Discontinuity (Re)^{2}visited: A Minimalist Approach to Pseudoprojective Constituent Parsing

Yannick Versley
LinkedIn*
yversley@linkedin.com

Abstract

In this paper, we use insights from Minimalist Grammars (Keenan and Stabler, 2003) to argue for a context-free approximation of discontinuous structures that is both easy to parse for state-of-the-art dynamic programming constituent parsers and has a simple and effective method for the reconstruction of discontinuous tree structures.

The results achieved on the Tiger treebank – paired with state-of-the-art constituent parsers such as the BLLIP and Berkeley parsers – both improve on existing transformation-based approaches for representing discontinuous structures and the state-of-the-art results of Fernández-González and Martins' (2015) parsing-as-reduction approach.

1 Introduction

For languages with free(r) word order and richer morphology, predicate-argument structure (dependencies of words and their heads/governors) and topology (contiguous phrases or regions in the sentence) do not always match up. Hence, constituency parsing techniques that rely on a context-free backbone either have to do with a language-dependent approximation that puts topology at the center (e.g. the TüBa-D/Z treebank of Telljohann et al., 2009, based on topological fields) or can only produce an approximation of the actual predicate-argument structures (as is the case with most parsing approaches targeting the Negra and Tiger treebanks, cf. Skut et al., 1997; Brants et al., 2002).

As an alternative to this procrustean choice, practitioners have traditionally preferred dependency structures, which today offer straightforward ways to deal with nonprojective structures in practical ways despite the fact that exact parsing of nonprojective dependencies with second-order factors is intractable in general (McDonald, 2006).

In the following, we present a principled treatment for approximating discontinuous syntactic structures by context-free ones. The resulting novel technique for pseudoprojective parsing is well suited for lexicalized as well as unlexicalized projective parsers, and yields a feasible solution for accurate probabilistic parsing of discontinuous structures.

2 Related work

Grammar-based approaches to constituent parsing based on minimally context-sensitive formalisms such as LCFRS/MCFG can give guarantees of polynomial-time parsability. However, the step to practical (i.e., fast and accurate) parsing is much larger than is the case for context-free structures: Specifically, binarization is a key to yielding probabilistic formulations with a good tradeoff between expressivity and sparsity e.g. in the parser of Charniak (2000). Binarization, however, is nonstraightforward for LCFRS (Gildea, 2010; van Cranenburgh, 2012), and, unlike for CFG, it is ineffective to obtain a grammar with lower parsing complexity. As a result, parsers directly based on LCFRS such as Kallmeyer and Maier (2013) are rather limited in terms of speed and accuracy.

Greedy (and beam-search) parsing of discontinuous constituents has recently seen some progress in the form of approaches that use swapping techniques in transition-based parsing of discontinuous constituents (Versley, 2014; Maier, 2015). A further strain of approaches uses other dependency parsing techniques by reducing discontinuous con-

*Work was done at the University of Heidelberg stituent parsing to a dependency labeling problem (Hall and Nivre, 2008; Fernández-González and Martins, 2015). However, none of these two groups of approaches easily lend themselves to reranking (Collins, 2000; Charniak and Johnson, 2005) or comparable techniques.

As in van Cranenburgh and Bod (2013), we assume that pairing k-best parsing of context-free structures with a deterministic back-transform to discontinuous structures is a feasible approach to yield k-best lists of discontinuous structures. While van Cranenburgh and Bod rely on a complex ranking mechanism as the second step, we show that refinements to the first steps can already yield results that surpass the state of the art without any separate ranking step.

3 Pseudoprojective parsing

After initial successes with head-lexicalized PCFG parsing for English (Collins, 1997; Charniak, 2000), researchers tried to apply these models to languages that show less configurationality than English. For Czech, (Collins et al., 1999) reordered the original word sequence into one where the corresponding tree becomes continuous. For German, (Dubey and Keller, 2003) transform the treebank by raising discontinuous parts to a higher node in the tree.

Both of these approaches will yield parsing models that (potentially) produce sensible trees for feasible sentences that have a continuous syntactic structure (which comprises the large majority of sentences in English, but may only cover less than half of sentences in typical text for German or Czech). As such, these approximations would be problematic for not offering a path from unparsed tokens as they occur in the text to (potentially) discontinuous trees reflecting predicate-argument structure.

Subsequent approaches such as Levy and Manning (2004) use a multiple-step algorithm to revert the changes introduced by node raising by (heuristically or automatically) identifying NULL elements in the tree, and in a second step identifying dislocated material and linking it up with an appropriate NULL position. Unlike work for heuristic reattachment of dependencies such as Hall and Novak (2005) or Nivre and Nilsson (2005), however, this work is relatively complex. Levy and Manning's evaluation is also performed on the individual steps rather than in a framework that looks at the complete process of parsing and reattachment.

Boyd (2007) proposes an approximation of LCFRS's block structure to get to a tree transformation that is, in difference to a pure node raising approach, reversible. Boyd labels the blocks of a discontinuous nodes by appending a special suffix (i.e. "VP*" instead of "VP"), and later merging these nodes in a top-down fashion. We will discuss this approach in more detail in section 3.2.

Boyd herself compares her approach of block-based transformation to a setting where node raising was applied to the training corpus but no back-transformation was used (similar to parsing models distributed with modern constituent parsers). Undoing the block-based transformation yielded a better result than node-raising without back-transformation using gold part-of-speech tags.

Contra Boyd, Rehbein and van Genabith (2009) find in an evaluation based on f-structure conversion that Boyd's way of transforming trees gives worse results to their approach. Hsu (2010) looks purely at how well parsers are able to reproduce the structures created by pseudoprojective transformations, and finds that parsers introduce more errors in Boyd's method than in the node-raising method.[1]

3.1 Formalizing pseudoprojectivity

For our purposes a tree $T = (\mathrm{NT} \cup \{t_1, \dots, t_n\}), E)$ over a sequence of terminals Term $= t_1, \dots, t_n$ is a directed acyclic graph such that (i) terminals have no children (ii) all nonterminal nodes have at least one child, (iii) all nodes have at most one parent and (iv) that there is a unique topmost node that dominates all other nodes. For the nodes of such a tree we can recursively assign a *yield* function such that the yield of t_i is $\{i\}$ and the yield of a nonterminal node is the union of the yields of the children.

We call a node *contiguous* if its yield is a contiguous subsequence; we call a tree contiguous if all its nodes are contiguous.

[1]A reviewer points out that these results should be seen in the context of the experimental framework used – Rehbein only uses the Berkeley parser, and Hsu only uses plain unbinarized PCFGs – and that particular transformations may be more or less appropriate for specific parsers.

A **pseudoprojective transform** over a set $\mathcal{T}$ of trees is a pair $(\textit{proj}, \textit{unproj})$ of functions with the following properties:

- *proj* is a total function from trees in $\mathcal{T}$ to trees. For any tree $T \in \mathcal{T}$, the value *proj*(T) is a projective tree over the same terminal sequence.
- *unproj* is a partial function from projective trees to trees from $\mathcal{T}$. For any $T' \in \textit{proj}(\mathcal{T})$, $\textit{proj}(\textit{unproj}(T')) = T'$
- *proj* (and by extension *unproj*) preserve contiguous nodes: if T contains a node with label a and a contiguous yield $i..j$, *proj*(T) must contain a node with label a and a contiguous yield $i..j$. If a contiguous node a_1 has a contiguous ancestor a_2, the ancestor relationship between a_1 and a_2 is preserved in *proj*(T).
- There is a set of *barrier nodes* B among the contiguous nodes from $T \cap T'$, minimally including the topmost node and all terminal nodes. If *proj*(T) contains a node not contained in T with a ancestor $n^a \in B$ and a descendent $n^d \in B$, then T must contain a noncontiguous node with ancestor n^a and a descendent n^d.

  (I.e. *proj*$(T) = T$ for all contiguous trees).

Note that we do not require *proj* to be injective (in many useful cases it is not), nor do we say anything about *unproj*'s behaviour on trees outside of *proj*$(\mathcal{T})$. Possibilities for *proj* include simply deleting discontiguous nodes, adjusting their spans by reparenting (as in node raising) and/or adding other nodes.

3.2 LCFRS-inspired approximations

A Linear Context-Free Rewriting System (LCFRS) is a grammar where nodes in the parse tree do not correspond to a single span, but to a fixed number of *blocks*, yielding productions such as

$$\mathrm{S}_1(uvwxy) \rightarrow \mathrm{VP}_2(u,x)\ \mathrm{V}_1(v)\ \mathrm{NP}_2(w,y)$$

In grammar-based parsing, the number of blocks (i.e. letter variables in the production rule) determines the parsing complexity; for pseudo-projective transforms, the most important distinction is between (contiguous) blockdegree 1 nodes and (discontiguous) nodes with a larger blockdegree.

Boyd (2007) proposes a pseudoprojective transformation that approximates LCFRS's block structure: Boyd labels the blocks of a discontinuous nodes by appending a special suffix (i.e. "VP*" instead of "VP"), and later merging these nodes in a top-down fashion.

Van Cranenburgh (2012) suggests a refinement of Boyd's approximation – independently disc where the different blocks of a discontinuous phrase receive different labels (yielding an approximate production of $\mathrm{S} \rightarrow \mathrm{VP}^{*1}\ \mathrm{V}\ \mathrm{NP}^{*1}\ \mathrm{VP}^{*2}\ \mathrm{NP}^{*2}$)

In practice, we found that van Cranenburgh's approach creates many rare categories such as VP^{*12}, and that limiting the numbers in the superscripts yields identical performance to Boyd's approach.

3.3 A new look at node raising

If we have a treebank where a VP is the projection of a verb, we (or the parser) have a concrete expectation of what to find inside a VP node. In contrast, LCFRS and Boyd's transforms treat the trees as non-lexicalized construct: for a VP, we would get two `VP*` nodes with rather different properties. The second (in our case) contains a verb and other children that we would normally expect under a VP (not VP*) node, and the first one contains topicalized material. Furthermore, the occurrence of both topicalization and extraposition can mean that there is no strong regularity among "first parts" and "second parts", which renders van Cranenburgh's refinement ineffective.

If we consider trees to have head terminals, and the hierarchical relations inducing dependency relations between terminals (see definition in the appendix), we could hope for **contiguous subtree preservation**, a stronger version of contiguity preservation:

Given two trees T_1 and T_2 where the terminal node subsequences $i_1..j_1$ of T_1 and $i_2..j_2$ of T_2 as well as the dependencies between the terminals in these subsequences are identical. If T_1 has a contiguous subtree with a yield of $i_1..j_1$, then *proj*(T_2) should also contain the same contiguous subtree.

Node raising (leaving aside unary productions) fulfills this subtree conservation property for treebanks such as Tiger that use sibling adjunction (see Carreras et al., 2008). The LCFRS-derived pseudoprojective transformations do not, which intuitively

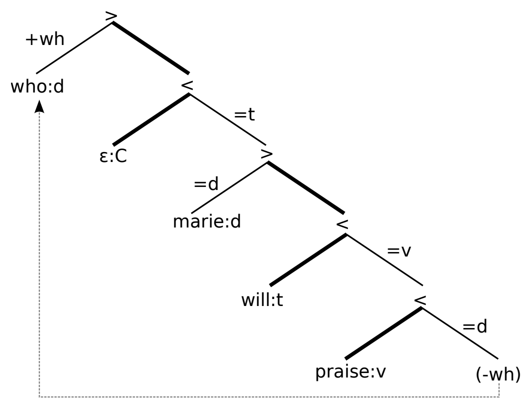

Figure 1: A Minimalist Grammar tree

explains the result of Hsu (2010) that the latter are harder for a context-free parser.

As a conclusion, we see two properties we want to maintain: on the one hand, subtree contiguity preservation seems to be beneficial for getting more accurate parses of the projectivized trees, on the other hand, some marking to aid deprojectivization seems very helpful.

4 Approximating Minimalist Grammars

The approach we present in this paper is based on an approximation of minimalist grammars (MG), stemming from a family of approaches that is popular in mainstream generative linguistics (Chomsky, 1986). Transformational grammar and its descendants have inspired some early work on parsing beyond context-free structures (Dorr, 1987; Lin, 1993), but have for a long time lacked a formalization that would enable more principled work.

Minimalist grammars (in the sense we will use here) are a grammar formalism introduced by Stabler (1997) and Keenan and Stabler (2003) that belongs to the class of *minimally context-sensitive grammars* (Vijay-Shanker et al., 1987; Michaelis, 1998), and carry (in comparison with LCFRS) the benefit of being a lexicalized formalism, and potentially of yielding more compact grammars (Michaelis, 1998) and a more natural model of probability assignments (Hunter and Dyer, 2014).

At the core of Minimalist Grammars are nodes/expressions that carry a category (e.g. x) together with valencies for certain categories (=x, x=) and attractee features (e.g. −w), which designate a node as being moved from its argument position to

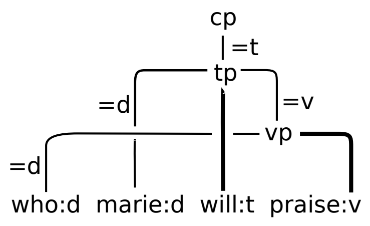

Figure 2: A Tiger-style tree

a higher node, and attractor features (e.g. +w) which designates a node as a host for moved constituent of the corresponding type. A MOVE operation extracts a $-x$ constituent into a $+x$ position, whereas a MERGE operation adjoins one node to another.

To illustrate the correspondence between the analysis assigned by a minimalist grammar and our target representation (flat discontinuous trees), let us compare Figures 1 and 2: On one hand, MG assumes binary adjunctions while the Tiger annotation scheme has flat phrases – this kind of variation in annotation scheme, while it may influence the structures preferred by PCFG and similar models, is semantically equivalent (Johnson, 1998). On the other hand, the representation with discontinuous phrases eliminates all empty nodes (both the empty complementizer and the trace of the moved *who*), sometimes yielding headless phrases.

We should emphasize here that, while many insights about constraints in movements that are valid in Minimalist Grammars are still valid in actual treebanks (whenever we take care to allow for notational differences), it is not true in general that annotated treebanks are designed using the criteria that are applied in the Minimalist Program (indeed, Minimalist Grammars have the goal of providing a formal grammar to express such ideas in a more theory-neutral way), nor do decisions about headedness, argumenthood, and the modeling of moved nodes necessarily correspond to those that one would make in an MG.

Beyond couching the relation between predicate-argument trees and surface trees in more principled terms (see also Boston et al., 2010 for a discussion of the relation to dependencies), what do we gain?

The **Shortest Move Constraint** posits that any material that is moved must attach at the first possible host node (i.e., for a moved phrase of type $-x$, attach to the lowest ancestor with the feature $+x$. With the caveat that SMC may not be appropriate for all types of movement (e.g. scrambling), it firstly provides an intuition on possible hosts for dislocated material (whereas the rule of "*raise the nodes up to a host node where the material can be attached without discontinuity*" is of a much more practical nature), while simultaneously predicting that any (potential) host phrase must also act as a barrier for movement.

Following the discussion of Boston et al. (2010), we find a plausible rationale for contiguous subtree preservation. A downside is that undoing *Move*-based dislocation – especially if we consider node raising as the protypical example – would need more information. Consider the following sentence:

(1) *Ich* *habe* [np *den* *Mann* [pp *mit*
I have the man with the
dem *Hamster*]] *gesehen* [*der* *grinst*]
hamster seen which grins
I have seen the man with the hamster which grins

In this case, the attachment of the relative clause to *Mann* (man) or to *Hamster* (hamster) is an instance of general attachment ambiguity, and while we can use very general preferences (attach low; prefer attaching as an argument rather than as an adjunct, cf. Hobbs and Bear, 1990), correct attachment often requires semantic or context-dependent information. (LCFRS-inspired transforms pass this problem onto the parser, in a form that poorly fits state-of-the-art parsing models).

4.1 Representing Moved Phrases

If we look at minimalist grammars from our *pseudoprojective transform* perspective, we see that even when we want a transform that fulfills the *contiguous subtree preservation* criterion (which includes Boyd's and van Cranenburgh's solution, even though they are pseudoprojective transforms in our sense), we can solve the problem differently from – and hopefully better than – node raising.

Marcus et al. (1994) and later Skut et al. (1997), in a solution that we will call the *trace-filler mechanism* try to ensure a relation between host phrase and the argument-structure parent by inserting an empty node in the latter and coindexing with the dislocated material; empty nodes are not supported by today's parsers, and the idea of encoding the coindexation by appending slash categories to phrases has been shown to be non-beneficial (Schiehlen, 2004).

We can preserve contiguous subtrees and yet want to add some marking, we could insert a phrase between the host phrase and the dislocated material, in the simplest case one single fixed category, as in

(2) [vp [np den Mann [pp mit dem Hamster]] gesehen [X [s der grinst]]]

where we inserted an additional `X` node between host and dislocated material. In difference to simple node raising, this would encode the information which nodes were dislocated and which ones were not in the tree rather than leaving it implicit.

The **LR** scheme for adding additional information to raised nodes, which we propose here, leaves the block of the sentence that includes its head with an unchanged node label, but labels the other part as being left- or right-dislocated, yielding, in the above example, `VP⋆L`, `VP` and `VP⋆R` as labels if the middle phrase contains the head of the VP. In our example, we would obtain the following:

(3) [vp [np den Mann [pp mit dem Hamster]] gesehen [NP*R [s der grinst]]

Similar to the *trace-filler* view, and different from the LCFRS view, we distinguish between the main part of a phrase (normally the one containing the head) and discontinuous (moved) parts. We mark the moved parts with the concatenation of the original parent label and a `⋆L` or `⋆R` suffix. In comparison with node raising, the additional node fulfills an important function as it both helps the parser recognize where such moved or extraposed material can occur, based on topology (i.e., outside context) as well as the material itself, and also provides useful information for reattachment.

4.2 Simple Heuristics for Backtransformation

Adopting a minimalist perspective on reattachment, we would assume that a parser for context-free structures can plausibly produce the *derived structures* that capture sentence topology but not necessarily the full set of argument and adjunct dependencies.

method	F_1
uncross, no reattach	94.99
LR + top-down reattach	97.90
LR + bottom-up reattach	99.20
+S barrier	99.08

Table 1: Heuristic reattachment: roundtrip results on Tiger sentences 40475-45474

To capture the discontinuous structures, we investigate two reattachment heuristics:

- The **top-down** heuristic chooses an appropriate node from the siblings of the dislocated phrase, similar to the way that Boyd's or van Cranenburgh's reattachment algorithm would operate.
- The **bottom-up** heuristic iterates over the yield of the parent node (limited either to the part left of the dislocated one for `*R` ones, otherwise the part right of it), to produce a sequence of all suitable descendents of the parent node in a close-to-far, low-to-high fashion.

Evidence from reconstructing gold data (see table 1) shows that the *bottom-up* heuristic is more accurate in the reconstruction, which is consistent with the preference on low attachment as formulated by, e.g., Hobbs and Bear (1990).

Considering the relation to Minimalist Grammar and the Shortest Move Constraint, we would expect that certain nodes act as **barriers** and can block dislocation or movement across them: In example (4) (see figure 3), such a barrier constraint would prevent the fronted subclause "*nehmen die Überlegungen Gestalt an*" (if the plans become more concrete) to the VP internal to the relative clause instead of the VP of the matrix sentence.

In our experiments with projectivising the development set of the Tiger treebank and subsequently reattaching dislocated phrases (see table 1), we see that top-down reattachment already provides a 60% error reduction with respect to simply leaving nodes unattached, and that using a preference for lower attachment ("*bottom-up reattach*") yields a further 60% error reduction in terms of phrase F_1. In contrast, we see that having sentence (`S`) nodes block movement leads to a slight decrease in accuracy.

5 Experimental set-up

In the following parsing experiments, we want to compare more directly the accuracy of our **LR** scheme of projectivization and reattachment to that of, e.g. Boyd's proposal while taking into account many of the concerns that occur in practical parsing today, in particular the compatibility with other techniques used to improve parsing accuracy (linguistic tree transformations, products of latent variable grammars, word clustering-based generalizations of words). As we are specifically concerned about the behaviour with more unknown words and different distributions of syntactic constructions that occurs in out-of-domain corpora, we exclusively use the part-of-speech tags assigned by the parsing model itself.

Corpora used As an in-domain corpus that we split into a training, development and testing set, we use the Tiger treebank (Brants et al., 2002), which encodes argument and adjunct relations in a discontinuous constituent structure with edge labels. We use two splits that were used in the literature for parsing experiments: the first, called the *SPMRL split* reproduces the train/development/test portions of Farkas and Schmid (2012) and was used in the SPMRL shared tasks of 2013 for dependency and constituency parsing, using sentences 1–40474 as training set, the next 5000 as development set, and the remaining 5000 as a test set. The second split was first used in the experiments of Hall and Nivre (2008) and uses folds 9 and 10 in a 10-fold setup as development and testing portions, respectively.

As an out-of-domain dataset, we use the Smultron treebanks of Volk et al. (2015), which include portions of a novel (`sophie`), business reports (`economy`), texts about mountaineering (`alpine`) as well as extracts from the manual of a DVD player (`dvdman`). The annotation of the Smultron treebanks is loosely based on the Tiger scheme but differs in two important respects: on one hand, the Tiger scheme merges PP nodes with the noun phrases that is the argument of the preposition, yielding one single PP phrase; on the other hand, the Smultron annotation scheme uses extra nodes for unary noun, verb and adjective phrases that would be elided in the Tiger scheme. For a sensible comparison, we use a transformed version of the Smultron treebanks where unary nodes are deleted and argu-

(4) **[VP*L** Nehmen die Überlegungen Gestalt an**]**, würden die Frankfurter, **[S** die im Januar
take the thoughts shape on, would the Frankfurt+ADJ, which in January
1988 **[VP** ihr 125jähriges Bestehen feiern**]** können**]**, **[VP** die Verbindung zu ihren historischen
1988 their 125-year existence celebrate can, the connection to their historical
Wurzeln kappen**]**.
roots clip.
"If the plans become more concrete, the Frankfurt group, which had its 125-year anniversary in January 1988 would cut the the connection to their historical roots."

Figure 3: An example sentence where the closest VP is not a suitable movement target

ment NPs of prepositional phrases are unwrapped.

Parsing models For reasons of simplicity, we limit ourselves to a small number of generative parsing models that have been shown to work well for context-free parsing.

The **BLLIP parser** (of which we use the generative model only and not the discriminative reranking part) uses a "maximum-entropy-inspired" probabilistic model that produces head-lexicalized constituents from the inside out while conditioning on the two previous neighbours, the grandparent constituents, and their heads (Charniak, 2000).

The **PCFG-LA** parsing model of Petrov et al. (2006) uses a zeroth-order right-markovized version of the treebank which is subsequently augmented with latent symbol refinements in order to improve the fit, using smoothing and a split-merge procedure to avoid overfitting in the EM-based refinement process. We found that four split-merge iterations (instead of the default six) gave the best results with the linguistically transformed trees.

As the PCFG-LA model learns a latent-variable augmentation of the treebank trees and most often reaches a non-unique local maximum of the EM objective, it is possible and useful to combine multiple PCFG-LA models to reach an even better performance using a **product grammar** approach where each rule is scored by a product-of-experts of multiple parsers (Petrov, 2010).

For the out-of-domain parsing, Candito and Seddah (2010) found that replacing words with clusters improved the generalization ability of parsers. After preliminary experiments showed that replacing all words with clusters actually had a negative effect, our setting using **word clusters** only replaces words that occur fewer than five times, using word clusters derived with the Marlin tool (Müller and Schuetze, 2015) and text from the DECOW corpus[2], limiting the vocabulary size to the most frequent 250 000 word types as done by Müller and Schütze.

Finally, we also include in our investigation the use of **linguistic tree transformations**, which e.g. Dubey (2005) as well as Versley and Rehbein (2009) found useful both for unlexicalized and discriminative PCFG parsing. In particular, we use lowering of parenthetic material as proposed by Maier et al. (2012) to reduce the complexity of discontinuities, but also markers for relative clauses and comparative phrases, linguistically motivated subcategorization information for sentences, added case information to noun phrases, as well as refing part-of-speech classes using some morphological information.

5.1 Comparing Boyd with LR

For each of the three parsers (BLLIP, PCFG-LA, PCFG-LA product grammar) we produce transformed trees with the non-annotated treebank trees (`orig`) as well as the enriched ones (`xform`), using Boyd's projectivization transform (`boyd`) and the one proposed here (`LR`).

Quite expectedly, we find that head-lexicalized parsing using the BLLIP parser is substantially helped by the linguistically motivated tree transformations. Somewhat less intuitively, as the earlier results of Petrov et al. (2006) for English indicate that basic transformations such as head annotation do not help PCFG-LA models, we find that our linguistically motivated transformations substantially help both single PCFG-LA and product grammars. We also see that the LR transform performs slightly worse than Boyd's transform in the BLLIP parser,

[2]`http://www.corporafromtheweb.org`

variant	*BLLIP*				*PCFG-LA*				*LA-product*			
	F1	EX	POS	discF1	F1	EX	POS	discF1	F1	EX	POS	discF1
orig-Boyd	80.32	43.75	97.55	68.41	82.35	42.95	97.63	70.55	83.11	45.66	97.76	72.68
orig-LR	81.28	45.71	97.68	70.02	82.27	43.57	97.71	71.46	83.29	45.81	97.80	72.95
xform-Boyd	82.01	44.67	97.70	71.55	81.35	42.71	97.55	70.92	83.62	46.16	97.72	74.43
xform-LR	81.93	45.27	**97.73**	71.64	82.43	43.49	97.68	72.77	84.18	46.90	**97.84**	75.43
LR + cleanup	82.19	45.53	**97.73**	71.85	82.43	43.49	97.68	72.77	**84.36**	47.08	**97.84**	75.56
LR + filter	**82.20**	**45.63**	97.72	**72.09**	**82.58**	**43.87**	**97.69**	**73.11**	84.09	**47.31**	97.79	**75.81**

Table 2: Comparison on the Tiger development set, sentences with ≤ 70 words

whereas the tendency is exactly reversed in PCFG-LA-based parsing. Looking at the evaluation results when ignoring continuous constituents (see table 2, *discF1* columns), we see that the LR scheme specifically improves the quality of discontinuities, sometimes by a slight amount, and sometimes by as much as one percent).

Dealing with invalid solutions Some of the decrease in performance for the BLLIP parser for the LR scheme is due to the occasional dislocated phrases (e.g. `NP*R`) that cannot be reattached and are then left behind. For the Boyd transform, our implementation already reverts discontinuity-marked phrases (e.g. `NP*`) to the original label (e.g. `NP`).

To deal with the 'invalid' phrases, we propose two solutions: the first one, which we call the **cleanup** strategy, involves a post-processing step in which all dislocated phrase nodes are deleted (and their daughters attached to the deleted phrase's parent). The second one, which we call the **filter** strategy, involves producing ranked list of parses, of which we delete any parse that includes dislocated nodes that cannot be reattached. Of the remaining parses, we chose the highest-scoring one.

For the BLLIP parser, we find that both the cleanup step and the kbest list filter result in improvements over the default version, while the advantage of the filtering-based approach over the simpler cleanup approach is very slight at best.

5.2 Comparing with the state of the art

Table 4 compares the result of our pseudoprojective parsing approach to other evaluation results on the Tiger treebank using parser-assigned POS tags.[3]

We find that our BLLIP parsing model already performs quite well, going beyond the results of van Cranenburgh and Bod (2013) that use a PCFG base parser followed by a DOP-inspired reranking step, or the discriminative parsing results of Versley (2014). Using the product-grammar approach, we find that our approach outperforms the parsing-as-reduction approach of Fernández-González and Martins (2015) by a small margin in the Hall/Nivre split, also yielding much improved (+1.9%) evaluation results for the case of the SPMRL split.

5.3 Experiments with Out-of-domain parsing

Table 3 shows the results on different treebanks from SMULTRON in comparison with those on Tiger. We see that out-of-domain results are substantially lower than those achievable on the Tiger development set: The novel (`sophie`) performs relatively well while performance on the other domains falls off even more. The most difficulties are due to the DVD manuals (`dvdman`), which already Seeker and Kuhn (2014) argue to be due to several phenomena not seen in well-edited text.

If we compare the accuracy of the POS assignment from the PCFG-LA parsing models to both the tagging results of Seeker and Kuhn (2014) and our own, in each case using the Marmot CRF tagger Müller et al. (2013), we see that in all cases (except for the Alpine domain) the parser is able to make use of the syntactic context to achieve improved part-of-speech accuracy, even if the overall difficulty of the out-of-domain texts is higher. Use of word clusters seems to be especially helpful in those cases where the parser has difficulty at the POS level.

6 Summary

In this paper, we have presented a new, linguistically well-motivated method for transforming discontinuous trees to context-free ones and back.

[3]Fernandez-Gonzalez and Martins report substantially higher results using gold part-of-speech tags.

Parser	LF1/70	EX/70	POS
TIGERDEV			
multi+sm4	84.18	46.90	97.84
multi+sm4+clust	84.31	47.51	97.74
ALPINE			
multi+sm4	74.18	32.70	93.79
multi+sm4+clust	74.80	33.96	93.96
marmot Seeker14			94.42
ECONOMY			
multi+sm4	74.38	22.05	91.73
multi+sm4+clust	74.50	22.44	92.40
marmot Seeker14			91.83
SOPHIE			
multi+sm4	77.71	38.56	96.51
multi+sm4+clust	77.53	38.19	96.41
marmot Seeker14			95.20
DVDMAN			
multi+sm4	71.54	25.78	90.55
multi+sm4+clust	72.45	26.56	90.22
marmot Seeker14			90.81

Table 3: Comparative results for parsing the SMULTRON treebanks (LR without cleanup)

Tiger-H&N (pred)	$L \leq 40$		all	
	F1	EX	F1	EX
Hall&Nivre 2008	75.33	32.63	—	—
van Cranenburgh '13	78.8-	40.8-	—	—
Fernandez&Martins '15	82.57	45.93	81.12	44.48
Ours, BLLIP	81.16	43.17	79.68	41.87
Ours, PCFGLA-prod	82.93	44.26	81.93	42.87
Tiger-SPMRL (pred)	$L \leq 70$		all	
	F1	EX	F1	EX
Versley 2014	73.90	37.00	—	—
Fernandez&Martins '15	77.72	38.75	77.32	38.64
Ours, BLLIP	76.96	35.52	76.52	35.42
Ours, PCFGLA-prod	79.84	39.61	79.50	39.50

Table 4: Test set results on the split of Hall and Nivre (2008) and on the SPMRL split

Because this method allows us to make effective use of state-of-the-art parsing for continuous trees (similar to the parsing-as-reduction approach leveraging state-of-the-art models for dependency parsing), this transformation approach has a substantial advantage over models that use a more complex grammar formalism but have to use a simpler statistical model.

Using three generative probabilistic models, we showed that our method performs better than the older transformation approach of Boyd (2007), and outperforms the current state of the art for discontinuous parsing on the Tiger treebank, the parsing-as-reduction approach of Fernández-González and Martins (2015). Future work will explore feature-based statistical models for reattachment and parse selection.

Acknowledgements The research described in this paper was supported by the Leibniz Society and the Baden Württemberg ministery of science, research and the arts as part of the ScienceCampus *Empirical Linguistics and Computational Natural Language Modeling*. The author is grateful to Jen Sikos and Alexis Palmer, as well as the three anonymous reviewers, for helpful comments on earlier versions of this paper.

References

Boston, M. F., Hale, J. T., and Kuhlmann, M. (2010). Dependency structures derived from Minimalist Grammars. In Ebert, C., Jäger, G., and Michaelis, J., editors, *MOL 10/11*, volume 6149 of *Lecture Notes in Computer Science*, pages 1–12. Springer-Verlag, Berlin Heidelberg.

Boyd, A. (2007). Discontinuity revisited: An improved conversion to context-free representations. In *Proceedings of the Linguistic Annotation Workshop (LAW 2007)*.

Brants, S., Dipper, S., Hansen, S., Lezius, W., and Smith, G. (2002). The TIGER treebank. In *Proc. TLT 2002*.

Candito, M.-H. and Seddah, D. (2010). Parsing word clusters. In *Proceedings of the First Workshop on Statistical Parsing of Morphologically-Rich Languages (SPMRL 2010)*.

Carreras, X., Collins, M., and Koo, T. (2008). TAG, dynamic programming, and the perceptron for efficient, feature-rich parsing. In *Proceedings of CoNLL*.

Charniak, E. (2000). A maximum-entropy-inspired parser. In *Sixth Applied Natural Language Processing Conference (ANLP-NAACL 2000)*.

Charniak, E. and Johnson, M. (2005). Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proc. ACL 2005*.

Chomsky, N. (1986). *Barriers*. Linguistic Inquiry Monographs. MIT Press.

Collins, M. (1997). Three generative, lexicalized models for statistical parsing. In *Proc. ACL 1997*.

Collins, M. (2000). Discriminative reranking for natural language parsing. In *ICML 2000*.

Collins, M., Hajič, J., Ramshaw, L., and Tillmann, C. (1999). A statistical parser for Czech. In *Proceedings of ACL 1999*.

Dorr, B. J. (1987). Principle-based parsing for machine translation. Technical report, Massachusetts Institute of Technology Artificial Intelligence Laboratory.

Dubey, A. (2005). What to do when lexicalization fails: parsing German with suffix analysis and smoothing. In *ACL-2005*.

Dubey, A. and Keller, F. (2003). Probabilistic parsing for German using sister-head dependencies. In *ACL'2003*.

Farkas, R. and Schmid, H. (2012). Forest reranking through subtree ranking. In *EMNLP-CoNLL 2012*.

Fernández-González, D. and Martins, A. F. T. (2015). Parsing as reduction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1523–1533, Beijing, China. Association for Computational Linguistics.

Gildea, D. (2010). Optimial parsing strategies for linear context-free rewriting systems. In *Proceedings of NAACL 2010*.

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

Hall, K. and Novak, V. (2005). Corrective modeling for non-projective dependency parsing. In *Proceedings of the Ninth International Workshop on Parsing Technology (IWPT 2005)*.

Hobbs, J. R. and Bear, J. (1990). Two principles of parse preference. In *Coling 1990*.

Hsu, Y.-Y. (2010). Comparing conversions of discontinuity in PCFG parsing. In *Proceedings of the Ninth International Workshop on Treebanks and Linguistic Theories*.

Hunter, T. and Dyer, C. (2014). Distributions on minimalist grammar derivations. In *Proceedings of the 13th Meeting on the Mathematics of Language (MoL 13)*.

Johnson, M. (1998). Pcfg models of linguistic tree representations. *Computational Linguistics*, 24(4):613–632.

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

Keenan, E. L. and Stabler, E. P. (2003). *Bare Grammar*. CSLI Publications, Stanford.

Levy, R. and Manning, C. (2004). Deep dependencies from context-free statistical parsers: correcting the surface dependency approximation. In *ACL 2004*.

Lin, D. (1993). Principle-based parsing without overgeneration. In *Proceedings of ACL 1993*.

Maier, W. (2015). Discontinuous incremental shift-reduce parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1202–1212, Beijing, China. Association for Computational Linguistics.

Maier, W., Kaeshammer, M., and Kallmeyer, L. (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).

Marcus, M., Kim, G., Marcinkiewicz, M. A., MacIntyre, R., Bies, A., Ferguson, M., Katz, K., and Schasberger, B. (1994). The Penn treebank: Annotating predicate argument structure. In *Workshop on Human Language Technology*.

McDonald, R. (2006). Online learning of approximate dependency parsing algorithms. In *EACL 2006*.

Michaelis, J. (1998). Derivational minimalism is mildly context-sensitive. In *Logical Aspects of Computational Linguistics*.

Müller, T., Schmid, H., and Schütze, H. (2013). Efficient higher-order CRFs for morphological tagging. In *Proceedings fo EMNLP 2013*.

Müller, T. and Schuetze, H. (2015). Robust morphological tagging with word representations. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 526–536, Denver, Colorado. Association for Computational Linguistics.

Nivre, J. and Nilsson, J. (2005). Pseudo-projective dependency parsing. In *Proceedings of ACL 2005*.

Petrov, S. (2010). Products of random latent variable grammars. In *HLT-NAACL 2010*.

Petrov, S., Barett, L., Thibaux, R., and Klein, D. (2006). Learning accurate, compact, and interpretable tree annotation. In *COLING-ACL 2006*.

Rehbein, I. and van Genabith, J. (2009). Automatic acquisition of lfg resources for german: As good as it gets. In *Proceedings of LFG09*.

Schiehlen, M. (2004). Annotation strategies for probabilistic parsing in German. In *Proc. Coling 2004*.

Seeker, W. and Kuhn, J. (2014). An out-of-domain test suite for dependency parsing of German. In *Proceedings of LREC 2014*.

Skut, W., Krenn, B., Brants, T., and Uszkoreit, H. (1997). An annotation scheme for free word order languages. In *Proceedings of the Fifth Conference on Applied Natural Language Processing (ANLP-97)*.

Stabler, E. (1997). Derivational minimalism. In *Logical Aspects of Computational Linguistics*, pages 68–95.

Telljohann, H., Hinrichs, E. W., Kübler, S., Zinsmeister, H., and Beck, K. (2009). Stylebook for the Tübingen Treebank of Written German (TüBa-D/Z). Technical report, Seminar für Sprachwissenschaft, Universität Tübingen.

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

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

Versley, Y. (2014). Experiments with easy-first nonprojective constituent parsing. In *Proceedings of the First Joint Workshop on Statistical Parsing of Morphologically Rich Languages and Syntactic Analysis of Non-Canonical Languages*, pages 39–53, Dublin, Ireland. Dublin City University.

Versley, Y. and Rehbein, I. (2009). Scalable discriminative parsing for German. In *Proc. IWPT 2009*.

Vijay-Shanker, K., Weir, D. J., and Joshi, A. K. (1987). Characterizing structural descriptions produced by various grammatical formalisms. In *Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics (ACL 1987)*.

Volk, M., Göhring, A., Rios, A., Marek, T., and Samuelsson, Y. (2015). SMULTRON (version 4.0) — The Stockholm MULtilingual parallel TReebank. An English-French-German-Quechua-Spanish-Swedish parallel treebank with sub-sentential alignments.

Appendix A. Notational clarifications

For two trees T and T' covering the same sequence of terminals, and containing no unary productions, we define the **intersection tree** $T \cap T'$ as follows:

- If we identify nonterminals with (label, yield) pairs, the nonterminals of $T \cap T'$ are exactly those (label, yield) pairs that correspond to nonterminals of both T and T'.

- The set of edges of $T \cap T'$ is determined as the cover relation of its descendent relation. A node n_1 is a descendent of a node n_2 iff $\text{yield}(n_1) \subseteq \text{yield}(n_2)$.

We can extend this definition to trees containing unary productions if we consider mappings L_T from yields to label sequences (i.e., $\mathcal{P}(\mathbb{N}_n) \rightarrow \Sigma^*$ if Σ is our set of labels).

We can define this mapping for a tree as follows:

- if there is no node in T with a yield y, then $L_T(y) = \varepsilon$

- if T contains one or more nodes with the same yield, they form a chain of unary productions. Given the sequence $l_1, \dots l_m \in \Sigma^*$ of the labels of this sequence (read from top to bottom), we then set $L_T(y) = l1, \dots, l_m$.

Taking (e.g.) the longest common prefix of two such sequences gives us an operation that is idempotent, commutative and associative.

If we identify nonterminals with (label, yield, order) triples, we can extend the definition of the intersection tree as follows:

- The nonterminals of $T \cap T'$ are those (label, yield, order) triples that correspond to nonterminals of both T and T' and whose unary parents (if any) are also nonterminals of $T \cap T'$.

- A node n_1 is a descendent of a node if either $\text{yield}(n_1) \subsetneq \text{yield}(n_2)$ or if $\text{yield}(n_1) = \text{yield}(n_2) \wedge \text{order}(n_1) \leq \text{order}(\text{n}_2)$.

We can assign **induced dependencies** to a node given a suitable head assignment function as follows:

Given a head assignment function $\text{headidx} : \Sigma \times \Sigma^* \rightarrow \mathbb{N}$ (such that $\text{headidx}(p, c_1, \dots, c_m) \in \{1, \dots, m\}$) we can recursively assign a head to each node by using:

- $\text{head}(n) = n$ for terminal nodes

- $\text{head}(n) = \text{head}(n_k)$ if n has label l_p and the children $n_1, \dots, n_m$ have labels $l_1, \dots, l_m$ and $\text{headidx}(l_p; l_1, \dots, l_m) = k$

The induced dependency graph then contains $\text{Range}(Term)$ as nodes. For any pair of a node n and its child n', the dependency graph contains a dependency edge $(\text{head}(n), \text{head}(n'))$ as long as $\text{head}(n) \neq \text{head}(n')$.