Discontinuous Data-Oriented Parsing: A mildly context-sensitive all-fragments grammar

Andreas van Cranenburgh, Remko Scha, and Federico Sangati
Institute for Logic, Language and Computation
University of Amsterdam
Science Park 904, 1098 XH Amsterdam, The Netherlands
acranenb@science.uva.nl, scha@uva.nl, f.sangati@uva.nl

Abstract

Recent advances in parsing technology have made treebank parsing with discontinuous constituents possible, with parser output of competitive quality (Kallmeyer and Maier, 2010). We apply Data-Oriented Parsing (DOP) to a grammar formalism that allows for discontinuous trees (LCFRS). Decisions during parsing are conditioned on all possible fragments, resulting in improved performance. Despite the fact that both DOP and discontinuity present formidable challenges in terms of computational complexity, the model is reasonably efficient, and surpasses the state of the art in discontinuous parsing.

1 Introduction

Many natural language phenomena are inherently not context-free, or call for structural descriptions that cannot be produced by a context-free grammar (Chomsky, 1957; Shieber, 1985; Savitch et al., 1987). Examples are extraposition, cross-serial dependencies and WH-inversion. However, relaxing the context-freeness assumption comes at the peril of combinatorial explosion.

This work aims to transcend two limitations associated with probabilistic context-free grammars. First in the sense of the representations produced by the parser, which allow constituents with gaps in their yields (see figure 1). Building on Kallmeyer and Maier (2010), we parse with a mildly context-sensitive grammar (LCFRS) that can be read off directly from a treebank annotated with discontinuous constituents.

Secondly, the statistical dependencies in our generative model are derived from arbitrarily large fragments from the corpus. We employ a Data-Oriented Parsing (DOP) model: a probabilistic tree-substitution grammar that employs probabilities derived from the frequencies of all connected fragments in the treebank (Bod, 1992; Bansal and Klein, 2010; Sangati and Zuidema, 2011). We generalize the DOP model to support discontinuity. This allows us to model complex constructions such as *NP kann man VVINF*.

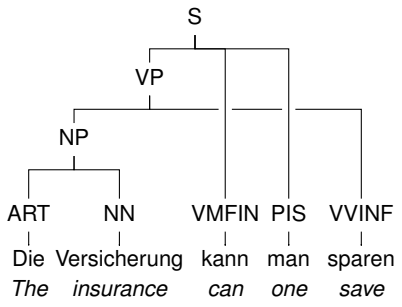


Figure 1: A discontinuous tree from the Negra corpus. Translation: *As for the insurance, one can save it.*

2 Motivation

Treebank grammars need not be mere exercises in machine learning; they may be significant steps toward cognitively viable models of human language processing. To develop the enterprise of corpus-based parsing in this more ambitious direction, substantial questions must be faced—methodological as well as technical ones.

All successful treebank grammars now employ very large numbers of rules; these rules, extracted from the corpus, are extremely specific and could never be motivated by abstract linguistic considerations. They either use large and complex fragments of the corpus trees (the Data-Oriented

Proceedings of the 2nd Workshop on Statistical Parsing of Morphologically-Rich Languages (SPMRL 2011), pages 34–44,
Dublin, Ireland, October 6, 2011.

Parsing approach), or they introduce specialized non-terminal labels which encode non-local information about the occurrence contexts of the nodes (the symbol-refinement approach). (These two methods are in fact not unrelated, as we shall discuss below.) Both approaches use "grammars" in the technical sense (bodies of rewrite rules), but not in the linguistic sense. The grammars do not encode general properties of the language, but specific properties of the corpus. All interesting treebank grammars are thus essentially exemplar-based. This characterization applies even more unavoidably to discriminative models that use the corpus trees directly without any kind of intervening grammar.

The exemplar-based models implemented by modern statistical parsers are of potential interest to the theory of language cognition, because they are the first formal alternatives to the rule-based models espoused by modern linguistics. But the tests that drive the development of statistical parsers are not sufficient to establish their plausibility as cognitive models. For corpus-based language processing work to become more relevant to language cognition, innovations are needed along several dimensions. One of them is evaluation methodology. F_1-scores on labeled bracketings constitute a poor criterion of excellence. More qualitative analyses are needed; we must try to understand what works and what doesn't.

In the present paper we do not deal with evaluation methods, but we take up another, related point. A precondition for a cognitively viable model of exemplar-based syntactic processing, is a cognitively viable definition of "syntactic Gestalts," i.e., of the kinds of objects that occur in the corpus and that are to be produced by the parser. It is customary to employ "syntactic surface trees" for this purpose, i.e., labeled trees with ordered branches, having the words of the sentence as their leaves. When we look at languages with a relatively free word order (which often correlates with a relatively non-trivial morphology), limitations of this approach become apparent. In such languages, the intuitive "parts" of the sentence need not coincide with contiguous surface constituents. By introducing movement features and allowing empty constituents, it is possible to encode non-local connections inside ordered surface trees; at the same time, functional feature labels may be added to the surface-syntactic categories. This approach was taken in the Penn Treebank (Taylor et al., 2003). In principle, this makes it possible to extract functional structures from the corpus-trees, and also, to evaluate parsers on their capacity to correctly construct the functional structures of test sentences. In practice, however, this is hardly ever done.

In English, the discrepancies between functional structure and surface constituents are less prominent than in many other languages. It is no coincidence that this issue was first squarely faced by the designers of the annotation conventions of the German Negra and Tiger corpora (Skut et al., 1997; Brants et al., 2002). They chose to use unordered trees (with crossing branches), allowing discontinuous constituents that correspond directly to the intuitively perceived argument structures. A model using a corpus annotated in this way, would be more interesting from a cognitive point of view, because it employs more plausible exemplars, and its output can be compared with more meaningful "gold trees."

Much of the work that used the Tiger and Negra corpora has failed to take advantage of this situation. Typically, these corpora are converted into traditional phrase-structure trees, so as to allow the application of the standard American techniques that were developed for English corpora. Exceptions were Plaehn (2004) and Maier (2010). The current paper follows up on that work, and integrates it with the Data-Oriented Parsing approach.

3 Discontinuity

Our symbolic grammar is a Linear Context-Free Rewriting System (Vijay-Shanker et al., 1987). LCFRS was introduced to subsume a wide variety of mildly context-sensitive formalisms (e.g., TAG, CCG, and even synchronous CFG). Intuitively it can be seen as a generalization of context-free grammar to other structures: rules are context-free, but instead of strings they rewrite tuples, trees or graphs. In our case 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 rules. 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.

A LCFRS is a tuple $G = \langle N, T, V, P, S\rangle$. N is a finite set of non-terminals; a function $\mathit{dim} : N \to \mathbb{N}$ 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 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 $1 \leq i \leq \mathit{dim}(A_i)$.

Rules 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.

A rule can be *instantiated* when its variables can be bound to 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 rules from a treebank (Maier and Søgaard, 2008), and the relative frequencies of rules form a maximum likelihood estimate, for a probabilistic LCFRS (PLCFRS). The tree in figure 1 decomposes into the following productions:

$\mathrm{S}(x_0x_1x_2x_3) \rightarrow \mathrm{VP}_2(x_0, x_3)\ \mathrm{VMFIN}(x_1)\ \mathrm{PIS}(x_2)$
$\mathrm{VP}_2(x_0, x_1) \rightarrow \mathrm{NP}(x_0)\ \mathrm{VVINF}(x_1)$
$\mathrm{NP}(x_0x_1) \rightarrow \mathrm{ART}(x_0)\ \mathrm{NN}(x_1)$
$\mathrm{ART}(\mathrm{Die}) \rightarrow \varepsilon$
$\mathrm{NN}(\mathrm{Versicherung}) \rightarrow \varepsilon$
$\mathrm{VVINF}(\mathrm{sparen}) \rightarrow \varepsilon$
$\mathrm{VMFIN}(\mathrm{kann}) \rightarrow \varepsilon$
$\mathrm{PIS}(\mathrm{man}) \rightarrow \varepsilon$

Discontinuous non-terminals are annotated with a number indicating their fan-out, to satisfy the restriction that each non-terminal type A can be mapped to a unique fan-out by $\mathit{dim}(A)$. A derivation proceeds by instantiating rules with subsequences of terminals in the input. Each non-terminal can be seen as a predicate holding over part of the sentence. Following the framework of deductive parsing (Nederhof, 2003), a sentence is parsed in a sequence of weighted deduction steps aiming at the goal theorem, viz., the start symbol covering the whole input in a single span.

Algorithm 1 A probabilistic agenda-based CKY parser for LCFRS.

```
initialize agenda A with POS tags
while A ≠ ∅
    ⟨I, x⟩ ← item with lowest score on agenda
    add ⟨I, x⟩ to C and F
    for all ⟨I', y⟩ deduced from
    {⟨I, J⟩, ⟨J, I⟩, ⟨I⟩ | J ∈ C}
        if I' ∉ A ∪ C
            enqueue ⟨I', y⟩ in A
        else if I' ∈ A ∧ y < score for I' in A
            add I' with old score to F
            update weight of I' in A to y
        else
            add ⟨I', y⟩ to F
```

This work employs an extended version of the agenda-based CKY parser for LCFRS in Kallmeyer and Maier (2010). The algorithm is Knuth's generalization of Dijkstra's shortest path algorithm to the case of hypergraphs, where the shortest path is the Viterbi derivation and the hypergraph is the chart defining possible derivations. A requirement of a CKY parser is that rules are binarized; we also restrict the parser to ordered rules for efficiency. The search space of a PCFG can be explored systematically from left to right with constituents of increasing size; this is what makes typical CKY parsers efficient. Unfortunately this approach does not translate to LCFRS because discontinuous constituents can cover any subsequence of the input. For this reason an explicit agenda has to be used, which we order by inside probability; alternatively the agenda can employ figures of merit or A* heuristics, but this is not explored in this work.

Our implementation[1] produces an exhaustive chart instead of stopping at the Viterbi derivation. The pseudo-code is given in algorithm 1. The

[1]The parser including source code is publicly available from `http://github.com/andreasvc/disco-dop`. Everything was written in Python, with pre-processing and evaluation making use of NLTK (Bird et al., 2009), and the parser and k-best extraction making use of Cython (Behnel et al., 2011) to translate Python code with static type annotations to native C code.

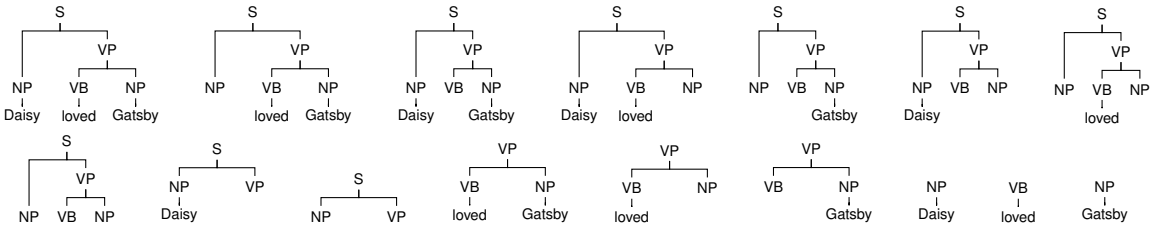


Figure 2: DOP1 fragments as extracted from the tree of "Daisy loved Gatsby."

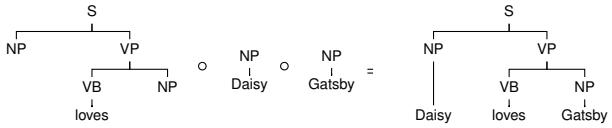


Figure 3: A DOP1 derivation. Note that "Daisy" becomes the subject, because fragments are combined with left-most substitution.

parser makes use of an agenda $\mathcal{A}$ (implemented as a priority queue with the decrease-key operation), a chart $\mathcal{C}$ with Viterbi probabilities, and the full chart $\mathcal{F}$ including suboptimal items. Both $\mathcal{A}$ and $\mathcal{C}$ store items I defined by a category and a span, paired with a weight x, while $\mathcal{F}$ stores weighted edges between items, representing a shared parse forest.

The algorithm is deceptively simple. Most of the work is in producing all items that can be deduced from a given item and items in the chart. This involves iterating over all grammar rules with matching labels, and verifying whether the yields of particular items can instantiate the rule. This can be optimized by representing yields as bit vectors and first verifying that the two yields do not overlap. The next step is to walk through both bit vectors in parallel and verifying the conditions for instantiating a rule. Rules that can be instantiated are given a score that is the sum of the weights (e.g., log probabilities) of their RHS.

Any LCFRS can be binarized (as required by the CKY parser) and parsed in $\mathcal{O}(|G| \cdot |w|^{3\varphi})$ time, where $|G|$ is the size of the grammar, $|w|$ is the length of the sentence, and φ is the fan-out after binarization (Gómez-Rodríguez et al., 2009). The fan-out may increase due to binarization, but in our experiments this was no cause for concern. The degree of the polynomial reflects the maximal number of comparisons needed to determine whether a rule can be applied; a binarized rule has three non-terminals with φ components each in the worst case.

This stands in contrast to previous formalisms for discontinuous parsing (Johnson, 1985; Plaehn, 2004), which have exponential time complexity. The difference is that in a LCFRS, the productions are formally context-free in the sense that they are applied without knowledge of the rest of the derivation, and they are restricted to cover certain spans in a particular order, which avoids having to enumerate the exponential number of possible discontinuous spans or permutations of the sentence.

4 Data-Oriented Parsing

Data-Oriented parsing (DOP) was introduced by Scha (1990) as both a cognitive and computational approach to analyzing language in terms of exemplars instead of rules. Concretely this works by allowing arbitrarily large fragments in the corpus to recombine with each other. The intuition is that, in terms of cognitive load, reuse is cheaper than computation, so remembering fragments is more effective than deriving them anew.

The first concrete DOP model was DOP1 (Bod, 1992). In DOP1, a fragment is defined as a connected subset of nodes in a tree, such that for every non-terminal node, the fragment either has all children in common with the tree, or none. In the latter case the node is a frontier node, which functions as a substitution site (this is analogous to an open slot in a construction). The weight of a fragment is its relative frequency in the training data. Figure 2 illustrates the DOP1 fragments extracted from the tree of an example sentence. Note that the smallest fragments

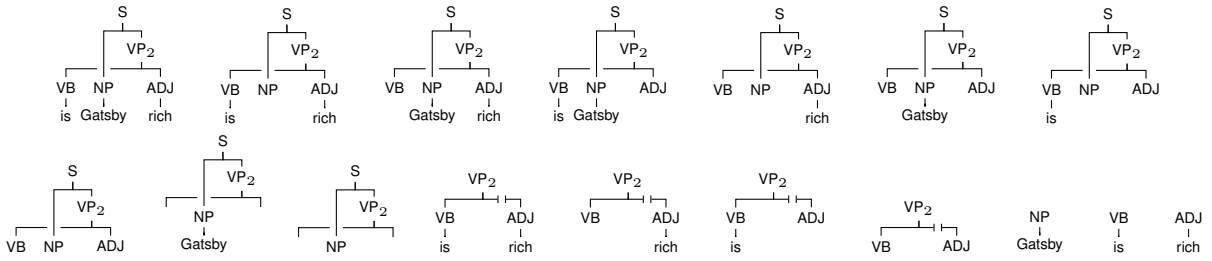


Figure 4: Discontinuous fragments as extracted from the tree of "is Gatsby rich?"

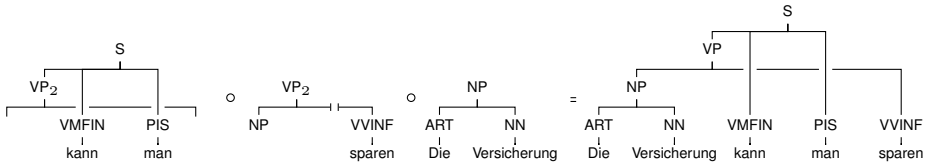


Figure 5: A discontinuous DOP derivation of the tree in figure 1.

correspond to CFG productions, such that a CFG is a DOP1 model restricted to fragments of depth 1.

A derivation is defined as a sequence of fragments combined through left-most substitution. Left-most substitution is defined for any two trees t_1 and t_2, such that t_1 has a frontier node labeled X and root$(t_2) = X$; the result of $t_1 \circ t_2$ is a new tree where t_2 is substituted for the first frontier node labeled X in t_1. The probability of a derivation is the product of the weights of its fragments. Figure 3 shows a derivation with the fragments in figure 2.

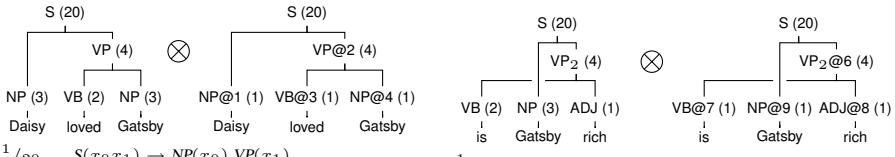
We can generalize the DOP1 model to the case of discontinuous trees. By substituting an LCFRS for the CFG backbone of DOP1, we obtain a discontinuous DOP model—Disco-DOP. The Disco-DOP model employs the same definition of a fragment, but applies it to a broader class of trees. Figure 4 shows the fragments extracted from a discontinuous tree. Note that when a discontinuous node becomes a frontier node, it specifies where its yield will end up with respect to the yield of other nodes in the fragment. Figure 5 shows a derivation with discontinuous fragments of the tree in figure 1.

Parsing with all fragments explicitly is not possible, as there are exponentially many. One solution is to select a subset of fragments (e.g., Sangati and Zuidema, 2011). In this work we employ the approach introduced by Goodman (1996, 2003), who defines a PCFG which decomposes the probabilities of fragments into several PCFG productions, such that the same parse trees can be recovered as in a DOP model with explicitly represented fragments. This reduction generalizes straightforwardly to a PLCFRS.

Each node A in the training corpus is decorated with a unique address A_j. Given a node A_j with children B_k and C_l, the number of fragments headed by A_j is given by $a_j = (b_k + 1)(c_l + 1)$. The total number of subtrees (fragments) for A is given by $a = \sum_j a_j$. We also apply a normalization factor $\bar{a}$ which is the frequency of non-terminals of type A in the training data. The probabilities of Disco-DOP derivations can then be encoded in the following reduction, applied to each production in the training corpus:

$$
\begin{array}{lcll}
A_j(\vec{\alpha}) & \rightarrow & B(\vec{\alpha_B})\ C(\vec{\alpha_C}) & (1/a_j) \\
A_j(\vec{\alpha}) & \rightarrow & B_k(\vec{\alpha_B})\ C(\vec{\alpha_C}) & (b_k/a_j) \\
A_j(\vec{\alpha}) & \rightarrow & B(\vec{\alpha_B})\ C_l(\vec{\alpha_C}) & (c_l/a_j) \\
A_j(\vec{\alpha}) & \rightarrow & B_k(\vec{\alpha_B})\ C_l(\vec{\alpha_C}) & (b_k c_l/a_j) \\
A(\vec{\alpha}) & \rightarrow & B(\vec{\alpha_B})\ C(\vec{\alpha_C}) & (1/(a\bar{a})) \\
A(\vec{\alpha}) & \rightarrow & B_k(\vec{\alpha_B})\ C(\vec{\alpha_C}) & (b_k/(a\bar{a})) \\
A(\vec{\alpha}) & \rightarrow & B(\vec{\alpha_B})\ C_l(\vec{\alpha_C}) & (c_l/(a\bar{a})) \\
A(\vec{\alpha}) & \rightarrow & B_k(\vec{\alpha_B})\ C_l(\vec{\alpha_C}) & (b_k c_l/(a\bar{a}))
\end{array}
$$

Where $\vec{\alpha}$ refers to the arguments of the LHS non-terminal. Each addressed non-terminal represents an internal node of a fragment, while the unaddressed nodes represent both the root and the frontier nodes of fragments. The latter allow a switch from one fragment to another during parsing, viz. they simulate substitution sites of DOP fragments.



$^{1}/_{20}$ $S(x_0x_1) \rightarrow NP(x_0)\ VP(x_1)$
$^{4}/_{20}$ $S(x_0x_1) \rightarrow NP(x_0)\ VP@2(x_1)$
$^{1}/_{20}$ $S(x_0x_1) \rightarrow NP@1(x_0)\ VP(x_1)$
$^{4}/_{20}$ $S(x_0x_1) \rightarrow NP@1(x_0)\ VP@2(x_1)$
$^{1}/_{4}$ $VP(x_0x_1) \rightarrow VB(x_0)\ NP(x_1)$
$^{1}/_{4}$ $VP(x_0x_1) \rightarrow VB(x_0)\ NP@4(x_1)$
$^{1}/_{4}$ $VP(x_0x_1) \rightarrow VB@3(x_0)\ NP(x_1)$
$^{1}/_{4}$ $VP(x_0x_1) \rightarrow VB@3(x_0)\ NP@4(x_1)$
$^{1}/_{4}$ $VP@2(x_0x_1) \rightarrow VB(x_0)\ NP(x_1)$
$^{1}/_{4}$ $VP@2(x_0x_1) \rightarrow VB(x_0)\ NP@4(x_1)$
$^{1}/_{4}$ $VP@2(x_0x_1) \rightarrow VB@3(x_0)\ NP(x_1)$
$^{1}/_{4}$ $VP@2(x_0x_1) \rightarrow VB@3(x_0)\ NP@4(x_1)$
$^{1}/_{3}$ $NP(Daisy) \rightarrow \epsilon$
$^{1}/_{1}$ $NP@1(Daisy) \rightarrow \epsilon$
$^{1}/_{2}$ $VB(loved) \rightarrow \epsilon$
$^{1}/_{1}$ $VB@3(loved) \rightarrow \epsilon$
$^{2}/_{3}$ $NP(Gatsby) \rightarrow \epsilon$
$^{1}/_{1}$ $NP@4(Gatsby) \rightarrow \epsilon$

$^{1}/_{20}$ $S(x_0x_2x_1) \rightarrow VP_2(x_0, x_2)\ NP(x_2)$
$^{1}/_{20}$ $S(x_0x_2x_1) \rightarrow VP_2(x_0, x_2)\ NP@9(x_2)$
$^{4}/_{20}$ $S(x_0x_2x_1) \rightarrow VP_2@6(x_0, x_2)\ NP(x_2)$
$^{4}/_{20}$ $S(x_0x_2x_1) \rightarrow VP_2@6(x_0, x_2)\ NP@9(x_2)$
$^{1}/_{4}$ $VP_2(x_0, x_1) \rightarrow VB(x_0)\ ADJ(x_1)$
$^{1}/_{4}$ $VP_2(x_0, x_1) \rightarrow VB(x_0)\ ADJ@8(x_1)$
$^{1}/_{4}$ $VP_2(x_0, x_1) \rightarrow VB@7(x_0)\ ADJ(x_1)$
$^{1}/_{4}$ $VP_2(x_0, x_1) \rightarrow VB@7(x_0)\ ADJ@8(x_1)$
$^{1}/_{4}$ $VP_2@6(x_0, x_1) \rightarrow VB(x_0)\ ADJ(x_1)$
$^{1}/_{4}$ $VP_2@6(x_0, x_1) \rightarrow VB(x_0)\ ADJ@8(x_1)$
$^{1}/_{4}$ $VP_2@6(x_0, x_1) \rightarrow VB@7(x_0)\ ADJ(x_1)$
$^{1}/_{4}$ $VP_2@6(x_0, x_1) \rightarrow VB@7(x_0)\ ADJ@8(x_1)$
$^{1}/_{1}$ $ADJ(rich) \rightarrow \epsilon$
$^{1}/_{1}$ $ADJ@8(rich) \rightarrow \epsilon$
$^{1}/_{1}$ $NP@9(Gatsby) \rightarrow \epsilon$
$^{1}/_{2}$ $VB(is) \rightarrow \epsilon$
$^{1}/_{1}$ $VB@7(is) \rightarrow \epsilon$

Figure 6: The pairwise Cartesian product of the productions in the original and the addressed tree gives the productions in the reduction.

The use of the normalization factor is called the Equal Weights Estimate; this formulation follows Bod (2003). Goodman (2003) first suggested this normalization but his formula appears to contain a mistake, having a_j in the denominator of the last four rules instead of a. The normalization is intended to counter the bias for large subtrees in DOP1—when all fragments are considered, the majority will consist of large fragments, which results in the majority of probability mass being assigned to rare, large fragments.

Intuitively, the reduction can be seen as state-splitting in the limit. A state-split partitions a non-terminal into two or more new non-terminals to cover more specific and fine-grained contexts. Taken to the extreme, we can keep splitting non-terminals until each resulting non-terminal refers to one specific occurrence of that non-terminal in a single sentence, which greatly increases the amount of hierarchical information that can be extracted and exploited from the training corpus. This is exactly what happens in Goodman's reduction. Compared to other automatic state-splitting approaches such as latent variable grammars, this approach has the advantage of being conceptually much simpler.

Figure 6 shows a concrete example of the reduction, using the tree from figure 1. The eight productions per node of the reduction can be considered as the pairwise Cartesian product of the original production and the one with addressed nodes. To get the reduction of a complete tree, this operation is applied to all productions of both trees in parallel. The probabilities are derived from the number of subtrees, shown in brackets after the node labels. The normalization step has been left out, for simplicity's sake. Productions without addressed nodes, i.e., the original productions, will recur, and their probabilities must be summed. In our case productions are considered equivalent when both the non-terminals and their arguments match.

The large number of non-terminals and productions in this grammar make parsing with this reduction inefficient. In order to optimize parsing with the DOP reduction, we apply coarse-to-fine inference in the spirit of Bansal and Klein (2010). The principle is to parse first with a coarse grammar, in this case the treebank PLCFRS, and use information from the resulting chart to prune parsing with the full grammar, in this case the DOP reduction. Figure 7 illustrates the approach. The fine grammar is *projected* onto the coarse grammar by mapping nodes A_j to the original nodes A. Pruning is implemented by blocking items $\langle A, \vec{a} \rangle$ which are not part of the n-best derivations in the coarse chart; such items are simply prevented from entering the agenda. Although this approach is reminiscent of re-ranking (Charniak and Johnson, 2005), in our approach items can recombine to form parse trees not present in the n-best derivations. We use $n = 50$ in all experiments.

We aim at maximizing the chance of obtaining the correct structure for a given sentence, viz. finding its

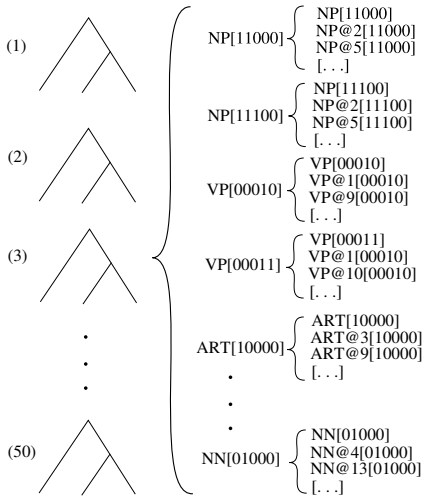


Figure 7: The coarse-to-fine inference. On the left are the n-best derivations from the coarse chart. In the middle the chart items (category and spans) extracted from those derivations. On the right are all items for the fine grammar that map to these coarse items. The latter will be the only items allowed to enter the agenda when parsing with the fine grammar.

most probable parse (MPP). In DOP the probability of a parse tree is the sum of all its possible derivations. However, as there can be exponentially many derivations for each parse tree, finding the MPP is NP-hard (Sima'an, 1996). Therefore the MPP must be approximated with a restricted set of derivations: a list of derivations is produced using the algorithm of Huang and Chiang (2005) which efficiently extracts the exact k-best list from an exhaustive chart with Viterbi probabilities. We use $k = 10,000$ in all experiments. After this list is obtained, we sum the probabilities of all derivations generating the same parse tree by applying the projection, and select the best one. Note that probabilities of DOP derivations are spread over multiple derivations in Goodman's reduction. In our experiments the list of derivations in the reduction often collapses to just 5 parse trees.

words	train	test	PLCFRS rules	Disco-DOP rules
≤ 15	9025	1015	24020	678659
≤ 25	14870	1639	53773	1769507
≤ 30	16490	1845	50381	1799797

Table 1: Number of sentences and rules.

5 Experiments

We evaluate on version 2 of the German Negra treebank (Skut et al., 1997). Results are for models based on splits of 90% training data and 10% test data. The setup follows Kallmeyer and Maier (2010) as much as possible. The parser is presented with (gold) part-of-speech tags from the treebank. The DOP model, however, does exploit its knowledge of lexical dependencies by using fragments with terminals. In a pre-processing step, function labels are discarded and all punctuation is lowered to the best matching constituent. Heads are marked using the head finding rules for the Negra corpus used by the Stanford parser, after which trees are binarized head-outward (Klein and Manning, 2003a,b). The markovization setting is v=1 (i.e., no parent annotation), and $h \in \{1, 2, \infty\}$, dictated by efficiency concerns. Lower values for h give better performance because they allow more flat structures to be covered through re-combinations of parts of different constituents. However, this also greatly increases the number of possible edges which have to be explored. For this reason we had to increase the value of h for parsing longer sentences, at the cost of decreased performance and coverage. Figure 8 illustrates the binarization. With these settings the grammar has a fan-out of 5 for the grammar of up to 15 word sentences, and a fan-out of 7 for the other two. Table 1 lists the size of the training & test corpora and their grammars for the respective length restrictions. Note that Kallmeyer and Maier (2010) apply the length restriction before the 90-10 split, but the difference is not more than 12 sentences.

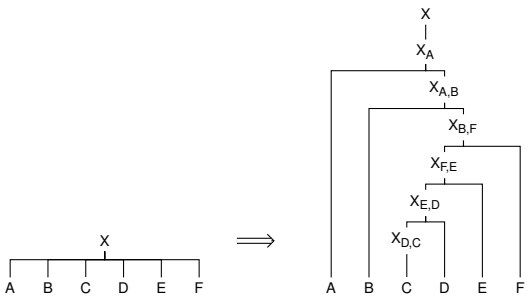


Figure 8: A head-outward binarization with h=2 v=1 markovization; C is the head node.

NEGRA	words	coverage	LP	LR	F_1	EX
Plaehn (2004): DPSG	≤ 15	96.04	73.61	72.72	73.16	39.0
Kallmeyer and Maier (2010): PLCFRS	≤ 15	-	-	-	81.27	-
This work: Disco-DOP v=1, h=1	≤ 15	99.90	84.09	85.03	**84.56**	54.68
Kallmeyer and Maier (2010): PLCFRS	≤ 25	99.45	73.03	73.46	73.25	-
This work: Disco-DOP v=1, h=2	≤ 25	98.90	78.26	79.37	**78.81**	39.11
Maier and Kallmeyer (2010): PLCFRS	≤ 30	97.00	72.39	70.68	71.52	-
This work: Disco-DOP v=1, h=∞	≤ 30	96.59	73.05	74.93	**73.98**	34.80

Table 2: Results for discontinuous parsing on the Negra treebank.

Evaluation is performed using a generalization of the PARSEVAL measures, which compares bracketings of the form $\langle A, \vec{a} \rangle$ where $\vec{a}$ is the yield described by a tuple of intervals (Maier, 2010); we used Maier's publicly available implementation.[2] We use PARSEVAL, in spite of its serious shortcomings (Rehbein and van Genabith, 2007), to enable comparison with previous work. Unparsed sentences are assigned a baseline parse with all tags directly under the root node.

Our model performs consistently better than previous results on discontinuous parsing; see table 2 for the results, including comparisons to previous work. Figure 9 plots the time required to parse sentences of different lengths with v=1 h=2, showing a strikingly steep curve, which makes clear why parsing sentences longer than 25 words was not feasible with these settings. The coarse-to-fine inference appears to work rather well, apparently displaying a linear observed time complexity on the DOP grammar; unfortunately exhaustive parsing with the coarse grammar forms a bottleneck. The total time to parse was 1, 16 and 52 hours for 15, 25, and 30 words respectively, using about 4 GB of memory.

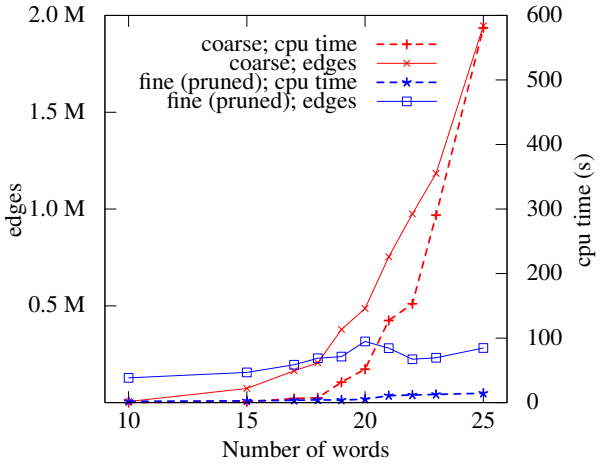


Figure 9: Efficiency as a function of the number of words in the coarse (PLCFRS) and the fine stage (Disco-DOP).The data are from parsing 10 Negra sentences, hand-picked to illustrate the worst case.

6 Remaining challenges

From these results it may appear as if the combination of the formalism and treebank parsing forms an inherent barrier to parsing longer sentences. Even Kallmeyer and Maier (2010), who employ a precomputed table of outside estimates, could not parse beyond 30 words, because of memory limitations. Their four-dimensional table is indexed on non-terminals, span length, length of gaps, and number of words to the left and right. This implies a space complexity of $\mathcal{O}(|N| \cdot n^3)$ where $|N|$ is the number of non-terminals and n the maximum sentence length. With approximately 12,000 non-terminals as cited by Kallmeyer and Maier (2010), a limit of 100 words per sentence, and double precision, this implies a table of 96 GB.

Another issue is that it is not clear whether obtaining k-best lists with these estimates works well or is possible at all. Pauls and Klein (2009) present an algorithm for extending an A* parser to a parser that builds the k-best derivations during parsing, but the estimates of Kallmeyer and Maier (2010) are not monotone, a property which is assumed by Pauls and Klein (2009). Monotonicity guarantees that the Viterbi parse will be found first.

We consider extending the coarse-to-fine approach to be more promising. Instead of only making the

[2] Cf. `http://www.wolfgang-maier.net/rparse`

categories coarser, we can also resort to a coarser formalism. Following Barthélemy et al. (2001), we can extract a grammar that defines a superset of the language we want to parse, but with a fan-out of 1. Concretely, a context-free grammar can be read off from discontinuous trees that have been transformed to context-free trees by the procedure introduced by 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 can mark components with an index to reduce overgeneration). Because Boyd's transformation is reversible, derivations from this grammar can be converted back to discontinuous trees, and can guide parsing with the LCFRS. Results with this approach will be reported in future work.

Aside from these technical issues, many linguistic features have been glossed over in this work to limit its scope. A proper parser and evaluation should work with grammatical functions as well, and parsing languages with less strict word-order implies that morphology provides important information about constituents that have been moved or extraposed. Movement and extraposition could also be modeled statistically, which can reduce data sparsity. Scrambling is known to be beyond the power of LCFRS (Becker et al., 1992); however, the question is whether it needs to be part of the formalism at all. As the work of Tsarfaty (2010) shows, it is possible to incorporate morphology, grammatical functions, and word-order in a statistical model built on a PCFG backbone. Perhaps this approach could be combined with the work presented here, such that it can produce discontinuous structures using LCFRS, and to weaken its independence assumptions through a DOP model.

7 Conclusion

A data-oriented model of discontinuous phrase-structure has been presented which outperforms all previously published results. This has been achieved by combining a variety of techniques: a linear context-free rewriting system as the symbolic grammar, data-oriented parsing as the probabilistic framework, a general method for enumerating k-best derivations from a chart, and a coarse-to-fine optimization to tame the complexity of DOP.

It turns out that using a grammar formalism with a parsing complexity that is well beyond cubic is not an impediment for making a DOP model with considerably better performance. The remaining difficulty with parsing longer sentences lies squarely on the side of discontinuity, not DOP. It is quite plausible that further innovations in binarization, pruning and estimates will enable parsing longer sentences.

Acknowledgments

We are grateful to Wolfgang Maier for correspondence and making the code of his parser `rparse` available.

References

Mohit Bansal and Dan Klein. 2010. Simple, accurate parsing with an all-fragments grammar. In *Proceedings of ACL*, pages 1098–1107.

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

Tilman Becker, Owen Rambow, and Michael Niv. 1992. The derivational generative power of formal systems or scrambling is beyond LCFRS. Technical Report IRCS-92-38, Institute for research in cognitive science. URL `ftp://ftp.cis.upenn.edu/pub/ircs/tr/92-38.ps.Z`.

Stefan Behnel, Robert Bradshaw, Craig Citro, Lisandro Dalcin, Dag Sverre Seljebotn, and Kurt Smith. 2011. Cython: The best of both worlds. *Computing in Science and Engineering*, 13:31–39.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly Media.

Rens Bod. 1992. A computational model of language performance: Data-oriented parsing. In *Proceedings COLING'92 (Nantes, France)*, pages 855–859.

Rens Bod. 2003. An efficient implementation of a new DOP model. In *Proceedings of EACL*, volume 1, pages 19–26. URL `http://www.ldc.upenn.edu/acl/E/E03/E03-1005.pdf`.

Pierre Boullier. 1998. Proposal for a natu-

ral 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 and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of ACL*, pages 173–180.

Noam Chomsky. 1957. *Syntactic Structures*. Mouton, The Hague.

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. 1996. Efficient algorithms for parsing the DOP model. In *Proceedings of EMNLP*, pages 143–152.

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.

Liang Huang and David Chiang. 2005. Better k-best parsing. In *Proceedings of IWPT 2009*, pages 53–64.

Mark Johnson. 1985. Parsing with discontinuous constituents. In *Proceedings of ACL*, pages 127–132.

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. 2003a. Accurate unlexicalized parsing. In *Proceedings of ACL*, volume 1, pages 423–430.

Dan Klein and Christopher D. Manning. 2003b. Fast exact inference with a factored model for natural language parsing. *Advances in neural information processing systems*, pages 3–10.

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 Laura Kallmeyer. 2010. Discontinuity and non-projectivity: Using mildly contextsensitive formalisms for data-driven parsing. In *Proceedings of TAG*, volume 10.

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

Mark-Jan Nederhof. 2003. Weighted deductive parsing and Knuth's algorithm. *Computational Linguistics*, 29(1):135–143.

Adam Pauls and Dan Klein. 2009. K-best A* parsing. In *Proceedings of ACL*, volume 2, pages 958–966.

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.

Ines Rehbein and Josef van Genabith. 2007. Evaluating evaluation measures. In *NODALIDA 2007 Conference Proceedings*, pages 372–379. URL http://doras.dcu.ie/15237.

Federico Sangati and Willem Zuidema. 2011. Accurate parsing with compact tree-substitution grammars: Double-DOP. In *Proceedings of EMNLP*, pages 84–95. URL http://www.aclweb.org/anthology/D11-1008.

Walter J. Savitch, Emmon Bach, W. Marsh, and Gila Safran-Naveh. 1987. *The Formal Complexity of Natural Language*, volume 33 of *Linguistics and Philosophy*. D. Reidel.

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.

Khalil Sima'an. 1996. Computational complexity of probabilistic disambiguation by means of tree-grammars. In *Proceedings of the 16th conference on Computational linguistics*, volume 2, pages 1175–1180.

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.

Ann Taylor, Mitchell Marcus, and Beatrice Santorini. 2003. The penn treebank: an overview. In Anne Abeillé, editor, *Treebanks: Building and using parsed corpora*, pages 5–22. Springer.

Reut Tsarfaty. 2010. *Relational-realizational parsing*. Ph.D. thesis, University of Amsterdam. URL http://dare.uva.nl/record/335795.

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