Experiments with Easy-first nonprojective constituent parsing

Yannick Versley
Department of Computational Linguistics
University of Heidelberg
versley@cl.uni-heidelberg.de

Abstract

Less-configurational languages such as German often show not just morphological variation but also free word order and nonprojectivity. German is not exceptional in this regard, as other morphologically-rich languages such as Czech, Tamil or Greek, offer similar challenges that make context-free constituent parsing less attractive.

Advocates of dependency parsing have long pointed out that the free(r) word order and non-projective phenomena are handled in a more straightforward way by dependency parsing. However, certain other phenomena in language, such as gapping, ellipses or verbless sentences, are difficult to handle in a dependency formalism.

In this paper, we show that parsing of discontinuous constituents can be achieved using easy-first parsing with online reordering, an approach that previously has only been used for dependencies, and that the approach yields very fast parsing with reasonably accurate results that are close to the state of the art, surpassing existing results that use treebank grammars. We also investigate the question whether phenomena where dependency representations may be problematic – in particular, verbless clauses – can be handled by this model.

1 Introduction

Automatic syntactic parsing has been fruitfully incorporated into sytems for information extraction (Miyao et al., 2008), question answering, machine translation (Huang and Chiang, 2007), among others, but we also see syntactic structures being used to communicate facts about language use in the digital humanities or in investigations of the language of language learners. In all of these applications, we see fruitful use both of constituent trees, and of dependency trees.

Depending on the application, different criteria may become important: on one hand, the ability to produce structures that are (intuitively) compatible with semantic composition, or where arguments and adjuncts are related to their predicate in the tree, which commonly requires dealing with nonprojectivity. Such a formalism should also deal with a wide range of constructions including verbless clauses. Finally, parsing speed is somewhat important for many application cases, and a parser that changes the tokenization of the input or inserts additional "null" tokens runs afoul many of the fundamental assumptions in pipelines for semantic processing or information extraction.

If we look at the current three largest treebanks for German, namely the Hamburg Dependency Treebank (Foth et al., 2014) with 101 000 sentences, the TüBa-D/Z treebank (Telljohann et al., 2009) with 85 000 sentences or the Tiger treebank (Brants et al., 2002) with about 50 000 sentences, we see find a continuum of the nonprojective single-parent dependencies of the HDT on one side and projective phrase structures of TüBa-D/Z, with Tiger straddling in the middle with a scheme that is neither projective nor limited to dependencies, and which represents, we'll argue, both the best and the worst of both worlds.

Because of its expressivity, the Negra/Tiger scheme has also been used for other languages such as Swedish Volk and Samuelsson (2004) as well as Georgian/Russian/Ukrainian (Kapanadze, 2012), and as Early New High German (Pauly et al., 2012).

The Tiger scheme is arguably more expressive than either of the alternatives since it can capture both elliptic clauses (which are difficult to represent in normal dependency schemes) and nonprojective constructions (which have to be added as a second annotation layer in purely projective treebanks such as TüBa-D/Z). It also makes it the most difficult to provide good automatic tool support, in terms of effective parsing components or of annotation tools, since parsing of discontinuous constituents has only recently become practical.

The straightforward approach of Kallmeyer and Maier (2013) to use a treebank-derived linear context-free rewriting system suffers from near-exponential observed time consumption in practice. Approaches that use context-free grammar approximation such as the ones of Schmid (2006), Cai et al. (2011) or van Cranenburgh and Bod (2013), still have cubic time complexity; especially in the latter case, it is not clear whether techniques that allow fast PCFG parsing such as those of Bodenstab et al. (2011) would be suitable for the subsequent steps with increased grammar complexity.

In this paper, we present a novel application of the easy-first parsing principle of Goldberg and Elhalad (2010) to discontinuous constituent parsing, which performs fast enough for interactive use (about 40 sentences per second) while giving an acceptable accuracy that is within the range normally seen with unmodified treebank grammars.

In the remainder of the paper, we will include a short discussion of the interrelation between constituency and dependency relations of syntax, as well as relevant prior work in section 2, and discuss the construction of the parser in section 3. Section 4 and following contain a discussion of quantitative results on the Tiger corpus, whereas the penultimate section contains a more detailed analysis of the parser behaviour on constructions that are problematic for either dependency parsers or projective constituent parsing.

2 Constituency and Dependency: Good friends?

Constituency and dependency structures are two formalisms that are frequently used for theory-neutral description of syntactic structures. In constituent structures, usually influenced by some version of X-bar theory (see Kornai and Pullum, 1990 for a discussion; most notably, phrases are supposed to be projections of a head), whereas in dependency structures it is usually assumed that each word has exactly one governor (except one or more words that are attached to a virtual *root* node).

The common subset of both can be described (in the words of Hockenmaier, 2007) as “Heads, arguments, modifiers, conjuncts”, which includes the grammatical function labels that are added in dependency structures, and to varying extent in phrase structure treebanks. Nivre (2011) goes further and asks whether we need constituents at all, since pure dependency parsing recovers arguments and adjuncts while being generally faster (and, at least for results published on Czech and French which Nivre cites, more accurate). Versley and Zinsmeister (2006) similarly argue that even “deep” dependency relations (including nonlocal ones) can be recovered from single-parent dependencies if subsequent disambiguation steps identify the scope of conjunctions, argument sharing in coordination, passive identification, and lexicalized control phenomena. However, verbless clauses as they may occur in coordination pose a problem to the idea that every phrase is headed by a preterminal, or the equivalent assumption in dependency grammar that every argument has a governing head word.

In constituent treebanks, the solution to this problem is rather simple: deviate from the descriptive-Xbar schema outlined earlier on and introduce headless projections for these clauses. Dependency treebanks lack this additional degree of freedom, and the choice is usually to either attach the respective nodes somewhere else (Böhmova et al., 2001; Foth, 2006) or introduce empty nodes that are the governors of the orphaned subtrees (Bosco and Lombardo, 2006; Vincze et al., 2010; Dipper et al., 2013).

In dependency parsing, good solutions for nonprojective edges have been found, including pseudoprojective parsing (Nivre and Nilsson, 2005), approximate weighted constraint solving (Koo et al., 2010), as well as deterministic online reordering (Nivre, 2009), which also has been applied to easy-first decoding strategies (Tratz and Hovy, 2011). Seeker et al. (2012) additionally employs an *attach-inner* operation which allows non-projective insertion into a structure that has already been built. Despite these very reasonable solutions, the treatment of elliptic phrases, whether it is done using the somewhere-else approach or by introducing empty nodes (see Seeker et al., 2012 and references therein) yields uninformative structures for subsequent processing components or even makes it necessary to re-engineer subsequent processing stages for dealing with the newly introduced empty nodes, or (equally impractical) require the refactoring of annotated corpus resources to accommodate a new tokenization whenever a null element is introduced or changed.

In constituency parsing, the problem of discontinuous constituents in parsing has, at least in German, first been met with a proposals of raising degrees of complexity (among others, van Noord, 1991; Plaehn, 2000) and then silently been ignored both in the building of parsers and in their evaluation: researchers from Dubey and Keller (2003) to the present day cite bracketing scores based on structures that would make the reconstruction of "Heads, arguments, modifiers, and conjuncts" – usually – rather difficult.

Only relatively recently has the problem of discontinuous constituent parsing been tackled head-on. Kallmeyer and Maier (2013) propose an approach that extracts a treebank LCFRS grammar, which is then used for probabilistic parsing, albeit with near-exponential time consumption. Maier et al. (2012) present an approach to make parsing in this approach more efficient by flattening coherent structures in a sentence to one single sentence node and thus eliminating scrambling as a source of discontinuities, together with other transformations, which allows a time complexity of $O(n^6)$ and parsing times of about 2 minutes for a 40-word sentence. van Cranenburgh and Bod (2013) use a more practical approach that first creates phrase candidates from the n-best list of a projective constituent parser, and uses these to construct LCFRS items that do not necessarily correspond to grammar rules seen in the training set, but which are then matched against a collection of tree fragments extracted from the training set.

There exists some work on transforming dependency structures into constituents that may help in the recovery of discontinuous constituents: Hall and Nivre (2008) propose to encode information about node labels in the dependency labels, whereas Carreras et al. (2008) show that an ILP-based combination of finding dependencies and adding phrase projections and adjunctions to a dependency backbone works well for constructing structures matching those of the Penn Treebank. Seddah (2010) found that similar spinal structures can be used for the French Treebank.

3 Incremental parsing

In general, statistical parsing follows one of several general approaches: one is the approach of item-based decoding, which is centered around the creation of a parse forest that implicitly stores a very large number of possible trees, followed by either dynamic programming in the case of projective parsing (e.g. (Collins, 2003)) or techniques that provide an approximate or exact solution to the intractable problem in the case of nonprojective parsing with second-order factors (Koo et al., 2010). The second large group of approaches is based on incremental structure building, including the approaches of Magerman (1995) or Sagae and Lavie (2006) in the case of constituent parsing, or of Nivre (2003) and following in the case of dependency parsing, with approaches such as Stolcke (1995) or Huang and Sagae (2010) occupying a middle ground.

While the idea of head lexicalization has played a large role in projective constituent parsing, there are rather few approaches that attempt to bridge the gap between dependency and constituency representations in a way that could be exploited for the efficient building of discontinuous constituent structures.

Among these, both the approaches of Hall and Nivre (2008) and of Carreras et al. (2008) could be described in terms of a **spinal transform**: each terminal in the input string is assigned a set of governing nodes that form its *spine*; parsing then consists of assigning a dependency structure among the terminal nodes and of assigning spines and the relation to each other.

In the remainder of this section, we describe two approaches that we used to perform nonprojective constituent parsing in expected linear time: one is relatively close to the approach of Hall and Nivre (2008), but instead of assigning nodes to the first terminal of their yield, uses a strategy more like the spinal tree adjoining grammr of Carreras et al. (2008). The other is an application of the principle

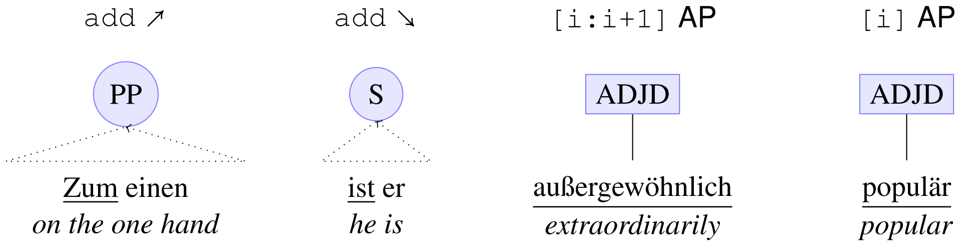

Figure 1: Example for an intermediate state in EaFi, with the preferred action candidates for each position

of easy-first parsing, which has been used for unlabeled dependency parsing by Goldberg and Elhalad (2010), and for non-projective labeled dependency parsing by Tratz and Hovy (2011), towards discontinuous constituency parsing. Because computed feature vectors can be memorized and only have to be recomputed in a small window around the last parser action, this latter approach, just as a left-to-right transition-based parser, has an expected time consumption that is linear in the number of words to be parsed.

3.1 ADG: Constituency-to-Dependency Reduction

Our baseline is an approach close in spirit to Hall and Nivre (2008): The tree with node labels is turned into a dependency graph that encodes, on the governor edge of each terminal, a combination of (i) the node labels on the spine of this node, and (ii) the level at which this node attaches to its parent's spine.

We change two parameters of Hall and Nivre's approach: on one hand, we do not use the first terminal in the yield of a node as its representative but the head according to the head table that we also use to assign the head in the easy-first parser. The reason for this is a practical one: using the head, we get a distribution of 531 different spine/level combinations when we use the head, whereas we would get about 1525 categories when we use the first terminal.

To ensure efficient parsing, this list is further pared down to 100 entries, with the remaining entries being replaced by an UNK placeholder. In decoding, terminals with these entries are assigned the most frequent combination of spine and parent category for the POS tags of the node and its governor, and the topmost spine node with a matching category (or simply the topmost one) would be chosen.

The decoding algorithm and parameter settings for MaltParser were then determined using the MaltOptimizer software (Ballesteros and Nivre, 2012). The settings selected use the stack-projective algorithm with head+path marking strategy for pseudoprojective parsing.[1]

Hall and Nivre's approach is more complex than the approach presented here, and involves interleaving of identifying dependency edges (using the nonprojective Covington parsing scheme) and the stepwise determination of the topmost edge label, then the path of edge labels, and finally the path of constituent labels and its attachment. However, we find that this approach of dependency reduction constitutes a very reasonable intelligent baseline, and is able to perform at a similar speed than our approach.

3.2 EaFi: Easy-first Constituency Parsing

The main approach that we will present here constitutes an adaptation of the *Easy-First* approach to nonprojective constituent parsing. The parser keeps track of a sequence of nodes, beginning with the terminals that are output by the preprocessing consisting of morphological analyzer and lemmatization, and at each point applies one of several actions:

- **Reduce-Unary**: one node is grouped under a unary node of a given category, with the restriction that the corresponding unary rule must have been observed in the treebank. (Additionally, we collapse any two nodes with the same category embedding each other, which sometimes occurs in the Tiger treebank when several empty-headed phrases are assumed to embed each other in coordination).

[1] MaltParser is able to do direct nonprojective parsing using the reordering approaches of Nivre (2009) and Nivre et al. (2009), however the pseudoprojective approach was selected in MaltOptimizer's parameter selection.

<u>*Basic*</u> <u>featureset</u>		
Unigram:	$n \in n_{i-2} \ldots n_{i+3}$	$\mathrm{C}n\mathrm{P}n$ $\mathrm{C}n\mathrm{W}n$ $\mathrm{C}n\mathrm{M}n$ $\mathrm{C}n\mathrm{L}n$
Left/Right Children:	$n \in n_{i,L} n_{i,R} n_{i+1,L} n_{i+1,R}$	$\mathrm{C}n\mathrm{P}n$
Bigram:	$m, n \in n_{i-1} n_i \ldots n_{i+2} n_{i+3}$	$\mathrm{W}m\mathrm{W}n$ $\mathrm{W}m\mathrm{C}n$ $\mathrm{C}m\mathrm{W}n$ $\mathrm{W}m\mathrm{W}n$
Trigram:	$r, m, n \in n_{i-1} n_i n_{i+1} \ldots n_{i+1} n_{i+2} n_{i+3}$	
<u>*Medium*</u> <u>featureset</u>		
Bigram+Child:	$m, n, r \in \{n_i n_{i+1} n_{i+1,L}; n_i n_{i+1} n_{i,R};$	
	$n_{i+1} n_{i+2} n_{i+1,R}; n_i n_{i-1} n_{i,L}\}$	$\mathrm{C}m\mathrm{C}n\mathrm{C}r$ $\mathrm{C}m\mathrm{C}n\mathrm{P}r$
Distance:	$\Delta \in \{\mathrm{dist}(n_i, n_{i+1}), \mathrm{gap}(n_i, n_{i+1})\}$	$\Delta\mathrm{C}m$ $\Delta\mathrm{C}n$ $\Delta\mathrm{P}m$ $\Delta\mathrm{P}n$
	$m, n = n_i, n_{i+1}$	$\Delta\mathrm{F}m$ $\Delta\mathrm{F}n$
<u>*Large*</u> <u>featureset</u>		
Gap bigram:	$m, n \in n_{i-1} n_{i+2}, n_i n_{i+2}$	$\mathrm{W}m\mathrm{W}n$ $\mathrm{W}m\mathrm{C}n$ $\mathrm{C}m\mathrm{W}n$ $\mathrm{W}m\mathrm{W}n$
Bigram+2child:	$m, n \in n_{i-1} n_i \ldots n_{i+2} n_{i+3}; C, D \in \mathrm{L}, \mathrm{R}$	$\mathrm{C}m\mathrm{C}n\mathrm{C}m_C\mathrm{C}n_D$ $\mathrm{C}m\mathrm{C}n\mathrm{C}m_C\mathrm{P}n_D$
		$\mathrm{C}m\mathrm{C}n\mathrm{P}m_C\mathrm{C}n_D$
Bigram-Dist	$m, n \in n_{i-1} n_i \ldots n_{i+2} n_{i+3}$	Δ
	$\Delta = \mathrm{dist}(m, n)$	$\Delta\mathrm{P}m\mathrm{P}n$

Table 1: Features used: W=word, C=phrase label, M=head morph, P=head pos

- **Reduce-Binary**: two nodes are grouped under one node, with the head of the new node being determined by a head table.
- **Add-Left/Add-Right**: one node is added as a child to a node on its left/to its right.
- **Swap**: if two nodes are in surface order (i.e., the head of the first node being left of the head of the second node in the normal word ordering), they can be swapped.

In addition, we experimented with a **retag** action which allows the parser to change the tag of a word that has been mistagged. While this has a positive effect on the parser's accuracy for verb phrases, it also results in a slight deterioration of other phrases, resulting in a very slight decline in performance.

To decide among different parsing actions, the parser uses a linear classifier with a pre-defined feature set (POS, word form, morphological tag and lemma in a two-token window around the two nodes that are being considered, the category and part-of-speech tag of the leftmost and rightmost dependent of the nodes that are being considered; bigrams of words, categories, and one of each, in a window of one around the two nodes being considered, and trigrams consisting of two category and one category, part-of-speech tag, or word form within said window).

Weights are learned by performing online learning with early stopping, similar to the strategy employed by Collins and Roark (2004). We use the Adaptive Gradients method (Duchi et al., 2011) for weight updates and averaging of the weight vector (in a fashion identical to the averaged perceptron). We found that 5-10 epochs of training on Tiger were sufficient to get a mostly usable model, and used 15 epochs of training for the results reported in the later section. Considering that Goldberg and Elhalad (2010) use a learning strategy that performs multiple perceptron updates until the constraint violation is fixed, we also tried this strategy but did not achieve convergence.

3.3 Reordering Oracles for Constituents

The basic idea for reordering oracles in deterministic dependency parsing has been presented by Nivre (2009). In the following, we present a straightforward adapation of the idea to constituent trees.

Given a set of terminals $T = \{w_1, \ldots, w_n\}$ that is totally ordered by a relation <, an unordered tree graph is a directed graph $(NT \cup T, \lhd)$ with nonterminal (NT) and terminal nodes (T), where the transitive hull $\lhd^*$ of the parent relation $\lhd$ is acyclic, no node has a parent from T, and exactly one node, v_{root}, has no parent.

An **node ordering** $\prec$ is *consistent* with $\lhd$ whenever, for any node u and an descendant $u' \rhd^* u$, and a node v with an descendant $v' \rhd^* v$, $u \prec v$ entails $u' \prec v'$.

A **tree cut** of a tree is a sequence $v_1, \ldots, v_n$ that contains exactly one node from each path $v_{\text{root}}, \ldots w_i$ from the root to a terminal. Nivre's insight, applied to constituent structures, is that sorting the terminals in a $\triangleleft$-compatible order $\prec$ will allow us to use normal projective parsing techniques to find a sequence of reductions that parses this tree, since any needed reduction would reduce one $\prec$-ordered cut to another $\prec$-ordered cut. In the following, two orderings $\prec$, $\prec'$ are considered equivalent iff they only differ on pairs of nodes u, v where one is the ancestor of the other.

Two subtrees under nodes u and v with yields yield$(u) = \{u' \in T | u' \triangleleft^* u\}$ and yield$(v) = \{v' \in T | v' \triangleleft^* v\}$ are *separated* by a surface ordering $<$ whenever any two terminals u' of u and v' of v fulfill $u' < v'$. Note that two nodes without gaps (i.e. block-degree one) either embed each other (in which case $u \triangleleft^* v$ or $v \triangleleft^* u$ is the case) or they are separated by $<$. In a slight abuse of notation, we extend $<$ from a total order of the terminals to a partial order of the nonterminals by writing $u < v$ whenever u and v are separated. For projective trees, this extension of $<$ specifies exactly one total relation (modulo equivalence), and which is also $\triangleleft$-compatible.

For trees that are non-projective, we can have the situation where two nodes u and v are *overlapping* in that u has descendants u', u'' and v has a descendant v' with $u' < v' < u''$. Then we cannot extend $<$ to an ordering of nodes that is $\triangleleft$-compatible. However, we can always find an ordering that respects $<$ locally such that, for two children u' and u'' of u, $u' < u''$ entails $u' \prec u''$. Nivre proposes the sequence assigned by an in-order traversal of the dependency tree. In our case, any function $h : NT \rightarrow (NT \cup T)$ that assigns a "head" child to each node will do the same, with an extension $h^*(w) = w$ for all terminals and $h^*(v) = h^*(h(v))$ otherwise, through $u \prec v :\Rightarrow h^*(u) \prec h^*(v)$.[2]

A transition sequence for parsing a tree is then a sequence consisting of **reductions** (leading from a cut $\ldots v_i, u', \ldots u'', v_j, \ldots$ with a contiguous subsequence of the children of u to the sequence $\ldots v_i, u, v_j, \ldots$ that contains u instead) and **swaps** (leading from a cut $\ldots v_i, v_j \ldots$ that has $h^*(v_i)$ and $h^*(v_j)$ ordered with respect to $<$ but not with respect to $\prec$ to a cut $\ldots v_j, v_i \ldots$ that is orders $h^*(v_i)$ and $h^*(v_j)$ with respect to $\prec$ but not $<$).

Nivre (2009) defines an oracle for shift-reduce parsing that is **swap-eager** in that it always allows swapping. In Nivre's case, the oracle is deterministic and always performs the swapping before any reduction.

Nivre et al. (2009) note that the swap-eager oracle performs too many swaps because it swaps groups of words that are later reduced. They propose a **swap-lazy** algorithm that does not swap two nodes if one of them is adjacent to another node that is within the same maximal projective subtree.

The perspective of parsing as a series of swap and reduce actions allows us to specify a strategy that performs less reductions in some cases: Consider that we need to reorder the $<$-contiguous sequence of terminals to the $\prec$-contiguous sequence that is needed for reducing the tree to its final form. The number of swaps performed, if we assume that we always swap adjacent constituents, is exactly equal to the number of terminal pairs v_i, v_j that are $<$-ordered but not $\prec$-ordered. Any reduction of the number of swaps relative to this baseline will come from a group of nodes with heads $v_{i1}, \ldots v_{ik}$ that are reduced to their parent v_i before being swapped with a node v_j.

We can take advantage of this fact by using any node with blockdegree one as a **barrier**: no node that is a descendant of this node can be swapped with a node that is not a descendant before the reduction that results in the barrier node has been carried out. Because any projective subtree has all nodes as barrier nodes, any pair of nodes whose swapping is delayed by the swap-lazy approach will be kept from swapping by a barrier. Conversely, nodes with a block-degree of one can also occur higher-up in the tree (e.g. as clause or sentence nodes), in which case they can act as a barrier even when their subtrees are not projective.

4 Quantitative Evaluation

In order to evaluate our approach, we used the Tiger treebank, with the split used in the SPMRL'2013 shared task (about 40 000 training sentences and 5 000 development and test sentences each; see also

[2] Note that the concrete choice of h is quite arbitrary: we could take the actual head child, but also the first or last child of a node.

	$\ell \leq 30$	$\ell \leq 40$					
	F_1	F_1	LA	EX	NP	PP	VP
EaFi: Preprocessing (large, barrier, noretag)							
gold	77.95	76.64	92.17	41.71	75.0	82.8	56.6
marmot	75.51	73.97	91.08	38.48	72.7	81.3	48.3
pred	74.71	73.18	90.81	37.67	72.1	80.6	48.9
ADG, marmot preprocessing							
marmot	73.42	72.24	90.95	33.77	68.0	77.4	52.1
EaFi: Train projective, evaluate on real data							
gold	76.86	75.50	92.13	38.38	74.4	81.7	48.2
marmot	74.43	72.98	91.20	36.52	72.1	79.8	42.6
pred	73.75	72.32	90.72	35.55	71.8	79.2	42.4
EaFi: Train projective, evaluate on projective							
gold	79.95	78.59	93.40	44.20	76.1	83.0	68.7
marmot	77.00	75.64	92.38	40.79	73.8	81.1	59.1
pred	76.25	74.94	91.87	39.60	73.5	80.5	58.4

Table 2: Results on SPMRL'13-dev (German, Tiger treebank) with varying preprocessing

	$\ell \leq 30$	$\ell \leq 40$					
	F_1	F_1	LA	EX	NP	PP	VP
EaFi: Feature set (barrier, noretag)							
basic	70.26	68.60	89.03	34.03	69.1	77.1	40.1
medium	73.31	71.75	90.13	36.22	70.5	79.9	45.8
large	74.71	73.18	90.81	37.67	72.1	80.6	48.9
EaFi: Reordering (large, noretag)							
eager	73.33	71.66	90.33	37.43	71.7	80.6	47.8
lazy	74.85	73.37	90.85	38.08	72.2	80.8	49.0
barrier	74.71	73.18	90.81	37.67	72.1	80.6	48.9
EaFi: Tag correction (large, barrier)							
noretag	74.71	73.18	90.81	37.67	72.1	80.6	48.9
retag	74.62	73.16	90.83	37.51	71.5	80.3	49.4

Table 3: Results on SPMRL'13-dev (German, Tiger treebank) with *pred* preprocessing

Seddah et al., 2013 for a more extensive description), with the state-of-the-art preprocessing results for part-of-speech and morphological tags[3] which were produced by Björkelund et al. (2013) using the MarMoT tagger (Müller et al., 2013), in addition to the gold-standard preprocessing (*gold*) and automatic predictions (*pred*) that are part of the official dataset of the SPMRL shared task.

We applied two transformations to the data, which are automatically reversed in the parser output: one is adding NPs into PPs, which is also done by Seeker et al. (2012), and the other is that we make parenthetical material subordinate to its embedding clause, as Maier et al. (2012) also advocate.

Evaluation was performed using the evaluator from the DiscoDop package of van Cranenburgh and Bod (2013), excluding punctuation and the ROOT label added by disco-dop from the evaluation. Training was run for 15 epochs. Parsing the 5000 development sentences took about 90-120 seconds for EaFi, which corresponds to 40-55 sentences per second (on a Core i7 2GHz) and is slightly faster than MaltParser using the ADG-derived model and a LibLinear classifier.

In the results in table 2, we see the results for the dependency-to-constiuents approach, as well as for the easy-first parsing with different reordering heuristics. As in Nivre et al. (2009), we notice that the *lazy* strategy that keeps projective constituents together yields better results than the *eager* strategy which allows moving right away. The overall results – around 76.6% f-score on gold tags and 73.1% f-score on predicted tags in sentences of 40 words and below – indicate the promise of this approach, even though they are significantly below the results of van Cranenburgh and Bod (2013) who achieve more than 78% f-measure using predicted tags on a different split of the Tiger treebank. Van Cranenburgh's approach is about 15-20 times slower than ours, using 10 seconds for a 40-word sentence.

For informative purposes, we also included results for projective parsing in table 2, using a conversion that first attaches punctuation and then projectivizes the tree by detaching non-head children.[4] Comparing the nonprojective parser and a variant that was trained on the projectivized version of the dataset, we see that the projective parser is about 1-2 percent worse than the nonprojective one, corresponding to our intuition that the reordering part improves the parsing on average. We also see that the projective evaluation yields an estimate of parser performance that is substantially more optimistic than evaluating on the original treebank.

4.1 Comparison with Related work

Tables 4 and 5 show previous results for discontinous constituent parsing on the Tiger and Negra treebanks. The current best results on the Tiger treebank have been achieved by van Cranenburgh and Bod (2013), whose approach yields 78.8% Parseval F_1 measure on the Tiger treebank in the split by Hall and Nivre (2008), and 76.8% on the Negra treebank, in both cases with above 40% of exact matches among the sentences of up to 40 words. Kallmeyer and Maier (2013) only report results on shorter sentences in Negra for their approach using a modified treebank LCFRS. They achieve 75.6% on sentences of up to 30 words.

A recent approach that attempts to speed up discontinuous constituent parsing is the one by Angelov and Ljunglöf (2014), whose parser takes about 100 seconds for a length-40 sentence, which can be reduced to 10 seconds for a length-40 sentence with an approximate search strategy. For sentences between 5 and 60 tokens, their approach reaches an F_1 score of 69.3%, which however deteriorates quickly when approximate search is used, to 61.9% F_1 in the latter case.

It is quite evident that pushing for more speed in these formalisms forcibly leads to a deterioration in the quality of the results. As such, we think that the speed/quality tradeoff achieved in our system is quite useful.

5 Qualitative Analysis

In the following, we will provide a categorization of the phenomena concerning verbless clauses on one hand, and discontinuous constituents on the other. Table 6 contains a breakdown on these types of

[3] Data from `http://www.cis.lmu.de/~muellets/marmot/marmot_spmrl.tar.bz2`, version with file dates of June 13th 2014. See `http://code.google.com/p/cistern/wiki/marmotSPMRL`

[4] The SPMRL shared task dataset is idiosyncratic in that it deprojectivizes before attaching punctuation, which leads to a result that is rather dissimilar to the original treebank.

	$\ell \leq 30$		$\ell \leq 40$	
	F_1	EX	F_1	EX
Hall and Nivre (2008), gold^a	—	—	79.93	37.78
Hall and Nivre (2008), pred^a	—	—	75.33	32.63
van Cranenburgh and Bod (2013), pred^a	—	—	78.8	40.8
This work, gold^a	76.47	40.61	74.23	37.32
Maier (2010), LCFRS gold^c	73.43	29.87	—	—
Maier (2010), CFG gold^c	75.57	31.80	—	—
This work, gold^b	77.95	43.81	76.64	41.71
This work, gold, eval w/ ROOT^{bc}	81.13	43.81	79.80	41.71

^{a}) Hall&Nivre split ^{b}) SPMRL split ^{c}) includes the ROOT node in the evaluation

Table 4: Previous results on the Tiger treebank

	$\ell \leq 30$		$\ell \leq 40$	
	F_1	EX	F_1	EX
Maier (2010), LCFRS gold^c	71.52	31.65	—	—
Maier (2010), CFG gold^c	74.04	33.43	—	—
van Craenburgh (2012), LCFRS, gold	—	—	67.26	27.90
van Craenburgh (2012), Disco-DOP, gold	—	—	72.33	33.16
Maier et al. (2012)	74.5	—	—	—
Kallmaier and Maier (2013), LCFRS, gold	75.75	—	—	—
van Cranenburgh and Bod (2013), gold	—	—	76.8	40.5

^{c}) includes the ROOT node in the evaluation

Table 5: Previous results on the NeGra treebank

phenomena according to whether they are:

- **correctly parsed** (+): when the incredients for the construction are present in the parse and they are combined in a suitable fashion.
- **missed** (o): when the ingredients for the construction are present, but combined in another way – for example, an extraposition where the extraposed item is misattached
- **broken** (-): when the ingredients for the construction are not present and the parse has a completely different structure.

Many of the same categories are discussed by Seeker and Kuhn (2012), who only discuss examples, and by Maier et al. (2014), who published a list of sentence numbers for each phenomenon that is, however, disjoint with the development portion considered here. Although the distinctions between "missed" and "broken" analyses are somewhat subjective, we think that it is still informative in the sense that it helps to compare the relative difficulty of the problems involved.

5.1 Types of Verbless Clauses

In their conversion Seeker and Kuhn (2012) found 3 035 sentences that contain at least one empty node in the Tiger treebank, or about one every 16 sentences. While this phenomenon may be more frequent in spontaneously-produced text such as it may occur in user-generated content, it is still quite frequent.

Seeker et al. only distinguish among edge labels, followed by a guess on the clause type that they need in order to place the inserted null element.

In this work, we will concentrate on verbless VP and S nodes, with rougly three categories:

The first consists of **verbless copula clauses** that mostly occur at top level,[5] and where the most obvious way to build a complete clause would be to add a *be* copula to the clause.

[5] sentences 40499, 41442, 41468, 41676, 41682, 41736, 41743, 42566, 42606, 42738

	ADG/marm			large/pred		
	+	o	-	+	o	-
copula clauses	3	4	3	2	5	3
gapping/ellipsis	0	4	6	0	4	6
parentheticals	1	6	4	0	4	6
extraposition	1	7	2	0	7	3
scrambling	3	2	4	3	1	5
topicalization	3	6	1	4	4	2

+) construction parsed ok, ○) construction missed, −) broken parse

Table 6: Qualitative analysis: Counts for *ok/missed/broken* examples

A second group consists of clauses with **gapping/ellipsis** which occur in a coordinated structure, but do not have a verb of their own.[6] Such cases can occur with a final constituent in clause coordination as well as with a non-final constituent in verb-last clauses:

(1) a. Die Anstalt soll [Anfang 1998 noch 1200 Beschäftigte] und [ein Jahr später 600
the institution shall start 1998 still 1200 employees and one year later 600
zählen].
count.
"*The institution will count 1200 employees at the start of 1998 and one year later, 600*".

b. [Die Zahl der Urlaubsreisen im Inland fiel laut Schörcher um zwei Prozent]
the number of holiday trips in interior fell according to Schörcher by two percent
und [damit nicht mehr so stark wie im Vorjahreszeitraum] .
and hence not anymore as strong as in the previous year period
"*The national number of holiday trips fell by two percent according to Schorcher, and hence not as strongly anymore as in the corresponding period from last year*".

Finally, we have **parentheticals**, which are rather rather similar to the examples listed under *verbless copula clauses*, except that they occur as parenthetical material in a larger clause rather than by themselves.[7]

5.2 Types of Non-projectivity Phenomena

For the purpose of this paper, we will make a three-way distinction in the phenomena that create discontinuities, according to the following questions:

- If we serialize the sorted (sub)tree, would the result yield a grammatical sequence? Or, to ask a related question, would anything be missing if we kept only the continuous block of the head?
- If we flatten the tree by introducing a common *ordering domain* for multiple heads (which would be the result of tree flattening as proposed by Uszkoreit, 1987 or of a common argument list as advocated by Hinrichs and Nakazawa, 1989; flattening the sentence is also the solution used in the German LFG grammar of Forst, 2007), would we have gotten rid of the problem?

Making these distinctions gives us three rather large categories that we can use to classify nonprojectivity phenomena:

Extraposition[8] is phenomenon where the sorted subtree would (usually) be grammatical, and where the continuous part only would (usually) be acceptable:

[6] sentences 40698, 40788, 40836, 41003, 41174, 41218, 41356, 41399, 41544, 41665
[7] sentences 40698, 40749, 40861, 40894, 40899, 40924, 41219, 41267, 41437, 41443
[8] Sentences 40506, 40507, 40517, 40528, 40567, 40583, 40589, 40594, 40622, 40672

(2) a. Ele hat mir [ein Buch] geschenkt [über die Savanne].
Ele has me a book given about the savannah.
"*Ele gave me a book about the savannah*".

b. Ich habe [Ele] ein Buch geschenkt [und Susi].
I have Ele a book given and Susi.
"*I gave a book to Ele and Susi*".

c. Heute ist [der Staubsauger] gekommen, [den Du bestellt hast].
Today is the vacuum cleaner come, which you ordered have.
"*Today, the vacuum cleaner that you ordered came.*"

Note that extraposition can be arbitrarily deep, as NPs can embed each other recursively:

(3) a. Heute ist (die Rechnung für [den Staubsauger]) gekommen, ([den Du bestellt hast]).
Today is the invoice for the vacuum cleaner$_j$ come, which$_j$ you ordered have.
"*Today, the invoice for the vacuum cleaner you ordered came*"

b. Ich habe (die Rechnung für [den Staubsauger]) gefunden, (die Du vermißt hast).
I have the invoice$_i$ for the vacuum cleaner found, which$_i$ you missed have.
"*I found the invoice for the vacuum cleaner which you were missing*"

Scrambling[9] is the effect that occurs when two verbs that both have arguments are in the same clause, and share the ordering domain in that clause, have crossing argument dependencies:

(4) …daß (dem Kunden) [den Kühlschrank] bisher noch niemand [zu reparieren] zu versuchen
…that the customer the fridge until now yet nobody to repair to try
(versprochen) hat.
promised has.
"*that no one has promised the customer to try repairing the fridge.*"

The example above is due to Becker et al. (1992), who claim that there is no bound on the distance over which each element can scramble, nor a bound on the number of unbounded dependencies that can occur in one sentence. Becker et al. further claim that no LCFRS can faithfully represent a sequence of m verbs which are each preceded by one argument, where the arguments can be permuted freely.[10]

Finally, **Topicalization**[11] or more generally V2-order phenomena are those where a part of the verb clause (either an argument, or an argument of a phrase within the verb clause, or part of the verb clause itself) is moved into clause-initial position.

(5) a. Ein Buch über die Savanne hat Ele mir geschenkt.
a book about the savannah has Ele me given.
"*Ele gave me a book about the savannah.*"

b. Über die Savanne hat mir Ele ein Buch geschenkt.
about the savannah has me Ele a book given.
"*Ele gave me a book about the savannah.*"

c. Ein Buch geschenkt hat Ele mir.
a book given has Ele me.
"*Give me a book, Ele did.*"

[9] Sentences 40524, 40567, 40572, 40588, 40594, 40595, 40601, 40885, 40966, 41025

[10] The practical consequence is that any LCFRS extracted from a treebank will either underspecify the dependencies in such a construction – this is the flattening solution – or yield rules with growing block-degree. van Cranenburgh (2012) shows that the sentences with up to 25 words in Negra can be parsed with an LCFRS that leads to $O(n^9)$ time complexity when a suitable binarization is used, where the original treebank grammar would mean a parsing complexity of $O(n^{19})$. Maier et al. (2012) point out that the observed time complexity of the arbitrary-block-degree parser used by Maier (2010) and Kallmeyer and Maier (2013) is due to necessary bookkeeping, and that their variant with fixed block degree yields a polynomial time complexity.

[11] Sentences 40513, 40521, 40528, 40544, 40546, 40548, 40551, 40572, 40580, 40585

5.3 Analysis of Parser behaviour

Using the movement actions, the parser is able to correctly attach topicalized nodes in simple sentences, and to sort out in most cases which nodes belong to the VP and which ones to the S node. In the presence of complex sentence structure, the very local view on the sentence that the parser has quickly becomes a hindrance. Extraposed material is attached correctly in the case of relative clauses, whereas infinitival constructions (which can plausibly attach to the verb) are often missed, and clauses that are extraposed modifiers of adverbs or adjectives are mostly missed. As with early treebank-based parsers, the presence of multiple verbs (as in coherent constructions) can mislead the parser into assuming a more complex structure than is actually present.

In general, verbless copula clauses, asyndetic coordination, and gapping/ellipsis, which are difficult for dependency parsing, are also especially prone to confuse the very local view of the easy-first parser, which is a rather anticlimactic, yet commonsensical conclusion.

In summary, simple material is often handled surprisingly well, whereas sentences with a complex topological structure – i.e., coordination, clauses embedded in a nominal phrase, or correlations, are rather challenging for easy-first parsing. Parsing algorithms with more context such as Sartorio et al. (2013) or an application of beam search might help in some of these cases.

6 Summary and Future Work

In this article, we presented a deterministic parser that uses an easy-first strategy to perform non-projective constituent parsing in expected linear time, with results that perform in a similar range as results for discontinuous treebank grammars, and provides a means to provide rather fast parsing in cases where discontinuous structure is required. We introduced the *barrier* formulation as an alternative to the lazy reordering of Nivre et al. (2009), which shows similar performance but which may reveal a closer connection to formalisms with restricted discontinuities.

While all experiments and the phenomen-oriented analysis have been performed on German data, the reordering oracle approach does not make any language-specific assumptions and constitutes a general technique for deterministic parsing of discontinuous constituent trees.

Acknowledgements The author would like to thank the three anonymous reviewers for their valuable comments, and Thomas Müller for providing the Marmot-tagged version of the SPMRL dataset.

References

Angelov, Krasimir and Peter Ljunglöf. 2014. Fast statistical parsing with multiple context-free grammars. In *Proceedings of EACL 2014*.

Ballesteros, Miguel and Joakim Nivre. 2012. MaltOptimizer: A system for MaltParser optimization. In *Proceedings of the Eigth International Conference on Language Resources and Evaluation (LREC 2012)*.

Becker, Tilman, Owen Rambow, and Michael Niv. 1992. The derivational generative power, or, scrambling is beyond LCFRS. Technical report, University of Pennsylvania. A version of this paper was presented at MOL3, Austin, Texas, November 1992.

Björkelund, Anders, Özlem Çetinoglu, Richard Farkas, Thomas Müller, and Wolfgang Seeker. 2013. (re)ranking meets morphosyntax: State-of-the-art results from the SPMRL 2013 shared task. In *Proc. SPMRL 2013*.

Bodenstab, Nathan, Aaron Dunlop, Keith Hall, and Brian Roark. 2011. Adaptive beam-width prediction for efficient CYK parsing. In *Proceedings of ACL/HLT 2011*.

Böhmova, A., Jan Hajič, Eva Hajičová, and B. Hladká. 2001. The Prague dependency treebank: Three-level annotaion scenario. In *Treebanks: Building and using syntactically annotated corpora*, Kluwer Academic Publishers, pages 103–127.

Bosco, Cristina and Vincenzo Lombardo. 2006. Comparing linguistic information in treebank annotations. In *LREC 2006*.

Brants, Sabine, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2002. The TIGER treebank. In *Proc. TLT 2002*.

Cai, Shu, David Chiang, and Yoav Goldberg. 2011. Language-independent parsing with empty elements. In *Proceedings of ACL 2011*.

Carreras, Xavier, Michael Collins, and Terry Koo. 2008. TAG, dynamic programming, and the perceptron for efficient, feature-rich parsing. In *Proceedings of CoNLL*.

Collins, Michael. 2003. Head-Driven statistical models for Natural Language parsing. *Computational Linguistics* 29(4):589–637.

Collins, Michael and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *ACL-04*.

Dipper, Stefanie, Anke Lüdeling, and Marc Resnicek. 2013. NoSta-D: A corpus of german non-standard varieties. In Marcos Zampieri and Sascha Diwersy, editors, *Non-standard DataSources in Corpus-based Research*, Shaker Verlag.

Dubey, Amit and Frank Keller. 2003. Probabilistic parsing for German using sister-head dependencies. In *ACL'2003*.

Duchi, John, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12:2121–2159.

Forst, Martin. 2007. Filling statistics with linguistics - property design for the disambiguation of German LFG parses. In *ACL 2007 workshop on deep linguistic processing*.

Foth, Kilian. 2006. Eine umfassende Dependenzgrammatik des Deutschen. Technical report, Fachbereich Informatik, Universität Hamburg.

Foth, Kilian, Arne Köhn, Niels Beuck, and Wolfgang Menzel. 2014. Because size does matter: The Hamburg dependency treebank. In *Proceedings of the Language Resources and Evaluation Conference (LREC 2014)*.

Goldberg, Yoav and Michael Elhalad. 2010. An efficient algorithm for easy-first non-directional dependency parsing. In *Proceedings of NAACL-2010*.

Hall, Johann and Joakim Nivre. 2008. Parsing discontinuous phrase structure with grammatical functions. In *Proceedings of the 6th International Conference on Natural Language Processing (GoTAL 2008)*.

Hinrichs, Erhard and Tsuneko Nakazawa. 1989. Flipped out: AUX in German. In *Papers from the 25th Annual Regional Meeting of the Chicago Linguistic Society*.

Hockenmaier, Julia. 2007. CCG grammar extraction from treebanks: translation algorithms and applications. Presentation from the Treebank Workshop, 2007, Rochester NY.

Huang, Liang and David Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In *Proceedings of ACL 2007*.

Huang, Liang and Kenji Sagae. 2010. Dynamic programming for linear-time incremental parsing. In *Proceedings of ACL 2010*.

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

Kapanadze, Oleg. 2012. Building parallel treebanks for the lesser-resourced languages. Technical report, Tbilisi State University.

Koo, Terry, Alexander M. Rush, Michael Collins, Tommi Jaakkola, and David Sontag. 2010. Dual decomposition for parsing with non-projective head automata. In *Proceedings of EMNLP 2010*.

Kornai, Andras and Geoff Pullum. 1990. The X-bar theory of phrase structure. *Language* 66:24–50.

Magerman, David M. 1995. Statistical decision-tree models for parsing. In *ACL'1995*.

Maier, Wolfgang. 2010. Direct parsing of discontinuous constituents in German. In *Proceedings of the NAACL-HLT First Workshop on Statistical Parsing of Morphologically Rich Languages*.

Maier, Wolfgang, Miriam Kaeshammer, Peter Baumann, and Sandra Kübler. 2014. Discosuite – a parser test suite for German discontinuous structures. In *Proceedings of LREC 2014*.

Maier, Wolfgang, Miriam Kaeshammer, and Laura Kallmeyer. 2012. PLCFRS parsing revisited: Restricting the fan-out to two. In *Proceedings of the 11th International Workshop on Tree Adjoining Grammar and Related Formalisms (TAG+11)*.

Miyao, Yusuke, Rune Sætre, Kenji Sagae, Takuya Matsuzaki, and Jun'ichi Tsujii. 2008. Task-oriented evaluation of syntactic parsers and their representations. In *ACL 2008*.

Müller, Thomas, Helmut Schmid, and Hinrich Schütze. 2013. Efficient higher-order CRFs for morphological tagging. In *Proceedings fo EMNLP 2013*.

Nivre, Joakim. 2003. An efficient algorithm for projective dependency parsing. In *8th International Workshop on Parsing Technologies*.

Nivre, Joakim. 2009. Non-projective dependency parsing in expected linear time. In *Proc. Joint ACL-AFNLP 2009*.

Nivre, Joakim. 2011. Bare-bones dependency parsing - a case for occam's razor? In *Nodalida 2011*.

Nivre, Joakim, Marco Kuhlmann, and Johan Hall. 2009. An improved oracle for dependency parsing with online reordering. In *Proceedings of the 11th International Conference on Parsing Technologies (IWPT)*.

Nivre, Joakim and Jens Nilsson. 2005. Pseudo-projective dependency parsing. In *Proceedings of ACL 2005*.

Pauly, Dennis, Ulyana Senyuk, and Ulrike Demske. 2012. Strukturelle Mehrdeutigkeit in frühneuhochdeutschen Texten. *Journal for Language Technology and Computational Linguistics* 27(2):65–82.

Plaehn, Oliver. 2000. Computing the most probable parse for a discontinuous phrase structure grammar. In *Proceedings of the 6th International Workshop on Parsing Technologies*.

Sagae, Kenji and Alon Lavie. 2006. A best-first probabilistic shift-reduce parser. In *Proceedings of the Human Language Technology Conference of the NAACL (NAACL/HLT 2006)*.

Sartorio, Francesco, Giorgio Satta, and Joakim Nivre. 2013. A transition-based dependency parser using a dynamic parsing strategy. In *Proceedings of ACL 2013*.

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

Seddah, Djamé. 2010. Exploring the spinal-tig model for parsing French. In *Proceedings of LREC 2010*.

Seddah, Djamé, Reut Tsarfaty, Sandra Kübler, Marie Candito, Jinho D. Choi, Richárd Farkas, Jennifer Foster, Iakes Goenaga, Koldo Gojenola Galletebeitia, Yoav Goldberg, Spence Green, Nizar Habash, Marco Kuhlmann, Wolfgang Maier, Joakim Nivre, Adam Przepiórkowski, Ryan Roth, Wolfgang Seeker, Yannick Versley, Veronika Vincze, Marcin Woliński, Alina Wróblewska, and Eric Villemonte de la Clergerie. 2013. Overview of the SPMRL 2013 Shared Task: A Cross-Framework Evaluation of Parsing Morphologically Rich Languages. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*. Seattle, WA, pages 146–182.

Seeker, Wolfgang, Richárd Farkas, Bernd Bohnet, Helmut Schmid, and Jonas Kuhn. 2012. Data-driven dependency parsing with empty heads. In *Proceedings of Coling 2012*.

Seeker, Wolfgang and Jonas Kuhn. 2012. Making ellipses explicit in dependency conversion for a german treebank. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*.

Stolcke, Andreas. 1995. An efficient probabilistic context-free parsing algorithm that computes prefix probabilities. *Computational Linguistics* 21(2):165–201.

Telljohann, Heike, Erhard W. Hinrichs, Sandra Kübler, Heike Zinsmeister, and Kathrin Beck. 2009. Stylebook for the Tübingen Treebank of Written German (TüBa-D/Z). Technical report, Seminar für Sprachwissenschaft, Universität Tübingen.

Tratz, Stephen and Eduard Hovy. 2011. A fast, accurate, non-projective, semantically-enriched parser. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP 2011)*.

Uszkoreit, Hans. 1987. *Word order and constituent structure in German*. Number 8 in CSLI Lecture Notes. Center for the Study of Language and Information.

van Cranenburgh, Andreas. 2012. Efficient parsing with linear context-free rewriting systems. In *EACL 2012*.

van Cranenburgh, Andreas and Rens Bod. 2013. Discontinuous parsing with an efficient and accurate DOP model. In *Proceedings of the International Conference on Parsing Technologies (IWPT 2013)*.

van Noord, Geertjan. 1991. Head corner parsing for discontinuous constituency. In *Proceedings of ACL 1991*.

Versley, Yannick and Heike Zinsmeister. 2006. From dependency parsing to deep(er) semantics. In *Proceedings of the Fifth International Workshop on Treebanks and Linguistic Theories (TLT 2006)*.

Vincze, Veronika, Dóra Szauter, Attila Almási adn György Móra, Zoltán Alexin, and János Csirik. 2010. Hungarian dependency treebank. In *Proceedings of the Seventh Conference on International Language Resources and Evaluation (LREC 2010)*.

Volk, Martin and Yvonne Samuelsson. 2004. Bootstrapping parallel treebanks. In *Proceedings of the 5th International Workshop on Linguistically Interpreted Corpora (LINC) at Coling 2004*.