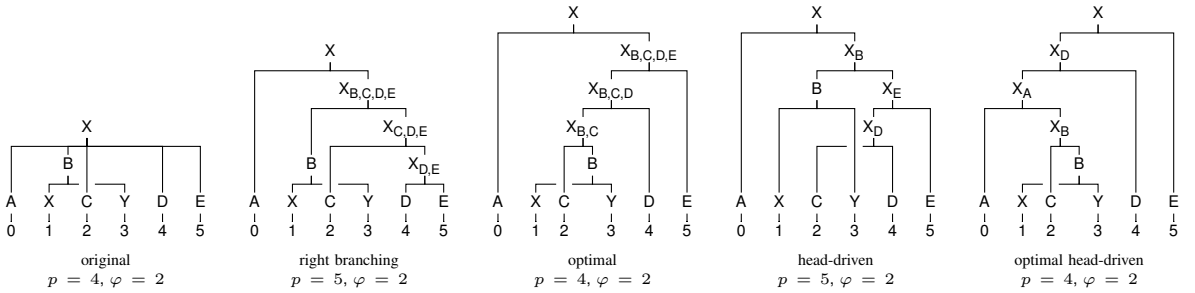

Figure 4: The four binarization strategies. C is the head node. Underneath each tree is the maximum parsing complexity and fan-out among its productions.

horizontal context for the artificial labels of a binarized production. In a normal form binarization, this parameter equals infinity, because the binarized production should only apply in the exact same context as the context in which it originally belongs, as otherwise the set of strings accepted by the grammar would be affected. An artificial label will have the form $X_{A,B,C}$ for a binarized production of a constituent X that has covered children A, B, and C of X. The other extreme, $h = 1$, enables generalizations by stringing parts of binarized constituents together, as long as they share one non-terminal. In the previous example, the label would become just X_A, i.e., the presence of B and C would no longer be required, which enables switching to any binarized production that has covered A as the last node. Limiting the amount of horizontal context on which a production is conditioned is important when the treebank contains many unique constituents which can only be parsed by stringing together different binarized productions; in other words, it is a way of dealing with the data sparseness about n-ary productions in the treebank.

The second parameter describes parent annotation, which will not be investigated in this work; the default value is $v = 1$ which implies only including the immediate parent of the constituent that is being binarized; including grandparents is a way of weakening independence assumptions.

Crescenzi et al. (2011) also remark that an *optimal* head-driven binarization allows for Markovization. However, it is questionable whether such a binarization is worthy of the name Markovization, as the non-terminals are not introduced deterministically from left to right, but in an arbitrary fashion dictated by concerns of parsing complexity; as such there is not a Markov process based on a meaningful (e.g., temporal) ordering and there is no probabilistic interpretation of Markovization in such a setting.

To summarize, we have at least four binarization strategies (cf. figure 4 for an illustration):

1. **right branching:** A right-to-left binarization. No regard for optimality or statistical tweaks.
2. **optimal:** A binarization which minimizes parsing complexity, introduced in Gildea (2010). Binarizing with this strategy is exponential in the resulting optimal fan-out (Gildea, 2010).
3. **head-driven:** Head-outward binarization with horizontal Markovization. No regard for optimality.
4. **optimal head-driven:** Head-outward binarization with horizontal Markovization. Minimizes parsing complexity. Introduced in and proven to be NP-hard by Crescenzi et al. (2011).

3.2 Finding optimal binarizations

An issue with the minimal binarizations is that the algorithm for finding them has a high computational complexity, and has not been evaluated empirically on treebank data.[2] Empirical investigation is interesting for two reasons. First of all, the high computational complexity may not be relevant with constant factors of constituents, which can reasonably be expected to be relatively small. Second, it is important to establish whether an asymptotic improvement is actually obtained through optimal binarizations, and whether this translates to an improvement in practice.

Gildea (2010) presents a general algorithm to binarize an LCFRS while minimizing a given scoring function. We will use this algorithm with two different scoring functions.

[2] Gildea (2010) evaluates on a dependency bank, but does not report whether any improvement is obtained over a naive binarization.

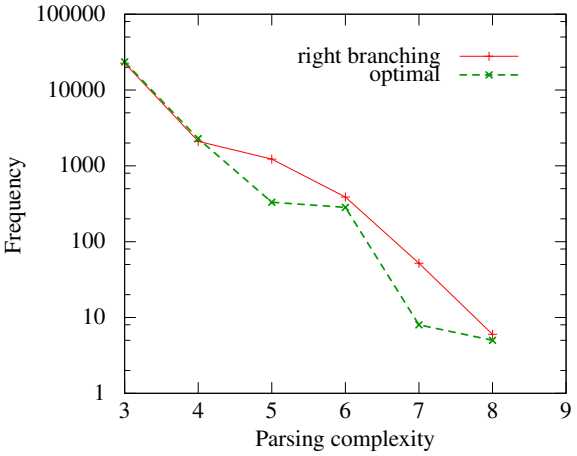

Figure 5: The distribution of parsing complexity among productions in binarized grammars read off from NEGRA-25. The y-axis has a logarithmic scale.

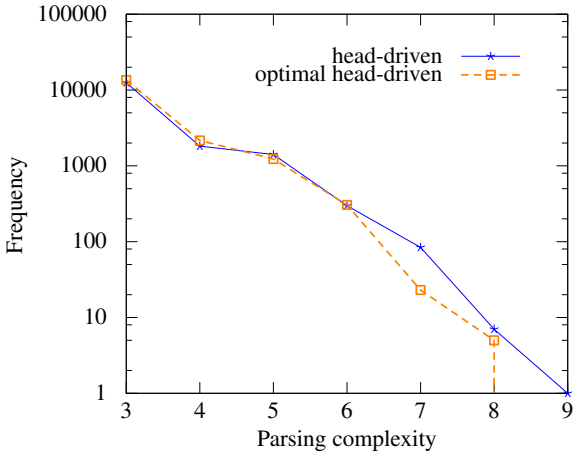

Figure 6: The distribution of parsing complexity among productions in Markovized, head-driven grammars read off from NEGRA-25. The y-axis has a logarithmic scale.

The first directly optimizes parsing complexity. Given a (partially) binarized constituent c, the function returns a tuple of scores, for which a linear order is defined by comparing elements starting from the most significant (left-most) element. The tuples contain the parsing complexity p, and the fan-out φ to break ties in parsing complexity; if there are still ties after considering the fan-out, the sum of the parsing complexities of the subtrees of c is considered, which will give preference to a binarization where the worst case complexity occurs once instead of twice. The formula is then:

$$\mathit{opt}(c) = \langle p, \varphi, s \rangle$$

The second function is the similar except that only head-driven strategies are accepted. A head-driven strategy is a binarization in which the head is introduced first, after which the rest of the children are introduced one at a time.

$$\mathit{opt\text{-}hd}(c) = \begin{cases} \langle p, \varphi, s \rangle & \text{if } c \text{ is head-driven} \\ \langle \infty, \infty, \infty \rangle & \text{otherwise} \end{cases}$$

Given a (partial) binarization c, the score should reflect the maximum complexity and fan-out in that binarization, to optimize for the worst case, as well as the sum, to optimize the average case. This aspect appears to be glossed over by Gildea (2010). Considering only the score of the last production in a binarization produces suboptimal binarizations.

3.3 Experiments

As data we use version 2 of the Negra (Skut et al., 1997) treebank, with the common training, development and test splits (Dubey and Keller, 2003). Following common practice, punctuation, which is left out of the phrase-structure in Negra, is reattached to the nearest constituent.

In the course of experiments it was discovered that the heuristic method for punctuation attachment used in previous work (e.g., Maier, 2010; van Cranenburgh et al., 2011), as implemented in `rparse`,[3] introduces additional discontinuity. We applied a slightly different heuristic: punctuation is attached to the highest constituent that contains a neighbor to its right. The result is that punctuation can be introduced into the phrase-structure without any additional discontinuity, and thus without artificially inflating the fan-out and complexity of grammars read off from the treebank. This new heuristic provides a significant improvement: instead of a fan-out of 9 and a parsing complexity of 19, we obtain values of 4 and 9 respectively.

The parser is presented with the gold part-of-speech tags from the corpus. For reasons of efficiency we restrict sentences to 25 words (including punctuation) in this experiment: NEGRA-25. A grammar was read off from the training part of NEGRA-25, and sentences of up to 25 words in the development set were parsed using the resulting PLCFRS, using the different binarization schemes. First with a right-branching, right-to-left binarization, and second with the minimal binarization according to parsing complexity and fan-

[3] Available from `http://www.wolfgang-maier.net/rparse/downloads`. Retrieved March 25th, 2011

	right branching	optimal	head-driven	optimal head-driven
Markovization	v=1, h=∞	v=1, h=∞	v=1, h=2	v=1, h=2
fan-out	4	4	4	4
complexity	8	8	9	8
labels	12861	12388	4576	3187
clauses	62072	62097	53050	52966
time to binarize	1.83 s	46.37 s	2.74 s	28.9 s
time to parse	246.34 s	193.94 s	2860.26 s	716.58 s
coverage	96.08 %	96.08 %	98.99 %	98.73 %
F_1 score	66.83 %	66.75 %	72.37 %	71.79 %

Table 1: The effect of binarization strategies on parsing efficiency, with sentences from the development section of NEGRA-25.

out. The last two binarizations are head-driven and Markovized—the first straightforwardly from left-to-right, the latter optimized for minimal parsing complexity. With Markovization we are forced to add a level of parent annotation to tame the increase in productivity caused by $h = 1$.

The distribution of parsing complexity (measured with eq. 6) in the grammars with different binarization strategies is shown in figure 5 and 6. Although the optimal binarizations do seem to have some effect on the distribution of parsing complexities, it remains to be seen whether this can be cashed out as a performance improvement in practice. To this end, we also parse using the binarized grammars.

In this work we binarize and parse with `disco-dop` introduced in van Cranenburgh et al. (2011).[4] In this experiment we report scores of the (exact) Viterbi derivations of a treebank PLCFRS; cf. table 1 for the results. Times represent CPU time (single core); accuracy is given with a generalization of PARSEVAL to discontinuous structures, described in Maier (2010).

Instead of using Maier's implementation of discontinuous F_1 scores in `rparse`, we employ a variant that ignores (a) punctuation, and (b) the root node of each tree. This makes our evaluation incomparable to previous results on discontinuous parsing, but brings it in line with common practice on the Wall street journal benchmark. Note that this change yields scores about 2 or 3 percentage points lower than those of `rparse`.

Despite the fact that obtaining optimal binarizations is exponential (Gildea, 2010) and NP-hard (Crescenzi et al., 2011), they can be computed relatively quickly on this data set.[5] Importantly, in the first case there is no improvement on fan-out or parsing complexity, while in the head-driven case there is a minimal improvement because of a single production with parsing complexity 15 without optimal binarization. On the other hand, the optimal binarizations might still have a significant effect on the average case complexity, rather than the worst-case complexities. Indeed, in both cases parsing with the optimal grammar is faster; in the first case, however, when the time for binarization is considered as well, this advantage mostly disappears.

The difference in F_1 scores might relate to the efficacy of Markovization in the binarizations. It should be noted that it makes little theoretical sense to 'Markovize' a binarization when it is not a left-to-right or right-to-left binarization, because with an optimal binarization the non-terminals of a constituent are introduced in an arbitrary order.

More importantly, in our experiments, these techniques of optimal binarizations did not scale to longer sentences. While it is possible to obtain an optimal binarization of the unrestricted Negra corpus, parsing long sentences with the resulting grammar remains infeasible. Therefore we need to look at other techniques for parsing longer sentences. We will stick with the straightforward

[4] All code is available from: `http://github.com/andreasvc/disco-dop`.

[5] The implementation exploits two important optimizations. The first is the use of bit vectors to keep track of which non-terminals are covered by a partial binarization. The second is to skip constituents without discontinuity, which are equivalent to CFG productions.

head-driven, head-outward binarization strategy, despite this being a computationally sub-optimal binarization.

One technique for efficient parsing of LCFRS is the use of context-summary estimates (Kallmeyer and Maier, 2010), as part of a best-first parsing algorithm. This allowed Maier (2010) to parse sentences of up to 30 words. However, the calculation of these estimates is not feasible for longer sentences and large grammars (van Cranenburgh et al., 2011).

Another strategy is to perform an online approximation of the sentence to be parsed, after which parsing with the LCFRS can be pruned effectively. This is the strategy that will be explored in the current work.

4 Context-free grammar approximation for coarse-to-fine parsing

Coarse-to-fine parsing (Charniak et al., 2006) is a technique to speed up parsing by exploiting the information that can be gained from parsing with simpler, coarser grammars—e.g., a grammar with a smaller set of labels on which the original grammar can be projected. Constituents that do not contribute to a full parse tree with a coarse grammar can be ruled out for finer grammars as well, which greatly reduces the number of edges that need to be explored. However, by changing just the labels only the grammar constant is affected. With discontinuous treebank parsing the asymptotic complexity of the grammar also plays a major role. Therefore we suggest to parse not just with a coarser grammar, but with a coarser grammar formalism, following a suggestion in van Cranenburgh et al. (2011).

This idea is inspired by the work of Barthélemy et al. (2001), who apply it in a non-probabilistic setting where the coarse grammar acts as a guide to the non-deterministic choices of the fine grammar. Within the coarse-to-fine approach the technique becomes a matter of pruning with some probabilistic threshold. Instead of using the coarse grammar only as a guide to solve non-deterministic choices, we apply it as a pruning step which also discards the most suboptimal parses. The basic idea is to extract a grammar that defines a superset of the language we want to parse, but with a fan-out of 1. More concretely, a context-free grammar can be read off from discontinuous trees that have been transformed to context-free trees by the procedure introduced in Boyd (2007). Each discontinuous node is split into a set of new nodes, one for each component; for example a node NP_2 will be split into two nodes labeled NP*1 and NP*2 (like Barthélemy et al., we mark components with an index to reduce overgeneration). Because Boyd's transformation is reversible, chart items from this grammar can be converted back to discontinuous chart items, and can guide parsing of an LCFRS. This guiding takes the form of a white list. After parsing with the coarse grammar, the resulting chart is pruned by removing all items that fail to meet a certain criterion. In our case this is whether a chart item is part of one of the k-best derivations—we use $k = 50$ in all experiments (as in van Cranenburgh et al., 2011). This has similar effects as removing items below a threshold of marginalized posterior probability; however, the latter strategy requires computation of outside probabilities from a parse forest, which is more involved with an LCFRS than with a PCFG. When parsing with the fine grammar, whenever a new item is derived, the white list is consulted to see whether this item is allowed to be used in further derivations; otherwise it is immediately discarded. This coarse-to-fine approach will be referred to as CFG-CTF, and the transformed, coarse grammar will be referred to as a split-PCFG.

Splitting discontinuous nodes for the coarse grammar introduces new nodes, so obviously we need to binarize after this transformation. On the other hand, the coarse-to-fine approach requires a mapping between the grammars, so after reversing the transformation of splitting nodes, the resulting discontinuous trees must be binarized (and optionally Markovized) in the same manner as those on which the fine grammar is based.

To resolve this tension we elect to binarize twice. The first time is before splitting discontinuous nodes, and this is where we introduce Markovization. This same binarization will be used for the fine grammar as well, which ensures the models make the same kind of generalizations. The second binarization is after splitting nodes, this time with a binary normal form (2NF; all productions are either unary, binary, or lexical).

Parsing with this grammar proceeds as follows. After obtaining an exhaustive chart from the coarse stage, the chart is pruned so as to only contain items occurring in the k-best derivations. When parsing in the fine stage, each new item is

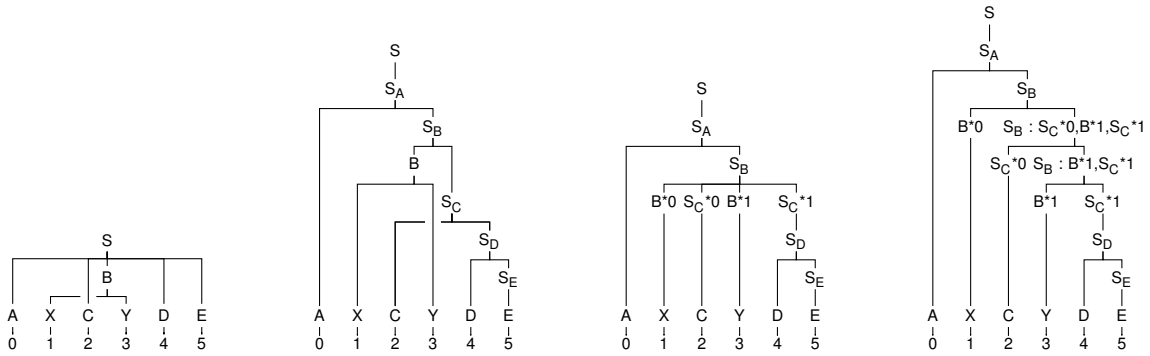

Figure 7: Transformations for a context-free coarse grammar. From left to right: the original constituent, Markovized with $v = 1$, $h = 1$, discontinuities resolved, normal form (second binarization).

model	train	dev	test	rules	labels	fan-out	complexity
Split-PCFG	17988	975	968	57969	2026	1	3
PLCFRS	17988	975	968	55778	947	4	9
Disco-DOP	17988	975	968	2657799	702246	4	9

Table 2: Some statistics on the coarse and fine grammars read off from NEGRA-40.

looked up in this pruned coarse chart, with multiple lookups if the item is discontinuous (one for each component).

To summarize, the transformation happens in four steps (cf. figure 7 for an illustration):

1. **Treebank tree:** Original (discontinuous) tree
2. **Binarization:** Binarize discontinuous tree, optionally with Markovization
3. **Resolve discontinuity:** Split discontinuous nodes into components, marked with indices
4. **2NF:** A binary normal form is applied; all productions are either unary, binary, or lexical.

5 Evaluation

We evaluate on Negra with the same setup as in section 3.3. We report discontinuous F_1 scores as well as exact match scores. For previous results on discontinuous parsing with Negra, see table 3. For results with the CFG-CTF method see table 4.

We first establish the viability of the CFG-CTF method on NEGRA-25, with a head-driven $v = 1$, $h = 2$ binarization, and reporting again the scores of the exact Viterbi derivations from a treebank PLCFRS versus a PCFG using our transformations. Figure 8 compares the parsing times of LCFRS with and without the new CFG-CTF method. The graph shows a steep incline for parsing with LCFRS directly, which makes it infeasible to parse longer sentences, while the CFG-CTF method is faster for

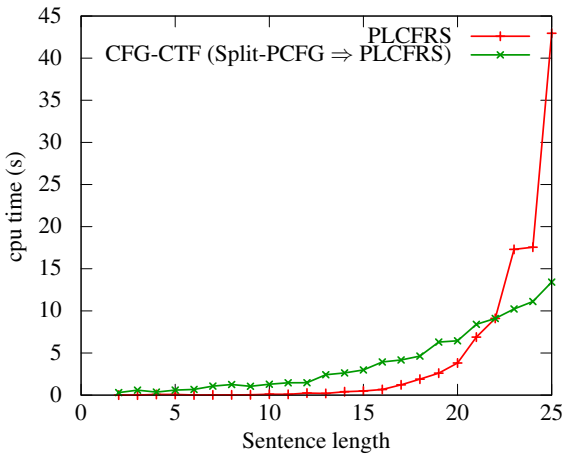

Figure 8: Efficiency of parsing PLCFRS with and without coarse-to-fine. The latter includes time for both coarse & fine grammar. Datapoints represent the average time to parse sentences of that length; each length is made up of 20–40 sentences.

sentences of length > 22 despite its overhead of parsing twice.

The second experiment demonstrates the CFG-CTF technique on longer sentences. We restrict the length of sentences in the training, development and test corpora to 40 words: NEGRA-40. As a first step we apply the CFG-CTF technique to parse with a PLCFRS as the fine grammar, pruning away all items not occurring in the 10,000 best derivations

	words	PARSEVAL (F_1)	Exact match
DPSG: Plaehn (2004)	≤ 15	73.16	39.0
PLCFRS: Maier (2010)	≤ 30	71.52	31.65
Disco-DOP: van Cranenburgh et al. (2011)	≤ 30	73.98	34.80

Table 3: Previous work on discontinuous parsing of Negra.

	words	PARSEVAL (F_1)	Exact match
PLCFRS, dev set	≤ 25	72.37	36.58
Split-PCFG, dev set	≤ 25	70.74	33.80
Split-PCFG, dev set	≤ 40	66.81	27.59
CFG-CTF, PLCFRS, dev set	≤ 40	67.26	27.90
CFG-CTF, Disco-DOP, dev set	≤ 40	74.27	34.26
CFG-CTF, Disco-DOP, test set	≤ 40	72.33	33.16
CFG-CTF, Disco-DOP, dev set	∞	73.32	33.40
CFG-CTF, Disco-DOP, test set	∞	71.08	32.10

Table 4: Results on NEGRA-25 and NEGRA-40 with the CFG-CTF method. NB: As explained in section 3.3, these F_1 scores are incomparable to the results in table 3; for comparison, the F_1 score for Disco-DOP on the dev set ≤ 40 is 77.13 % using that evaluation scheme.

from the PCFG chart. The result shows that the PLCFRS gives a slight improvement over the split--pcfg, which accords with the observation that the latter makes stronger independence assumptions in the case of discontinuity.

In the next experiments we turn to an all-fragments grammar encoded in a PLCFRS using Goodman's (2003) reduction, to realize a (discontinuous) Data-Oriented Parsing (DOP; Scha, 1990) model—which goes by the name of Disco-DOP (van Cranenburgh et al., 2011). This provides an effective yet conceptually simple method to weaken the independence assumptions of treebank grammars. Table 2 gives statistics on the grammars, including the parsing complexities. The fine grammar has a parsing complexity of 9, which means that parsing with this grammar has complexity $\mathcal{O}(|w|^9)$. We use the same parameters as van Cranenburgh et al. (2011), except that unlike van Cranenburgh et al., we can use $v = 1, h = 1$ Markovization, in order to obtain a higher coverage. The DOP grammar is added as a third stage in the coarse-to-fine pipeline. This gave slightly better results than substituting the the DOP grammar for the PLCFRS stage. Parsing with NEGRA-40 took about 11 hours and 4 GB of memory. The same model from NEGRA-40 can also be used to parse the full development set, without length restrictions, establishing that the CFG-CTF method effectively eliminates any limitation of length for parsing with LCFRS.

6 Conclusion

Our results show that optimal binarizations are clearly not the answer to parsing LCFRS efficiently, as they do not significantly reduce parsing complexity in our experiments. While they provide some efficiency gains, they do not help with the main problem of longer sentences.

We have presented a new technique for large-scale parsing with LCFRS, which makes it possible to parse sentences of any length, with favorable accuracies. The availability of this technique may lead to a wider acceptance of LCFRS as a syntactic backbone in computational linguistics.

Acknowledgments

I am grateful to Willem Zuidema, Remko Scha, Rens Bod, and three anonymous reviewers for comments.

References

François Barthélemy, Pierre Boullier, Philippe Deschamp, and Éric de la Clergerie. 2001. Guided parsing of range concatenation languages. In *Proc. of ACL*, pages 42–49.

Pierre Boullier. 1998. Proposal for a natural language processing syntactic backbone. Technical Report RR-3342, INRIA-Rocquencourt, Le Chesnay, France. URL `http://www.inria.fr/RRRT/RR-3342.html`.

Adriane Boyd. 2007. Discontinuity revisited: An improved conversion to context-free representations. In *Proceedings of the Linguistic Annotation Workshop*, pages 41–44.

Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2002. The Tiger treebank. In *Proceedings of the workshop on treebanks and linguistic theories*, pages 24–41.

Eugene Charniak, Mark Johnson, M. Elsner, J. Austerweil, D. Ellis, I. Haxton, C. Hill, R. Shrivaths, J. Moore, M. Pozar, et al. 2006. Multilevel coarse-to-fine PCFG parsing. In *Proceedings of NAACL-HLT*, pages 168–175.

Joan Chen-Main and Aravind K. Joshi. 2010. Unavoidable ill-nestedness in natural language and the adequacy of tree local-mctag induced dependency structures. In *Proceedings of TAG+*. URL `http://www.research.att.com/~srini/TAG+10/papers/chenmainjoshi.pdf`.

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

Pierluigi Crescenzi, Daniel Gildea, Aandrea Marino, Gianluca Rossi, and Giorgio Satta. 2011. Optimal head-driven parsing complexity for linear context-free rewriting systems. In *Proc. of ACL*.

Amit Dubey and Frank Keller. 2003. Parsing german with sister-head dependencies. In *Proc. of ACL*, pages 96–103.

Kilian Evang and Laura Kallmeyer. 2011. PLCFRS parsing of English discontinuous constituents. In *Proceedings of IWPT*, pages 104–116.

Daniel Gildea. 2010. Optimal parsing strategies for linear context-free rewriting systems. In *Proceedings of NAACL HLT 2010.*, pages 769–776.

Carlos Gómez-Rodríguez, Marco Kuhlmann, and Giorgio Satta. 2010. Efficient parsing of well-nested linear context-free rewriting systems. In *Proceedings of NAACL HLT 2010.*, pages 276–284.

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 2009*, pages 539–547.

Joshua Goodman. 2003. Efficient parsing of DOP with PCFG-reductions. In Rens Bod, Remko Scha, and Khalil Sima'an, editors, *Data-Oriented Parsing*. The University of Chicago Press.

Laura Kallmeyer. 2010. *Parsing Beyond Context-Free Grammars*. Cognitive Technologies. Springer Berlin Heidelberg.

Laura Kallmeyer, Timm Lichte, Wolfgang Maier, Yannick Parmentier, Johannes Dellert, and Kilian Evang. 2008. Tulipa: Towards a multi-formalism parsing environment for grammar engineering. In *Proceedings of the Workshop on Grammar Engineering Across Frameworks*, pages 1–8.

Laura Kallmeyer and Wolfgang Maier. 2010. Data-driven parsing with probabilistic linear context-free rewriting systems. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 537–545.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proc. of ACL*, volume 1, pages 423–430.

Marco Kuhlmann and Giorgio Satta. 2009. Treebank grammar techniques for non-projective dependency parsing. In *Proceedings of EACL*, pages 478–486.

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

Wolfgang Maier. 2010. Direct parsing of discontinuous constituents in German. In *Proceedings of the SPMRL workshop at NAACL HLT 2010*, pages 58–66.

Wolfgang Maier and Timm Lichte. 2009. Characterizing discontinuity in constituent treebanks.

In *Proceedings of Formal Grammar 2009*, pages 167–182. Springer.

Wolfgang Maier and Anders Søgaard. 2008. Treebanks and mild context-sensitivity. In *Proceedings of Formal Grammar 2008*, page 61.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

James D. McCawley. 1982. Parentheticals and discontinuous constituent structure. *Linguistic Inquiry*, 13(1):91–106.

Oliver Plaehn. 2004. Computing the most probable parse for a discontinuous phrase structure grammar. In Harry Bunt, John Carroll, and Giorgio Satta, editors, *New developments in parsing technology*, pages 91–106. Kluwer Academic Publishers, Norwell, MA, USA.

Remko Scha. 1990. Language theory and language technology; competence and performance. In Q.A.M. de Kort and G.L.J. Leerdam, editors, *Computertoepassingen in de Neerlandistiek*, pages 7–22. LVVN, Almere, the Netherlands. Original title: *Taaltheorie en taaltechnologie; competence en performance*. Translation available at `http://iaaa.nl/rs/LeerdamE.html`.

Stuart M. Shieber. 1985. Evidence against the context-freeness of natural language. *Linguistics and Philosophy*, 8:333–343.

Wojciech Skut, Brigitte Krenn, Thorten Brants, and Hans Uszkoreit. 1997. An annotation scheme for free word order languages. In *Proceedings of ANLP*, pages 88–95.

Andreas van Cranenburgh, Remko Scha, and Federico Sangati. 2011. Discontinuous data-oriented parsing: A mildly context-sensitive all-fragments grammar. In *Proceedings of SPMRL*, pages 34–44.

K. Vijay-Shanker and David J. Weir. 1994. The equivalence of four extensions of context-free grammars. *Theory of Computing Systems*, 27(6):511–546.

K. Vijay-Shanker, David J. Weir, and Aravind K. Joshi. 1987. Characterizing structural descriptions produced by various grammatical formalisms. In *Proc. of ACL*, pages 104–111.

David J. Weir. 1988. *Characterizing mildly context-sensitive grammar formalisms*. Ph.D. thesis, University of Pennsylvania. URL `http://repository.upenn.edu/dissertations/AAI8908403/`.