Discontinuity Revisited: An Improved Conversion to Context-Free Representations

Adriane Boyd
Department of Linguistics
The Ohio State University
1712 Neil Ave.
Columbus, OH 43210
adriane@ling.osu.edu

Abstract

This paper introduces a new, reversible method for converting syntactic structures with discontinuous constituents into traditional syntax trees. The method is applied to the Tiger Corpus of German and results for PCFG parsing requiring such context-free trees are provided. A labeled dependency evaluation shows that the new conversion method leads to better results by preserving local relationships and introducing fewer inconsistencies into the training data.

1 Introduction

Unlike traditional treebanks, the Negra and Tiger Corpora (Brants et al., 2002) allow crossing branches in the syntactic annotation to handle certain features of German. In order to use the Negra or Tiger Corpus data to train a PCFG parser, it is necessary to convert the syntactic annotation into context-free syntax trees. In previous work (see section 3.1), a non-reversible method has been used that raises nodes in the tree to eliminate discontinuities. This method effectively introduces inconsistencies into the data and disrupts the grammatical dependency annotation in the trees. This paper presents a new, reversible method for converting Negra and Tiger syntactic structures into context-free syntax trees appropriate for training a PCFG parser. A reversible conversion allows the original grammatical dependency relations to be reconstructed from the PCFG parser output. This paper focuses on the newer, larger Tiger Corpus, but methods and results are very similar for the Negra Corpus.

2 Tiger Corpus

The Tiger Corpus was a joint project between Saarland University, the University of Stuttgart, and University of Potsdam. The Tiger Corpus Version 2 contains 50,474 sentences of newspaper text. The Tiger annotation combines features from phrase structure grammar and dependency grammar using a tree-like syntactic structure with grammatical functions labeled on the edges of the tree (Brants et al., 2002). Flat sentence structures are used in many places to avoid attachment ambiguities and non-branching phrases are not allowed. The annotation scheme emphasizes the use of the tree structure to encode all grammatical relations in local trees regardless of whether a grammatical dependency is local within in the sentence. This leads to the use of discontinuous constituents to handle flexible word order, extraposition, partial constituent fronting, and other phenomena. An example of a Tiger tree with discontinuous constituents (both VPs) is shown in Figure 1.

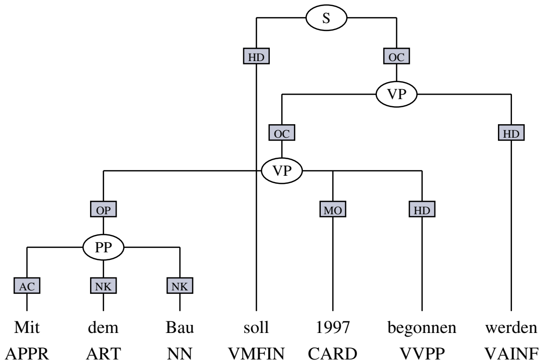
3 Conversion to Context-Free Syntax Trees

For research involving PCFG parsing models trained on Tiger Corpus data, it is necessary to convert the syntax graphs with crossing branches into traditional syntax trees in order to extract context-free grammar rules from the data. Approximately 30% of sentences in Tiger contain at least one discontinuous constituent.

3.1 Existing Tiger Corpus Conversion

In previous research, crossing branches have been resolved by raising non-head nodes out of discon-

Proceedings of the Linguistic Annotation Workshop, pages 41–44,
Prague, June 2007.



'Construction should start in 1997.'
(lit. *with the construction should 1997 begun be*)

Figure 1: Discontinuous Tiger tree

tinuous constituents until no more branches cross. The converted sentence from Figure 1 is shown in Figure 2. In any sentence, multiple nodes could each be raised one or more times, so it is difficult to automatically reconstruct the original sentence. Previous work on PCFG parsing using Negra or Tiger has either used the provided Penn Treebank-style versions of the corpora included with Negra and Tiger Version 1 (Dubey and Keller, 2003; Dubey, 2004) or used a program provided with the Negra/Tiger *Annotate* software (Plaehn and Brants, 2000) which performs the raising algorithm (Kübler, 2005; Kübler et al., 2006). This conversion will be referred to as the "raising method".

3.2 A New Approach to Eliminating Discontinuities

The raising method has the advantages of preserving the number of nodes in the tree, but it is not easily reversible and disrupts local trees. Raising non-head nodes is not an ideal way of eliminating discontinuities because it does not preserve the relationship between a head and a dependent that is represented in a local tree in the Tiger annotation. After raising one or more nodes in 30% of the sentences in the corpus, local trees are no longer consistent across the treebank. Some VPs may contain all their objects while others do not. For example, in Figure 2 the PP object *Mit dem Bau* is no longer in the local tree with its head *begonnen*. The PCFG has lessened chance of capturing generalizations from the resulting inconsistent training data.

Preferable to the raising method is a conversion that is reversible and that preserves local trees as much as possible. The new approach to the conversion involves splitting discontinuous nodes into smaller "partial nodes". Each subset of the original children with a continuous terminal yield becomes a partial node. In this way, it is possible to remove crossing branches while preserving the parent relationships from the original tree. Because partial nodes retain their original parents, the reverse conversion is greatly simplified.

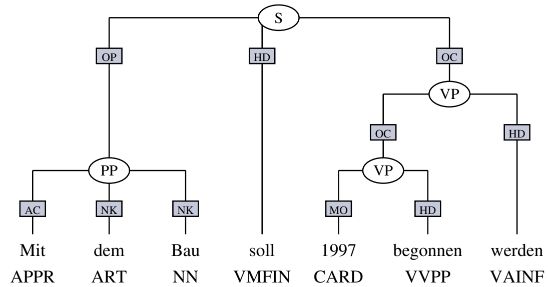


Figure 2: Result of conversion by raising

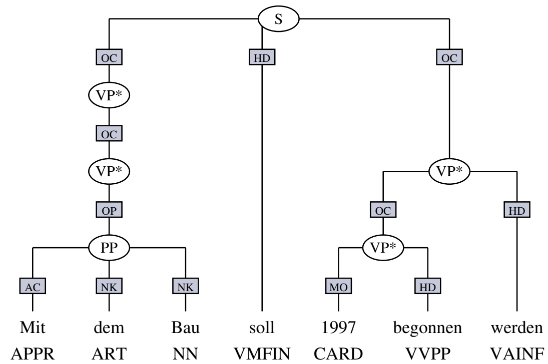


Figure 3: Result of conversion by splitting

In order to make the conversion easily reversible, the partial nodes need to be marked in some way so that they can be identified in the reverse conversion. A simple method is to use a single mark (*) on all partial nodes.[1] For example, a discontinuous VP with the children NN-OA (noun acc. obj.) and VVINF-HD (infinitive) would be converted into a VP* with an NN-OA child and a VP* with a VVINF-HD child. The method of creating partial nodes with a single mark will be called the "splitting method". It is completely reversible unless there are two discontinuous sisters with the same label. While it is not unusual for a Tiger tree to have multiple dis-

[1] This approach was inspired by Joakim Nivre's paper *Pseudo-Projective Dependency Parsing* (Nivre, 2005), in which non-projective dependency structures are converted to easier-to-parse projective dependency structures in a way that limits the number of new labels introduced, but is mostly reconstructible.

continuous nodes with same label (as in Figure 1), two nodes with the same label are never sisters so the conversion is reversible for all sentences. Each tree is converted with the following algorithm, which is a postorder traversal that starts at the root node of the tree. The postorder traversal guarantees that every child of a node is continuous before the node itself is evaluated, so splitting the node under consideration into partial nodes will resolve the discontinuity.

SPLIT-DISC-NODES($Node$)
 for each $Child$ of $Node$
  SPLIT-DISC-NODES($Child$)
 if $Node$'s terminal yield is discontinuous
  $Children$:= immediate children of $Node$
  $ContSets$:= divide $Children$ into subsets
        with continuous terminal yields
  for each $ChildSubset$ in $ContSets$
   $PNode$:= new node
   $PNode$'s label := $Node$'s label with mark (*)
   $PNode$'s parent := $Node$'s parent
   for each $Child$ in $ChildSubset$
    $Child$'s parent := $PNode$
  remove $Node$ from tree

The splitting conversion of the sentence from Figure 1 can be seen in Figure 3. To convert the split version back to the original version, the tree is examined top-down, rejoining any marked sister nodes with the same label.

4 Results

All parsing was performed using the unlexicalized parsing model from the left corner parser LoPar Schmid (2000). The input data was labeled with perfect tags from the corpus to prevent errors in tagging from affecting the parsing results.

4.1 Data Preparation

For the following experiments, the Tiger Corpus Version 2 was divided into training, development, and testing sections. Following the data split from Dubey (2004), 90% of the corpus was used as training data, 5% as development data, and 5% as test data. In preprocessing, all punctuation was removed because it is not attached within the sentence. 6.5% of sentences are excluded because they contain no annotation beyond the word level or because they contain multiple root nodes. After preprocessing, there are 42,612 sentences in the training set. For evaluation, only sentences with 40 words or fewer are used, leaving 2,312 test sentences. The raised version is created using the *Annotate* software and the split version is created using the method described in section 3.2. For the split version, partial nodes are rejoined before evaluation.

In the Penn Treebank-style versions of the corpus appropriate for training a PCFG parser, each edge label has been joined with the phrase or POS label on the phrase or word immediately below it. Because of this, the edge labels for single-word arguments (e.g., pronoun subjects) are attached to the POS tag of the word, which provides the parser with the perfect grammatical function label when perfect lexical tags are provided. This amounts to providing the perfect grammatical function labels for approximately one-third of arguments in Tiger, so to avoid this problem, non-branching phrase nodes are introduced for single-word arguments. Phrase nodes are introduced above all single-word subjects, accusative objects, dative objects, and genitive objects. The category of the inserted phrase depends on the POS tag on the word (NP, VP, or AP as appropriate).

4.2 Experiment 1: Reversibility of Splitting Conversion

All sentences in the test set were converted into syntax trees by splitting discontinuous nodes according to the algorithm in section 3.2. All 2,312 sentences in the test set can be converted back to their original versions with no errors. The most frequently split nodes are VP (∼55%) and NP (∼20%).

4.3 Experiment 2: Labeled Dependency Evaluation

A labeled dependency evaluation is chosen instead of a typical PARSEVAL evaluation for two reasons: 1) PARSEVAL is unable to evaluate trees with discontinuous constituents; 2) a bracketing evaluation examines all types of brackets in the sentence and may not reflect how accurately significant grammatical dependencies have been identified.

It is useful to look at an evaluation on grammatical functions that are important for determining the functor-argument structure of the sentence. In this evaluation, subjects, accusative objects, prepo-

	Raised			Split		
GF	P	R	F	P	R	F
Subj	74.8	71.6	73.2	74.7	73.5	74.1
AccObj	46.3	48.9	47.4	49.2	53.7	51.4
PPObj	20.4	10.7	15.6	31.9	15.6	23.8
DatObj	20.1	11.5	15.8	25.5	14.3	19.9

Table 1: Labeled Dependency Evaluation

sitional objects, and dative objects are considered as part of labeled dependency triples consisting of the lexical head verb, the grammatical function label, and the dependent phrase bearing the grammatical function label. The internal structure of the dependent phrase is not considered.

In Tiger annotation, the head of an argument is the sister marked with the grammatical function label HD. HD labels are found with an f-score of 99% by the parser, so this evaluation mainly reflects how well the arguments in the dependency triple are identified. This evaluation uses lexical heads, so if the sister with the label HD is a phrase, then a recursive search for heads within that phrase finds the lexical head. For 5.7% of arguments in the gold standard, it is not possible to find a lexical head. Further methods could be applied to find the remaining heads heuristically, but the additional parameters this introduces for the evaluation are avoided by ignoring these cases.

The results for a labeled dependency evaluation on important grammatical function labels are shown in Table 4.3. Grammatical functions are listed in order of decreasing frequency. The results for subjects remain similar between the raised and split version, as expected, and the results for all other types of arguments improve 4-8% for the split version.

Subjects are rarely affected by the raising method because S nodes are rarely discontinuous, so it is not surprising that the results for subjects are similar for both methods. However, VPs are by far the most frequently discontinuous nodes, and since the raising method can move an object away from its head, the difference between the two conversion methods is most evident in the object relations. Data sparsity plays a role in the lower scores for the objects, since there are approximately twice as many subjects as accusative objects and twelve times as many subjects as dative objects.

5 Future Work

Further research will extend the dependency evaluation presented in this paper to include more or all of the grammatical functions. There is significant work on a dependency conversion for Negra by the Partial Parsing Project (Daum et al., 2004) that could be adapted for this purpose.

6 Conclusion

By using an improved conversion method to remove crossing branches from the Negra/Tiger corpora, it is possible to generate trees without crossing branches that can be converted back to the original format with no errors. This is a significant improvement over the previously used conversion by raising, which was not reversible and had the effect of introducing inconsistencies into the corpus. The new splitting conversion method shows a 4-8% improvement in a labeled dependency evaluation on accusative, prepositional, and dative objects.

References

Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius and George Smith, 2002. The TIGER Treebank. In *Proceedings of TLT 2002*.

Michael Daum, Kilian Foth and Wolfgang Menzel, 2004. Automatic transformation of phrase treebanks to dependency trees. In *Proceedings of LREC 2004*.

Amit Dubey, 2004. Statistical Parsing for German: Modeling Syntactic Properties and Annotation Differences. Ph.D. thesis, Universität des Saarlandes.

Amit Dubey and Frank Keller, 2003. Probabilistic Parsing Using Sister-Head Dependencies. In *Proceedings of ACL 2006*.

Sandra Kübler, 2005. How do treebank annotation schemes influence parsing results? Or how not to compare apples and oranges. In *Proceedings of RANLP 2005*.

Sandra Kübler, Erhard W. Hinrichs and Wolfgang Maier, 2006. Is it really that difficult to parse German? In *Proceedings of EMNLP 2006*.

Joakim Nivre, 2005. Pseudo-Projective Dependency Parsing. In *Proceedings of ACL 2005*.

Oliver Plaehn and Thorsten Brants, 2000. Annotate – An Efficient Interactive Annotation Tool. In *Proceedings of ANLP 2000*.

Helmut Schmid, 2000. *LoPar: Design and Implementation*. Technical report, Universität Stuttgart.