Discontinuous Parsing with an Efficient and Accurate DOP Model

Andreas van Cranenburgh[*†]
[*]Huygens ING
Royal Dutch Academy of Science
P.O. box 90754, 2509 LT, The Hague, The Netherlands
`andreas.van.cranenburgh@huygens.knaw.nl`

Rens Bod[†]
[†]Institute for Logic, Language and Computation
University of Amsterdam
Science Park 904, 1098 XH, The Netherlands
`rens.bod@uva.nl`

Abstract

We present a discontinuous variant of tree-substitution grammar (TSG) based on Linear Context-Free Rewriting Systems. We use this formalism to instantiate a Data-Oriented Parsing model applied to discontinuous treebank parsing, and obtain a significant improvement over earlier results for this task. The model induces a TSG from the treebank by extracting fragments that occur at least twice. We give a direct comparison of a tree-substitution grammar implementation that implicitly represents all fragments from the treebank, versus one that explicitly operates with a significant subset. On the task of discontinuous parsing of German, the latter approach yields a 16 % relative error reduction, requiring only a third of the parsing time and grammar size. Finally, we evaluate the model on several treebanks across three Germanic languages.

1 Introduction

A Probabilistic Context-Free Grammar (PCFG) extracted from a treebank (Charniak, 1996) provides a simple and efficient model of natural language syntax. However, its independence assumptions are too strong to form an accurate model of language syntax. A tree-substitution grammar (TSG) provides a generalization of context-free grammar (CFG) that operates with larger chunks than just single grammar productions. A probabilistic TSG can be seen as a PCFG in which several productions may be applied at once, capturing structural relations between those productions.

Tree-substitution grammars have numerous applications. They can be used for statistical parsing, such as with Data-Oriented Parsing (DOP; Scha, 1990; Bod et al., 2003; Sangati and Zuidema, 2011) and Bayesian TSGs (Post and Gildea, 2009; Cohn et al., 2010; Shindo et al., 2012). Other applications include grammaticality judgements (Post, 2011), multi-word expression identification (Green et al., 2011), stylometry and authorship attribution (Bergsma et al., 2012; van Cranenburgh, 2012c), and native language detection (Swanson and Charniak, 2012).

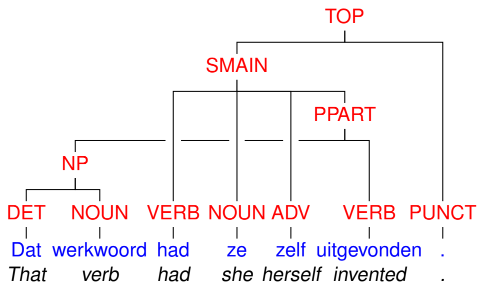

TOP
SMAIN
PPART
NP
DET
NOUN
VERB
NOUN
ADV
VERB
PUNCT
Dat werkwoord had ze zelf uitgevonden .
That verb had she herself invented .

Figure 1: A tree from the Dutch Alpino treebank (van der Beek et al., 2002). PPART is a discontinuous constituent (indicated with crossing branches) due to its extraposed NP object. Translation: *She had invented that verb herself.*

An orthogonal way to extend the domain of locality of PCFG is to employ a formalism that produces richer derived structures. An example of this is to allow for producing trees with discontinuous constituents (cf. figure 1 for an example). This can be achieved with (string rewriting) Linear Context-Free Rewriting Systems (LCFRS; Vijay-Shanker et al., 1987). Kallmeyer and Maier (2010, 2013) use this formalism for statistical parsing with discontinuous constituents.

The notion of discontinuous constituents in annotation is useful to bridge the gap between the information represented in constituency and dependency structures. Constituency structures capture the hierarchical structure of phrases—which is useful for identifying re-

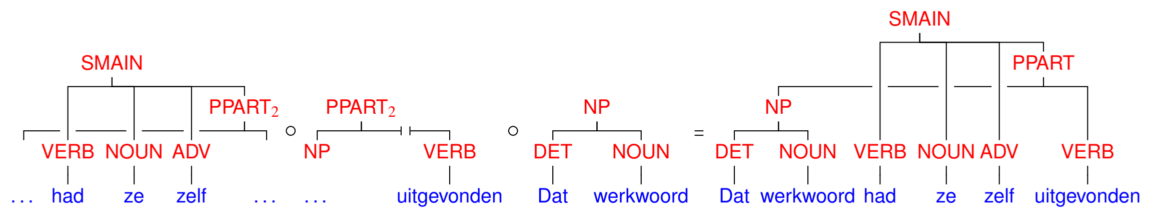


Figure 2: A discontinuous tree-substitution derivation of the tree in figure 1. Note that in the first fragment, which has a discontinuous frontier non-terminal, the destination for the discontinuous spans is marked in advance, shown as ellipses.

usable elements; discontinuous constituents extend this to allow for arbitrary long-distance relations that may arise due to such phenomena as extraposition and word-order freedom. The essential difference between traditional phrase-structure trees and discontinuous ones is that the former is a two-dimensional (planar) structure of a one-dimensional surface form, while the latter allows for a higher dimensional structure. This can be contrasted with the alternative of introducing artificial empty categories, which encode the same information but in the labels instead of the tree structure.

The two approaches of tree-substitution and discontinuity have been synthesized into a discontinuous all-fragments grammar (van Cranenburgh et al., 2011; van Cranenburgh, 2012a) defined implicitly through a reduction (Goodman, 2003). The present paper extends this work. We present a grammar transformation to parse with an arbitrary discontinuous TSG and present results with this new implementation, using TSGs induced by extracting fragments from treebanks. Our method outperforms previous results for discontinuous constituency parsing.

2 Grammar formalisms

In this section we formulate how a discontinuous Tree-Substitution Grammar can be implemented using a Linear Context-Free Rewriting System as the base grammar.

2.1 Linear Context-Free Rewriting Systems

LCFRS can be seen as the discontinuous equivalent of CFG, and its probabilistic variant can be used as a discontinuous treebank grammar. LCFRS generalizes over CFG by rewriting a fixed number of strings at a time for each non-terminal. This number, the measure of discontinuity in a constituent, is called the fan-out. A CFG is an LCFRS with a maximum fan-out of 1. In this paper we use the simple RCG notation (Boullier, 1998) for LCFRS. We will define a restricted variant that operates on unary and binary productions.

A binarized, string-rewriting LCFRS is a 4-tuple $G = \langle N, T, P, S \rangle$. N and T are disjoint finite sets of non-terminals and terminals, respectively. A function $\varphi : N \rightarrow \{1, 2, \ldots, \}$ specifies the unique fan-out for every non-terminal symbol. S is the distinguished start symbol with $S \in N, \varphi(S) = 1$. We assume a set V of variables of the form b_i and c_i with $i \in \mathbb{N}$. P is a finite set of productions, which come in three forms:

$$A(\alpha_1, \ldots, \alpha_{\varphi(A)}) \rightarrow B(b_1, \ldots, b_{\varphi(B)})\ C(c_1, \ldots, c_{\varphi(C)})$$
$$A(\alpha_1, \ldots, \alpha_{\varphi(A)}) \rightarrow B(b_1, \ldots, b_{\varphi(B)})$$
$$D(t) \rightarrow \varepsilon$$

where $A, B, C, D \in N$, $\alpha_i \in V^*$ for $1 \leqslant i \leqslant \varphi(A_i)$, $t \in T$, and $\varphi(D) = 1$.

Productions must be *linear*: if a variable occurs in a production, it occurs exactly once on the left hand side (LHS), and exactly once on the right hand side (RHS). A production 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 production 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. In the remainder we will notate discontinuous non-terminals with a subscript indicating their fan-out.

LCFRS productions may be induced from a discontinuous tree, using a procedure described in Maier and Søgaard (2008). We extend this procedure to handle frontier non-terminals, i.e., non-terminals that do not dominate terminals or non-terminals, because they are part of a tree fragment extracted from a treebank.

Given a discontinuous tree, we extract a grammar production for each non-leaf non-terminal node. The node itself forms the LHS non-terminal, and the non-terminals that are immediately dominated by it forms the RHS. The yield of each node is converted to a sequence of indices reflecting sentence order; this includes the spans of any frontier non-terminals. For each span in the yield, identified as a maximally continuous range in the sequence of indices, a variable is introduced. The variables become the arguments of the LHS and RHS non-terminals, ordered as in the original yield. For the RHS non-terminals, each argument is a single variable. The arguments to the LHS non-terminal consist of a tuple of one or more variables corresponding to consecutive ranges of the sequence of indices. Pre-terminals yield a production with their terminal as a direct argument to the pre-terminal, and an empty RHS. Frontier non-terminals only appear on the RHS of a production. See figure 3 for examples of LCFRS productions extracted from discontinuous trees.

2.2 Tree-Substitution Grammar

In this section we present a reduction of an arbitrary discontinuous TSG to a string-rewriting LCFRS. We first look at general strategies for reducing a TSG to a simpler formalism, and then show that these also apply for the discontinuous case.

A TSG is a tuple $\langle N, T, R, S\rangle$. N and T are disjoint finite sets of non-terminals and terminals, respectively. R is a finite set of elementary trees of depth greater than or equal to 1. Elementary trees from R are combined by substituting a derived tree rooted at a non-terminal X at some leaf node in an elementary tree with a frontier node labeled with X. Derived trees rooted at the start symbol S with leaf nodes labeled by terminal symbols are taken to be the trees generated by the grammar. See figure 2 for an example of a TSG derivation; this derivation contains discontinuous constituents, how these are combined with a TSG shall be explained below.

Goodman (2003) gives a reduction to a PCFG for the special case of a TSG based on all fragments from a given treebank. This reduction is stochastically equivalent after the summation of probabilities from equivalent derivations—however, it does not admit parsing of TSGs with arbitrary sets of elementary trees or arbitrary probability models.

We use a transformation based on the one given in Sangati and Zuidema (2011). Internal nodes are removed from elementary trees, yielding a flattened tree of depth 1. We binarize this flattened tree with a left-factored binarization that adds unique identifiers to every intermediate node introduced by the binarization. In order to separate phrasal and lexical productions, a new POS tag is introduced for each terminal, which selects for that specific terminal. A sequence of productions is then read off from the transformed tree. The unique identifier in the first production is used to look up the original elementary tree in a backtransform table,[1] which is used to restore the internal nodes after parsing. The weight associated with an elementary tree carries over to the first production it produces; the rest of the productions are assigned a weight of 1.

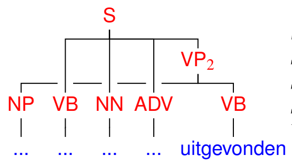
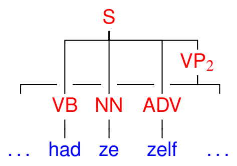
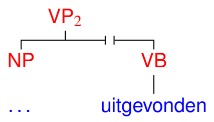
Elementary tree	Productions
 S VP$_2$ NP VB NN ADV VB ... uitgevonden	$S(ab) \rightarrow S^1(a)\ VB(b)$ $S^1(ab) \rightarrow S^2(a)\ ADV(b)$ $S^2(ab) \rightarrow S^3(a)\ NN(b)$ $S^3(ab) \rightarrow NP(a)\ VB^4(b)$ $VB^4(\text{uitgevonden}) \rightarrow \varepsilon$
 S VP$_2$ VB NN ADV ... had ze zelf ...	$S(abc) \rightarrow S_2^5(a,c)\ ADV^6(b)$ $S_2^5(ab,c) \rightarrow S_2^7(a,c)\ NN(b)$ $S_2^7(ab,c) \rightarrow VP_2(a,c)\ VB^8(b)$ $VB^8(\text{had}) \rightarrow \varepsilon$ $NN^7(\text{ze}) \rightarrow \varepsilon$ $ADV^6(\text{zelf}) \rightarrow \varepsilon$
	$VP_2(a,b) \rightarrow NP(a)\ VB^9(b)$ $VB^9(\text{uitgevonden}) \rightarrow \varepsilon$

Figure 3: Transforming a tree-substitution grammar into an LCFRS. The elementary trees are extracted from the tree in figure 1 with abbreviated labels. The right column shows the productions after transforming each elementary tree. Note that the productions for the first elementary tree contain no discontinuity, because the discontinuous internal node is eliminated. Conversely, the transformation may also introduce more discontinuity, due to the binarization.

The transformation defined above assumes that a sequence of productions can be read off from a syntactic tree, such as a standard phrase-structure tree that can be converted into a sequence of context-free grammar productions. Using the method for inducing LCFRS productions from syntactic trees given in the previous section, we can apply the TSG transformation for discontinuous trees as well. Figure 3 illustrates the transformation of a discontinuous TSG.

[1]Note that only this first production requires a globally unique identifier; to reduce the grammar constant, the other identifiers can be merged for equivalent productions.

3 Inducing a TSG from a treebank

In Data-Oriented Parsing the grammar is the treebank itself, and in principle all possible fragments from its trees can be used to derive new sentences. Grammar induction is therefore conceptually simple (although the grammar is very large), as there is no training or learning involved. A fragment of a tree T is defined as a connected subgraph of T with two or more nodes, where each node in the fragment either has no children or the same children as the corresponding node in T.

The use of all possible fragments allows for multiple derivations of the same tree; this spurious ambiguity is seen as a virtue in DOP, because it combines the advantages of specific larger fragments and the smoothing of smaller fragments. This is in contrast to more parsimonious approaches which decompose each tree in the training corpus into a sequence of fragments representing a single derivation, such as in Bayesian TSG (Post and Gildea, 2009; Cohn et al., 2010)

Representing all possible fragments of a treebank is infeasible, since the number of fragments is exponential in the number of nodes. A practical solution is to define a subset. A method called Double-DOP (2DOP; Sangati and Zuidema, 2011) realizes this without compromising on the principle of data-orientation by restricting the set to recurring fragments, i.e., fragments that occur at least twice. These are found by considering every pair of trees and extracting the largest tree fragments they have in common. It is feasible to do this exhaustively for the whole treebank. This is in contrast to the sampling of fragments in earlier DOP models (Bod, 2001) and Bayesian TSGs. Since the space of fragments is enormous (viz. exponential), it stands to reason that a sampling approach will not extract all relevant fragments in a reasonable time frame.

Sangati et al. (2010) presents a tree-kernel method for extracting maximal recurring fragments that operates in time quadratic in the number of nodes in the treebank. However, using a different tree kernel, tree fragments can be obtained from a treebank in linear average time (van Cranenburgh, 2012b).

3.1 Discontinuous fragments

The aforementioned fragment extraction algorithms can be adapted to support trees with discontinuous constituents, using a representation where leaf nodes are decorated with indices indicating their ordering. This

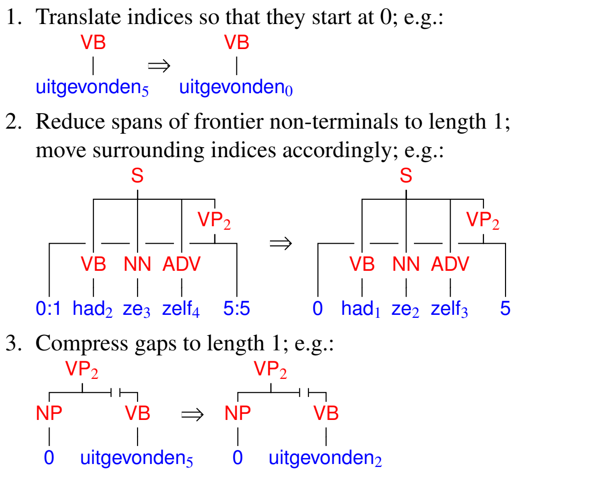

1. Translate indices so that they start at 0; e.g.:
2. Reduce spans of frontier non-terminals to length 1; move surrounding indices accordingly; e.g.:
3. Compress gaps to length 1; e.g.:

Figure 4: Canonicalization of fragments extracted from parse trees. The example fragments have been extracted from the tree in figure 1. The fragments are visualized here as discontinuous tree structures, but since the discontinuities are encoded in the indices of the yield, they can be represented in a standard bracketing format as used by the fragment extractor.

makes it possible to use the same data structures as for continuous trees, as long as the child nodes are kept in a canonical order (induced from the order of the lowest index of each child). Indices are used not only to keep track of the order of lexical nodes in a fragment, but also for that of the contribution of frontier non-terminals. This is necessary in order to preserve the configuration of the yield in the original sentence. The indices are based on those in the original sentence, but need to be decoupled from this original context. This process is analogous to how LCFRS productions are read off from a tree with discontinuous constituents. The canonicalization of fragments is achieved in three steps, described and illustrated in figure 4. In the examples, frontier non-terminals have spans denoted with inclusive *start:end* intervals, as extracted from the original parse tree, which are reduced to variables denoting a contiguous spans whose relation to the other spans is reflected by their indices.

When binarized with $h = \infty$, $v = 1$ markovization (Klein and Manning, 2003), about 8.5 % of fragments extracted from the Negra treebank (cf. sec. 5.1) contain a discontinuous root or internal node, compared to 30 % of sentences in the treebank that contain one or more discontinuous constituents.

4 Parsing

After extracting fragments, we apply the grammar transformation to turn them into grammar productions. In order to achieve full coverage, we augment the set of fragments with cover fragments of depth 1 corresponding to all single productions in the treebank. Productions corresponding to fragments are assigned a probability based on the frequency of the respective fragment, productions introduced by the transformation are given a probability of 1.

We parse with the transformed grammar using the `disco-dop` parser (van Cranenburgh et al., 2011; van Cranenburgh, 2012a). This is an agenda-based parser for PLCFRS based on the algorithm in Kallmeyer and Maier (2010, 2013), extended to produce *n*-best derivations (Huang and Chiang, 2005) and exploit coarse-to-fine pruning (Charniak et al., 2006).

4.1 Probabilities and disambiguation

To instantiate the probabilistic model we use the relative frequency estimate (RFE), since it has shown good results with the Double-DOP model (Sangati and Zuidema, 2011). The frequency of fragments is obtained by the fragment extractor, divided by the total frequency mass of fragments with the same root node.

In DOP many derivations may produce the same parse tree, and it has been shown that approximating the most probable parse, which considers all derivations for a tree, yields better results than the most probable derivation (Bod, 1995). To select a parse tree from a derivation forest, we marginalize the 10,000-best DOP derivations and select the tree with the most probability mass.

4.2 Coarse-to-fine pruning

In order to tame the complexity of LCFRS and DOP, we apply the same coarse-to-fine pruning as in van Cranenburgh (2012a). Namely, we parse in three stages:

1. Split-PCFG: A PCFG approximation of the discontinuous treebank grammar; rewrites spans of discontinuous constituents independently
2. PLCFRS: The discontinuous treebank grammar; rewrites discontinuous constituents in a single operation
3. The discontinuous DOP grammar: tree fragments instead of individual productions from treebank

The first stage is necessary because without pruning, the PLCFRS generates too many discontinuous spans, the majority of which are implausible or not even part of a complete derivation. The second stage is not necessary for efficiency but gives slightly better accuracy on discontinuous constituents.

The PCFG approximation splits discontinuous constituents into several non-terminals related through their labels; e.g.:

$$\begin{aligned} \text{PLCFRS production:}\ & \mathrm{VP}_2(a,b) \rightarrow \mathrm{VB}(a)\ \mathrm{PRT}(b) \\ \text{PCFG approximation:}\ & \{\ \mathrm{VP}_2{*}1 \rightarrow \mathrm{VB}, \\ & \quad \mathrm{VP}_2{*}2 \rightarrow \mathrm{PRT}\ \} \end{aligned}$$

In a post-processing step PCFG derivations are converted to discontinuous trees by merging siblings marked with '*'. This approximation overgenerates compared to the respective LCFRS; e.g., two components $\mathrm{VP}_2{*}1$ and $\mathrm{VP}_2{*}2$ may be generated which where extracted from different discontinuous constituents, such that their combination could not be generated by the LCFRS.

Pruning is achieved by limiting the second and third stages to the labeled spans occurring in the *k*-best derivations of the previous stage. The initial values for *k* are 10,000 and 50, for the PLCFRS and DOP grammar respectively. These values are chosen to be able to directly compare the new approach with the results in van Cranenburgh (2012a). However, experimenting with a higher value for *k* for the DOP stage has shown to yield improved performance.

4.3 Reconstructing derivations

After a derivation forest is obtained and a list of *k*-best derivations has been produced, the backtransform is applied to these derivations to recover their internal structure. This proceeds by doing a depth-first traversal of the derivations, and expanding each non-intermediate[2] node into a template of the original fragment. These templates are stored in a backtransform table indexed by the first binarized production of the fragment in question. The template fragment has its substitution sites marked, which are filled with values obtained by recursively expanding the children of the current constituent. To reconstruct 10,000 derivations takes 2 seconds on average.

[2]An intermediate node is a node introduced by the binarization.

Language	treebank	train	dev	test
GERMAN	Negra	sent. 1–18,602	sent. 19,603–20,602	sent. 18,603–19,602
GERMAN	Tiger	sec. 2–9 / 1–9	sec. 1	sec. 0
ENGLISH	PTB: WSJ	sec. 2-21	sec. 24	sec. 23
DUTCH	Alpino	16,319 sents.	extra: 446 sents.	CoNLL2006: 386 sents.
DUTCH	Lassy small	52,157 sents.	6,520 sents.	6,523 sents.

Table 1: The discontinuous treebanks used in the experiments.

5 Experimental setup

In this section we describe the experimental setup for benchmarking our discontinuous Double-DOP implementation on several discontinuous treebanks.

5.1 Data

We evaluate on three languages: the German Negra & Tiger treebanks (Skut et al., 1997; Brants et al., 2002), a discontinuous version of the Penn treebank (Evang and Kallmeyer, 2011), and the Dutch Alpino & Lassy treebanks (van der Beek et al., 2002; Van Noord, 2009); cf. table 1. Negra contains discontinuous annotations by design, as a strategy to cope with the relatively free word-order of German. The discontinuous Penn treebank consists of the WSJ section in which traces have been converted to discontinuous constituents; we use the version used in Evang and Kallmeyer (2011, sec. 5.1-5.2) without restrictions on the transformations. The Alpino treebank is referred to as a dependency treebank but when discontinuity is allowed it can be directly interpreted as a constituency treebank. Finally, Tiger and Lassy use similar annotation schemes as Negra and Alpino, respectively. The train-dev-test splits we employ are as commonly used for the Penn treebank; for Negra we use the one defined in Dubey and Keller (2003); for Tiger we follow Hall and Nivre (2008) who define sections 0–9 where sentence i belongs to section i mod 10; for Alpino and Lassy the split is our own.[3]

For purposes of training we remove grammatical functions from the treebanks, and binarize the trees in the training sets head-outward with $h = 1, v = 1$ markovization ($v = 2$ for PTB) and heuristics for head assignment (Klein and Manning, 2003); i.e., n-ary nodes are factored into nodes specifying an immediate sibling and parent. We add fan-out markers to guarantee unique fan-outs for non-terminal labels, e.g., {VP, VP_2, VP_3, ...}, which are removed again for evaluation. We apply a few simple manual state splits.[4] In order to compare the results on Negra with previous work, we do not apply the state splits when working with gold POS tags.

The complexity of parsing with an LCFRS depends on the maximal sum of the non-terminal fan-outs of its productions (Gildea, 2010). Using this measure, parsing with the DOP grammars extracted from Negra, WSJ, and Alpino has a worst-case time complexity of $O(n^9)$. The complexities for Tiger and Lassy are $O(n^{10})$ and $O(n^{12})$ respectively, due to a handful of anomalous sentences; by discarding these sentences, a grammar with a complexity of $O(n^9)$ can be obtained with no or negligible effect on accuracy.

5.2 Unknown words

In initial experiments we present the parser with the gold part-of-speech tags, as in previous experiments on discontinuous parsing. Later we show results when tags are assigned automatically with a simple unknown word model, based on the Stanford parser (Klein and Manning, 2003). Tags that rewrite more than σ words are considered open class tags, and words they rewrite are open class words. Open class words in the training

[3]The Alpino training set consists of all manually corrected sentences distributed with the Alpino parser, except for the Lassy corpus samples, `gen_g_suite`, and our development and test set, `extra` and `CoNNL2006` respectively. The Lassy split derives from 80-10-10 partitions of the canonically ordered sentence IDs in each subcorpus (viz. `dpc`, `WR`, `WS`, and `wiki`).

[4]For English we apply the state splits described in Evang and Kallmeyer (2011, sec. 4.2). S nodes with a WH-element are marked as such. VPs with as head a bare infinitive, to-infinitive, or particle verb are marked as such. The marking for VPs headed by a bare or to-infinitive is percolated to the parent S-node.

For Dutch and German we split the POS tags for sentence-ending punctuation '.!?'. For German we additionally split S nodes that are relative clauses, based on the respective grammatical function label.

set that do not occur more than 4 times are replaced with features; words in the test set which are not part of the known words from the training set are replaced with the same features. The features are defined in the Stanford parser as model 4, which is relatively language independent. ϵ probability mass is handed out for combinations of known open class words with unseen tags. For ϵ we use 0.01; σ is tuned on each training set to ensure that no closed class words are identified as open class words; for English and German we use 150, and 100 for Dutch.

5.3 Discontinuity without LCFRS

The idea up to now has been to generate discontinuous constituents using formal rewrite operations of LCFRS. However, the PCFG approximation used in the pruning stage encodes discontinuities as part of the labels. Instead of using this technique only as a crutch for pruning, it can also be combined with the use of fragments to obtain a purely cubic time pipeline. While the PCFG approximation increases the independence assumptions for discontinuous constituents, the use of large fragments can mitigate this increase. We shall refer to this alternative approach as 'Split-2DOP.'

5.4 Metrics

We employ the exact match and the Parseval measures (Black et al., 1992) as evaluation metrics. The latter can be straightforwardly generalized to discontinuous spans by representing spans of bracketings as sets of indices (Maier, 2010). Unfortunately it is not always made explicit in previous work on Negra parsing what kind of evaluation parameters are being used. We use the evaluation parameters typically used with `EVALB` on the Penn treebank. Namely, the root node, as well as punctuation, are not counted towards the score (similar to `COLLINS.prm`, except that we discount all punctuation, including brackets). Counting the root node as a constituent should not be done because it is not part of the corpus annotation and the parser is able to generate it without doing any work; when the root node is counted it inflates the F-score by several percentage points. Punctuation should be ignored because in the original annotation of the Dutch and German treebanks, punctuation is attached directly under the root node instead of as part of constituents. Punctuation can be re-attached using heuristics for the purposes of parsing, but evaluation should not be affected by this.

Model	k=50	k=5000
DOP reduction: disco-DOP	74.3	73.5
Double-DOP: disco-2DOP	76.3	**77.7**

Table 2: Comparing the DOP reduction (implicit fragments) with Double-DOP (explicit fragments) on the Negra development set with different amounts of pruning (higher k means less pruning; gold POS tags).

6 Evaluation

Table 2 compares previous results of Disco-DOP to the new Disco-2DOP implementation. The second column shows the accuracy for different values of k, i.e., the number of coarse derivations that determine the allowed labeled spans for the fine stage. While increasing this value did not yield improvements using the DOP reduction, with Disco-2DOP there is a substantial improvement in performance, with $k = 5000$ yielding the best score among the handful of values tested.

Table 3 lists the results for discontinuous parsing of three Germanic languages, with unknown word models. The cited work by Kallmeyer and Maier (2013) and Evang and Kallmeyer (2011) also uses LCFRS for discontinuity but employs a treebank grammar with relative frequencies of productions. Hall and Nivre (2008) use a conversion to dependencies from which discontinuous constituents can be recovered. For English and German the results improve upon the best known discontinuous constituency parsing results. The new system achieves a 16 % relative error reduction over the previous best result for discontinuous parsing on sentences $\leqslant$ 40 in the Negra test set. In terms of efficiency the Disco-2DOP model is more than three times as fast as the DOP reduction, taking about 3 hours instead of 10 on a single core. The grammar is also more compact: the size of the Disco-2DOP grammar is only a third of the DOP reduction, at 6 MB versus 18 MB compressed size.

The substantial improvements on the larger German and Dutch treebanks Tiger and Lassy suggest that providing more training data will keep improving accuracy. The results for Dutch are not comparable to earlier work because such work has only been evaluated on dependency relations of grammatical functions, which our model does not produce. Earlier work on recovering empty categories and their antecedents in the Penn treebank (Johnson, 2002; Gabbard et al., 2006;

Parser, treebank	$\lvert w \rvert$	DEV			TEST		
		POS	F1	EX	POS	F1	EX
GERMAN							
*vanCra2012, Negra	$\leqslant 40$	100	74.3	34.3	100	72.3	33.2
†*KaMa2013, Negra	$\leqslant 30$				100	75.8	
*this paper, Negra	$\leqslant 40$	100	**77.7**	**41.5**	100	**76.8**	**40.5**
this paper, Negra	$\leqslant 40$	96.7	76.4	39.2	96.3	74.8	38.7
HaNi2008, Tiger	$\leqslant 40$				97.0	75.3	32.6
this paper, Tiger	$\leqslant 40$	97.6	78.7	40.5	97.6	**78.8**	**40.8**
ENGLISH							
†*EvKa2011, WSJ	< 25				100	79.0	
this paper, WSJ	$\leqslant 40$	96.0	**85.2**	28.0	96.6	**85.6**	31.3
DUTCH							
this paper, Alpino	$\leqslant 40$	90.1	74.5	37.2	85.2	65.9	23.1
this paper, Lassy	$\leqslant 40$	94.1	79.0	37.4	94.6	77.0	35.2

Table 3: Discontinuous parsing of three Germanic languages. POS is the part-of-speech tagging accuracy, F1 is the labeled bracketing F1-score, EX is the exact match score. Results marked with * use gold POS tags; those marked with † do not discount the root node and punctuation. NB: KaMa, EvKa, and HaNi use a different test set and length restriction. Key to citations: vanCra: van Cranenburgh (2012a); KaMa: Kallmeyer and Maier (2013); HaNi: Hall and Nivre (2008); EvKa: Evang and Kallmeyer (2011).

Pipeline	F1 %	EX %
Split-PCFG (no LCFRS, no TSG)	65.8	28.0
Split-PCFG ⇒ PLCFRS (no TSG)	65.9	28.6
Split-PCFG ⇒ PLCFRS ⇒ 2DOP	77.7	41.5
Split-PCFG ⇒ Split-2DOP (no LCFRS)	78.1	42.0

Table 4: Parsing discontinuous constituents is possible without LCFRS (Negra dev. set, gold POS tags; results are for final stage).

Schmid, 2006) has recovered long-distance dependencies by producing the traces and co-indexation as in the original annotation; unfortunately the results are not directly comparable because their evaluation method depends on having both traces and antecedents, while our model directly generates discontinuous constituents.

Table 4 shows a comparison of coarse-to-fine pipelines with and without LCFRS, showing that, surprisingly, the use of a formalism that explicitly models discontinuity as an operation does not give any improvement over a simpler model in which discontinuities are only modeled probabilistically by encoding them into labels and fragments. This demonstrates that given the use of tree fragments, discontinuous rewriting through LCFRS comes at a high computational cost without a clear benefit over CFG.

From the results it is clear that a probabilistic tree-substitution grammar is able to provide much better results than a simple treebank grammar. However, it is not obvious whether the improvement is due to the more fine-grained statistics (i.e., weakened independence assumptions), or because of the use of larger chunks. A serendipitous discovery during development of the parser provides insight into this: during an experiment, the frequencies of fragments were accidentally permuted and assigned to different fragments, but the resulting decrease in accuracy was surprisingly low, from 77.7 % to 74.1 % F1—suggesting that most of the improvement over the 65.9 % of the PLCFRS treebank grammar comes from memorizing larger chunks, as opposed to statistical reckoning.

7 Conclusion

We have shown how to parse with discontinuous tree-substitution grammars along with a practical implementation. We have presented a fragment extraction tool that finds recurring structures in treebanks efficiently, and supports discontinuous treebanks. This enables a data-oriented parsing implementation providing a compact, efficient, and accurate model for discontinuous parsing in a single generative model that improves upon previous results for this task.

Surprisingly, it turns out that the formal power of LCFRS to describe discontinuity is not necessary, since equivalent results can be obtained with a probabilistic tree-substitution grammar in which long-distance dependencies are encoded as part of non-terminal labels.

The source code of the parser used in this work has been released as `disco-dop 0.4`, available at:
`https://github.com/andreasvc/disco-dop`

Acknowledgments

We are grateful to Kilian Evang for supplying the discontinuous Penn treebank, to Remko Scha for extensive feedback and fruitful discussions, to the reviewers for detailed comments, and to Corina Koolen and Kim Jautze for reading drafts.

This work is part of The Riddle of Literary Quality, a project supported by the Royal Netherlands Academy of Arts and Sciences through the Computational Humanities Program.

References

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

Shane Bergsma, Matt Post, and David Yarowsky. 2012. Stylometric analysis of scientific articles. In *Proceedings of NAACL*, pages 327–337. `http://aclweb.org/anthology/N12-1033`.

Ezra Black, John Lafferty, and Salim Roukos. 1992. Development and evaluation of a broad-coverage probabilistic grammar of English-language computer manuals. In *Proceedings of ACL*, pages 185–192. `http://aclweb.org/anthology/P92-1024`.

Rens Bod. 1995. The problem of computing the most probable tree in data-oriented parsing and stochastic tree grammars. In *Proceedings of EACL*, pages 104–111. `http://aclweb.org/anthology/E95-1015`.

Rens Bod. 2001. What is the minimal set of fragments that achieves maximal parse accuracy? In *Proceedings of ACL*, pages 69–76.

Rens Bod, Remko Scha, and Khalil Sima'an, editors. 2003. *Data-Oriented Parsing*. The University of Chicago Press.

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

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. 1996. Tree-bank grammars. In *Proceedings of the National Conference on Artificial Intelligence*, pages 1031–1036.

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.

Trevor Cohn, Phil Blunsom, and Sharon Goldwater. 2010. Inducing tree-substitution grammars. *The Journal of Machine Learning Research*, 11(Nov):3053–3096.

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

Kilian Evang and Laura Kallmeyer. 2011. PLCFRS parsing of English discontinuous constituents. In *Proceedings of IWPT*, pages 104–116. `http://aclweb.org/anthology/W11-2913`.

Ryan Gabbard, Seth Kulick, and Mitchell Marcus. 2006. Fully parsing the Penn treebank. In *Proceedings of NAACL-HLT*, pages 184–191.

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

Joshua Goodman. 2003. Efficient parsing of DOP with PCFG-reductions. In Bod et al. (2003).

Spence Green, Marie-Catherine de Marneffe, John Bauer, and Christopher D. Manning. 2011. Multiword expression identification with tree substitution grammars: A parsing tour de force with French. In *Proceedings of EMNLP*, pages 725–735.

Johan Hall and Joakim Nivre. 2008. Parsing discontinuous phrase structure with grammatical functions. In Bengt Nordstrm and Aarne Ranta, editors, *Advances in Natural Language Processing*, volume 5221 of *Lecture Notes in Computer Science*, pages 169–180. Springer Berlin / Heidelberg. `http://dx.doi.org/10.1007/978-3-540-85287-2_17`.

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

Mark Johnson. 2002. A simple pattern-matching algorithm for recovering empty nodes and their antecedents. In *Proceedings of ACL*, pages 136–143.

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.

Laura Kallmeyer and Wolfgang Maier. 2013. Data-driven parsing using probabilistic linear context-free rewriting systems. *Computational Linguistics*, 39(1):87–119.

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

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 Anders Søgaard. 2008. Treebanks and mild context-sensitivity. In *Proceedings of Formal Grammar 2008*, pages 61–76.

Matt Post. 2011. Judging grammaticality with tree substitution grammar derivations. In *Proceedings of the ACL-HLT 2011*, pages 217–222.

Matt Post and Daniel Gildea. 2009. Bayesian learning of a tree substitution grammar. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 45–48.

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

Federico Sangati, Willem Zuidema, and Rens Bod. 2010. Efficiently extract recurring tree fragments from large treebanks. In *Proceedings of LREC*, pages 219–226. `http://dare.uva.nl/record/371504`.

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`.

Helmut Schmid. 2006. Trace prediction and recovery with unlexicalized PCFGs and slash features. In *Proceedings of COLING-ACL*, pages 177–184.

Hiroyuki Shindo, Yusuke Miyao, Akinori Fujino, and Masaaki Nagata. 2012. Bayesian symbol-refined tree substitution grammars for syntactic parsing. In *Proceedings of ACL*, pages 440–448. `http://aclweb.org/anthology/P12-1046`.

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.

Benjamin Swanson and Eugene Charniak. 2012. Native language detection with tree substitution grammars. In *Proceedings of ACL*, pages 193–197. `http://aclweb.org/anthology/P12-2038`.

Andreas van Cranenburgh. 2012a. Efficient parsing with linear context-free rewriting systems. In *Proceedings of EACL*, pages 460–470. `http://aclweb.org/anthology/E12-1047`.

Andreas van Cranenburgh. 2012b. Extracting tree fragments in linear average time. Technical Report PP-2012-18, FNWI/FGw: Institute for Logic, Language and Computation (ILLC). `http://dare.uva.nl/en/record/421534`.

Andreas van Cranenburgh. 2012c. Literary authorship attribution with phrase-structure fragments. In *Proceedings of the NAACL-HLT 2012 Workshop on Computational Linguistics for Literature*, pages 59–63. `http://aclweb.org/anthology/W12-2508`.

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. `http://aclweb.org/anthology/W11-3805`.

Leonoor van der Beek, Gosse Bouma, Robert Malouf, and Gertjan van Noord. 2002. The Alpino dependency treebank. *Language and Computers*, 45(1):8–22.

Gertjan Van Noord. 2009. Huge parsed corpora in Lassy. In *Proceedings of TLT7*. LOT, Groningen, The Netherlands.

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. `http://www.aclweb.org/anthology/P87-1015`.