Data-Oriented Parsing with Discontinuous Constituents and Function Tags

van Cranenburgh, A.; Scha, R.; Bod, R.

DOI
10.15398/jlm.v4i1.100
Publication date
2016
Document Version
Final published version
Published in
Journal of Language Modelling

Data-oriented parsing with discontinuous constituents and function tags

Andreas van Cranenburgh[1,2]*, Remko Scha*[2]*, and Rens Bod*[2]
[1] Huygens ING, Royal Netherlands Academy of Arts and Sciences
[2] Institute for Logic, Language and Computation, University of Amsterdam

ABSTRACT

Statistical parsers are effective but are typically limited to producing projective dependencies or constituents. On the other hand, linguistically rich parsers recognize non-local relations and analyze both form and function phenomena but rely on extensive manual grammar engineering. We combine advantages of the two by building a statistical parser that produces richer analyses.

We investigate new techniques to implement treebank-based parsers that allow for discontinuous constituents. We present two systems. One system is based on a Linear Context-Free Rewriting System (LCFRS), while using a Probabilistic Discontinuous Tree-Substitution Grammar (PDTSG) to improve disambiguation performance. Another system encodes discontinuities in the labels of phrase-structure trees, allowing for efficient context-free grammar parsing.

The two systems demonstrate that tree fragments as used in tree-substitution grammar improve disambiguation performance while capturing non-local relations on an as-needed basis. Additionally, we present results for models that produce function tags, resulting in a more linguistically adequate model of the data. We report substantial accuracy improvements in discontinuous parsing for German, English, and Dutch, including results on spoken Dutch.

Keywords: discontinuous constituents, statistical parsing, tree-substitution grammar

This article is a substantially revised and extended version of van Cranenburgh and Bod (2013). While finishing this article, we learned with great sadness of the passing of our co-author Remko Scha. We dedicate this article to his memory.

1 INTRODUCTION

Probabilistic algorithms for parsing and disambiguation select the most probable analysis for a given sentence in accordance with a certain probability distribution. A fundamental property of such algorithms is thus the definition of the space of *possible sentence structures* that constitutes the domain of the probability distribution. Modern statistical parsers are often automatically derived from corpora of syntactically annotated sentences ("treebanks"). In this case, the "linguistic backbone" of the probabilistic grammar naturally depends on the convention for encoding syntactic structure that was used in annotating the corpus.

When different parsing and disambiguation algorithms are applied to the same treebank, their relative accuracies can be objectively assessed if the treebank is split into a training set (that is used to induce a grammar and its probabilities) and a test set (that provides a "gold standard" to assess the performance of the system). This is common practice now. In many cases, however, the linguistic significance of these evaluations may be questioned, since the test sets consist of phrase-structure trees, i.e., part-whole structures where all parts are contiguous chunks. Non-local syntactic relations are not represented in these trees; utterances in which such relations occur are therefore skipped or incorrectly annotated.

For certain practical applications this restriction may be harmless, but from a linguistic (and cognitive) viewpoint it cannot be defended. Since Chomsky's transformational-generative grammar, there have been many proposals for formal grammars with a less narrow scope. Some of these formalisms have been employed to annotate large corpora; in principle, they can thus be used in treebank grammars extracted from these corpora.

The Penn treebank, for instance, enriches its phrase-structure representations with "empty constituents" that share an index with the constituent that, from a transformational perspective, would be analyzed as originating in that position. Most grammars based on the Penn treebank ignore this information, but it was used by, e.g., Johnson (2002), Dienes and Dubey (2003), and Gabbard *et al.* (2006).

Another perspective on non-local syntactic dependencies generalizes the notion of a "syntactic constituent," in that it allows "dis-

continuous constituent structures," where a non-terminal node dominates a lexical yield that consists of different non-contiguous parts (McCawley 1982). Several German and Dutch treebanks have been annotated in terms of discontinuous constituency, and some statistical parsers have been developed that use these treebanks. Also, phrase structures with co-indexed traces can be converted into discontinuous constituent structures; the Penn treebank can therefore be transformed and used in the discontinuous constituency approach (Evang and Kallmeyer 2011). Figure 1 shows an example of a tree with discontinuous constituents.

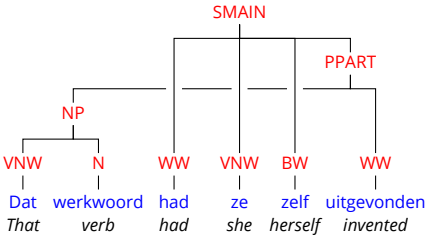


Figure 1:
A tree from the Dutch Alpino treebank (van der Beek *et al.* 2002). PPART is a discontinuous constituent (indicated with crossing branches) due to its extraposed NP object. Part-of-speech tags: VNW = pronoun, N = noun, WW = verb, BW = adverb. The tags also contain additional morphological features not shown here that distinguish personal pronouns from others, auxiliary verbs from main verbs, etc.

It is an annotation choice to employ discontinuous constituents; some treebanks elect not to model non-local phenomena, while others may choose different mechanisms. For example, two German treebanks employ discontinuous constituents (Skut *et al.* 1997; Brants *et al.* 2002), while another German treebank does not (Telljohann *et al.* 2004, 2012). The annotation scheme of the latter treebank lacks information expressed in the former two. For instance, it cannot encode the heads of non-local modifiers; with discontinuous constituents, a modifier is a sibling of its head, regardless of their configuration. On the other hand, the co-indexed traces of the Penn treebank provide more information than discontinuous constituents, because they assume that constituents have been moved from somewhere else in the tree and encode the original position. Discontinuous constituents describe surface structure without making such assumptions. Some phenomena that can be analyzed with discontinuous constituents are extraposition, topicalization, scrambling, and parentheticals; cf. Maier *et al.* (2014) for an overview of such phenomena in German.

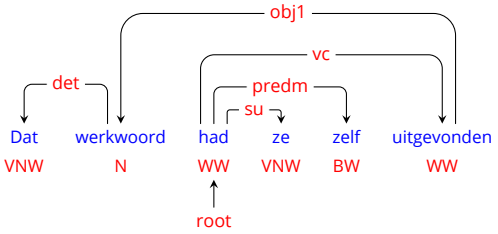


Figure 2:
A dependency structure derived from the tree in Figure 1. The *obj1* arc makes this structure non-projective.

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-usable elements; discontinuous constituents extend this to allow for arbitrary non-local relations that may arise due to such phenomena as extraposition and free word order. There is a close relation of discontinuous constituency to non-projectivity in dependency structures (Maier and Lichte 2011). Compare Figure 2, which shows a dependency structure for the constituency tree in Figure 1. Note that in this dependency structure, the edge labels are grammatical functions present in the original treebank, while the constituent labels in Figure 1 are syntactic categories. The dependency structure encodes the non-local relations within the discontinuous constituent. On the other hand, it does not represent the hierarchical grouping given by the NP and PPART constituents. By encoding both hierarchical and non-local information, trees with discontinuous constituents combine the advantages of constituency and dependency structures. We will also come back to grammatical function labels.

This paper is concerned with treebank-based parsing algorithms that accept discontinuous constituents. It takes as its point of departure work by Kallmeyer and Maier (2010, 2013) that represents discontinuous structures in terms of a string-rewriting version of Linear Context-Free Rewriting Systems (Section 3.1). In addition, we employ Tree-Substitution Grammar (TSG). We make the following contributions:

1. We discuss the notions of competence and performance in (computational) linguistics (Section 2). We argue that instead of focussing on the search for the formal (competence) grammar with the right capacity for natural language, we can consider performance aspects such as cognitive limitations and pruning strategies.

2. We show that Tree-Substitution Grammar can be applied to discontinuous constituents (Section 3.2) and that it is possible, using a transformation, to parse with a Tree-Substitution Grammar without having to write a separate parser for this formalism (Section 4.2).
3. We induce a tree-substitution grammar from a treebank (Section 5) using a method called Double-DOP (Sangati and Zuidema 2011). This method extracts a set of recurring tree fragments. We show that compared to another method which implicitly works with all possible fragments, this explicit method offers advantages in both accuracy and efficiency (Section 4.2.1, Section 9).
4. Fragments make it possible to treat discontinuous constituency as a statistical phenomenon within an encompassing context-free framework (Section 4.1, Section 7); this yields a considerable efficiency improvement without hurting accuracy (Section 9).
5. Finally, we present an evaluation on three languages. We employ manual state splits from previous work for improved performance (Section 8) and discuss methods and results for grammars that produce function tags in addition to phrasal labels (Section 8.3).

This work explores parsing discontinuous constituents with Linear Context-Free Rewriting Systems and Context-Free Grammar, as well as with and without the use of tree fragments through tree substitution. Figure 3 gives an overview of these systems and how they are combined in a coarse-to-fine pipeline (cf. Section 6.4).

2 THE DIVISION OF LABOR BETWEEN COMPETENCE AND PERFORMANCE

Traditionally, two aspects of language cognition have been distinguished: competence and performance (Chomsky 1965). Linguistic competence comprises a language user's "knowledge of language," usually described as a system of rules, while linguistic performance includes the details of the user's production and comprehension behavior. For a computational model, its syntactic competence defines the set of possible sentences that it can process in principle, and the structures it may assign to them, while its performance includes such

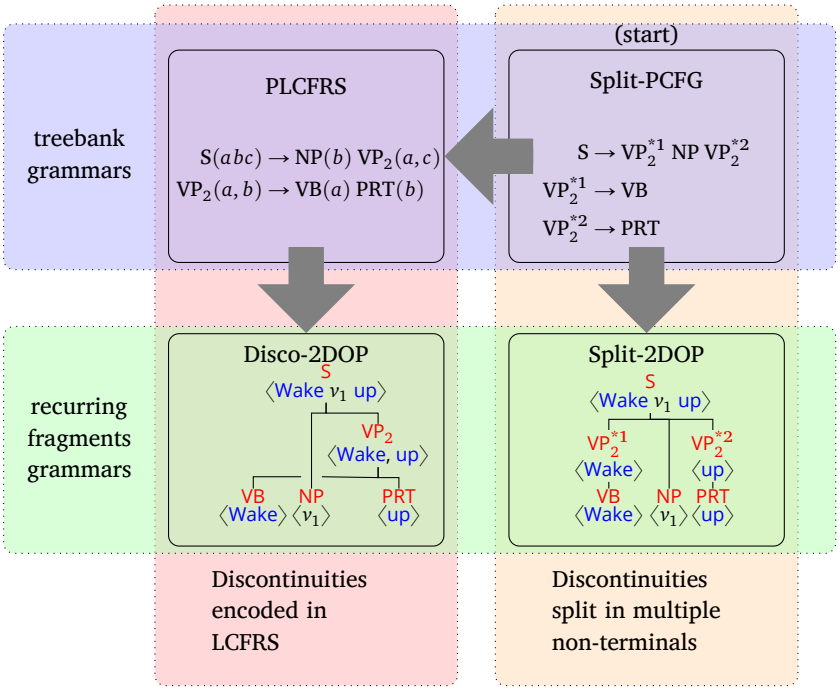


Figure 3: The systems explored in this work

aspects as disambiguation using occurrence frequencies of grammatical constructions. Thus, the choice of a formalism to describe the system's competence grammar depends on one's decisions on how syntax should be formalized. Regular and context-free grammars have been argued to be too limited (Chomsky 1956; Shieber 1985), while richer alternatives – context-sensitive and beyond – are considered too powerful to allow for an efficient computational implementation; this applies to Transformational Grammar (Peters and Ritchie 1973), Lexical-Functional Grammar, and Head-Driven Phrase Structure Grammar (Trautwein 1995). We may therefore wish to strike a balance and find a grammar formalism that is just powerful enough to describe the syntax of natural language. Joshi (1985) proposes Mildly Context-Sensitive grammars, which are beyond context-free, but avoid the computational complexity that comes with the full class of context-sensitive grammars. The first formalism developed in this framework was Tree-Adjoining Grammar (TAG; Joshi 1985). There has been

work on automatic extraction of tree-adjoining grammars from corpora (Chiang 2000; Xia *et al.* 2001; Kaeshammer and Demberg 2012), and formal extensions such as multi-component TAG (Weir 1988; Schuler *et al.* 2000; Kallmeyer 2009). Linear Context-Free Rewriting Systems (LCFRSs), as employed in the work reported below, are instances of Mildly Context-Sensitive grammar. LCFRS appears to be a *lingua franca* among mildly context-sensitive formalisms, since several formalisms have been shown to be equivalent to it (Vijay-Shanker and Weir 1994).

Irrespective of whether one accepts the competence-performance dichotomy, a practical natural language system needs to deal with phenomena that depend on world knowledge reflected in language use (e.g., the fact that in *"eat pizza with a fork"*, *with a fork* is prototypically related to *eat* rather than to *pizza*). This has led to a statistical turn in computational linguistics, in which models are directly induced from treebanks (Scha 1990; Charniak 1996; Bod *et al.* 2003; Geman and Johnson 2004). If the end goal is to make an adequate model of language *performance*, there is actually no need to have a competence grammar which is 'just right.' Instead, we might reduce some of the formal complexity by encoding it in statistical patterns. Concretely, we can opt for a grammar formalism that deliberately overgenerates, and count on grammatical analyses having a higher probability of being selected during disambiguation. This operationalizes the idea of there being a spectrum between ungrammaticality, markedness, and felicity. In Section 4.1 we introduce an approximation of LCFRS that makes it possible to produce discontinuous constituents in cubic time using a context-free grammar, by encoding information in non-terminal labels. A probabilistic variant of the resulting grammar makes stronger independence assumptions than the equivalent LCFRS, but as a component in a larger statistical system this does not have to pose a problem.

In the debate about the context-freeness of language, cross-serial dependencies have played an important role (Huybregts 1976; Bresnan *et al.* 1982; Shieber 1985). Consider the following example in Dutch:

(1) Jan zag dat Karel hem haar laat leren zwemmen.
Jan saw that Karel him her lets teach swim.
'Jan saw that Karel lets him teach her to swim.'

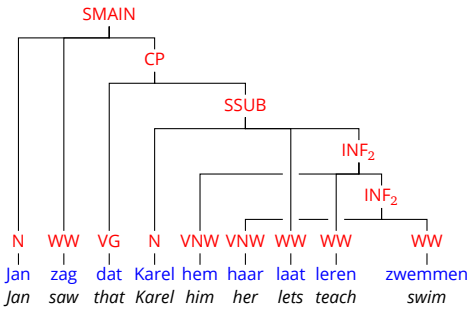


Figure 4:
Cross-serial dependencies in Dutch expressed with discontinuous constituents

Ojeda (1988) gives an account using discontinuous constituents; cf. Figure 4. In Section 4.1 we show how such analyses may be produced by an overgenerating context-free grammar.

This is an instance of the more general idea of approximating rich formal models in formally weaker but statistically richer models, i.e., descriptive aspects of language that can be handled as a performance rather than a competence problem. Another instance of this is constituted by the various restricted versions of TAG, whose string languages form a proper subset of those of LCFRS. Restricted variants of TAG that generate context-free string languages are Tree-Insertion Grammar (Schabes and Waters 1995; Hoogweg 2003; Yamangil and Shieber 2012), and off-spine TAG (Swanson *et al.* 2013); TSG is an even more restricted variant of TAG in which the adjunction operation is removed altogether. These results suggest that there is a trade-off to be made in the choice of formalism. While on the one hand Mild Context-Sensitivity already aims to limit formal complexity to precisely what is needed for adequate linguistic description, a practical, statistical implementation presents further opportunities for constraining complexity.

Another performance aspect of language relevant for computational linguistics is pruning. While normally considered an implementation aspect made necessary by practical hardware limitations, finding linguistically and psychologically plausible shortcuts in language processing forms an interesting research question. Schuler *et al.* (2010) present a parser with human-like memory constraints based on a finite-state model. Although Roark *et al.* (2012) are not concerned with cognitive plausibility, they also work with finite-state methods and show that CFG parsing can

be done in quadratic or even linear time with finite-state pruning methods.

As a specific example of a cognitive limitation relevant to parsing algorithms, consider center embedding. Karlsson (2007) reports from a corpus study that center embeddings only occur up to depth 3 in written language, and up to depth 2 in spoken language. If a statistical parser would take such cognitive limitations into account, many implausible analyses could be ruled out from the outset. More generally, it is worthwhile to strive for an explicit performance model that incorporates such cognitive and computational limitations as first class citizens.

In this work we do not go all the way to a finite-state model, but we do show that the non-local relations expressed in discontinuous constituents can be expressed in a context-free grammar model. We start with a mildly context-sensitive grammar formalism to parse discontinuous constituents, augmented with tree substitution. We then show that an approximation with context-free grammar is possible and effective. We find that the reduced independence assumptions and larger contexts taken into account as a result of tree substitution make it possible to capture non-local relations without going beyond context-free. Tree substitution thus increases the capabilities of the performance side without increasing the complexity of the competence side. A performance phenomenon that is modeled by this is that non-local relations are only faithfully produced as far as observed in the data.

3 GRAMMAR FORMALISMS

In this section we describe two formalisms related to discontinuous constituents; (string rewriting) Linear Context-Free Rewriting Systems and Discontinuous Tree-Substitution Grammar.

(String rewriting) Linear Context-Free Rewriting Systems (LCFRS; Vijay-Shanker *et al.* 1987) can produce such structures. An LCFRS generalizes CFG by allowing non-terminals to rewrite tuples of strings instead of just single, contiguous strings. This property makes LCFRS suitable for directly parsing discontinuous constituents (Kallmeyer and Maier 2010, 2013), as well as non-projective dependencies (Kuhlmann and Satta 2009; Kuhlmann 2013).

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 1992; Bod *et al.* 2003; Bansal and Klein 2010; Sangati and Zuidema 2011) and Bayesian TSGs (O'Donnell *et al.* 2009; Post and Gildea 2009; Cohn *et al.* 2009, 2010; Shindo *et al.* 2012). Other applications include grammaticality judgements (Post 2011), multi-word expression identification (Green *et al.* 2011), stylometry (Bergsma *et al.* 2012; van Cranenburgh 2012b), and native language detection (Swanson and Charniak 2012).

Before defining these formalisms, we first define the tree structures they operate on. The notion of a "discontinuous tree" stems from a long linguistic tradition (Pike 1943, Sections 4.12–14; Wells 1947, Sections 55–62; McCawley 1982). It generalizes the usual notion of a phrase-structure tree in that it allows a non-terminal node to dominate a lexical span that consists of non-contiguous chunks. In our interpretation of this idea, it results in three formal differences:

1. A non-terminal with non-contiguous daughters does not have a non-arbitrary place in the left-to-right order with respect to its sibling nodes. Therefore, it is not obvious anymore that the left-to-right order of the terminals is to be described in terms of their occurrence in a tree with totally ordered branches. Instead, we employ trees with *unordered* branches, while every node is augmented with an explicit representation of its (ordered) yield.
2. An "ordinary" (totally ordered) tree has a contiguous string of leaf nodes as its yield. When we allow discontinuities, this property still applies to the (totally lexicalized) complete trees of complete sentences. But for tree fragments, it fails; their yields may contain gaps. In the general case, the yield of a discontinuous tree is thus a tuple of strings.
3. Extracting a fragment from a tree now consists of two steps:
   (a) Extracting a connected subset of nodes, and

(b) Updating the yield tuples of the nodes. In the yield tuple of every non-terminal leaf node, every element (a contiguous chunk of words) is replaced by a *terminal variable*. This replacement is percolated up the tree, to the yield tuples of all nodes. Different occurrences of the same word carry a unique index, to allow for the percolation to proceed correctly.

We now proceed to give a more formal definition of our notion of a discontinuous tree.

DEFINITION 1. A *discontinuous syntactic tree* is a rooted, unordered tree. Each node consists of a label and a yield. A yield is a tuple of strings composed of lexical items; the tuple of strings denotes a subsequence of the yield at the root of the tree. We write ⟨a b⟩ to denote a yield consisting of the contiguous sequence of lexical items 'a' and 'b', while ⟨a b, c⟩ denotes a yield containing 'a b' followed by 'c' with an intervening gap. Given a node X,

- the yield of X is composed of the terminals in the yields of the children of X;
- conversely, the yield of each child of X is a subsequence of the yield of X;
- the yields of siblings do not overlap.

Figure 5 shows a tree according to this definition in which discontinuities are visualized with crossing branches as before. The same tree is rendered in Figure 6, without crossing branches, to highlight the fact that the information about discontinuities is encoded in the yields of the tree nodes.

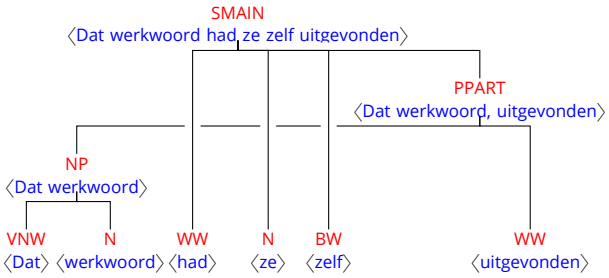
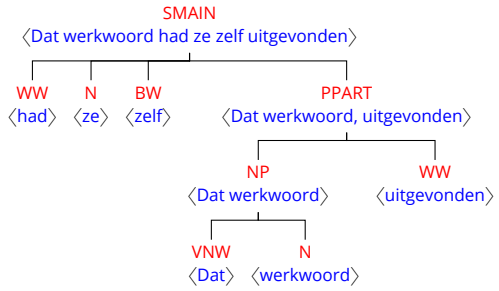


Figure 5:
A discontinuous tree with yield tuples

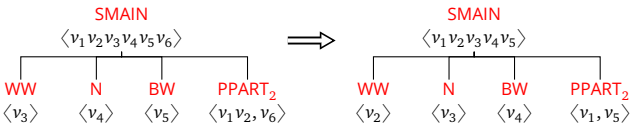
Figure 6:
An equivalent representation of the tree in Figure 5, without crossing branches



DEFINITION 2. An *incomplete tree* is a discontinuous tree in which the yields may contain variables v_n with $n \in \mathbb{N}$ in addition to lexical items. Variables stand in for any contiguous string of lexical items. An incomplete tree contains 2 or more nodes, or a single node with only lexical items in its yield. A node without children whose yield consists solely of variables is called a *substitution site*.

An incomplete tree may be derived from an extracted tree fragment. The tree fragment may contain variables for substrings which needed to be distinguished in other parts of the tree, but only occur contiguously in the fragment. We reduce these strings of contiguous variables to single variables; i.e., we abstract fragments from their original context by reducing strings of variables that appear contiguously across the fragment into single variables (e.g. Figure 7).

Figure 7:
Reducing variables in a fragment extracted from the tree in Figure 5



The *fan-out* of a non-terminal node equals the number of terminals in its yield that are not directly preceded by another terminal in the same yield; i.e., the number of contiguous substrings (components) of which the yield consists.[1] From here on we denote the fan-out of a discontinuous non-terminal with a subscript that is part of its label.

[1] Note that a distinction is often made between the *fan-out* of non-terminals in grammar productions, and the *block degree* of nodes of a syntactic tree (Maier and Lichte 2011; Kuhlmann 2013). Due to the fact that the productions of a TSG are trees, these notions coincide for our purposes.

3.1 *Linear Context-Free Rewriting Systems*

String-rewriting LCFRS can be seen as the discontinuous counterpart of CFG, and its probabilistic variant can be used to articulate a discontinuous treebank grammar. LCFRS productions differ from CFG productions in that they generate for a given non-terminal one or more strings at a time in potentially non-adjacent positions in the sentence. The number of these positions, the measure of discontinuity in a constituent, is called the fan-out. A CFG is an LCFRS with a maximum fan-out of 1. Together with the number of non-terminals on the right-hand side, the fan-out defines a hierarchy of grammars with increasing complexity, of which CFG is the simplest case. In this paper we use the simple RCG notation (Boullier 1998) for LCFRS. We focus on string-rewriting LCFRS and use the tree produced as a side-effect of a string's derivation as its syntactic analysis. It is possible to define an LCFRS that rewrites trees or graphs; however, the formalisms used in this paper are all expressible as string-rewriting LCFRSs.

DEFINITION 3. A string-rewriting LCFRS is a tuple $G = \langle N, T, V, P, S \rangle$. N and T are disjoint finite sets of non-terminals and terminals, respectively. A function $\varphi : N \to \{1, 2, \ldots, \}$ specifies the unique fan-out for every non-terminal symbol. V is a finite set of variables; we refer to the variables as x_j^i with $i, j \in \mathbb{N}$. S is the distinguished start symbol with $S \in N$ and $\varphi(S) = 1$. P is a finite set of productions, of the form:

$$A(\alpha_1, \ldots, \alpha_{\varphi(A)}) \to B_1(x_1^1, \ldots, x_{\varphi(B_1)}^1) \ldots B_r(x_1^r, \ldots, x_{\varphi(B_r)}^r)$$

for $r \geqslant 0$, where A, B_1, ..., $B_r \in N$, each $x_j^i \in V$ for $1 \leqslant i \leqslant r$, $1 \leqslant j \leqslant \varphi(B_i)$, and $\alpha_j \in (T \cup V)^+$ for $1 \leqslant j \leqslant \varphi(A)$. Observe that a component α_j is a concatenation of one or more terminals and variables.

The rank r refers to the number of non-terminals on the right-hand side of a production, while the fan-out φ of a non-terminal refers to the number of components it covers. A rank of zero implies a lexical production; in that case the right-hand side (RHS) is notated as ε implying no new non-terminals are produced (not to be confused with generating the empty string), and the left-hand side (LHS) argument is composed only of terminals.

Productions must be *linear* and *non-erasing*: if a variable occurs in a production, it occurs exactly once on the LHS, and exactly once on

the RHS. A production is *monotone*[2] 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. Due to our method of grammar extraction from treebanks, (cf. Section 3.1.1 below) all productions in this work are monotone and, except in some examples, at most binary ($r \leqslant 2$); lexical productions ($r = 0$) have fan-out 1 and introduce only a single terminal.

A production is *instantiated* when its variables are bound to spans such that for each component α_j of the LHS, the concatenation of the strings that its terminals and bound variables point to forms a contiguous, non-overlapping span in the input. In the remainder we will notate discontinuous non-terminals with a subscript indicating their fan-out.

When a sentence is parsed by an LCFRS, its derivation tree (Boullier 1998, Section 3.3; Kallmeyer 2010, pp. 115–117) is a discontinuous tree. Conversely, given a set of discontinuous trees, a set of productions can be extracted that generate those trees.

In a probabilistic LCFRS (PLCFRS), each production is associated with a probability and the probability of derivation is the product of the probabilities of its productions. Analogously to a PCFG, a PLCFRS may be induced from a treebank by using relative frequencies as probabilities (Maier and Søgaard 2008).

DEFINITION 4. The *language* of an LCFRS G is defined as follows (Kallmeyer and Maier 2013, pp. 92–93):

1. For every $A \in N$, we define the yield of A, $\text{yield}_G(A)$, as follows:
   (a) For every production $A(t) \to \varepsilon$ with $t \in T$, $\langle t \rangle \in \text{yield}_G(A)$
   (b) For every production

$$A(\alpha_1, \ldots, \alpha_{\varphi(A)}) \to B_1(x_1^1, \ldots, x_{\varphi(B_1)}^1) \ldots B_r(x_1^r, \ldots, x_{\varphi(B_r)}^r)$$

   and all tuples $\tau_1 \in \text{yield}_G(B_1), \ldots, \tau_r \in \text{yield}_G(B_r)$:

$$\langle f(\alpha_1), \ldots, f(\alpha_{\varphi(A)}) \rangle \in \text{yield}_G(A)$$

   where f is defined as follows:
   i. $f(t) = t$ for all $t \in T$,

[2] This property is called *ordered* in the RCG literature.

ii. $f(x_j^i) = \tau_i[j]$ for all $1 \leqslant i \leqslant r, 1 \leqslant j \leqslant \varphi(B_i)$, and
iii. $f(ab) = f(a)f(b)$ for all $a, b \in (T \cup V)^+$.
f is the *composition function* of the production.

(c) Nothing else is in $\text{yield}_G(A)$.

2. The language of G is then $L(G) = \text{yield}_G(S)$.

3.1.1 Extracting LCFRS productions from trees

LCFRS productions may be induced from a discontinuous tree, using a procedure described in Maier and Søgaard (2008). We extend this procedure to handle substitution sites, i.e., non-terminals with only variable terminals in their yield, but no lexical items; such nodes occur in tree fragments extracted from a treebank. The procedure is as follows:

Given a discontinuous tree, we extract a grammar production for each non-leaf non-terminal node. The label of the node forms the LHS non-terminal, and the labels of the nodes immediately dominated by it form the RHS non-terminals. The arguments of each RHS non-terminal are based on their yield tuples. Adjacent variables in the yield of the RHS non-terminals are collapsed into single variables and replaced on both LHS and RHS. Consider the tree fragment in Figure 7, which gives the following LCFRS production:

$$\text{SMAIN}(abcde) \rightarrow \text{PPART}(a, e)\ \text{WW}(b)\ \text{N}(c)\ \text{BW}(d)$$

Pre-terminals yield a production with their terminal as a direct argument to the pre-terminal, and an empty RHS. Substitution sites in a tree only appear on the RHS of extracted productions, since it is not known what they will expand to. See Figure 8 for examples of LCFRS productions extracted from a discontinuous tree.

3.2 *Discontinuous Tree-Substitution Grammar*

We now employ string-rewriting LCFRS, introduced in the previous section, to replace the CFG foundation of TSGs. Note that the resulting formalism directly rewrites elementary trees with discontinuous constituents, making it an instantiation of the more general notion of a tree-rewriting LCFRS. Tree-rewriting LCFRSs are more general because they allow other rewriting operations besides substitution. However, since we limit the operations in the formalism

Figure 8: The LCFRS $G = \langle N, T, V, P, S \rangle$ extracted from the tree in Figure 5

$$
\begin{aligned}
N &= \{\text{SMAIN, PPART, NP, VNW, N, WW, BW}\} \\
T &= \{\text{Dat, had, uitgevonden, werkwoord, ze, zelf}\} \\
V &= \{a, b, c, d, e\} \\
\varphi &= \{\text{SMAIN} : 1, \text{PPART} : 2, \text{NP} : 1, \\
&\quad \text{VNW} : 1, \text{N} : 1, \text{WW} : 1, \text{BW} : 1\} \\
S &= \text{SMAIN} \\
P &= \{\text{SMAIN}(abcde) \rightarrow \text{WW}(b)\ \text{N}(c)\ \text{BW}(d)\ \text{PPART}(a, e), \\
&\quad \text{PPART}(a, b) \rightarrow \text{NP}(a)\ \text{WW}(b), \\
&\quad \text{NP}(ab) \rightarrow \text{VNW}(a)\ \text{N}(b), \\
&\quad \text{VNW(Dat)} \rightarrow \varepsilon,\ \ \text{N(werkwoord)} \rightarrow \varepsilon, \\
&\quad \text{WW(had)} \rightarrow \varepsilon,\ \ \text{N(ze)} \rightarrow \varepsilon,\ \ \text{BW(zelf)} \rightarrow \varepsilon, \\
&\quad \text{WW(uitgevonden)} \rightarrow \varepsilon\}
\end{aligned}
$$

to substitution, it remains possible to specify a direct mapping to a string-rewriting grammar, as we shall see in the next section. As noted before, a TSG can be seen as a TAG without the adjunction operation. A discontinuous TSG may be related to a special case of set-local multi-component TAG (Weir 1988; Kallmeyer 2009). A multi-component TAG is able to specify constraints that require particular elementary trees to apply together; this mechanism can be used to generate the non-local elements of discontinuous constituents.

The following definitions are based on the definition for continuous TSG in Sima'an (1997).

DEFINITION 5. A *probabilistic, discontinuous TSG* (PDTSG) is a tuple $\langle N, T, V, S, \mathscr{C}, P \rangle$, where N and T are disjoint finite sets that denote the set of non-terminal and terminal symbols, respectively; V is a finite set of variables; S denotes the start non-terminal; and $\mathscr{C}$ is a finite set of elementary trees. For all trees in $\mathscr{C}$ it holds that for each non-terminal, there is a unique fan-out; this induces a function $\varphi \subset N \times \{1, 2, \ldots\}$ with $\varphi(A)$ being the unique fan-out of $A \in N$. For convenience, we abbreviate $\varphi(\text{root}(t))$ for a tree t as $\varphi(t)$. The function P assigns a value $0 < P(t) \leqslant 1$ (probability) to each elementary tree t such that for every non-terminal $A \in N$, the probabilities of all elementary trees whose root node is labelled A sum to 1.

The tuple $\langle N, T, V, S, \mathscr{C} \rangle$ of a given PDTSG $\langle N, T, V, S, \mathscr{C}, P \rangle$ is called the DTSG underlying the PDTSG.

DEFINITION 6. *Substitution:* The substitution $A \circ B$ is defined iff the label of the left-most substitution site of A equals the label of the root node of B. The left-most substitution site of an incomplete tree A is the leaf node containing the first occurrence of a variable in the yield of the root of A. When defined, the result of $A \circ B$ equals a copy of the tree A with B substituted for the left-most substitution site of A. In the yield argument of A, each variable terminal is replaced with the corresponding component of one or more contiguous terminals from B. For example, given $\text{yield}(A) = \langle l_1 v_2, l_4 \rangle$ and $\text{yield}(B) = \langle l_2 l_3 \rangle$ where l_n is a lexical terminal and v_n a variable, $\text{yield}(A \circ B) = \langle l_1 l_2 l_3, l_4 \rangle$.

DEFINITION 7. A *left-most derivation* (derivation henceforth) d is a sequence of zero or more substitutions $T = (\ldots(f_1 \circ f_2) \circ \ldots) \circ f_m$, where $f_1, \ldots, f_m \in \mathscr{C}, \text{root}(T) = \text{root}(f_1) = S, \varphi(T) = 1$ and T contains no substitution sites. The probability $P(d)$ is defined as:

$$P(f_1) \cdot \ldots \cdot P(f_m) = \prod_{i=1}^{m} P(f_i)$$

Refer to Figure 9 for an example.

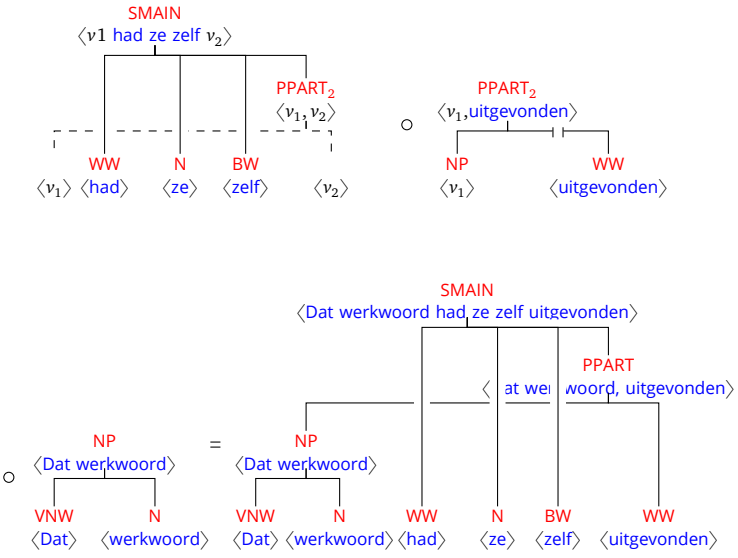


Figure 9:
A discontinuous tree-substitution derivation of the tree in Figure 1. Note that in the first fragment, which has a discontinuous substitution site, the destination for the discontinuous spans is marked in advance, shown with variables (v_n) as placeholders.

DEFINITION 8. A *parse* is any tree which is the result of a derivation. A parse can have various derivations. Given the set $D(T)$ of derivations yielding parse T, the probability of T is defined as $\sum_{d \in D(T)} P(d)$.

4 GRAMMAR TRANSFORMATIONS

CFG, LCFRS, and DTSG can be seen as natural extensions of each other. This makes it possible to define transformations that help to make parsing more efficient. Specifically, we define simplified versions of these grammars that can be parsed efficiently, while their productions or labels map back to the original grammar.

4.1 *A CFG approximation of discontinuous LCFRS parsing*

Barthélemy *et al.* (2001) introduced a technique to guide the parsing of a range concatenation grammar (RCG) by a grammar with a lower parsing complexity. Van Cranenburgh (2012a) applies this idea to probabilistic LCFRS parsing and extends the method to prune unlikely constituents in addition to filtering impossible constituents.

The approximation can be formulated as a tree transformation instead of a grammar transformation. The tree transformation by Boyd (2007) encodes discontinuities in the labels of tree nodes.[3] The resulting trees can be used to induce a PCFG that can be viewed as an approximation to the corresponding PLCFRS grammar of the original, discontinuous treebank. We will call this a Split-PCFG.

DEFINITION 9. A Split-PCFG is a PCFG induced from a treebank transformed by the method of Boyd (2007); that is, discontinuous constituents have been split into several non-terminals, such that each new non-terminal covers a single contiguous component of the yield of the discontinuous constituent. Given a discontinuous non-terminal

[3] Hsu (2010) compares three methods for resolving discontinuity in trees: (*a*) node splitting, as applied here; (*b*) node adding, a simpler version of node splitting that does not introduce new non-terminal labels; and (*c*) node raising, the more commonly applied method of resolving discontinuity. While the latter two methods yield better performance, we use the node splitting approach because it provides a more direct mapping to discontinuous constituents, which, as we shall later see, makes it a useful source of information for pruning purposes.

X_n in the original treebank, the new non-terminals will be labelled X_n^{*m}, with m the index of the component, s.t. $1 \leqslant m \leqslant n$.

For example:

$$\begin{aligned} \textsc{lcfrs} \text{ productions: } \mathrm{S}(abc) &\rightarrow \mathrm{NP}(b)\ \mathrm{VP}_2(a,c) \\ \mathrm{VP}_2(a,b) &\rightarrow \mathrm{VB}(a)\ \mathrm{PRT}(b) \\ \textsc{cfg} \text{ approximation: } \mathrm{S} &\rightarrow \mathrm{VP}_2^{*1}\ \mathrm{NP}\ \mathrm{VP}_2^{*2} \\ \mathrm{VP}_2^{*1} &\rightarrow \mathrm{VB} \\ \mathrm{VP}_2^{*2} &\rightarrow \mathrm{PRT} \end{aligned}$$

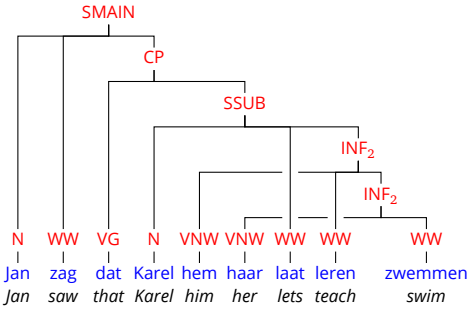
In a post-processing step, PCFG derivations are converted to discontinuous trees by merging siblings marked with '*'. This approximation overgenerates compared to the respective LCFRS, i.e., it licenses a superset of the derivations of the respective LCFRS. For example, a component VP_2^{*1} may be generated without generating its counterpart VP_2^{*2}; such derivations can be filtered in post-processing. Furthermore, two components VP_2^{*1} and VP_2^{*2} may be generated which were extracted from different discontinuous constituents, such that their combination could not be generated by the LCFRS.[4] Another problem would occur when productions contain discontinuous constituents with the same label; the following two productions map to the same productions in the CFG approximation:

$$\begin{aligned} \mathrm{VP}(adceb) &\rightarrow \mathrm{VP}_2(a,b)\ \mathrm{CNJ}(c)\ \mathrm{VP}_2(d,e) \\ \mathrm{VP}(adcbe) &\rightarrow \mathrm{VP}_2(a,b)\ \mathrm{CNJ}(c)\ \mathrm{VP}_2(d,e) \end{aligned}$$

However, such productions do not occur in any of the treebanks used in this work. The increased independence assumptions due to rewriting discontinuous components separately are more problematic, especially with nested discontinuous constituents. They necessitate the use of non-local statistical information to select the most likely structures, for instance by turning to tree-substitution grammar (cf. Section 2 above). (Note that the issue is not as problematic when the approximation is only used as a source of pruning information).

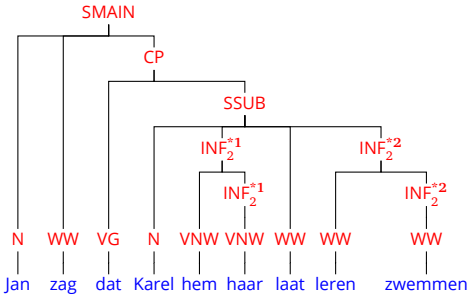
As a specific example of the transformation, consider the case of cross-serial dependencies. Figure 10 shows the parse tree for the

[4] A reviewer points out that if discontinuous rewriting is seen as synchronous rewriting (synchronous CFGs are equivalent to LCFRSs with fan-out 2), the split transformation is analogous to taking out the synchronicity.



discontinuous constituents:

$$\begin{aligned}
\text{SMAIN}(abc) &\rightarrow \text{N}(a)\ \text{WW}(b)\ \text{CP}(c)\\
\text{CP}(ab) &\rightarrow \text{VG}(a)\ \text{SSUB}(b)\\
\text{SSUB}(abcd) &\rightarrow \text{N}(a)\\
&\quad\ \text{INF}_2(b,d)\ \text{WW}(c)\\
\text{INF}_2(ab,cd) &\rightarrow \text{VNW}(a)\ \text{INF}_2(b,d)\\
&\quad\ \text{WW}(c)\\
\text{INF}_2(a,b) &\rightarrow \text{VNW}(a)\ \text{WW}(b)
\end{aligned}$$



PCFG approximation:

$$\begin{aligned}
\text{SMAIN} &\rightarrow \text{N WW CP}\\
\text{CP} &\rightarrow \text{VG SSUB}\\
\text{SSUB} &\rightarrow \text{N INF}_2^{*1}\ \text{WW INF}_2^{*2}\\
\text{INF}_2^{*1} &\rightarrow \text{VNW INF}_2^{*1}\\
\text{INF}_2^{*2} &\rightarrow \text{WW INF}_2^{*2}\\
\text{INF}_2^{*1} &\rightarrow \text{VNW}\\
\text{INF}_2^{*2} &\rightarrow \text{WW}
\end{aligned}$$

Figure 10: Cross-serial dependencies in Dutch expressed with discontinuous constituents (top); and the same parse tree, after discontinuities have been encoded in node labels (bottom)

example sentence from the previous section, along with the grammar productions for it, before and after applying the CFG approximation of LCFRS. Note that in the approximation, the second level of INF nodes may be rewritten separately, and a context-free grammar cannot place the non-local constraint that each transitive verb should be paired with a direct object. On the other hand, through the use of tree substitution, an elementary tree may capture the whole construction of two verbs cross-serially depending on two objects, and the model needs only to prefer an analysis with this elementary tree. Once an elementary tree contains the whole construction, it no longer matters whether its internal nodes contain discontinuous constituents or indexed node labels, and the complexity of discontinuous rewriting is weakened to a statistical regularity.

A phenomenon which cannot be captured in this representation, not even with the help of tree-substitution, is recursive synchronous rewriting (Kallmeyer *et al.* 2009). Although this phenomenon is rare, it does occur in treebanks.

4.2 TSG compression

Using grammar transformations, it is possible to parse with a TSG without having to represent elementary trees in the chart explicitly, but instead work with a parser for the base grammar underlying the TSG (typically a CFG, in our case an LCFRS).

In this section we present such a transformation for an arbitrary discontinuous TSG to a string-rewriting LCFRS. We first look at well-established strategies for reducing a continuous TSG to a CFG, and then show that these carry over to the discontinuous case. Previous work was based on probabilistic TSG without discontinuity; this special case of PDTSG is referred to as PTSG.

4.2.1 Compressing PTSG to PCFG

Goodman (2003) gives a reduction to a PCFG for the special case of a PTSG based on all fragments from a given treebank and their frequencies. This reduction is stochastically equivalent to an all-fragments PTSG after the summation of probabilities from equivalent derivations; however, it does not admit parsing with TSGs consisting of arbitrary sets of elementary trees or assuming arbitrary probability models. Perhaps counter-intuitively, restrictions on the set of fragments increase the size of Goodman's reduction (e.g., depth restriction, Goodman 2003, p. 134). While Goodman (2003) gives instantiations of his reduction with various probability models, the limitation is that probability assignments of fragments have to be expressible as a composition of the weights of the productions in each fragment. Since each production in the reduction participates in numerous implicit fragments, it is not possible to adjust the probability of an individual fragment without affecting related fragments. We leave Goodman's reduction aside for now, because we would prefer a more general method.

A naive way to convert any TSG is to decorate each internal node of its elementary trees with a globally unique number, which can be removed from derivations in a post-processing step. Each elementary tree then contributes one or more grammar productions, and because of the unique labels, elementary trees will always be derived as a whole. However, this conversion results in a large number of non-terminals, which are essentially 'inert': they never participate in substitution but deterministically rewrite to the rest of their elementary tree.

A more compact transformation is used in Sangati and Zuidema (2011), which can be applied to arbitrary PTSGs, but adds a minimal number of new non-terminal nodes. Internal nodes are removed from elementary trees, yielding a flattened tree of depth 1. Each flattened tree is then converted to a grammar production. Each production and original fragment is stored in a backtransform table. This table makes it possible to restore the original fragments of a derivation built from flattened productions. Whenever two fragments would map to the same flattened production, a unary node with a unique identifier is added to disambiguate them. 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.

4.2.2 Compressing PDTSG to PLCFRS

The transformation defined by Sangati and Zuidema (2011) 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 Section 4.2.1, we can apply the same TSG transformation to discontinuous trees as well.

Due to the design of the parser we will use, it is desirable to have grammar productions in binarized form, and to separate phrasal and lexical productions. We therefore binarize the flattened trees 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 the backtransform table.[5]

Figure 11 illustrates the transformation of a discontinuous TSG. The middle column shows the productions after transforming each ele-

[5] 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.

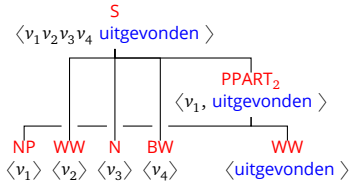
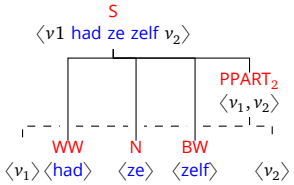
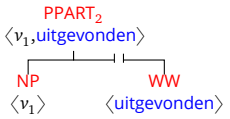
Elementary tree	Productions	Weight
	$\mathrm{S}(ab) \to \mathrm{S}^1(a)\ \mathrm{WW}(b)$ $\mathrm{S}^1(ab) \to \mathrm{S}^2(a)\ \mathrm{BW}(b)$ $\mathrm{S}^2(ab) \to \mathrm{S}^3(a)\ \mathrm{N}(b)$ $\mathrm{S}^3(ab) \to \mathrm{NP}(a)\ \mathrm{WW}^4(b)$ $\mathrm{WW}^4(\textbf{uitgevonden}) \to \varepsilon$	f/f' 1 1 1 1
	$\mathrm{S}(abc) \to \mathrm{S}_2^5(a,c)\ \mathrm{BW}^6(b)$ $\mathrm{S}_2^5(ab,c) \to \mathrm{S}_2^7(a,c)\ \mathrm{N}(b)$ $\mathrm{S}_2^7(ab,c) \to \mathrm{PPART}_2(a,c)\ \mathrm{WW}^8(b)$ $\mathrm{WW}^8(\textbf{had}) \to \varepsilon$ $\mathrm{N}^7(\textbf{ze}) \to \varepsilon$ $\mathrm{BW}^6(\textbf{zelf}) \to \varepsilon$	f/f' 1 1 1 1 1
	$\mathrm{PPART}_2(a,b) \to \mathrm{NP}(a)\ \mathrm{WW}^9(b)$ $\mathrm{WW}^9(\textbf{uitgevonden}) \to \varepsilon$	f/f' 1

Figure 11: Transforming a discontinuous tree-substitution grammar into an LCFRS and backtransform table. The elementary trees are extracted from the tree in Figure 1 with labels abbreviated. The first production of each fragment is used as an index to the backtransform table so that the original fragments in derivations can be reconstructed.

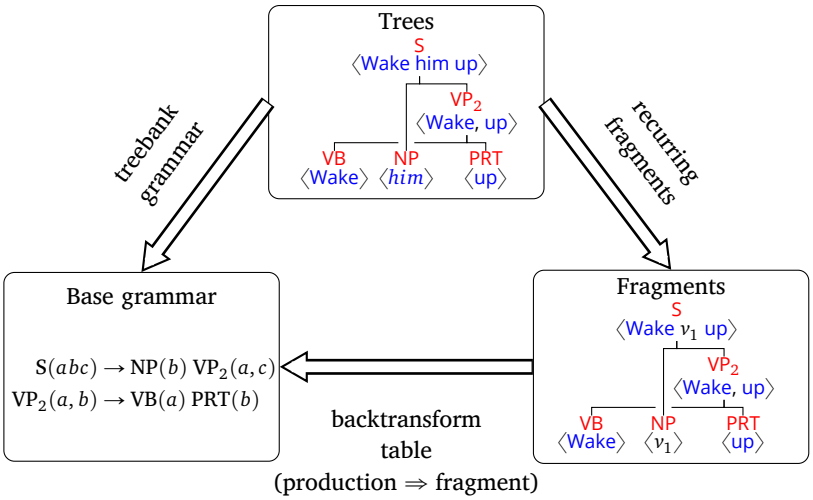


Figure 12: Diagram of the methods of grammar induction.

mentary tree. The rightmost column shows how relative frequencies can be used as weights, where f is the frequency of the elementary tree in the treebank, and f' is the frequency mass of elementary trees with the same root label. 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 (but cf. Section 8.1 below).

Figure 12 presents an overview of the methods of grammar induction presented thus far, as well as the approach for finding recurring fragments that will be introduced in the next section.

5 INDUCING A TSG FROM A TREEBANK

In Data-Oriented Parsing the grammar is implicit in 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 (even though the grammar may be very large), as there is no training or learning involved. This maximizes re-use of previous experience.

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 specificity of larger fragments and the smoothing of smaller fragments. This is in contrast to parsimonious approaches which decompose each tree in the training corpus into a sequence of fragments representing a single derivation.

5.1 *Extracting recurring fragments*

Representing all possible fragments of a treebank is not feasible, since the number of fragments is exponential in terms of the number of nodes. A practical solution is to define a subset. A method called Double-DOP (2DOP; Sangati and Zuidema 2011) implements this without compromising on the principle of data-orientation. It restricts the fragment set to recurring fragments, i.e., fragments that occur in at least two different contexts. 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

(that is, exponential in terms of sentence length), it stands to reason that a sampling approach will not discover 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 quadratic time in terms of the number of nodes in the treebank. A faster version of this method was presented in van Cranenburgh (2014), which uses a linear average time tree kernel, and introduces the ability to handle discontinuous trees. We obtain a further increase in speed by implementing an inverted index with a compressed bitmap (Chambi *et al.* 2015).

5.2 *Discontinuous fragments*

The aforementioned fragment extraction algorithms can be adapted to support trees with discontinuous constituents. Instead of implementing a new version with data structures for discontinuous trees following Definitions 1 and 2, we apply a representation that makes it possible to add discontinuous trees as a special case.

In the representation, leaf nodes are decorated with indices indicating their ordering. Just as in Figure 6, a discontinuous tree may be represented as a continuous tree, as long as information about the yield is encoded somehow. We do this by storing indices as leaf nodes, which denote an ordering and refer to a separate list of tokens. This 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 substitution sites. This is necessary in order to preserve the configuration of the yield in the original sentence. When leaf nodes are compared, the indices stand in for the token at the sentence position referred to. After a fragment is extracted, any indices need to be canonicalized. The indices originate from 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, in which contiguous intervals of indices are replaced by variables.

The canonicalization of fragments is achieved in three steps, as defined in the pseudocode of Algorithm 1; Figure 13 illustrates the

process. In the examples, substitution sites have spans denoted with inclusive *start:end* intervals, as extracted from the original parse tree, which are reduced to variables denoting contiguous spans whose relation to the other spans is reflected by their indices.

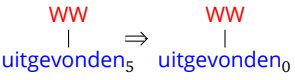
Algorithm 1 Canonicalizing discontinuous fragments.

INPUT: A tree fragment t with indexed terminals w_i or intervals $\langle i:j,\ldots\rangle$ as leaves ($0 \leqslant i < j < n$)
OUTPUT: A tree fragment with modified indices.

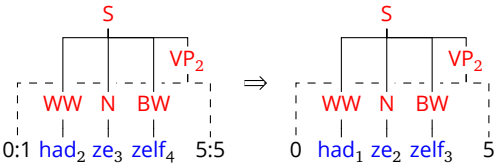
```
k ← the smallest index in t
subtract k from each index in t
FOR ALL intervals I = ⟨i:j,...⟩ of the substitution sites in t
    FOR ALL i:j ∈ I
        replace i:j with i
        subtract j − i from all indices k s.t. k > j
FOR ALL indices i in t
    IF the indices i + 1 and i + 2 are not in t
        k ← the smallest index in t s.t. k > i
        subtract k − i from all indices y s.t. y > i
```

Figure 13: Canonicalization of fragments extracted from parse trees. These sample 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.

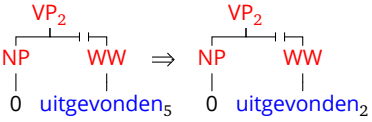
1. Translate indices so that they start at 0; e.g.:



2. Reduce spans of frontier non-terminals to length 1; move surrounding indices accordingly; e.g.:



3. Compress gaps to length 1; e.g.:



We will refer to the combination of Double-DOP with discontinuous constituents as Disco-2DOP. When recurring fragments are extracted from the Tiger treebank (cf. Section 8.1), we find that 10.4%

of fragment types contain a discontinuous node (root, internal, or substitution site). This can be contrasted with the observation that 30% of sentences in the Tiger treebank contain one or more discontinuous constituents, and that 20.9% of production types in the PLCFRS treebank grammar of Tiger contain a discontinuous non-terminal. On the other hand, when occurrence frequencies are taken into account, both the fragments and productions with discontinuities account for around 6.5% of the total frequency mass.

6 PARSING WITH PLCFRS AND PDTSG

After extracting fragments by means of the method of Section 5, we augment the set of fragments with all depth 1 fragments, in order to preserve complete coverage of the training set trees. Since depth 1 fragments are equivalent to single grammar productions, this ensures strong equivalence between the TSG and the respective treebank grammar.[6] We then apply the grammar transformation (cf. Section 4.2.1) to turn the fragments into productions. Productions corresponding to fragments are assigned a probability based on the relative frequency of the respective fragment; productions introduced by the transformation are given a probability of 1. For an example, please refer back to Figure 11.

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).

Parsing with LCFRS can be done with a weighted deduction system and an agenda-based parser. The deduction steps are given in Figure 14; for the pseudo-code of the parser see Algorithm 2, which is an extended version of the parser in Kallmeyer and Maier (2010, 2013) that obtains the complete parse forest as opposed to just the Viterbi derivation.

[6] Previous DOP work such as Zollmann and Sima'an (2005) adds all possible tree fragments up to depth 3. Preliminary experiments on 2DOP gave no improvement on performance, while tripling the grammar size; therefore we do not apply this in further experiments.

Figure 14: Weighted deduction system for binarized LCFRS

Lexical:	$\dfrac{}{p : [A, \langle\langle w_i \rangle\rangle]}$	$p : A(w_i) \rightarrow \varepsilon \in \mathscr{G}$
Unary:	$\dfrac{x : [B, \alpha]}{p \cdot x : [A, \alpha]}$	$p : A(\alpha) \rightarrow B(\alpha)$ is an instantiated rule from $\mathscr{G}$
Binary:	$\dfrac{x : [B, \beta],\ y : [C, \gamma]}{p \cdot x \cdot y : [A, \alpha]}$	$p : A(\alpha) \rightarrow B(\beta)\ C(\gamma)$ is an instantiated rule from $\mathscr{G}$
Goal:	$[S, \langle\langle w_1 \cdots w_n \rangle\rangle]$	

In Section 6.1 we describe the probabilistic instantiation of DTSG and the criterion to select the best parse. Section 6.2 describes how derivations from the compressed TSG are converted back into trees composed of the full elementary trees. Section 6.4 describes how coarse-to-fine pruning is employed to make parsing efficient.

Algorithm 2 A probabilistic agenda-based parser for LCFRS.

INPUT: A sentence $w_1 \cdots w_n$, a grammar $\mathscr{G}$
OUTPUT: A chart $\mathscr{C}$ with Viterbi probabilities, a parse forest $\mathscr{F}$.

```
initialize agenda A with all possible POS tags for input
WHILE A not empty
    ⟨I, x⟩ ← pop item with best score on agenda
    add ⟨I, x⟩ to C
    FOR ALL ⟨I′, z⟩ that can be deduced from ⟨I, x⟩ and items in C
        IF I′ ∉ A ∪ C
            enqueue ⟨I′, z⟩ in A
        ELSE IF I′ ∈ A ∧ z > score for I′ in A
            update weight of I′ in A to z
        add edge for I′ to F
```

6.1 *Probabilities and disambiguation*

Our probabilistic model uses the relative frequency estimate (RFE), which has shown good results with the Double-DOP model (Sangati and Zuidema 2011). The relative frequency of a fragment is the number of its occurrences, divided by the total number of occurrences 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 compute tree probabilities on the basis of the 10,000 most probable DOP derivations, and select the tree with the largest probability. Although the algorithm of Huang and Chiang (2005) makes it is possible to extract the exact k-best derivations from a derivation forest, we apply pruning while building the forest.

6.2 *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[7] 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.

6.3 *Efficient discontinuous parsing*

We review several strategies for making discontinuous parsing efficient. As noted by Levy (2005, p. 138), the intrinsic challenge of discontinuous constituents is that a parser will generate a large number of potential discontinuous spans.

6.3.1 Outside estimates

Outside estimates (also known as context-summary estimates and figures-of-merit) are computed offline for a given grammar. During parsing they provide an estimate of the outside probability for a given constituent, i.e., the probability of a complete derivation with that constituent divided by the probability of the constituent. The estimate can be used to prioritize items in the agenda. Estimates were first introduced for discontinuous LCFRS parsing in Kallmeyer and Maier (2010, 2013). Their estimates are only applied up to sentences of 30 words. Beyond 30 words the table grows too large.

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

A different estimate is given by Angelov and Ljunglöf (2014), who succeed in parsing longer sentences and providing an A* estimate, which is guaranteed to find the best derivation.

6.3.2 Non-projective dependency conversion

Hall and Nivre (2008), Versley (2014), and Fernández-González and Martins (2015) apply a reversible dependency conversion to the Tiger treebank, and use a non-projective dependency parser to parse with the converted treebank. The method has the advantage of being fast due to the greedy nature of the arc-eager transition-based dependency parser that is employed. The parser copes with non-projectivity by reordering tokens during parsing. Experiments are reported on the full Tiger treebank without length restrictions.

6.3.3 Reducing fan-out

The most direct way of reducing the complexity of LCFRS parsing is to reduce the fan-out of the grammar.

Maier *et al.* (2012) introduces a linguistically motivated reduction of the fan-outs of the Negra and Penn treebanks to fan-out 2 (up to a single gap per constituent). This enables parsing of sentences of up to length 40.

Nederhof and Vogler (2014) introduce a method of synchronous parsing with an LCFRS and a definite clause grammar. A parameter allows the fan-out (and thus parsing complexity) of the LCFRS to be reduced. Experiments are reported on sentences of up to 30 words on a small section of the Tiger treebank.

6.3.4 Coarse-to-fine pruning

We will focus on coarse-to-fine pruning, introduced in Charniak *et al.* (2006) and applied to discontinuous parsing by van Cranenburgh (2012a), who reports parsing results on the Negra treebank without length restrictions. Compared to the previous methods, this method does not change the grammar, but adds several new grammars to be used as preprocessing steps. Compared to the outside estimates, this method exploits sentence-specific information, since pruning information is collected during parsing with the coarser grammars.

Pauls and Klein (2009) present a comparison of coarse-to-fine and (hierarchical A*) outside estimates, and conclude that except when

near-optimality is required, coarse-to-fine is more effective as it prunes a larger number of unlikely constituents.

A similar observation is obtained from a comparison of the discontinuous coarse-to-fine method and the outside estimates of Angelov and Ljunglöf (2014): coarse-to-fine is faster with longer sentences (30 words and up), at the cost of not always producing the most likely derivation (Ljunglöf, personal communication).

6.4 *Coarse-to-fine pipeline*

In order to tame the complexity of LCFRS and DOP, we apply coarse-to-fine pruning. Different grammars are used in the sequel, each being an overgenerating approximation of the next. That is, a coarse grammar will generate a larger set of constituents than a fine grammar. Parsing with a coarser grammar is more efficient, and all constituents which can be ruled out as improbable with a coarser grammar can be discarded as candidates when parsing with the next grammar. A constituent is ruled improbable if it does not appear in the k-best derivations of a parse forest. We use the same setup as in van Cranenburgh (2012a); namely, we parse in three stages, using three different grammars:

1. Split-PCFG: A CFG 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. A discontinuous span $X_n\langle x_1, \ldots, x_n\rangle$ is added to the chart only if all of $X_n^{*m}\langle x_m\rangle$ with $1 \leqslant m \leqslant n$ are part of the k-best derivations of the chart of the previous stage.
3. Disco-DOP: The discontinuous DOP grammar; uses tree fragments instead of individual productions from the treebank. A discontinuous span $X_n\langle x_1, \ldots, x_n\rangle$ is added to the chart only if $X_n\langle x_1, \ldots, x_n\rangle$ is part of the k-best derivations of the chart of the previous stage, or if X_n is an intermediate symbol introduced by the TSG compression.

The first stage is necessary because without pruning, the PLCFRS generates too many discontinuous spans, the majority of which are improbable or not even part of a complete derivation. The second stage

is not necessary for efficiency but gives slightly better accuracy on discontinuous constituents.

For example, while parsing the sentence "Wake your friend up," the discontinuous VP "Wake … up" may be produced in the PLCFRS stage. Before allowing this constituent to enter into the agenda and the chart, the chart of the previous stage is consulted to see if the two discontinuous components "Wake" and "up" were part of the k-best derivations. In the DOP stage, multiple elementary trees may be headed by this discontinuous constituent, and again they are only allowed on the chart if the previous stage produced the constituent as part of its k-best derivations.

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. In other coarse-to-fine work the pruning criterion is based on a posterior threshold (e.g., Charniak *et al.* 2006; Bansal and Klein 2010); the k-best approach has the advantage that it does not require the computation of inside and outside probabilities.

For the initial PCFG stage, we apply beam search as in Collins (1999). The highest scoring item in each cell is tracked and only items up to 10,000 times less probable are allowed to enter the chart.

Experiments and results are described in Sections 8–9.

7 DISCONTINUITY WITHOUT LCFRS

The idea up to now has been to generate discontinuous constituents using formal rewrite operations of LCFRS. It should be noted, however, that the PCFG approximation used in the pruning stage reproduces discontinuities using information derived from the non-terminal 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 pipeline that runs in cubic time. While the CFG approximation increases the independence assumptions for discontinuous constituents, the use of large fragments in the DOP approach can mitigate this increase. To create the CFG approximation of the discontinuous treebank grammar, the treebank is transformed by splitting discontinuous constituents into several non-

terminal nodes (as explained in Section 4.1), after which grammar productions are extracted. This last step can also be replaced with fragment extraction to obtain a DOP grammar from the transformed treebank. We shall refer to this alternative approach as 'Split-2DOP.' The coarse-to-fine pipeline is now as follows:

1. Split-PCFG: A treebank grammar based on the CFG approximation of discontinuous constituents; rewrites spans of discontinuous constituents independently.
2. Split-2DOP grammar: tree fragments based on the same transformed treebank as above.

Since every discontinuous non-terminal is split up into a new non-terminal for each of its spans, the independence assumptions for that non-terminal in a probabilistic grammar are increased. While this representation is not sufficient to express the full range of nested discontinuous configurations, it appears adequate for the linguistic phenomena in the treebanks used in this work, since their trees can be unambiguously transformed back and forth into this representation. Moreover, the machinery of Data-Oriented Parsing mitigates the increase in independence assumptions through the use of large fragments. We can therefore parse using a DOP model with a context-free grammar as the symbolic backbone, and still recover discontinuous constituents.

8 EXPERIMENTAL SETUP

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

8.1 *Treebanks and preprocessing*

We evaluate on three languages: for German, we use the Negra (Skut *et al.* 1997) and Tiger (Brants *et al.* 2002) treebanks; for English, we use a discontinuous version of the Penn treebank (Evang and Kallmeyer 2011); and for Dutch, we use the Lassy (Van Noord 2009) and CGN (van der Wouden *et al.* 2002) treebanks; cf. Table 1. Negra and Tiger contain 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

Table 1: The discontinuous treebanks used in the experiments and the number of sentences used for development, training, and testing

Treebank	train (sentences)	dev (sentences)	test (sentences)
GERMAN			
Negra	18,602 (#1–18,602)	1000 (#19,603–20,602)	1000 (#18,603–19,602)
Tiger	40,379 / 45,427	5048	5047
ENGLISH			
PTB: WSJ	39,832	1346	2416
DUTCH			
Lassy small	52,157	6520	6523
CGN	70,277	2000	2000

been converted to discontinuous constituents; we use the version used in Evang and Kallmeyer (2011, Sections 5.1–5.2) without restrictions on the transformations. The Lassy treebank is referred to as a dependency treebank but when discontinuity is allowed it can be directly interpreted as a constituency treebank. The *Corpus Gesproken Nederlands* (CGN, Spoken Dutch Corpus; van der Wouden *et al.* 2002) is a Dutch spoken language corpus with the same syntactic annotations. We use the syntactically annotated sentences from the Netherlands (i.e., without the Flemish part) of up to 100 tokens. The train-dev-test splits we employ are as commonly used for the Penn treebank: sec. 2–21, sec. 24, sec. 23, respectively. 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, sec. 0 is used as test, sec. 1 as development, and 2–9 as training. When parsing the Tiger test set, the development set is added to the training set as well; while this is not customary, it ensures the results are comparable with Hall and Nivre (2008).

The same split is applied to the CGN treebank but with a single training set. For Lassy the split is our own.[8]

[8] The Lassy split derives from 80–10–10 partitions of the canonically ordered sentence IDs in each subcorpus (viz. `dpc`, `WR`, `WS`, and `wiki`). Canonically ordered refers to a 'version sort' where an identifier such as '2.12.a' is treated as a tuple of three elements compared consecutively.

For purposes of training we apply heuristics for head assignment (Klein and Manning 2003) and binarize the trees in the training sets head-outward with $h = 1$, $v = 1$ markovization; i.e., n-ary nodes are factored into nodes specifying an immediate sibling and parent. Note that for LCFRS, a binarization may increase the fan-out, and thus the complexity of parsing. It is possible to select the binarization in such a way as to minimize this complexity (Gildea 2010). However, experiments show that this increase in fan-out does not actually occur, regardless of the binarization strategy (van Cranenburgh 2012a). Head-outward means that constituents are binarized in a right-factored manner up until the head child, after which the rest of the binarization continues in a left-factored manner.

We add fan-out markers to guarantee unique fan-outs for non-terminal labels, e.g., $\{\text{VP}, \text{VP}_2, \text{VP}_3, \ldots\}$, which are removed again for evaluation.

For the Dutch and German treebanks, punctuation is not part of the syntactic annotations. This causes spurious discontinuities, as the punctuation interrupts the constituents dominating its surrounding tokens. Additionally, punctuation provides a signal for constituent boundaries, and it is useful to incorporate it as part of the rest of the phrase structures. We use the method described in van Cranenburgh (2012a): punctuation is attached to the highest constituent that contains a neighbor to its right. With this strategy there is no increase in the amount of discontinuity with respect to a version of the treebank with punctuation removed. The CGN treebank contains spoken language phenomena, including disfluencies such as interjections and repeated words. In preprocessing, we treat these as if they were punctuation tokens; i.e., they are moved to an appropriate constituent (as defined above) and are ignored in the evaluation.

The complexity of parsing with a binarized LCFRS is $O(n^{3\varphi})$ with φ the highest fan-out of the non-terminals in the grammar (Seki *et al.* 1991). For a given grammar, it is possible to give a tighter upper bound on the complexity of parsing. Given the unique fan-outs of non-terminals in a grammar, the number of operations it takes to apply a production is the sum of the fan-outs in the production (Gildea 2010):

$$c(p) = \varphi(A) + \sum_{i=1}^{r} \varphi(B_i)$$

The complexity of parsing with a grammar is then the maximum value of this measure for productions in the grammar. In our experiments we find a worst-case time complexity of $O(n^9)$ for parsing with the DOP grammars extracted from Negra and WSJ. The following sentence from Negra contributes a grammar production with complexity 9. The production is from the VP of *vorgeworfen*; bracketed words are from other constituents, indicating the discontinuities:

(2) Den Stadtteilparlamentariern [ist] immer wieder ["Kirchturmpolitik"]
The district-MPs have always again "parochialism"
vorgeworfen [worden], weil sie nicht über die Grenzen des
accused been, because they not beyond the boundaries of-the
Ortsbezirks hinausgucken würden.
local-district look-out would.
'Time and again, the district MPs have been accused of "parochialism" because they would not look out beyond the boundaries of the local district.'

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.

8.2 *Unknown words*

In initial experiments the parser is trained and evaluated on gold standard 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). An open class threshold σ determines which tags are considered open class tags; 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 set that do not occur more than 4 times are replaced with signatures based on a list of features; words in the test set which are not part of the known words from the training set are replaced with similar signatures. The features are defined in the Stanford parser as *Model 4*, which is relatively language independent; cf. Table 2 for the list of features.[9] Signatures are formed by concatenating the names of features that apply

[9] This table is based on code from the Stanford parser (release 2014-08-27), specifically the method `getSignature4` in the file `EnglishUnknownWordModel.java`.

Feature	Description
AC	All capital letters
SC	Initial capital, first word in sentence
C	Initial capital, other position
L, U	Has lower / upper case letter
S	No letters
N, n	All digits / one or more digits
H, P, C	Has dash / period / comma
x	Last character if letter and length > 3

Table 2:
Unknown word features,
Stanford parser *Model 4*

to a word; e.g., 'forty-two' gives `_UNK-L-H-o`. A probability mass ϵ is assigned for combinations of known open class words with unseen tags. We use $\epsilon = 0.01$. We tuned σ on each training set to ensure that no closed class words are identified as open class words; for English and German we use $\sigma = 150$, and we use $\sigma = 100$ for Dutch.

8.3 *Function tags*

We investigated two methods of having the parser produce function tags in addition to the usual phrase labels. The first method is to train a separate discriminative classifier that adds function tags to parse trees in a post-processing step. This approach is introduced in Blaheta and Charniak (2000). We employed their feature set.

Another approach is to simply append the function tags to the non-terminal labels, resulting in, e.g., NP-SBJ and NP-OBJ for subject and object noun phrases. While this approach introduces sparsity and may affect the performance without function tags, we found this approach to perform best and therefore report results with this approach. Gabbard *et al.* (2006) and Fraser *et al.* (2013) use this approach as well. Compared to the classifier approach, it does not require any tuning, and the resulting model is fully generative. We apply this to the Tiger, WSJ, and Lassy treebanks.

The Penn treebank differs from the German and Dutch treebanks with respect to function tags. The Penn treebank only has function tags on selected non-terminals (never on preterminals) and each non-terminal may have several function tags from four possible categories; whereas the German and Dutch treebanks have a single function tag

on most non-terminals. The tag set also differs considerably: the Penn treebank has 20 function tags, Lassy has 31, and Tiger has 43.

8.4 *Treebank refinements*

We apply a set of manual treebank refinements based on previous work. In order to compare the results on Negra with previous work, we do not apply the state splits when working with gold standard POS tags.

For Dutch and German we split the POS tags for the sentence-ending punctuation '.!?'. For all treebanks we add the feature 'year' to the preterminal label of tokens with numbers in the range 1900–2040, and replace the token with 1970. Other numbers are replaced with 000.

8.4.1 Tiger

For Tiger we apply the refinements described in Fraser *et al.* (2013). Since the Negra treebank is only partially annotated with morphological information, we do not apply these refinements to that treebank.

8.4.2 WSJ

We follow the treebank refinements of Klein and Manning (2003) for the Wall Street Journal section of the Penn treebank.

8.4.3 Lassy

The Lassy treebank contains fine-grained part-of-speech tags with morphological features. It is possible to use the full part-of-speech tags as the preterminal labels, but this introduces sparsity. We select a subset of features to add to the preterminal labels:

- nouns: proper/regular;
- verbs: auxiliary/main, finite/infinite;
- conjunctions: coordinating/subordinating;
- pronouns: personal/demonstrative;
- pre- vs. postposition.

Additionally, we percolate the feature identifying finite and infinite verbs to the parents and grandparents of the verb.

For multi-word units (MWU), we append the label of its head child. This helps distinguish MWUs as being nominal, verbal, prepositional, or otherwise.

The last two transformations are based on those for Tiger. Unary NPs are added for single nouns and pronouns in sentential, prepositional and infinitival constituents. For conjuncts, the function tag of the parent is copied. Both transformations can be reversed.

Since the CGN treebank uses a different syntax for the fine-grained POS tags, we do not apply these refinements to that treebank.

8.5 *Metrics*

We employ the exact match and Parseval measures (Black *et al.* 1992) as evaluation metrics. Both are based on bracketings that identify the label and yield of each constituent. The exact match is the proportion of sentences in which all labelled bracketings are correct. The Parseval measures consist of the precision, recall, and F-measure of the correct labelled bracketings averaged across the treebank. Since the POS accuracy is crucial to the performance of a parser and neither of the previous metrics reflect it, we also report the proportion of correct POS tags.

We use the evaluation parameters typically used with `EVALB` on the Penn treebank. Namely, the root node and punctuation are not counted towards the score (similar to `COLLINS.prm`,[10] 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.

It is not possible to directly compare evaluation results from discontinuous parsing to existing state-of-the-art parsers that do not produce discontinuous constituents, since parses without discontinuous constituents contain a different set of bracketings; cf. Figure 15, which compares discontinuous bracketings to the bracketings extracted from a tree in which discontinuity has been resolved by attaching non-head siblings higher in the tree, as used in work on parsing Negra.

[10] This is part of the EVALB software, cf. `http://nlp.cs.nyu.edu/evalb/`

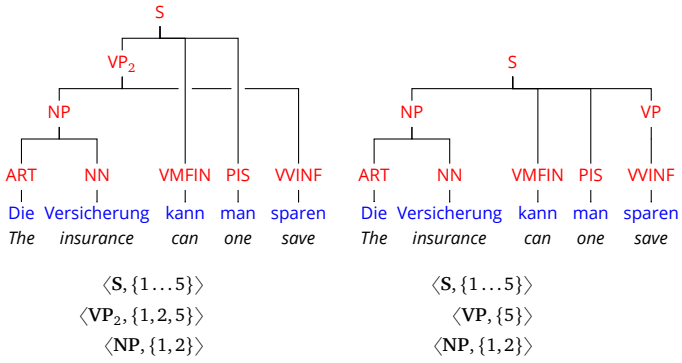


Figure 15: Bracketings from a tree with and without discontinuous constituents

Compared to an evaluation of bracketings without discontinuous constituents, an evaluation including discontinuous bracketings is more stringent. This is because bracketings are scored in an all-or-nothing manner, and a discontinuous bracketing includes non-local elements that would be scored separately when discontinuity is removed in a preprocessing step.

For function tags we use two metrics:

1. The non-null metric of Blaheta and Charniak (2000), which is the F-score of function tags on all correctly parsed bracketings. Since the German and Dutch treebanks include function tags on pre-terminals, we also include function tags on correctly tagged words in this metric.
2. A combined F-measure on bracketings of the form $\langle C, F, \text{span} \rangle$, where C is a syntactic category and F a function tag.

9 EVALUATION

This section presents an evaluation on three languages, and with respect to the use of function tags, tree fragments, pruning, and probabilities.

9.1 *Main results on three languages*

Table 3 lists the results for discontinuous parsing of three Germanic languages, with unknown word models. The cited works by Kallmeyer and Maier (2013) and Evang and Kallmeyer (2011) also use LCFRS

for discontinuity but employ a treebank grammar with relative frequencies of productions. Hall and Nivre (2008), Versley (2014), and Fernández-González and Martins (2015) use a conversion to dependencies discussed in Section 6.3.2. For English and German our 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 of size $\leqslant 40$ 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 three hours instead of ten on a single core. The grammar is also more compact: the Disco-2DOP grammar is only a third the size of that of the DOP reduction, at 6 MB versus 18 MB compressed size.

Table 3 also includes results from van Cranenburgh and Bod (2013) who do not add function tags to non-terminal labels nor apply the extensive treebank refinements described in Sections 8.3–8.4. Although the refinements and some of the function tags would be expected to improve performance, the rest of the function tags increase sparsity and consequently the resulting F-scores are slightly lower; but this tradeoff seems to be justified in order to get parse trees with function tags. The results on CGN show a surprisingly high exact match score. This is due to a large number of interjection utterances, e.g., "uhm."; since such sentences only consist of a root node and POS tags, the bracketing F_1-score is not affected by this.

9.2 *Function tags*

Table 4 reports an evaluation including function tags. For these three treebanks, the models reproduce most of the information in the original treebank. The following parts are not yet incorporated. The German and Dutch treebanks contain additional lexical information consisting of lemmas and morphological features. These could be added to the non-terminal labels of the model or obtained from an external POS tagger. Lastly, some non-terminals have multiple parents; these occur in the German and Dutch treebanks and are referred to as secondary edges.

9.3 *All-fragments vs. recurring fragments*

The original Disco-DOP model (van Cranenburgh *et al.* 2011) is based on an all-fragments model, while Disco-2DOP is based on recurring

Table 3: Discontinuous parsing of three Germanic languages. POS is the part-of-speech tagging accuracy; F_1 is the labelled bracketing F_1-score; EX is the exact match score. Results marked with * use gold standard POS tags; those marked with † do not discount the root node and punctuation. NB: Kallmeyer and Maier (2013) and Evang and Kallmeyer (2011) use a different test set and length restriction. 'vanCraBod2013' refers to van Cranenburgh and Bod (2013), and 'FeMa2015' to Fernández-González and Martins (2015)

		DEV			TEST		
Treebank and parser	$\|w\|$	POS	F_1	EX	POS	F_1	EX
GERMAN							
Negra							
van Cranenburgh (2012a)*	$\leqslant 40$	100	74.3	34.3	100	72.3	33.2
Kallmeyer and Maier (2013)*†	$\leqslant 30$				100	75.8	
this work, Disco-2DOP*	$\leqslant 40$	100	**77.7**	**41.5**	100	**76.8**	**40.5**
this work, Disco-2DOP	$\leqslant 40$	96.7	76.4	39.2	96.3	74.8	38.7
Tiger							
Hall and Nivre (2008)	$\leqslant 40$				97.0	75.3	32.6
Versley (2014)	$\leqslant 40$				100	74.2	37.3
FeMa2015	$\leqslant 40$					**82.6**	**45.9**
vanCraBod2013, Disco-2DOP	$\leqslant 40$	97.6	78.7	40.5	97.6	78.8	40.8
this work, Disco-2DOP	$\leqslant 40$	96.6	78.3	40.2	96.1	78.2	40.0
this work, Split-2DOP	$\leqslant 40$	96.6	78.1	39.2	96.2	78.1	39.0
ENGLISH							
WSJ							
Evang and Kallmeyer (2011)*†	< 25				100	79.0	
vanCraBod2013, Disco-2DOP	$\leqslant 40$	96.0	85.2	28.0	96.6	85.6	31.3
this work, Disco-2DOP	$\leqslant 40$	96.1	**86.9**	**29.5**	96.7	**87.0**	**34.4**
this work, Split-2DOP	$\leqslant 40$	96.1	86.7	**29.5**	96.7	87.0	33.9
DUTCH							
Lassy							
vanCraBod2013, Disco-2DOP	$\leqslant 40$	94.1	**79.0**	**37.4**	94.6	**77.0**	**35.2**
this work, Disco-2DOP	$\leqslant 40$	96.7	78.3	36.2	96.3	76.6	34.0
this work, Split-2DOP	$\leqslant 40$	96.8	78.0	34.9	96.3	76.2	32.7
CGN							
this work, Disco-2DOP	$\leqslant 40$	96.7	**72.6**	**64.1**	96.7	**73.0**	**63.8**
this work, Split-2DOP	$\leqslant 40$	96.6	71.2	63.4	96.7	72.2	63.3

Language, treebank	phrase labels	function tags	combined
German, Tiger	78.2	93.5	68.1
English, WSJ	87.0	86.3	82.5
Dutch, Lassy	76.6	92.8	70.0

Table 4:
Evaluation of function tags on sentences $\leqslant 40$ words, test sets; F_1 scores as defined at the end of Section 8.5

fragments. Table 5 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 labelled 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. Figure 16 shows the average time spent in each stage using the latter model on WSJ. The average time to parse a sentence ($\leqslant 40$ words) for this grammar is 7.7 seconds. Efficiency could be improved significantly by improving the PCFG parser using better chart representations such as packed parse forests and bit vectors (Schmid 2004).

Model	k = 50 F_1	k = 5000 F_1
DOP reduction: disco-DOP	74.3	73.5
Double-DOP: disco-2DOP	76.3	**77.7**

Table 5:
Comparing F-scores for 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 standard POS tags

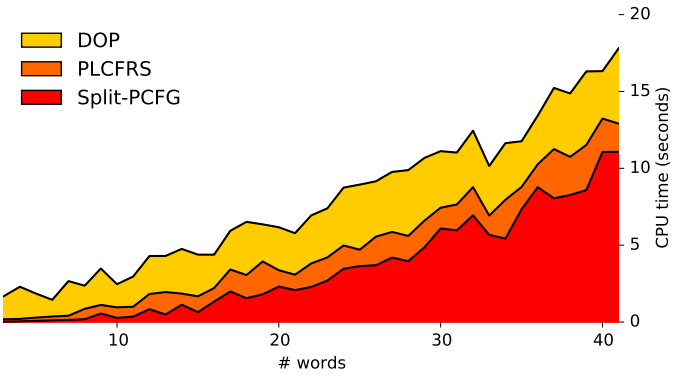


Figure 16:
Average time spent in each stage for sentences by length; disco-2DOP, WSJ development set

9.4 *Effects of pruning*

The effects of pruning can be further investigated by comparing different levels of pruning. We first parse the sentences in the Negra development set that are up to 30 words long with a PLCFRS treebank grammar, with $k = 10,000$ and without pruning. Out of 897 sentences, the Viterbi derivation is pruned on only 14 occasions, while the pruned version is about 300 times faster.

Table 6 shows results for different levels of pruning on sentences of all lengths. For sentences of all lengths it is not feasible to parse with the unpruned PLCFRS. However, we can compare the items in the parse forest after pruning and the best derivation to the gold tree from the treebank. From the various measures, it can be concluded that the pruning has a large effect on speed and the number of items in the resulting parse forest, while having only a small effect on the quality of the parse (forest).

Table 6: Results for different levels of pruning; mean over 1000 sentences

	(PCFG)	$k=100$	$k=1000$	$k=5000$	$k=10,000$
CPU time (seconds)	2.461	0.128	0.193	0.444	0.739
Number of items in chart	69,570.5	207.6	282.7	378.2	436.5
Percentage of gold standard items in chart	94.7	94.2	97.2	98.1	98.4
F_1 score	69.3	69.8	69.9	69.9	69.8

9.5 *Without LCFRS*

Table 3 shows that the Disco-2DOP and Split-2DOP techniques have comparable performance, demonstrating that the complexity of LCFRS parsing can be avoided. Table 7 shows the performance in each step of the coarse-to-fine pipelines, with and without LCFRS. 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.

Pipeline	F_1	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 7:
Parsing discontinuous constituents is possible without LCFRS (Negra development set, gold standard POS tags; results are for final stage)

9.6 *The role of probabilities*

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 specifically due to the more fine-grained statistics (i.e., frequencies of more specific events), or generally 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 performance was surprisingly low, from 77.7 to 74.1 F_1 – suggesting that most of the improvement over the 65.9 F_1 score of the PLCFRS treebank grammar comes from memorizing larger chunks, as opposed to statistical reckoning.

9.7 *Previous work*

Earlier work on recovering empty categories and their antecedents in the Penn treebank (Johnson 2002; Levy and Manning 2004; Gabbard *et al.* 2006; Schmid 2006; Cai *et al.* 2011) has recovered nonlocal dependencies by producing the traces and co-indexation as in the original annotation. If the results include both traces and antecedents (which holds for all but the last work cited), the conversion to discontinuous constituents of Evang and Kallmeyer (2011) could be applied to obtain a discontinuous F-score. Since this would require access to the original parser output, we have not pursued this.

As explained in Section 8.5, it is not possible to directly compare the results to existing parsers that do not produce discontinuous constituents. However, the F-measures do give a rough measure, since the majority of constituents are not discontinuous.

For English, there is a result with 2DOP by Sangati and Zuidema (2011) with an F_1 score of 87.9. This difference can be attributed to the absence of discontinuous bracketings, as well as their use of the Max-

imum Constituents Parse instead of the Most Probable Parse; the former optimizes the F-measure instead of the exact match score. Shindo *et al.* (2012) achieve an F_1 score of 92.9 with a Bayesian TSG that uses symbol refinement through latent variables (i.e., automatic state splitting).

For German, the best results without discontinuity and no length restriction are F_1 scores of 84.2 for Negra (Petrov 2010) and 76.8 for Tiger (Fraser *et al.* 2013; note that this result employs a different train-dev-test split than the one in this work).

10 CONCLUSION

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

Surprisingly, it turns out that the formal power of LCFRS is not necessary to describe discontinuity, since equivalent results can be obtained with a probabilistic tree-substitution grammar in which non-local relations are encoded in the non-terminal labels. In other words, it is feasible to produce discontinuous constituents without invoking mild context-sensitivity.

We have presented parsing results on three languages. Compared to previous work on statistical parsing, our models are linguistically richer. In addition to discontinuous constituents, our models also reproduce function tags from the treebank. While there have been previous results on reproducing non-local relations or function tags, this work reproduces both using models derived straightforwardly from treebanks, while exploiting ready-made treebank transformations for improved performance.

The source code of the parser used in this work is available at `https://github.com/andreasvc/disco-dop`.

ACKNOWLEDGMENTS

We are grateful to Kilian Evang for supplying the discontinuous Penn treebank, to the reviewers for detailed comments, and to Dave Carter and Adam Przepiórkowski for copy-editing suggestions.

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

REFERENCES

Krasimir Angelov and Peter Ljunglöf (2014), Fast statistical parsing with parallel multiple context-free grammars, in *Proceedings of EACL*, pp. 368–376, http://aclweb.org/anthology/E14-1039.

Mohit Bansal and Dan Klein (2010), Simple, accurate parsing with an all-fragments grammar, in *Proceedings of ACL*, pp. 1098–1107, http://aclweb.org/anthology/P10-1112.

François Barthélemy, Pierre Boullier, Philippe Deschamp, and Éric de la Clergerie (2001), Guided parsing of range concatenation languages, in *Proceedings of ACL*, pp. 42–49, http://aclweb.org/anthology/P01-1007.

Shane Bergsma, Matt Post, and David Yarowsky (2012), Stylometric analysis of scientific articles, in *Proceedings of NAACL*, pp. 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*, pp. 185–192, http://aclweb.org/anthology/P92-1024.

Don Blaheta and Eugene Charniak (2000), Assigning function tags to parsed text, in *Proceedings of NAACL*, pp. 234–240, http://aclweb.org/anthology/A00-2031.

Rens Bod (1992), A computational model of language performance: data-oriented parsing, in *Proceedings COLING*, pp. 855–859, http://aclweb.org/anthology/C92-3126.

Rens Bod (1995), The problem of computing the most probable tree in data-oriented parsing and stochastic tree grammars, in *Proceedings of EACL*, pp. 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*, pp. 69–76, http://aclweb.org/anthology/P01-1010.

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.

Adriane BOYD (2007), Discontinuity revisited: An improved conversion to context-free representations, in *Proceedings of the Linguistic Annotation Workshop*, pp. 41–44, http://aclweb.org/anthology/W07-1506.

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*, pp. 24–41, http://www.bultreebank.org/proceedings/paper03.pdf.

Joan BRESNAN, Ronald M. KAPLAN, Stanley PETERS, and Annie ZAENEN (1982), Cross-serial dependencies in Dutch, *Linguistic Inquiry*, 13(4):613–635.

Shu CAI, David CHIANG, and Yoav GOLDBERG (2011), Language-independent parsing with empty elements, in *Proceedings of ACL-HLT*, pp. 212–216, http://aclweb.org/anthology/P11-2037.

Samy CHAMBI, Daniel LEMIRE, Owen KASER, and Robert GODIN (2015), Better bitmap performance with Roaring bitmaps, *Software: Practice and Experience*, ISSN 1097-024X, doi:10.1002/spe.2325, http://arxiv.org/abs/1402.6407, to appear.

Eugene CHARNIAK (1996), Tree-bank grammars, in *Proceedings of the National Conference on Artificial Intelligence*, pp. 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*, pp. 168–175, http://aclweb.org/anthology/N06-1022.

David CHIANG (2000), Statistical parsing with an automatically-extracted tree adjoining grammar, in *Proceedings of ACL*, pp. 456–463, http://aclweb.org/anthology/P00-1058.

Noam CHOMSKY (1956), Three models for the description of language, *IRE Transactions on Information Theory*, 2(3):113–124.

Noam CHOMSKY (1965), *Aspects of the Theory of Syntax*, MIT press.

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

Trevor COHN, Sharon GOLDWATER, and Phil BLUNSOM (2009), Inducing compact but accurate tree-substitution grammars, in *Proceedings of NAACL-HLT*, pp. 548–556, http://aclweb.org/anthology/N09-1062.

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

Peter Dienes and Amit Dubey (2003), Deep syntactic processing by combining shallow methods, in *Proceedings of ACL*, pp. 431–438, http://aclweb.org/anthology/P03-1055.

Amit Dubey and Frank Keller (2003), Probabilistic parsing for German using sister-head dependencies, in *Proceedings of ACL*, pp. 96–103, http://aclweb.org/anthology/P03-1013.

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

Daniel Fernández-González and André F. T. Martins (2015), Parsing as reduction, in *Proceedings of ACL*, pp. 1523–1533, http://aclweb.org/anthology/P15-1147.

Alexander Fraser, Helmut Schmid, Richárd Farkas, Renjing Wang, and Hinrich Schütze (2013), Knowledge sources for constituent parsing of German, a morphologically rich and less-configurational language, *Computational Linguistics*, 39(1):57–85, http://aclweb.org/anthology/J13-1005.

Ryan Gabbard, Mitchell Marcus, and Seth Kulick (2006), Fully parsing the Penn treebank, in *Proceedings of NAACL-HLT*, pp. 184–191, http://aclweb.org/anthology/N06-1024.

Stuart Geman and Mark Johnson (2004), Probability and statistics in computational linguistics, a brief review, in Mark Johnson, Sanjeev P. Khudanpur, Mari Ostendorf, and Roni Rosenfeld, editors, *Mathematical foundations of speech and language processing*, pp. 1–26, Springer.

Daniel Gildea (2010), Optimal parsing strategies for linear context-free rewriting systems, in *Proceedings of NAACL-HLT*, pp. 769–776, http://aclweb.org/anthology/N10-1118.

Joshua Goodman (2003), Efficient parsing of DOP with PCFG-reductions, in Bod *et al.* (2003), pp. 125–146.

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*, pp. 725–735, http://aclweb.org/anthology/D11-1067.

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

Lars HOOGWEG (2003), Extending DOP with insertion, in Bod *et al.* (2003), pp. 317–335.

Yu-Yin HSU (2010), Comparing conversions of discontinuity in PCFG parsing, in *Proceedings of Treebanks and Linguistic Theories*, pp. 103–113, http://hdl.handle.net/10062/15954.

Liang HUANG and David CHIANG (2005), Better k-best parsing, in *Proceedings of IWPT*, pp. 53–64, NB corrected version on author homepage: http://www.cis.upenn.edu/~lhuang3/huang-iwpt-correct.pdf.

Marinus A.C. HUYBREGTS (1976), Overlapping dependencies in Dutch, *Utrecht Working Papers in Linguistics*, 1:24–65.

Mark JOHNSON (2002), A simple pattern-matching algorithm for recovering empty nodes and their antecedents, in *Proceedings of ACL*, pp. 136–143, http://aclweb.org/anthology/P02-1018.

Aravind K. JOSHI (1985), How much context sensitivity is necessary for characterizing structural descriptions: Tree adjoining grammars, in David R. DOWTY, Lauri KARTTUNEN, and Arnold M. ZWICKY, editors, *Natural language parsing: Psychological, computational and theoretical perspectives*, pp. 206–250, Cambridge University Press, New York.

Miriam KAESHAMMER and Vera DEMBERG (2012), German and English treebanks and lexica for tree-adjoining grammars, in *Proceedings of LREC*, pp. 1880–1887, http://www.lrec-conf.org/proceedings/lrec2012/pdf/398_Paper.pdf.

Laura KALLMEYER (2009), A declarative characterization of different types of multicomponent tree adjoining grammars, *Research on Language and Computation*, 7(1):55–99.

Laura KALLMEYER (2010), *Parsing Beyond Context-Free Grammars*, Cognitive Technologies, Springer.

Laura KALLMEYER and Wolfgang MAIER (2010), Data-driven parsing with probabilistic linear context-free rewriting systems, in *Proceedings of COLING*, pp. 537–545, http://aclweb.org/anthology/C10-1061.

Laura KALLMEYER and Wolfgang MAIER (2013), Data-driven parsing using probabilistic linear context-free rewriting systems, *Computational Linguistics*, 39(1):87–119, http://aclweb.org/anthology/J13-1006.

Laura KALLMEYER, Wolfgang MAIER, and Giorgio SATTA (2009), Synchronous rewriting in treebanks, in *Proceedings of IWPT*, http://aclweb.org/anthology/W09-3810.

Fred KARLSSON (2007), Constraints on multiple centre-embedding of clauses, *Journal of Linguistics*, 43(2):365–392.

Dan KLEIN and Christopher D. MANNING (2003), Accurate unlexicalized parsing, in *Proceedings of ACL*, volume 1, pp. 423–430, http://aclweb.org/anthology/P03-1054.

Marco KUHLMANN (2013), Mildly non-projective dependency grammar, *Computational Linguistics*, 39(2):355–387, http://aclweb.org/anthology/J13-2004.

Marco KUHLMANN and Giorgio SATTA (2009), Treebank grammar techniques for non-projective dependency parsing, in *Proceedings of EACL*, pp. 478–486, http://aclweb.org/anthology/E09-1055.

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

Roger LEVY and Christopher D. MANNING (2004), Deep dependencies from context-free statistical parsers: correcting the surface dependency approximation, in *Proceedings of ACL*, pp. 327–334, http://aclweb.org/anthology/P04-1042.

Wolfgang MAIER, Miriam KAESHAMMER, Peter BAUMANN, and Sandra KÜBLER (2014), Discosuite – A parser test suite for German discontinuous structures, in *Proceedings of LREC*, http://www.lrec-conf.org/proceedings/lrec2014/pdf/230_Paper.pdf.

Wolfgang MAIER, Miriam KAESHAMMER, and Laura KALLMEYER (2012), PLCFRS parsing revisited: Restricting the fan-out to two, in *Proceedings of TAG*, volume 11, http://wolfgang-maier.net/pub/tagplus12.pdf.

Wolfgang MAIER and Timm LICHTE (2011), Characterizing discontinuity in constituent treebanks, in *Proceedings of Formal Grammar 2009*, pp. 167–182, Springer.

Wolfgang MAIER and Anders SØGAARD (2008), Treebanks and mild context-sensitivity, in *Proceedings of Formal Grammar 2008*, pp. 61–76.

James D. MCCAWLEY (1982), Parentheticals and discontinuous constituent structure, *Linguistic Inquiry*, 13(1):91–106, http://www.jstor.org/stable/4178261.

Mark-Jan NEDERHOF and Heiko VOGLER (2014), Hybrid grammars for discontinuous parsing, in *Proceedings of COLING*, pp. 1370–1381, http://aclweb.org/anthology/C14-1130.

Timothy J. O'DONNELL, Joshua B. TENENBAUM, and Noah D. GOODMAN (2009), Fragment grammars: Exploring computation and reuse in language, Technical Report MIT-CSAIL-TR-2009-013, MIT CSAIL, http://hdl.handle.net/1721.1/44963.

Almerindo E. OJEDA (1988), A linear precedence account of cross-serial dependencies, *Linguistics and Philosophy*, 11(4):457–492.

Adam Pauls and Dan Klein (2009), Hierarchical search for parsing, in *Proceedings of NAACL-HLT*, pp. 557–565, http://aclweb.org/anthology/N09-1063.

P. Stanley Peters and R. W. Ritchie (1973), On the generative power of transformational grammars, *Information Sciences*, 6:49–83, http://dx.doi.org/10.1016/0020-0255(73)90027-3.

Slav Petrov (2010), Products of random latent variable grammars, in *Proceedings of NAACL-HLT*, pp. 19–27, http://aclweb.org/anthology/N10-1003.

Kenneth L. Pike (1943), Taxemes and immediate constituents, *Language*, 19(2):65–82, http://www.jstor.org/stable/409840.

Matt Post (2011), Judging grammaticality with tree substitution grammar derivations, in *Proceedings of the ACL-HLT 2011*, pp. 217–222, http://aclweb.org/anthology/P11-2038.

Matt Post and Daniel Gildea (2009), Bayesian learning of a tree substitution grammar, in *Proceedings of the ACL-IJCNLP 2009 Conference, Short Papers*, pp. 45–48, http://aclweb.org/anthology/P09-2012.

Brian Roark, Kristy Hollingshead, and Nathan Bodenstab (2012), Finite-state chart constraints for reduced complexity context-free parsing pipelines, *Computational Linguistics*, 38(4):719–753, http://aclweb.org/anthology/J12-4002.

Federico Sangati and Willem Zuidema (2011), Accurate parsing with compact tree-substitution grammars: Double-DOP, in *Proceedings of EMNLP*, pp. 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*, pp. 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*, pp. 7–22, LVVN, Almere, the Netherlands, original title: Taaltheorie en taaltechnologie; competence en performance. English translation: http://iaaa.nl/rs/LeerdamE.html.

Yves Schabes and Richard C. Waters (1995), Tree insertion grammar: cubic-time, parsable formalism that lexicalizes context-free grammar without changing the trees produced, *Computational Linguistics*, 21(4):479–513, http://aclweb.org/anthology/J95-4002.

Helmut Schmid (2004), Efficient parsing of highly ambiguous context-free grammars with bit vectors, in *Proceedings of COLING '04*, http://aclweb.org/anthology/C04-1024.

Helmut SCHMID (2006), Trace prediction and recovery with unlexicalized PCFGs and slash features, in *Proceedings of COLING-ACL*, pp. 177–184, http://aclweb.org/anthology/P06-1023.

William SCHULER, Samir ABDELRAHMAN, Tim MILLER, and Lane SCHWARTZ (2010), Broad-coverage parsing using human-like memory constraints, *Computational Linguistics*, 36(1):1–30, http://aclweb.org/anthology/J10-1001.

William SCHULER, David CHIANG, and Mark DRAS (2000), Multi-component TAG and notions of formal power, in *Proceedings of ACL*, pp. 448–455, http://aclweb.org/anthology/P00-1057.

Hiroyuki SEKI, Takahashi MATSUMURA, Mamoru FUJII, and Tadao KASAMI (1991), On multiple context-free grammars, *Theoretical Computer Science*, 88(2):191–229.

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

Hiroyuki SHINDO, Yusuke MIYAO, Akinori FUJINO, and Masaaki NAGATA (2012), Bayesian symbol-refined tree substitution grammars for syntactic parsing, in *Proceedings of ACL*, pp. 440–448, http://aclweb.org/anthology/P12-1046.

Khalil SIMA'AN (1997), Efficient Disambiguation by means of stochastic tree substitution grammars, in D. JONES and H. SOMERS, editors, *New Methods in Language Processing*, pp. 178–198, UCL Press, UK.

Wojciech SKUT, Brigitte KRENN, Thorsten BRANTS, and Hans USZKOREIT (1997), An annotation scheme for free word order languages, in *Proceedings of ANLP*, pp. 88–95, http://aclweb.org/anthology/A97-1014.

Ben SWANSON, Elif YAMANGIL, Eugene CHARNIAK, and Stuart SHIEBER (2013), A context free TAG variant, in *Proceedings of the ACL*, pp. 302–310, http://aclweb.org/anthology/P13-1030.

Benjamin SWANSON and Eugene CHARNIAK (2012), Native language detection with tree substitution grammars, in *Proceedings of ACL*, pp. 193–197, http://aclweb.org/anthology/P12-2038.

Heike TELLJOHANN, Erhard HINRICHS, and Sandra KÜBLER (2004), The Tüba-D/Z Treebank: Annotating German with a context-free backbone, in *Proceedings of LREC*, pp. 2229–2235, http://www.lrec-conf.org/proceedings/lrec2004/pdf/135.pdf.

Heike TELLJOHANN, Erhard W HINRICHS, Sandra KÜBLER, Heike ZINSMEISTER, and Kathrin BECK (2012), Stylebook for the Tübingen treebank of written German (TüBa-D/Z), technical report, Seminar für Sprachwissenschaft, Universität Tübingen, Germany, http://www.sfs.uni-tuebingen.de/fileadmin/static/ascl/resources/tuebadz-stylebook-1201.pdf.

Marten H. TRAUTWEIN (1995), *Computational pitfalls in tractable grammar formalisms*, Ph.D. thesis, University of Amsterdam, http://www.illc.uva.nl/Research/Publications/Dissertations/DS-1995-15.text.ps.gz.

Andreas VAN CRANENBURGH (2012a), Efficient parsing with linear context-free rewriting systems, in *Proceedings of EACL*, pp. 460–470, corrected version: http://andreasvc.github.io/eacl2012corrected.pdf.

Andreas VAN CRANENBURGH (2012b), Literary authorship attribution with phrase-structure fragments, in *Proceedings of CLFL*, pp. 59–63, revised version: http://andreasvc.github.io/clfl2012.pdf.

Andreas VAN CRANENBURGH (2014), Extraction of phrase-structure fragments with a linear average time tree kernel, *Computational Linguistics in the Netherlands Journal*, 4:3–16, ISSN 2211-4009, http://www.clinjournal.org/sites/default/files/01-Cranenburgh-CLIN2014.pdf.

Andreas VAN CRANENBURGH and Rens BOD (2013), Discontinuous parsing with an efficient and accurate DOP model, in *Proceedings of IWPT*, pp. 7–16, http://www.illc.uva.nl/LaCo/CLS/papers/iwpt2013parser_final.pdf.

Andreas VAN CRANENBURGH, Remko SCHA, and Federico SANGATI (2011), Discontinuous data-oriented parsing: A mildly context-sensitive all-fragments grammar, in *Proceedings of SPMRL*, pp. 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.

Ton VAN DER WOUDEN, Heleen HOEKSTRA, Michael MOORTGAT, Bram RENMANS, and Ineke SCHUURMAN (2002), Syntactic analysis in the spoken Dutch corpus (CGN), in *Proceedings of LREC*, pp. 768–773, http://www.lrec-conf.org/proceedings/lrec2002/pdf/71.pdf.

Gertjan VAN NOORD (2009), Huge parsed corpora in Lassy, in *Proceedings of TLT7*, LOT, Groningen, The Netherlands.

Yannick VERSLEY (2014), Experiments with easy-first nonprojective constituent parsing, in *Proceedings of SPMRL-SANCL 2014*, pp. 39–53, http://aclweb.org/anthology/W14-6104.

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 *Proceedings of ACL*, pp. 104–111, http://aclweb.org/anthology/P87-1015.

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

Rulon S. WELLS (1947), Immediate constituents, *Language*, 23(2):81–117, http://www.jstor.org/stable/410382.

Fei XIA, Chung-Hye HAN, Martha PALMER, and Aravind JOSHI (2001), Automatically extracting and comparing lexicalized grammars for different languages, in *Proceedings of IJCAI*, pp. 1321–1330.

Elif YAMANGIL and Stuart SHIEBER (2012), Estimating compact yet rich tree insertion grammars, in *Proceedings of ACL*, pp. 110–114, http://aclweb.org/anthology/P12-2022.

Andreas ZOLLMANN and Khalil SIMA'AN (2005), A consistent and efficient estimator for data-oriented parsing, *Journal of Automata Languages and Combinatorics*, 10(2/3):367–388, http://staff.science.uva.nl/~simaan/D-Papers/JALCsubmit.pdf.