Master's Thesis

A German Treebank and Lexicon for Tree-Adjoining Grammars

submitted in partial fulfillment of the requirements
for the degree of Master of Science in Language Science and Technology

Miriam Kaeshammer

Supervisors:
Dr. Vera Demberg
Prof. Dr. Hans Uszkoreit

April 18, 2012

Abstract

In this thesis, a treebank and a lexicon which we have developed for PLTAG parsing of German are presented. PLTAG is a psycholinguistically motivated, incremental version of tree-adjoining grammar (TAG), which explicitly models upcoming structures and lexemes. The created resources are however also applicable to parsing with other variants of TAG, and hence of interest to a broad audience. The German Tiger corpus is used as the basis for our system which performs the treebank conversion to PLTAG format and the automatic extraction of the lexicon. To the best of our knowledge, this work provides the first scalable German TAG grammar.

Declaration

Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Declaration

I hereby confirm that the thesis presented here is my own work, with all assistance acknowledged.

Saarbrücken, April 18, 2012

Miriam Kaeshammer

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my adviser, Vera Demberg, for giving me the opportunity to work on this interesting topic and for providing support throughout all stages of the thesis work. Thank you for being available, for answering my questions and for the collaboration which resulted in a conference paper.

I would also like to thank Laura Kallmeyer who made me discover and like tree-adjoining grammars back then in Tübingen. Thanks for giving me the time to finish this thesis.

My thanks also go to the wonderful people that I have met during my studies, to Katerina, Prash, Andreas and Dominikus who read and commented on parts of this work. I am especially grateful to Dominikus for the encouragement, help and joint work throughout the complete course of studies and for making things more fun.

I thank my family for their endless love and the unconditional support they provided, even though I know that they are still wondering what I have studied. I thank Lukas because of staying by my side and making me happy.

Contents

Chapter 1

Introduction

Grammars play a key role in natural language modeling and processing. While traditionally they have been created by hand, the availability of annotated resources such as treebanks has rendered it possible to automatically derive wide-coverage grammars for various formalisms, for example CFG (Charniak, 1996), LTAG (Xia et al., 2000) and LFG (Cahill, 2004) to name just a few approaches. Treebank grammars furthermore have the crucial advantage of holding statistical information that is necessary to estimate the parameters of stochastic parsers. A parser assigns syntactic structure to previously unseen sentences.

The treebank and lexicon presented in this work were developed for a recent version of TAG, called "PsychoLinguistically motivated Tree-Adjoining Grammar" (PLTAG, Demberg and Keller (2008)). Recent psycholinguistic research suggests that humans process sentences in a strictly incremental fashion, integrating incoming words eagerly with the incremental analysis, and make predictions about the upcoming structure and lexemes. Those key properties are modeled within PLTAG.

PLTAG and standard lexicalized tree-adjoining grammar (LTAG) generate the same derived trees, but the PLTAG grammar is a superset of a standard LTAG grammar: in addition to the standard initial trees and auxiliary trees of an LTAG grammar, it includes unlexicalized so-called prediction trees, which are necessary in order to make explicit predictions about upcoming material in a sentence. The German PLTAG resources presented in this work are hence also useful for other variants of TAG. They are induced automatically from the Tiger Treebank (Brants et al., 2002), thereby making extensive use of the encoded linguistic knowledge. The two main steps of the procedure are (1) to convert the specific format of the Tiger Treebank to (P)LTAG format and (2) to extract canonical elementary trees as well as prediction trees.

Similar work has been done for English by Demberg-Winterfors (2010) who induced the lexicon from the Penn Treebank (PTB, (Marcus et al., 1993)). The current thesis builds upon this approach.

The created (P)LTAG resources are of interest to several fields for various reasons: First of all, to the best of our knowledge, no broad-coverage lexicalized tree-adjoining grammar (and treebank) for German is currently available. The induced lexicon together with the

converted treebank will fill this gap. They constitute *the* required resources for probabilistic TAG parsing of German. Additionally, on a more theoretical level, this work reveals and discusses issues that appear when adding the sister-adjunction operation (inserting an elementary tree as a new daughter of some node) to the PLTAG formalism.

Furthermore, PLTAG can be combined with a sentence processing theory (Demberg-Winterfors, 2010) that models human processing difficulties. The theory has already been validated for English (Demberg-Winterfors, 2010) using a PLTAG parser based on the English resources from the PTB. Evidence from other languages should follow. The theory should for example be able to account for anti-locality effects (processing of the verb is facilitated if there is intervening material between the verbal arguments and the verb itself) that have been observed in German (e.g. Konieczny and Döring, 2003).

Finally, an implemented PLTAG parser can also play a role in language technology applications in the future. Incremental analysis of input text is desirable, for example, in human-machine dialogue systems, where the processing should start before the input is complete (Schlangen and Skantze, 2009). A PLTAG parser can furthermore determine processing difficulties, which could be used to assess text readability (e.g. for e-learning) or to optimize machine-generated text.

The goal of this work is to create resources for broad-coverage PLTAG parsing of German: a lexicon and a corpus of derived trees. This thesis is therefore organized as follows: Chapter 2 serves as a general introduction to the topics covered throughout this work. Besides motivating incrementality and prediction, it introduces the (P)LTAG formalism and presents previous work on extracting TAG lexica from treebanks. Aspects of German syntax and characteristics of the Tiger Treebank are presented as well. In Chapter 3, the target grammar (i.e. the elementary trees that will be induced) is specified and various aspects of it are discussed. Chapter 4 comprises the steps that are taken to convert the Tiger Treebank trees to derived (P)LTAG trees and the procedures that are used to induce the lexica. An evaluation of the extracted grammar in size and coverage is provided in Chapter 5. Chapter 6 concludes and presents directions for future work.

The main points of this work are also published in (Kaeshammer and Demberg, 2012), which presents the German PLTAG lexicon and treebank alongside with the corresponding resources for English.

Chapter 2

Background

This chapter provides the background knowledge that is assumed for this thesis. The goal is to extract a German (P)LTAG from the Tiger Treebank. We will start with concepts centered around incremental processing and key properties of the human parser. The cognitively plausible grammar formalism PLTAG, that emerged from those concepts, and its base formalism LTAG are introduced. In what follows, previous approaches to extract tree-adjoining grammars from treebanks are presented. The last part is dedicated to the German language in particular. We review properties of German syntax as well as the limits of describing them with tree-adjoining grammars. Finally, the Tiger Treebank and its annotation is introduced.

2.1 Psycholinguistic Motivation

The theory of human sentence processing proposed by Demberg-Winterfors (2010) assumes incrementality, prediction and full connectedness, and a verification mechanism. Exactly those properties are modeled in a dedicated grammar formalism, namely PLTAG, which is presented in detail in Section 2.2.2. A PLTAG parser can thus be used to automatically derive syntactic analyses for input text which comply with these assumptions. In this section, the assumed key properties of human language processing and selected psycholinguistic studies that lead to those assumptions are introduced. For an in-depth motivation, see (Demberg-Winterfors, 2010, Section 6).

Psycholinguistic research, e.g. self-paced reading and eye-tracking studies, provide evidence that human language comprehension works *incrementally*, i.e. an interpretation is built from left to right on a word by word basis (Tanenhaus et al., 1995; Konieczny, 2000). When a word is perceived, it is integrated into the current representation of the sentence, leading to different processing difficulties depending on the word and the context. Findings by Sturt and Lombardo (2005) support an even stronger claim, namely that of *full connectedness*: all words must be connected into one syntactic structure at any point of the incremental processing. Thus, the human parser does not create unconnected tree fragments. This is referred to as *strict incrementality*. The experiments by Sturt and Lombardo (2005)

are based on a gender mismatch. They found that in a sentence like *The pilot embarrassed Mary and put herself in an awkward situation* the effect of the gender mismatch between *herself* and *the pilot* (e.g elevated reading times) occurs directly when the reflexive pronoun is read, and not at some later point in the sentence. The interpretation is that people have already established the subject as the antecedent of the reflexive pronoun. This means that the subject and the pronoun are in a certain relation in the syntactic structure: the subject c-commands the pronoun. It is thus concluded that the pronoun is eagerly incorporated into the syntactic structure of the sentence prefix.

Strict incrementality is closely related to the concept of prediction, "the process of forming expectations about upcoming input" (Demberg-Winterfors, 2010). In order to guarantee full connectedness, it can be necessary to anticipate phrases for which no direct evidence has been encountered yet, for example in form of the head word. In a visual-world experiment, participants' eye-movements revealed that they anticipated the post-verbal argument based on the semantics of the preceding verb (Kamide et al., 2003).

Results by Staub and Clifton (2006) furthermore show that predictions also occur on the lexical level. Following the word *either*, the second conjunct as well as *or* are predicted, which is revealed by facilitated processing (shorter reading times in those regions compared to the same stimuli without *either*). Furthermore, in the *either* condition, coordination on the sentence level is not misanalyzed as NP coordination. Demberg-Winterfors (2010) assumes that predicted structures have to be verified at a later step. Aspects of processing difficulty can then be explained as being incurred by the verification.

The PLTAG model of human sentence processing offers a direct incorporation of the mentioned properties. Competing analyses during processing are furthermore conceptualized as ranked parallel parsing with a finite beam that corresponds to memory restrictions. This unified framework can therefore not only explain very specific structures (such as garden path sentences etc.), but it is aimed at scaling up and accounting for a wide range of naturally occurring structures.

2.2 Tree-Adjoining Grammars

Tree-adjoining grammar (TAG) is a linguistically motivated grammar formalism introduced by Joshi et al. (1975). Its primitive units for rewriting are trees, instead of symbols as in context-free grammars (CFG). Two operations, substitution and adjunction, are employed to derive larger tree structures.

Various versions of TAG have been developed over time, one of the most widely used ones for natural language grammars being LTAG. A very recent, psycholinguistically motivated variant is PLTAG. Since grammars of both formalisms are the aim of this work, they will be presented in the following sections.

2.2.1 LTAG Formalism

Formally, a tree-adjoining grammar (Joshi and Schabes, 1997) is a quintuple $\langle T, N, I, A, S\rangle$ where

- T and N are disjoint alphabets of terminal and non-terminal symbols,
- $S \in N$ is a specific start symbol (the root of the derived tree),
- I is a finite set of *initial* trees and A a finite set of *auxiliary* trees.

Trees in $I \cup A$ are called *elementary trees*. Their interior nodes are labeled by non-terminal symbols. Exactly one node on the frontier in the trees in A is marked as the *foot*, indicated by an asterisk. The label of the foot must be the same as the label of the root. All other nodes on the frontier of elementary trees are labeled by terminals or non-terminals, where non-terminals are marked for substitution, which, by convention, is indicated by a down arrow. See Figure 2.1 for examples of elementary trees. In a *lexicalized* tree-adjoining grammar (LTAG), at least one node on the frontier of each elementary tree must have a terminal label, the lexical *anchor*. The *spine* of an elementary tree is the path from the root to its lexical anchor leaf, which usually is the linguistic head of the tree in a linguistically motivated LTAG.

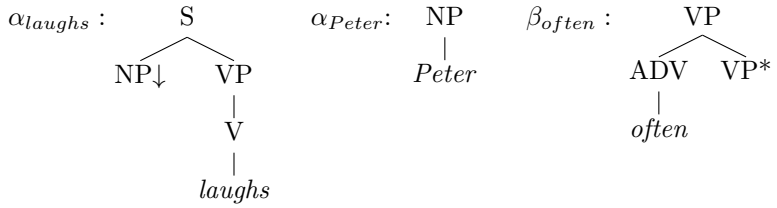


Figure 2.1: TAG elementary trees: α_{laughs} and α_{Peter} are initial trees, β_{often} is an auxiliary tree.

Elementary trees are combined with subsequent applications of the substitution and adjunction operations to build the *derived tree*. In the *substitution* operation, a substitution node n is replaced by an initial tree whose root node has the same label as n. For the *adjunction* operation, an auxiliary tree β is inserted into a tree at a node n whose label matches the label of the root node and the foot node of β. The subtree dominated by n is excised, leaving behind a copy of n, to which the auxiliary tree β is attached. Its root node is identified with the copy of n, while the excised subtree is attached to its foot node. Figure 2.2 shows how the sentence *Peter often laughs* is parsed: starting with α_{laughs}, α_{Peter} is substituted at its NP node and β_{often} is adjoined to its VP node. The derived tree may not contain any open foot or substitution nodes.

With a view to feature-based TAG (Vijay-Shanker and Joshi, 1988), the combination of elementary trees can be described as completing "half" nodes. Foot nodes as well as substitution nodes are thought of top halves, while root nodes are considered as bottom

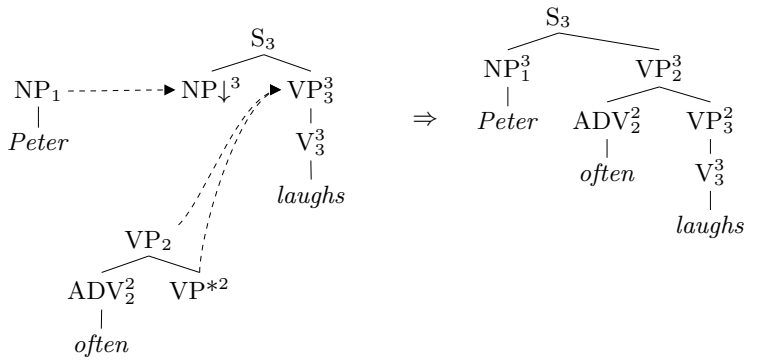


Figure 2.2: TAG derivation for *Peter often laughs* with the derived tree. Indices indicate top and bottom halves of nodes.

halves. Inner nodes consist of both a top and a bottom half. When elementary trees are combined, the corresponding half nodes are completed. The node to which an auxiliary tree is adjoined is thereby divided into its two halves. The indices in the elementary trees in Figure 2.2 represent the half nodes. The derived tree demonstrates how the VP is split due to the adjunction operation and how the half nodes are completed.

2.2.2 PsychoLinguistically Motivated TAG

Standard LTAG with typical linguistically motivated elementary trees, as for example in the XTAG lexicon for English (XTAG Research Group, 2001), can often not be used to derive a sentence strictly incrementally. For instance, when processing the sentence *Peter often ...* using the grammar in Figure 2.1, *often* and *Peter* cannot be combined into the same tree structure because the VP node necessary for adjunction of β_{often} has not been generated yet. In the most incremental LTAG derivation that can be obtained ($\alpha_{Peter} \rightarrow \alpha_{laughs} \rightarrow \beta_{often}$, see Figure 2.2 for the derivation), the word *laughs* is not combined in the incremental order (which is $\alpha_{Peter} \dashrightarrow \alpha_{often} \dashrightarrow \beta_{laughs}$). Since incrementality has been shown to be a key property of human sentence processing (Section 2.1), Demberg and Keller (2008) propose a psycholinguistically motivated variant of LTAG which supports strictly incremental derivations and models predictions.

PLTAG Formalism

PsychoLinguistically motivated tree-adjoining grammar (PLTAG, Demberg-Winterfors (2010)) extends the inventory of elementary trees of standard LTAG (henceforth called *canonical* elementary trees) with a *prediction lexicon*. It specifies so-called *prediction trees* (initial or auxiliary), which are usually unlexicalized and whose nodes are marked as predicted by prediction markers (a superscript and/or subscript, see the tree in Figure 2.3(b) as an example). Similarly to the features in feature-based TAG (Vijay-Shanker and Joshi, 1988),

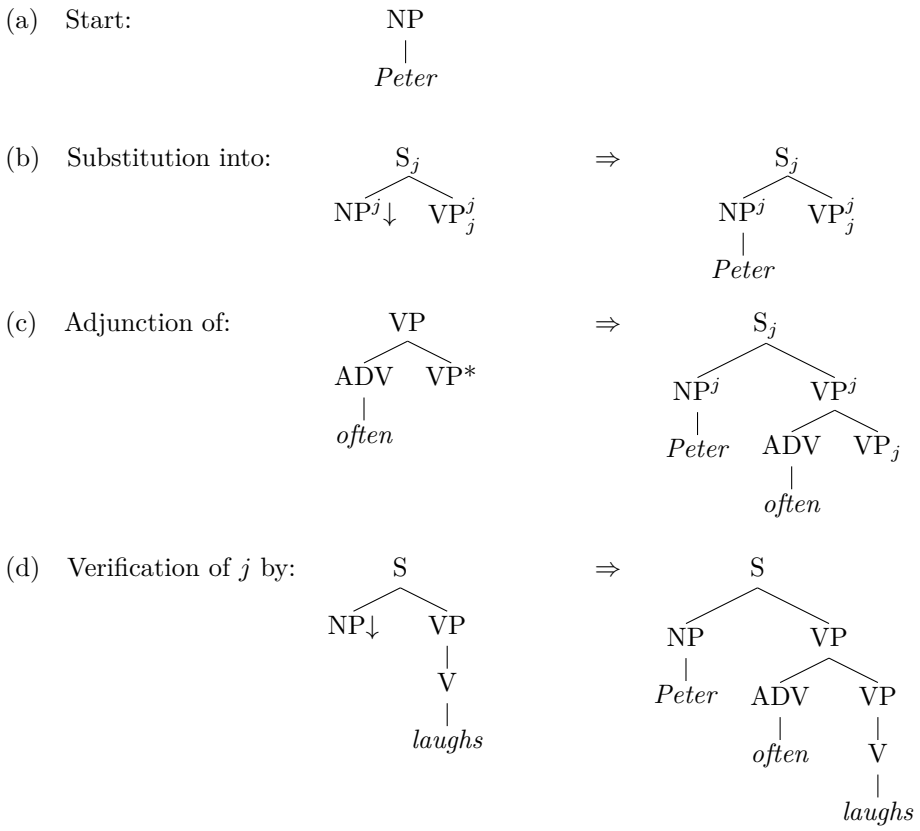


Figure 2.3: PLTAG derivation for *Peter often laughs*

substitution and foot nodes only have superscripts, and root nodes only have subscripts, while other nodes have both. This is because root, foot and substitution nodes are conceptualized as half nodes which are completed when elementary trees combine with adjunction or substitution, as explained in the previous section for LTAG.

The standard TAG operations, substitution and adjunction, are applied to the prediction trees of PLTAG was well. Additionally, PLTAG has a *verification* operation with which nodes that have been previously introduced via the integration of a prediction tree are validated, i.e. their prediction markers are removed. The *verification tree*, the canonical elementary tree that is used for the verification, has to "match" all predicted nodes of a unique, identical index: the predicted tree and the verification tree must have the same shape, meaning having all nodes in the same order. Additional nodes in the verification tree can only be at the bottom of the spine or to the right of the spine. This characteristic is necessary for incremental parsing. In Figure 2.3(d) a sample verification operation is shown (index j). Note the correspondence between the verification tree (*laughs*) and the prediction tree in (b).

A derivation in PLTAG always starts with the tree that corresponds to the first input word and then a sequence of substitution, adjunction and verification operations is applied. A new elementary tree can either be integrated into the partially derived tree or the

partially derived tree can be integrated into the new elementary tree. If a prediction tree is used during the derivation, it will have to be validated with a verification tree such that substitution or adjunction of a prediction tree and the verification operation always occur pairwise.

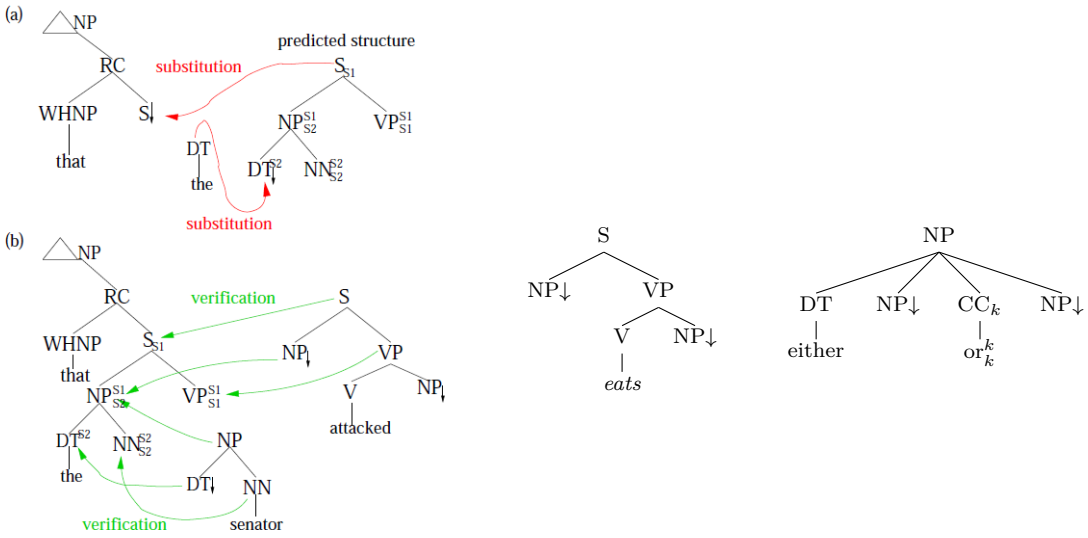
To ensure incrementality, in a partial PLTAG derivation for words $w_1 \dots w_i$, there are only terminal (i.e. lexicalized) nodes on the frontier to the left of the rightmost anchor. This means that no nodes with prediction markers or open substitution nodes are allowed on the frontier before word w_i. In case the partially derived tree is an auxiliary tree, the foot has to be right of w_i. The final derived tree may not contain any nodes which carry a prediction marker. All nodes on the frontier have to be terminals and the root label must be S. Figure 2.3 provides the incremental PLTAG derivation for *Peter often laughs*.

Intuitively, LTAG and PLTAG derivations directly correspond to each other. For each elementary tree which is out of the incremental order in an LTAG derivation (e.g. α_{laughs} in Figure 2.2), a corresponding prediction tree is used in PLTAG (step (b) in Figure 2.3). The original elementary tree is "moved" to the correct incremental position of its anchor, where it will validate the prediction tree with the verification operation (step (d)). Going the other way round from PLTAG to LTAG, each prediction tree used in a PLTAG derivation can be replaced with its verification tree. Demberg-Winterfors (2010) shows that, given an adequate PLTAG prediction lexicon, for each LTAG grammar, there exists a PLTAG grammar such that the derived LTAG trees are identical to the derived PLTAG trees.

Predictions in PLTAG

As Demberg and Keller (2008) point out, PLTAG incorporates several mechanisms to implement predictions about upcoming structures and lexemes. First of all, predictions are needed to ensure fully connected structures when parsing a sentence incrementally, as indicated at the beginning of this section. This happens for example if two dependents precede a head or if a grandparent and a child have been encountered, but the head of the intervening parent node has not been encountered yet. An example for the second case is presented in Figure 2.4(a): the determiner in the object relative clause in a sentence like *The reporter [that <u>the</u> senator attacked] admitted the error* cannot be integrated with the previously built tree without predicting the structures anchored by *senator* and *attacked*. Such predictions due to connectivity are realized in PLTAG with the prediction trees. Prediction trees may be pre-combined in order to make parsing more efficient. In Figure 2.4(a), the prediction tree is a pre-combined one (indices $s1$ and $s2$). Node halves with index $s1$ are validated by the elementary tree of *attacked*, while the tree of *senator* validates node halves with index $s2$.

Even though the PLTAG formalism does not impose any constraints on the shape or size of the prediction trees, it of course only makes sense to include those which are the same or smaller than some canonical elementary tree. Otherwise those prediction trees can



(a) Predictions to ensure connectivity and the corresponding verification operation; figure taken from (Demberg-Winterfors, 2010, p. 162)

(b) Prediction via an open substitution node on the right of the anchor

(c) Prediction via a multi-anchored tree

Figure 2.4: PLTAG predictions

never be validated by the verification mechanism.

Since the optimal granularity of predictions and the desired level of generalization is still an open question, the issue of the exact size of the prediction trees is not easy to solve. Demberg-Winterfors (2010) decides for minimal predictions that only predict upcoming structures as far as needed for full connectivity or subcategorization. Prediction trees are therefore defined as having the same shape as the canonical elementary trees, except that they do not have nodes to the right of the spine, and that the spine does not end in a terminal node. They furthermore do not have unary nodes at the bottom of the spine. For auxiliary trees, the foot node and nodes between the foot and the root of the canonical tree are also included in the prediction tree.

In addition to the prediction trees, predictions also stem from the lexicon entries themselves thanks to TAG's *extended domain of locality*. This refers to the fact that elementary trees in TAG can be of arbitrary size. They are consequently larger than corresponding context-free rules, which only describe trees of depth one. In LTAGs for natural languages, this domain of locality is used to describe co-occurrence relationships and dependencies between nodes that are further apart (see also Section 3.1). For example, subcategorization frames of a lexical item are encoded through substitution nodes in its elementary tree. Accordingly, open substitution nodes to the right of the lexical anchor function as predictions. Consider for instance the elementary tree of a transitive verb in Figure 2.4(b). It predicts an upcoming object via the right NP substitution node.

Multi-anchored trees constitute a source for predictions on the lexical level, as for

example in strong collocations like *either... or* and idioms. Since the first word, *either*, is predictive for the other word, *or*, the corresponding lexical entry looks like depicted in Figure 2.4(c). With its help, the facilitated processing of an *either... or* disjunction compared to the simple *or* disjunction can be explained: when processing the sequence *Peter eats either an apple or*, the *or* has already been predicted at the word *either*. It therefore has lower integration costs than the unexpected *or* in *Peter eats the apple or*. Furthermore, there is also an attachment ambiguity in the latter case (*or* can attach to the NP or S node depending on the second conjunct), but not in the first.

Modeling Human Processing Difficulties

To give the complete picture, it should also be mentioned here that the PLTAG formalism can be combined with a sentence processing theory (Demberg-Winterfors, 2010). It builds on top of a PLTAG parser which performs ranked parallel processing. The processing difficulty D at a word w_i has two components. A surprisal component corresponds to the difficulty of integrating w_i with all previous structures and can be calculated from the probabilities of the elementary trees. A memory decay component models the difficulty of retrieving previously predicted structures from memory. It is based on the probabilities of prediction trees, time stamps of the predicted nodes and a decay factor.

2.3 Previous Work: Creating Natural Language Tree-Adjoining Grammars

LTAGs for a large number of languages have been created. Traditionally, they have been hand-crafted, so building them was a labor-intensive and time-consuming task. The two existing LTAGs that have a grammatical coverage of interest are for English (XTAG Research Group, 2001) and French (Abeillé et al., 1999). For German, no manually built, broad-coverage tree-adjoining grammar is available to the best of our knowledge. In fact, it has been shown that certain phenomena, e.g. long-distance scrambling, cannot be described adequately with standard TAG (Becker et al., 1991), see also Section 2.4.2. Theoretical work has then concentrated on "interesting" phenomena that go beyond TAG and how they can be modeled within TAG extensions, e.g. V-TAG (Rambow, 1994), MCTAG (Kallmeyer and Yoon, 2004) or TT-MCTAG (Lichte, 2007; Lichte and Kallmeyer, 2008). Implementations therefore cover only fragments of the German grammar (e.g. GerTT[1], a grammar fragment of German in TT-MCTAG). The same holds for DTAG (Gerdes, 2002) which was not developed for parsing, but for generation, and can therefore exclude certain word order variations and phenomena without further complications.

The availability of annotated resources such as treebanks has changed the picture, since it has become possible to automatically derive wide-coverage grammars for various languages

[1] `http://www.sfs.uni-tuebingen.de/emmy/res-en.html`

and formalisms, amongst others LTAG. Besides the economical benefits, treebank grammars have the crucial advantage over hand-crafted grammars of holding statistical information which is required to train the parameters of stochastic parsers.

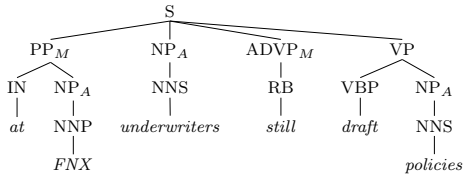
In what follows, we present previous work about extraction of tree-adjoining grammars for English from the Penn Treebank (PTB, Marcus et al. (1993)) which builds the ground for our work. Even for German, there exist two approaches for LTAG extraction from a treebank. It will be argued that those grammars could not be "recycled" to build a German PLTAG.

2.3.1 LTAG Extraction from PTB

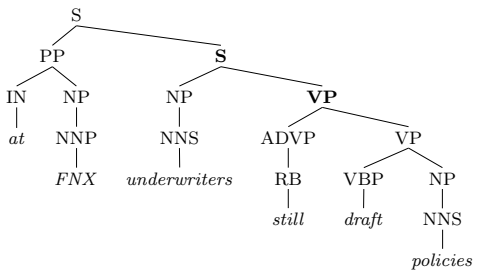
As so often, English with the PTB has been the pioneer language for extracting a TAG from a treebank. Various, slightly different approaches exist, of which the most relevant ones for this work will be presented. The overall idea is (a) to consider the treebank trees as derived trees, or to first convert them to derived trees, and (b) to use linguistic knowledge and heuristics to reconstruct the derivation, i.e. to identify elementary trees.

Xia's Approach (Xia et al., 2000; Xia, 2001)

In Xia's work, the flat structures of the treebank trees are first converted to derived trees, in which arguments and modifiers attach to different levels. The so-called *fully bracketed trees* are obtained by inserting the necessary inner nodes. For illustration, see Figure 2.5, especially the boldface nodes in Figure 2.5(b). This step is necessary in order to be able to encode modifiers as auxiliary trees in the following extraction procedure.



(a) Original PTB tree with arguments (A) and modifiers (M) marked. The remaining nodes are heads.



(b) Converted (= derived) tree

Figure 2.5: Xia's approach: PTB and derived tree; example inspired by Xia et al. (2000)

.

It thus first needs to be determined for each node, which of the three it is: (a) the head child of its parent, (b) an argument of its parent's head or (c) a modifier of its parent's head. This distinction is made by using treebank-specific heuristics which are stored in three tables: a head percolation table in the style of Magerman (1994), an argument table

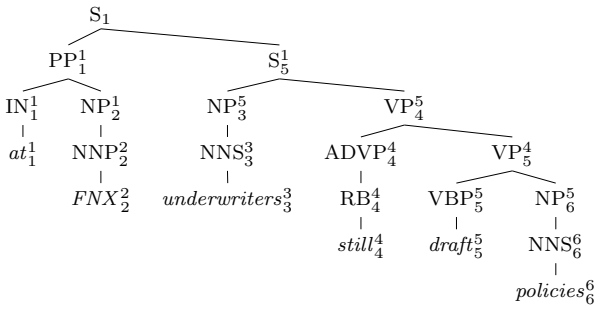


Figure 2.6: Xia's approach: the same tree as in Figure 2.5(b), but the nodes are indexed (top and bottom halves) according to which elementary tree they belong to.

that specifies which categories can serve as an argument of the head and a tagset table that decides which functional labels always indicate a head/argument/modifier. Note that such generalized heuristics might have problems providing correct results, depending on the granularity of annotation of the treebank. For example, it is difficult to classify a PP dominated by a VP as either argument or modifier if such a distinction is not provided in the treebank.

The extraction algorithm finds the spine of an elementary tree by following the head path from the root down to the leaf, which becomes the lexical anchor. If a sibling of a head child *hd* is a modifier, the sibling is factored out of the current elementary tree, the "host" tree, as an auxiliary tree where *hd* is the foot node and *hd*'s parent the root node. In the current elementary tree, those two nodes consequently appear collapsed. As an example, consider the ADVP node in Figure 2.5. It is a modifier sibling of the lower VP, which is a head. Figure 2.7(d) shows the extracted auxiliary tree. The initial tree in Figure 2.7(e) contains the VP node from which the auxiliary tree has been excised.

Since the extraction is the reverse process of parsing, the procedure can be described with the help of "cutting nodes into halves": *hd*'s parent (e.g. VP_4^5 in Figure 2.6) is cut in half, and the lower half constitutes the auxiliary tree's root node. *hd* (VP_5^4) is also cut in half. The upper part becomes the foot node of the new auxiliary tree, whereas the bottom part is completed to a full node in the host tree by *hd*'s parent's upper half. For illustration, see the nodes with index 4, especially the two VP nodes, in Figure 2.6.

If a sibling of *hd* is an argument, the argument node itself is cut into two halves: the top half constitutes a substitution site, whereas the lower half becomes the root of a new initial tree, see for example the node NP_3^5 in Figure 2.6. Coordination relations are factored into auxiliary trees. The algorithm is recursively applied to all extracted subtrees. The elementary trees extracted from the converted example tree are shown in Figure 2.7. Xia et al. (2000) extract 6926 tree templates[2] from the PTB.

The grammar extracted by Xia (2001) however does not include *predicative auxiliary*

[2]A template is an elementary tree without lexical anchor.

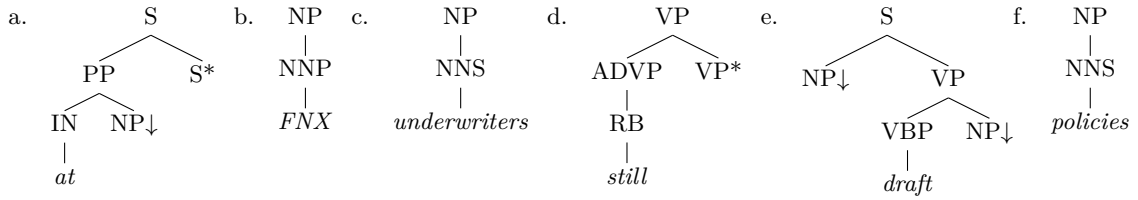


Figure 2.7: Xia's approach: elementary trees extracted from the tree in Figure 2.5(b); see Figure 2.6 for the correspondence. Example taken from (Xia et al., 2000).

trees, which are auxiliary trees where the foot node is an argument of the lexical anchor. Xia (2001) argues that they are too difficult to detect given the PTB annotation. Predicative auxiliary trees are usually used (for example within the XTAG project) to encode verbs like *believe* which subcategorize for a sentence. In contrast to the following approaches, each trace with its filler[3] is extracted as a tree set if they are not located within the same elementary tree.

Chen's Approach (Chen, 2001; Chen and Vijay-Shanker, 2004)

Even though Chen's work was conducted independently, it is very similar to Xia's approach. A head percolation table is used as well to identify head children. The head path provides the spine of the elementary tree. The procedure also distinguishes between arguments and modifiers, though in a slightly different way, but also based on the annotation of the PTB. Predicative auxiliary trees are extracted to localize long distance movement.

However, no conversion except some merging of node labels takes place prior to the extraction of elementary trees. Instead the additional nodes to encode a modifier as an auxiliary tree are generated "on the fly". For example, following Chen, the elementary trees in Figure 2.7 would be generated directly from 2.5(a). As a result, no derived trees are immediately available.

LTIG with sister-adjunction (Chiang, 2000, 2004)

Chiang induces a *lexicalized tree insertion grammar* (LTIG) from the PTB. LTIG is a restricted variant of LTAG in which no *wrapping* auxiliary trees nor an adjunction operation which creates such trees are allowed. A wrapping auxiliary tree has non-empty leaf nodes on *both* sides of the foot node. Like CFG, TIG is $O(n^3)$-time parsable, which stands in contrast to TAG (parsing complexity of $O(n^6)$).

Chiang's grammar is directly extracted from the PTB without introducing new nodes at any step. This is possible because an additional composition operation, *sister-adjunction*, is added to the standard definition of TAG. When an elementary tree α is sister-adjoined under a node n at position i, the root node of α becomes a new daughter of n. α is added

[3]Trace and filler are annotated as a pair of co-indexed nodes in the PTB.

between the ith and $i+1$th child of n (assuming an imaginary child to the left of the leftmost child and one to the right of the rightmost child). For example, if $i = 0$, α is added as the new leftmost daughter, as in the example in Figure 2.8. Chiang (2004) considers the set of initial trees to be eligible for being sister-adjoined. It is important to note that sister-adjunction, as it is defined, is not restricted at all. Only the probability model of a parser will constrain the sister-adjunction operation (Chiang, 2000).

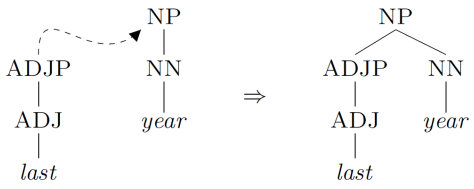


Figure 2.8: Sister-adjunction (according to Chiang) at position $i = 0$

Very similarly to the previous approaches, Chiang's extraction procedure determines the head child of each node using a head percolation table and classifies the remaining children as arguments or modifiers depending on their phrasal label and function tag. As commonly done, arguments are encoded as initial trees that leave behind a substitution node. Modifiers, however, are excised as sister-adjoining (initial) trees, see Figure 2.9(a) and (d). Recursive structures such as auxiliary and modal VPs and coordinating structures are identified and form auxiliary trees. Thus, due to encoding modifiers as sister-adjoining trees instead of auxiliary trees, which deviates from the XTAG analysis, the treebank trees constitute LTIG derived trees. Chiang's (2000) grammar is furthermore very compact (only about ∼2100 elementary tree templates), which is due to the fact that the trees used for modification are initial trees.

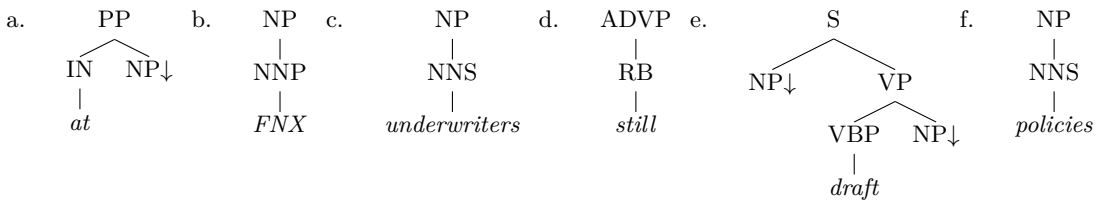


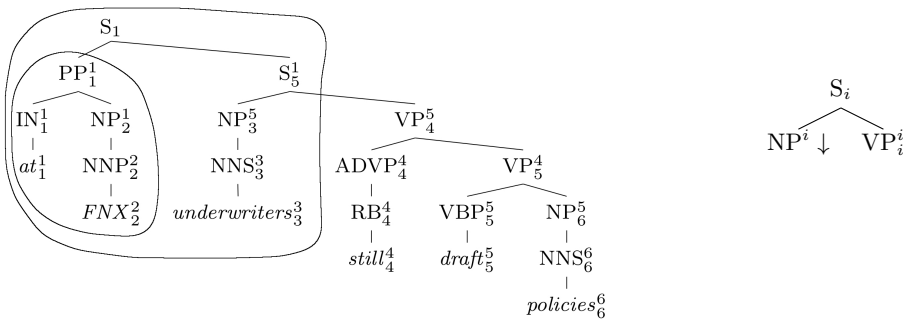
Figure 2.9: Chiang's approach: elementary trees extracted from the tree in Figure 2.5(a), assuming that the algorithm makes the same head-argument-modifier distinction.

2.3.2 PLTAG Extraction from PTB

Since a PLTAG lexicon is a superset of a corresponding LTAG lexicon, Demberg-Winterfors (2010) also extracts an LTAG lexicon from the PTB. The head-driven procedures in (Xia et al., 2000) are followed and the output are canonical elementary trees and a (P)LTAG

treebank. Demberg's grammar and treebank still differ from Xia's resources in that Demberg-Winterfors (2010) adds the linguistically motivated NP disambiguation from Vadas and Curran (2007) (as opposed to heuristically annotated NPs) and subcategorization information from PropBank (Palmer et al., 2005). Furthermore, auxiliaries and copula verbs are identified and re-annotated with a specific part-of-speech (POS) tag, which allows to specifically treat them during the extraction procedure. It turns out that the extracted grammar does not contain wrapping auxiliary trees, so in fact an LTIG lexicon (without sister-adjunction) is extracted.

In addition, Demberg-Winterfors (2010) also induces the PLTAG prediction lexicon. Due to reasons presented in Section 2.2.2, the desired prediction trees only include the tree structure left of the spine of a canonical elementary tree, including the spine itself, except unary nodes at the bottom of the spine. Transforming all canonical elementary trees to corresponding prediction trees would result in a large lexicon and thus a big search space during parsing. The prediction lexicon therefore only includes prediction trees which are necessary in order to derive the trees in the PLTAG treebank strictly incrementally. Going back to the example sentence from the previous section, reproduced here as Figure 2.10(a), a prediction tree corresponding to the elementary tree of *draft* (Figure 2.7(e)) is required, before being able to integrate the elementary tree for *underwriters* into the partially derived tree of *at FNX*.



(a) Connection paths for the first three words. Nodes are numbered according to the canonical tree to which they belong.

(b) Extracted prediction tree

Figure 2.10: Creating the prediction lexicon

In order to systematically identify the tree structures that are required for an incremental derivation of a treebank tree, the connection path (Lombardo and Sturt, 2002) for each prefix is computed. A connection path for words $w_1 \dots w_n$ is the minimal amount of structure that is needed to connect the words $w_1 \dots w_n$ into the same syntactic tree. This amount of structure is indicated with circles in Figure 2.10(a). The structure which is required by the connection path of words $w_1 \dots w_n$ but which is not part of the elementary trees that are anchored in words $w_1 \dots w_n$ leads to a prediction tree. In Figure 2.10(a), this happens at the

word *underwriters*: the S node and the NP node of the elementary tree with index 5 (*draft*) have to be predicted in order to connect all seen words. According to the specification of the shape of prediction trees currently used in (Demberg-Winterfors, 2010) as given above, the VP node (as part of the spine) is also included in the prediction tree. Nodes in the prediction tree are marked as predicted.

It can happen that nodes from more than one unseen canonical elementary tree are required by the connection path. This is for example the case in Figure 2.4(a). Demberg-Winterfors (2010) creates a pre-combined prediction tree in those cases, in which the nodes have unique markers according to the elementary tree from which they stem. The advantage is that the parsing algorithm can be restricted to the use of only one, possibly pre-combined, prediction tree in a row, which makes the search space considerably smaller. The consequence is that only previously observed combinations of prediction trees are possible during parsing. This means that not all, possibly indefinitely recursive, phenomena of language can be analyzed. A slightly lower coverage has to be accepted with such a compromise. However, it was found that in the PTB data at most five prediction trees were needed in order to achieve full connectivity, and this happened only once. In most cases ($\sim 90\%$) only nodes from one lexical anchor were required.

Demberg-Winterfors (2010) extracts 7100 tree templates from the PTB (Sections 02–21), which is roughly the same order of magnitude as the LTAG lexica by Xia (2001) and Chen (2001). The induced prediction lexicon contains 2800 trees. On the converted treebank trees from Section 23 of the PTB, a coverage of $> 90\%$ is achieved.

2.3.3 German LTAG Extraction from Negra

Neumann (2003) as well as Frank (2001) extract a tree grammar from the German Negra treebank (Skut et al., 1997). It is the predecessor of the Tiger Treebank, which has a slightly extended annotation scheme (Brants and Hansen, 2002). (See Section 2.4.3 for more information about the Tiger annotation.) We found that the resources to reproduce Frank's (2001) work are not available and that Neumann's (2003) grammar lacks linguistic motivation. We therefore expect that the latter does not generalize well to unseen data and that parsing will be problematic because of the size of the grammar.

Neumann (2003)

Neumann (2003) obtains stochastic lexicalized tree grammars from (untransformed) German and English treebanks. As Xia, Chen and Chiang, Neumann also uses a recursive, head-driven extraction procedure and factors non-head children into separate trees. In the German Negra treebank, heads are marked by functional labels, so no heuristic head percolation table is needed. Leaf nodes that are sisters of a lexical anchor are considered co-anchors.

The most severe shortcoming is the treatment of modification. Because of the flatness of the PTB trees, it seems to be difficult to distinguish between modifiers and arguments,

so Neumann (2003) treats them equally in the PTB grammar. In the Negra trees, modifiers can be identified via their edge label. But, again due to the flat annotation structures, modification cannot easily be factored out in auxiliary trees.[4] Instead, the extraction procedure produces copies of the respective elementary trees from which modifiers have been recursively deleted. This should improve generalization of the grammar since it can analyze phrases with less modifiers than those which have been encountered already.

It is clear, however, that such grammars grow very large in size, which is problematic for parsing, and that their generalization capacity is very limited. Indeed, Neumann (2003) extracts 12k tree templates from Sections 02–04 of the PTB as opposed to the roughly 6k tree templates extracted by Xia et al. (2000) and Chen and Vijay-Shanker (2004) from Sections 02–21, and 10k tree templates from a portion of the Negra corpus of the same size as the PTB data (only 4270 sentences).

Frank (2001)

In contrast to Neumann (2003), Frank (2001) first heavily restructures the German Negra treebank to be able to extract a linguistically sound LTAG lexicon that generalizes well to unseen data. The approach taken is also different from all previously presented work in that no recursive, head-driven extraction procedure is used.

As a very first step, Frank (2001) compiles the treebank structures into a Prolog-like constraint language. In this representation, trees can be described with predicates: for example `arc(A, LA, B, LB)` matches (a partial configuration of) a tree in which node `B` with label `LB` is the daughter of node `A` with label `LA`. Afterwards, a cascade of fine-grained conversion rules, in which application constraints and actions are specified in terms of predicates, rewrites the trees. Elementary trees are finally obtained by applying fragmentation rules of the same format as the conversion rules to the converted tree representations. They use the linguistic knowledge contained in the annotations (phrasal and functional categories) as well as external linguistic information and heuristics such as the German sentence topology (cf. Section 2.4.1).

Even though Frank's (2001) approach and the presented elementary trees seem to be appealing with respect to generalization and linguistic adequacy, following the same strategy turns out to be difficult. Unfortunately, neither the conversion and extraction rules nor the lexicon itself are available as resources. Given the information in the paper that presents the approach (Frank, 2001), it is not straightforward to reproduce her work.

From 10k Negra treebank trees, 113,500 lexicalized elementary trees are extracted, but according to the author this result is only preliminary. The grammar size, however, cannot be compared to other TAG lexica, since the overall number of extracted tree templates is not reported. It is reported that 75% of the elementary trees coincide with one of the 2,155

[4] See however the solutions presented in Section 2.3.1 that have been found for English which has the same issue of flat structures.

pre-defined tree templates. However, nothing is known about the number of tree templates of the remaining 25%.

2.4 German Syntax and the Tiger Treebank

In this section, the properties of German syntax that complicate grammar design and parsing with respect to English will be introduced. We will furthermore briefly review previous findings concerning modeling German grammar with TAG. Since our grammar will be extracted from a treebank, the specific treebank annotation scheme strongly influences the extraction procedure and the amount of conversion that is necessary to obtain a linguistically motivated tree-adjoining grammar with good generalization to unseen data.

2.4.1 Properties of German

In contrast to the more configurational word order of English, the German word order is *semi-free*. Some components, like the verb placement, are largely fixed, whereas others can permute almost freely.

In German, the sentence type determines the position of the finite verb. Non-embedded, declarative clauses are verb-second, as in (2.1). Imperatives and yes-no questions are verb-initial, as in (2.2). In subordinate clauses, the finite verb occupies the final position, see (2.3). A cluster of non-finite verbs (*gegessen*) is placed at the end of the clause.

(2.1) *Peter* *hat* *gestern* *einen* *Apfel* *gegessen.*
Peter has yesterday an apple eaten
'Peter ate an apple yesterday.'

(2.2) *Hat* *Peter* *gestern* *einen* *Apfel* *gegessen?*
Has Peter yesterday an apple eaten
'Did Peter eat an apple yesterday?'

(2.3) *dass* *Peter* *gestern* *einen* *Apfel* *gegessen* *hat.*
that Peter yesterday an apple eaten has
'... that Peter ate an apple yesterday.'

In the German sentence structure the verbal elements frame arguments and adjuncts. This is often formulated within the *topological field model* (e.g. Höhle, 1986). The verbal elements fill the left and right *Satzklammer* (sentence brackets), which split the sentence into three fields: the *Vorfeld* (initial field, *VF*), which contains exactly one constituent, the *Mittelfeld* (middle field, *MF*) and the *Nachfeld* (final field, *NF*). The left Satzklammer (*LK*) is occupied by the finite verb or, in verb-final clauses, by the complementizer, while we find the verbal complex (a verbal particle or one or several verbs) in the right Satzklammer (*RK*). Example (2.4) shows how the sentence from (2.1) would be partitioned into topological fields.

(2.4) [$_{VF}$ Peter] [$_{LK}$ hat] [$_{MF}$ gestern einen Apfel] [$_{RK}$ gegessen]

Apart from the fixed verb positions, the order of complements and adjuncts is flexible in German. For example, each of the two arguments or the adjunct in (2.1) can be realized in the Vorfeld, see (2.5). Furthermore, the remaining two constituents in the Mittelfeld could also swap, a phenomenon called *local scrambling*[5]. This yields six different linearizations for two arguments and an adjunct. Several syntactic and non-syntactic factors like pronominalization or pragmatic constraints influence the preference for one of the possible word orders.

(2.5) a. Einen Apfel hat Peter gestern gegessen.

b. Gestern hat Peter einen Apfel gegessen.

Objects of an embedded verb can even be moved into the matrix clause (Rambow, 1994), as shown in (2.6). The term *long-distance scrambling* has been coined for this phenomenon.

(2.6) *dass [den Kühlschrank]$_1$ niemand [zu reparieren]$_1$ versprochen hat.*
that the fridge nobody to repair promised has
'... that nobody promised to repair the fridge.'

Such *discontinuous constituents* and their treatment pose a challenge to grammar writing and parsing. Example (2.7) shows a relative clause which has been extraposed to the Nachfeld. This phenomenon is more frequent than in English (Gamon et al., 2002), in which clausal constituents (relative, infinitival and complement clauses) can also be moved to the right periphery.

(2.7) *Peter hat gestern [einen Apfel]$_i$ gegessen, [der faul war]$_i$.*
Peter has yesterday an apple eaten, which rotten was
'Peter ate an apple yesterday which was rotten.'

Another difference to English, related to the freer word order, is the richer morphology. Most notably, German noun phrases are inflected for case. Subjects have nominative case, whereas the direct object is marked for accusative and the indirect object for dative. The big morphological variation means that there are many low-frequency word forms in any data, which has to be taken into account when training a stochastic parser and should also influence the form of the extracted grammar.

2.4.2 German and TAG

It has been shown by Becker et al. (1991) that lexicalized TAGs cannot describe all occurrences of long-distance scrambling adequately. This means that the predicate-argument co-occurrence principle (Section 3.1) will not always be obeyed in such an analysis.

While a sequence like NP_1 NP_2 V_1 V_2 corresponding to the ordering in sentence (2.6), leaving aside the complementizer *dass*, can still be represented satisfactorily using for

[5]The term *scrambling* will be used in this thesis descriptively, without implying a certain linguistic theory of movement.

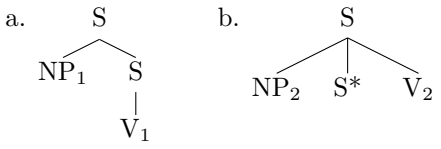


Figure 2.11: Schematic elementary trees that can describe some amount of long-distance scrambling. With reference to the sentence in (2.6), (a) would correspond to the elementary tree of *zu reparieren*, while the wrapping auxiliary tree (b) corresponds to *versprochen*. (b) adjoins to the lower S node in (a) to derive NP_1 NP_2 V_1 V_2.

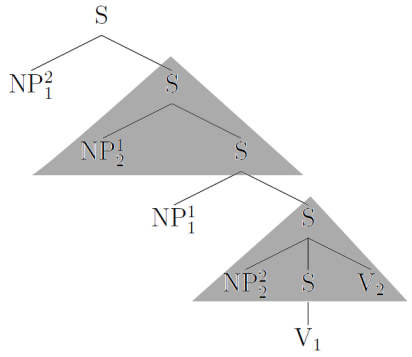


Figure 2.12: Derived tree with the yield NP_1^2 NP_2^1 NP_1^1 NP_2^2 V_1 V_2

example elementary trees like in Figure 2.11, the sequence NP_1^2 NP_2^1 NP_1^1 NP_2^2 V_1 V_2[6] corresponding to the sentence (2.8) (taken from Becker et al. (1991)) cannot. The desired derived tree might look like in Figure 2.12. We assume that the elementary tree anchored in V_2 features substitution nodes for both its arguments and that corresponding subtrees have been substituted. If it adjoins into the elementary tree of V_1, the yield[7] of the first (shaded nodes) would have to be divided into three parts (NP_2^1, NP_2^2 and V_2), as can be seen in Figure 2.12. This is not possible with TAG where the yield of an auxiliary tree can at most be divided into two parts, as it happens for example for tree (b) in Figure 2.11.

(2.8) *dass* that *[des* the *Verbrechens]*$_1$ crime *der* the *Detektiv* detective *[den* the *Verdächtigen]*$_1$ suspect *dem* the *Klienten* client *[zu* to *überführen]*$_1$ indict *versprochen* promised *hat* has

'. . . that the detective has promised the client to indict the suspect of the crime'

Becker et al. (1991) suggest to represent the two parts of the elementary tree anchored in V_2 as a set of trees (multi-component TAG or MCTAG). However, a non-local MCTAG is required in order to describe all scrambling phenomena, i.e. the nodes to which the members

[6] Superscripts are used to distinguish between the arguments of one verb, while subscripts indicate the predicate-argument dependency. Neither is part of the actual alphabet.

[7] The yield of a node is the concatenation of the terminal labels that it dominates from left to right. The yield of a tree is the yield of its root node.

of a tree set adjoin are not required to be part of one elementary tree or set. This would be too powerful and is assumed to be not polynomially parsable.

In contrast, MCTAG with shared nodes (SN-MCTAG, Kallmeyer and Yoon (2004)) stays tree-local, while relaxing the notion of tree-locality: after adjunction at a node n, n is "shared" between the adjoining tree and the host tree and therefore considered to be part of both. Typically, a scrambled constituent in SN-MCTAG is encoded together with its trace, which substitutes into the base position, in one set. Consequently, SN-MCTAG is only usable if a linguistic theory assuming a base word order and movement is adopted. Alternatively, Lichte (2007) suggests SN-MCTAG with tree tuples (TT-MCTAG). Elementary structures in TT-MCTAG are tuples of the form $\langle \gamma, \{\beta_1 \dots \beta_n\} \rangle$, where γ is a lexicalized elementary tree which is understood as the linguistic head, while the β-trees are auxiliary trees that correspond to the subcategorization frame of the head. During the derivation, the arguments $\beta_1 \dots \beta_n$ must adjoin to γ, or they must be linked to γ by a chain of root adjunctions. In contrast to pure SN-MCTAG, the trees of the set of the tuple do not have to be adjoined simultaneously. If TT-MCTAG is restricted to at most k pending β-trees, it has been shown to be polynomially parsable (Kallmeyer and Parmentier, 2008). The tree-adjoining grammar presented in the current work will not follow any of these approaches, as explained in Sections 3.2.1 and 3.3.4.

2.4.3 Tiger Treebank Annotation

The Tiger Treebank (Brants et al., 2002) contains more than 50,000 sentences ($\sim$ 900,000 tokens) taken from German newspaper texts that have been annotated with phrase structures and dependency information. The annotation scheme was designed to be theory-independent. The constituents of a sentence are represented within a graph whose leaf nodes carry part-of-speech (POS) tags, morphological information and lemmata, and whose inner nodes are labeled with phrasal categories. Its directed edges express syntactic functions.[8] See Table A.1 and Table A.2 for the complete inventory of phrasal and functional labels, and Table A.3 for the slightly modified version of the STTS tagset that has been used (Albert et al., 2003). Figure 2.13 shows a typical graph from Tiger for the sentence in (2.9).

Different from the PTB, in Tiger, *crossing branches* are employed to express long-distance relationships, which occur amongst others due to the semi-free word order and the discontinuous constituents presented in Section 2.4.1. More specifically, they handle extraposition of relative, comparative and complement clauses and appositions, object topicalization, insertions and repeated elements. Almost 30% of all graphs contain at least one crossing branch. In Figure 2.13, the VP constituent is discontinuous because of its topicalized object (*OA*).

Since context-free grammars cannot express the non-local dependencies of discontinuous constituents, tree conversions are necessary before training a (PCFG) parser on the Tiger

[8] We will write X-*Y* for a node of phrasal category X and function *Y*.

(2.9) *Versprechungen auf künftige Loyalität kann jeder Wind leicht umblasen.*
Promises about future loyalty can every wind easily blow down.
'Promises about future loyalty can be easily blown away with the wind.'

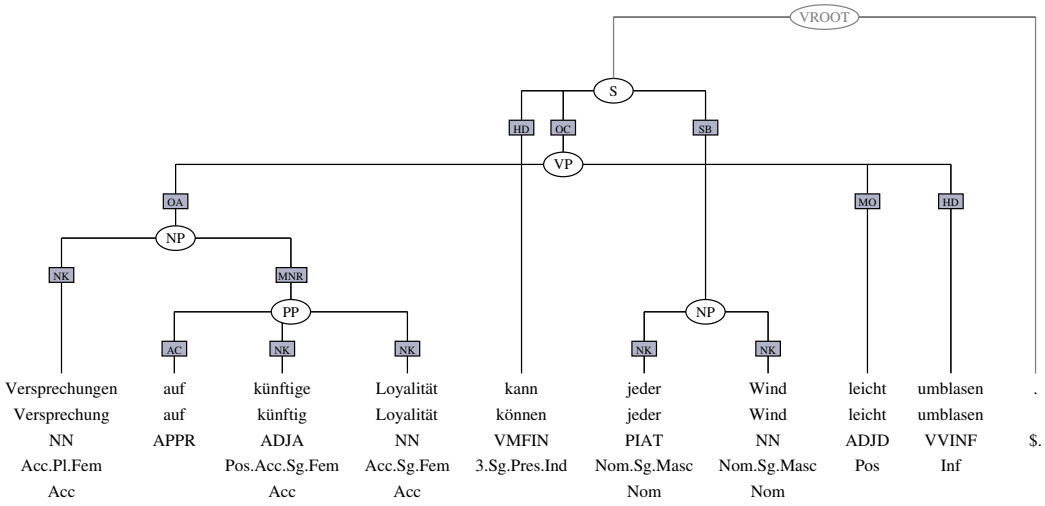


Figure 2.13: A sample graph from Tiger. The translation is provided in (2.9).

Treebank (or its predecessor Negra (Skut et al., 1997)). Previous work, for example (Schiehlen, 2004), (Dubey and Keller, 2003) or (Cahill, 2004), applies a standard procedure in which non-head daughters of the discontinuous constituents are re-attached minimally higher in the graph, such that a tree is obtained in which all crossing branches are resolved. As a result, the object in the graph in Figure 2.13 would be directly attached to the S node as a sister of the finite verb. An indexed trace could be left in the original location of the re-attached constituent, but it is usually suppressed for PCFG parsing (e.g. in (Dubey and Keller, 2003)). This transformation clearly involves a loss of the original non-local information, since the dependency structure is changed. It has therefore been tried to implement some mechanisms in order to preserve the trace information, but this resulted in a decrease in performance (cf. Schiehlen, 2004). As will be shown in Section 3.3.4, TAG can express some of the non-local dependencies thanks to its extended domain of locality.

The constituency annotation of the Tiger trees is extremely flat: no intermediate phrasal projections ("X-bar" levels) are annotated and (with a few exceptions) no unary branches are allowed. Categories do not have a maximal projection unless they have their own dependents. As an example, consider the adjective *leicht* in Figure 2.13 which does not project to an adjectival phrase AP. Most strikingly in comparison to the PTB, the subject (*SB*) as well as the finite verb (*kann* in Figure 2.13) are always immediate daughters of the sentence node S, which is a way of accounting for the relatively free word order of German.

Furthermore, prepositional phrases (PPs) do not embed a noun phrase (NP) constituent. Instead the noun (and determiners, adjectives and other modifiers if present) are sisters of the

adposition, as can be seen in Figure 2.13 (*auf künftige Loyalität*). Noun phrases themselves are also flat (*jeder Wind*) and their components have the functional label NK (noun kernel). In (Albert et al., 2003) it is argued that NPs can be structurally disambiguated based on the corresponding POS information and phrasal labels. Main clauses and subordinate clauses (including relative clauses) all have the same phrasal category S.

(2.10) *Erwachsene zahlen 15 Mark, Kinder elf Mark Eintritt.*
Adults pay 15 Mark, children eleven Mark entrance fee
'Adults pay 15 Mark, children 11 Mark entrance fee.'

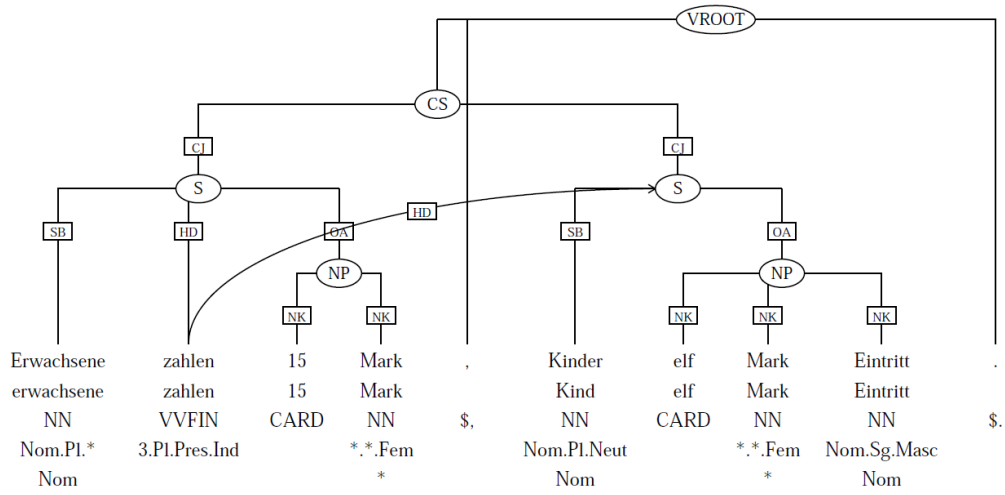


Figure 2.14: A Tiger graph with a secondary edge. The translation is provided in (2.10).

For coordinated categories, the Tiger annotation employs specific labels, such as CS for the coordination of sentences (see Figure 2.14 for an example), CNP for coordinated NPs etc. (Table A.1 lists all of them.) This drastically increases the overall number of labels. Double dependencies in coordinations can be annotated thanks to *secondary edges*, an additional layer that makes structure sharing in the graph possible. Figure 2.14 shows an example of gapping, where the finite verb of the first conjunct also serves as the head (*HD*) of the second conjunct. Around 7% of all Tiger graphs have one or more secondary edges.

Punctuation marks have been left unattached during the annotation phase and are now thought of being attached to a virtual root (VROOT). They therefore create additional crossing branches.

2.5 Summary

Since there is evidence that humans process sentences strictly incrementally and predict upcoming structures and lexemes based on what has already been processed, it is a natural consequence to also design and use a grammar formalism which incorporates those notions. This has been done with PLTAG. A German PLTAG lexicon and a German PLTAG treebank

are required in order to build a PLTAG parser for German for validation of the human language processing theory and for language technology applications. Those resources are also applicable to parsing with other variants of TAG. Corresponding approaches for English that induce such resources from existing constituent treebanks have shown to be fruitful, so the German parsing resources will also be automatically extracted from an available treebank: the large German Tiger Treebank provides syntactic function information besides the phrase structure annotation. Particularities of the German language (e.g. sentence structure, discontinuous constituents and case marking) and of the Tiger annotation (very flat graphs) will require special consideration. Previously extracted German TAG lexica are not available, and it is questionable whether the corresponding approaches scale up to broad-coverage parsing.

Chapter 3

The Target Grammar

This chapter will provide details on the shape of elementary trees that are induced from the Tiger Treebank. The exact procedures of how the treebank is converted to derived trees and how the elementary trees are extracted will be given in Chapter 4, but certain aspects are motivated here separately. Since our target PLTAG grammar is a superset of a standard LTAG grammar and since they generate the same derived trees, as it has been discussed in Section 2.2.2, the canonical elementary trees are the same for both. The following characteristics therefore hold for the extracted LTAG as well as PLTAG grammar, except, of course, the specifications about the prediction trees, which LTAG does not use.

For English, the linguistically motivated elementary trees from the XTAG project are usually considered as the target grammar or at least as a good inspiration when extracting an LTAG from the PTB. However, for German no such wide-coverage base framework is available (cf. Section 2.3) that could be taken as the target for an extraction.

Since English and German are both Germanic languages, some analyses (e.g. for auxiliary verbs or for adjectives) can be adopted from English. However, for example for the sentence structure, the analysis has to follow a different path: German has a more flexible word order as was explained in Section 2.4.1 and the standard assumptions for English do not hold (e.g. the substructure of a sentence is a subject NP and a VP). Section 3.2 therefore details and motivates the design decisions that we take for the German grammar. Section 3.3 provides the target elementary trees for certain linguistic phenomena. The chapter starts with principles for TAG grammar writing which are not language-specific.

3.1 Linguistic Principles in TAG

Tree-adjoining grammars for natural languages usually follow certain principles, even though these linguistic constraints are not part of the grammar formalism itself. Originally they have been defined for manually written grammars, but the automatically induced grammar presented in this work should also adhere to the most important principles. The following ones are taken from Kallmeyer (2010).

Lexicalization has already been mentioned in Section 2.2.1. In each elementary tree,

there is at least one non-empty lexical item, its anchor. Elementary trees are allowed to contain empty words, but only if also an overt lexical item is present. There are several reasons for lexicalization: First of all, a lexicalized grammar is finitely ambiguous, i.e. it provides finitely many analyses for each string, which is relevant for efficient processing. Lexicalization furthermore allows to filter the grammar before parsing a string w. The search space can thus be limited to the elementary trees that have lexical anchors which occur in w. A linguistic motivation for lexicalization is that, together with the next principle, certain idiosyncrasies of expressions can be directly encoded in the elementary tree.

Another principle which is commonly accepted among grammar writers is the one of **predicate-argument co-occurrence**. The elementary tree of a predicate contains argument slots (substitution or foot nodes) for all and only its arguments. Elementary trees thus encode subcategorization frames. This principle indicates the essential distinction that is drawn in TAG-based natural language grammars between arguments, that are subcategorized for by a head, and modifiers[1], optional constituents which can be viewed as choosing the elements with which they combine themselves. The prepositional phrase *in Berlin* is an argument in Example (3.1a), and the elementary tree anchored in *lebt* provides a PP substitution node for it (Figure 3.1(a)). In contrast, the same phrase in Example (3.1b) is an adjunct, which is typically encoded as an auxiliary tree in TAG that adjoins to a VP or S node. The elementary tree anchored in the transitive verb *isst* therefore only has substitution nodes for its two arguments, but not for the PP (Figure 3.1(b)). The fact that adjuncts (and other recursive structures, e.g. coordination) can be encoded in separate auxiliary trees is usually referred to as *factoring of recursion*.

(3.1) a. *Peter lebt [in Berlin].*
Peter lives in Berlin
'Peter lives in Berlin.'

b. *Peter isst einen Apfel [in Berlin].*
Peter eats an apple in Berlin
'Peter eats an apple in Berlin.'

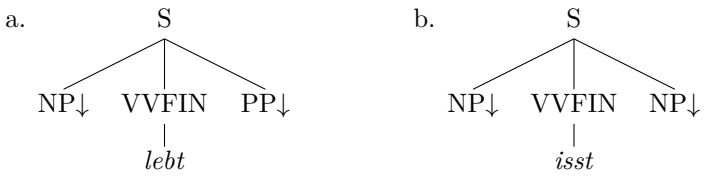


Figure 3.1: Lexicalized elementary trees that adhere to the predicate-argument co-occurrence constraint

From a linguistic point of view, elementary trees are maximal projections of lexical items, with adjuncts having additional nodes to specify where they can adjoin (Abeillé and Rambow, 2000). Lexicalization thus corresponds to the notion of a *head*. However, without

[1]Instead of *modifier*, sometimes the term *adjunct* is used.

having a particular linguistic theory in mind, the head is viewed from a lexical perspective, since it determines which other constituents (i.e. its dependents) are represented in its tree (following the co-occurrence constraint). It is a clear goal of this work to obey the two foregoing generally accepted linguistic TAG principles.

There are a handful of other principles which are more arguable and are not followed in every natural language tree-adjoining grammar: Kallmeyer (2010) explains that elementary trees should not be semantically void (**semantic anchoring**) and that an elementary tree should correspond to a single semantic unit (**compositionality principle**). The grammar presented in this work will not fully adopt the semantic anchoring principle, as it contains, for example, an elementary tree for the German particle *zu*, which is used to form the *zu*-marked infinitive and which does not carry its own semantics. (XTAG, which uses for example separate auxiliary trees for complementizers, does not obey it either.) Since the extracted grammar does not include a semantic theory - semantics in PLTAG is not yet fully defined -, it is difficult to verify the compositionality principle. For an idiomatic expression, however, whose meaning is clearly not composed by the meaning of its constituents, the extracted grammar will provide a multi-anchored tree, as far as it can by automatically detected given the treebank annotation (see Section 3.2.3). An example would be *jemanden ins Visier nehmen* ('to target somebody') in its figurative sense.

3.2 Grammar Design Decisions

Global grammar design decisions concerning sentence structure modeling, multi-anchored elementary trees and the method of how to encode the flat Tiger structures in TAG are presented and discussed in the following sections.

3.2.1 Sentence Structure Modeling?

In the context of generative grammar and X-bar theory, German syntax and the topological model are described in terms of movement of constituents.[2] The verb-final sentence type (see Example (2.3)) is considered as the base word order. Verb-first sentences (2.2) are generated from it by moving the finite verb into the left Satzklammer, the C (complementizer) position. (This is called *Finitumvoranstellung.*) For verb-second sentences (2.1), an additional movement, namely topicalization, is needed to fill the Vorfeld (the specifier of C) with one of the constituents from the Mittelfeld. Each movement leaves behind a trace, which is phonetically empty. Figure 3.2 provides a sketch of such an analysis.

The Tiger Treebank with its extremely flat graphs does neither include X-bar level annotations nor traces for moved finite verbs. Topicalization is sometimes indicated with a trace, but only if it results from a crossing branch, that leaves a trace behind when being

[2]More information on the generative model and syntactic analysis in the Chomskyan tradition is beyond the scope of this thesis. The reader is advised to consult some syntax introduction, for example the one in (Grewendorf et al., 1989).

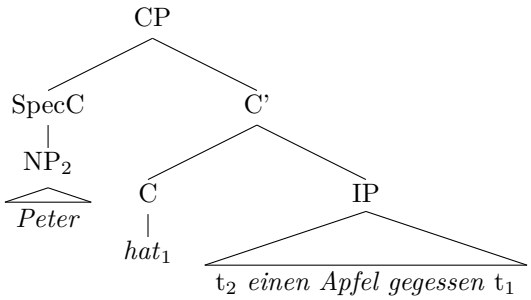


Figure 3.2: Analysis of a verb-second sentence in the generative model

resolved, as in Figure 2.13 for example. A subject in the Vorfeld for instance does not cause a discontinuous S or VP and therefore has no trace (Figure 2.14). Topological fields are not explicitly annotated in the Tiger corpus either.

From the information that is present in the Tiger Treebank, it is possible to identify the three different sentence types and to infer topological fields with rules. One can then furthermore enrich the tree structures such that the LTAG analysis is close to the one provided by generative syntax. This is roughly the approach that has been taken by Frank (2001). In the verbal trees that she extracts, a binary branching VP serves as the Mittelfeld. In verb-initial and verb-second sentences the finite verb is co-indexed with its (newly inserted) empty base position. For verb-second sentences, the Vorfeld is realized either by an argument (substitution node), a verbal projection of the tree itself (e.g. a past participle in compound tenses) or a modifier. To avoid multiple constituents in the Vorfeld, the modifier is not encoded as an auxiliary tree in this case, but as an initial tree with a substitution node for the sentence.

This neat linguistic account has the severe disadvantage of resulting in a huge lexicon. Since TAG does not provide the explicit notion of movement, elementary trees in which the transformation has already been performed are needed. This means that for a specific verb, the lexicon does not only contain different elementary trees for the three sentence types plus for different orderings of the arguments, but even more trees for the different constituents that can fill the Vorfeld. Additionally, modifiers will not only be associated with an auxiliary tree, but they also require an initial tree for the case that they have been seen in the Vorfeld. Unfortunately, Frank (2001) does not report the final number of tree templates (i.e. unlexicalized trees) that she extracts from the Negra corpus. Comparing lexicalized elementary trees is difficult since the version of Negra she uses contains only 10,000 sentences: Frank (2001) extracts 113k trees, which appears to be a large lexicon compared to the roughly 150k trees in our final grammar induced from five times more data (see Section 5.1). Furthermore, to the best of our knowledge, Frank's (2001) grammar has never been used for phrase-structure parsing.

It has been shown that the size of the grammar is a decisive factor for the TAG parsing complexity (Mazzei and Lombardo, 2004; Sarkar et al., 2000). The larger the grammar,

the less feasible becomes TAG parsing. Since our future goal is PLTAG parsing, we take a different strategy than Frank (2001) and decided against modeling topological fields in the grammar. Instead, the extraction procedure used is very similar to the one for English where only a distinction between heads, arguments and modifiers is required on each level of the (converted) treebank tree. It is provided in detail in Section 4.2.2. The main benefit is that we are able to extract verbal trees that generalize to several sentence types. As an example, consider the verb-initial sentence in (3.2) and its corresponding Tiger graph in Figure 3.3. The initial tree for the transitive particle verb in Figure 3.4(a) has been extracted from it. The annotation of the elementary tree does not restrict it to be used for verb-initial sentences only. The same initial tree could also provide a verb-second sentence with a modifier in the Vorfeld, such as (3.3).

(3.2) *Tauschen Sie sich mit Ihren West-Kolleginnen mal aus?*
Exchange you yourself with your colleagues from the West sometimes PARTICLE
'Do you sometimes exchange ideas with your colleagues from the West?'

(3.3) *Heute tauschen Sie sich mit Ihren West-Kolleginnen aus.*
Today exchange you yourself with your colleagues from the West PARTICLE
'Today you exchange ideas with your colleagues from the West.'

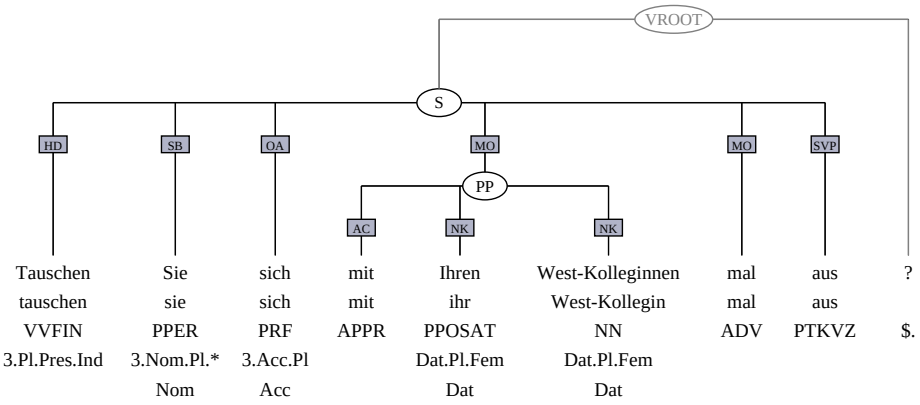


Figure 3.3: Verb-initial sentence. The translation is provided in (3.2).

Of course, we still need several elementary trees for different orderings of the arguments with respect to the verb. For *austauschen*, the elementary trees that corresponds to the verb-second sentence with a topicalized subject and to the verb-final sentence are shown in Figure 3.4(b) and (c). If the finite verb is an auxiliary or modal, the proposed strategy leads to one tree which can be used for all three sentence types, as will be shown in Section 3.3.1.

An alternative to having a separate elementary tree for every possible relative order of arguments to the verb is using TT-MCTAG as proposed in (Lichte and Kallmeyer, 2008). Arguments are factored into a tree set and the order in which they adjoin is not specified. (The sentence structure is modeled via features.) A short introduction is given as a part of Section 2.4.2. While this approach facilitates manual grammar implementation, which we

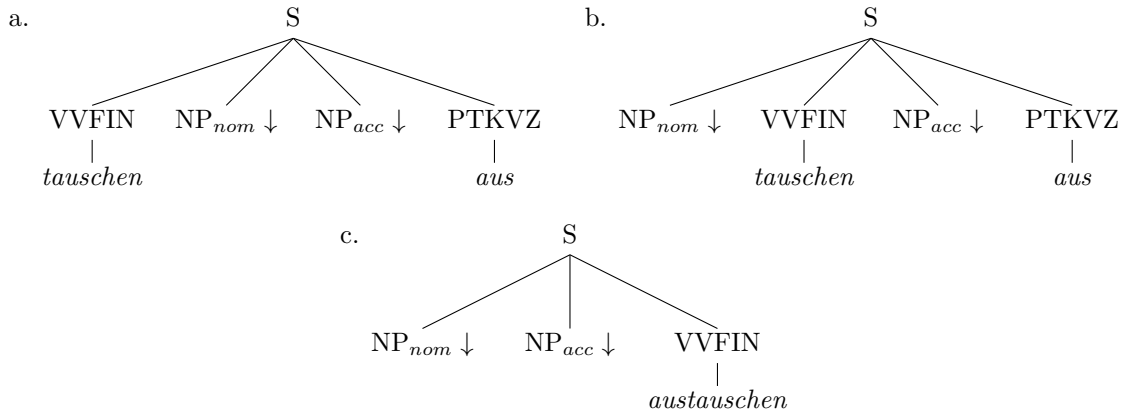


Figure 3.4: Elementary trees for *austauschen*

are not concerned about, the task of building the final tree structures (putting the factorized arguments together again) is shifted to parsing. For grammar extraction from a treebank, factorizing the complements as in TT-MCTAG could help to alleviate the data sparsity issue since word orders that have not been seen during training are also possible. A similar effect, however, could be achieved by grouping the verbal elementary trees into tree families or by having a good back-off model for parsing.

It should be mentioned that the extracted grammar, which does not model the topological fields, will overgenerate. For example, nothing prevents several modifiers to adjoin from the left to the elementary trees in Figure 3.4(a) and (b), resulting in a Vorfeld with more than one constituent. However, this is not a problem which is specific to our TAG grammar. Any CFG that is extracted from a treebank has a similar issue of overgeneration. Furthermore, the purpose of the grammar at hand is parsing and not generation anyway. Since the grammar has been extracted for stochastic and not for symbolic parsing, the probability model will impose further constraints and might for example learn that adjoining several modifiers to the S node at the beginning of a sentence is not very probable. As one can see from the Tiger Treebank (Examples (3.4) and (3.5)), having one constituent in the Vorfeld is rather a soft constraint anyway.

(3.4) *[$_{VF}$ [Nahrungsmittelsicherheit]$_{SB}$ [aber]$_{MO}$] [$_{RK}$ ist] die erste Voraussetzung ...*
food safety however is the first requirement ...
'Food safety, however, is the first requirement ...'

(3.5) *[$_{VF}$ [Jetzt]$_{MO}$ [plötzlich]$_{MO}$] [$_{RK}$ stellt] er sich an*
now suddenly "makes a fuss" he himself PARTICLE
'He suddenly makes a fuss now'

3.2.2 Sister-Adjunction

The decision against modelling of the topological field structure goes hand in hand with employing *sister-adjunction*. Sister-adjunction is a composition operation that is not found

in the standard definition of TAG as given in Section 2.2.1. It is a mechanism borrowed from D-Tree Grammar (Rambow et al., 1995).

Chiang (2000) proposes the sister-adjunction operation in order to be able to derive the flat syntax structures of the PTB with a TAG, without first converting the treebank trees to a format (fully bracketed trees) from which modifiers can be factored out as auxiliary trees. See Section 2.3.1 for his definition of sister-adjunction. Chiang's (2000) grammar still contains auxiliary trees, but only for recursion that was already present in the PTB, e.g. VP auxiliary trees for modal and auxiliary verbs. Those can be categorized as *predicative auxiliary trees*, in which the foot node is subcategorized for by the lexical anchor. In *modifier auxiliary trees*, in contrast, the foot node represents the modified category. Instead of modifier auxiliary trees, Chiang (2000) uses sister-adjoining modifier trees.

We follow Chiang's (2000) idea and keep the flat annotation of Tiger, instead of introducing additional nodes in order to enable regular TAG adjunction of modifier auxiliary trees. Since in the Tiger graphs sentence level nodes (S) do not even necessarily embed a verbal phrase (VP) (see for example Figure 3.3), many additional projections, more than in English since there always is a VP, would be required for a classical TAG analysis.

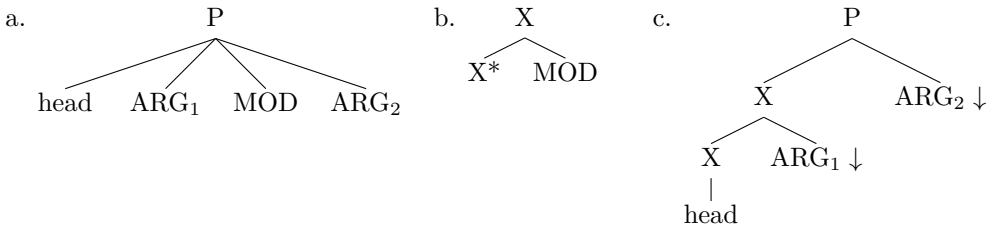


Figure 3.5: Schematic representation of the issue that occurs with regular adjunction when a modifier daughter is located between two arguments (inspired by (Chen, 2001, p.83))

From an issue reported by Chen (2001) as well as Demberg-Winterfors (2010), it can also be inferred that in the German Tiger Treebank more unflattening than in English would be necessary for regular adjunction. Both do not completely binarize the treebank trees, but only create the necessary foot and root nodes for modifier auxiliary trees. They cannot adequately describe the case in which a modifier daughter occurs between two arguments, as in Figure 3.5(a). Additional projections as in (c) would be required such that an auxiliary tree (b) can adjoin. Otherwise there is no structure available to which the modifier auxiliary tree could adjoin back in. The problematic configuration is more frequent in German than in English because of the missing VP level in the sentence and because of the free word order that allows virtually any ordering of arguments and modifiers in the Mittelfeld. For illustration, the Tiger graph in Figure 3.6 shows a modifier adverbial phrase between a direct (*OA*) and an indirect (*DA*) object. It consequently seems that in order to adopt their extraction approach (modifiers as auxiliary trees), only complete binarization of the Tiger trees would solve such problems. However, the desired

(3.6) *Politisches Bewußtsein fügt sich nicht mehr dieser sozialen Mechanik*
political awareness submit itself not anymore this social mechanism
'Political awareness does not submit itself to this social mechanism any longer.'

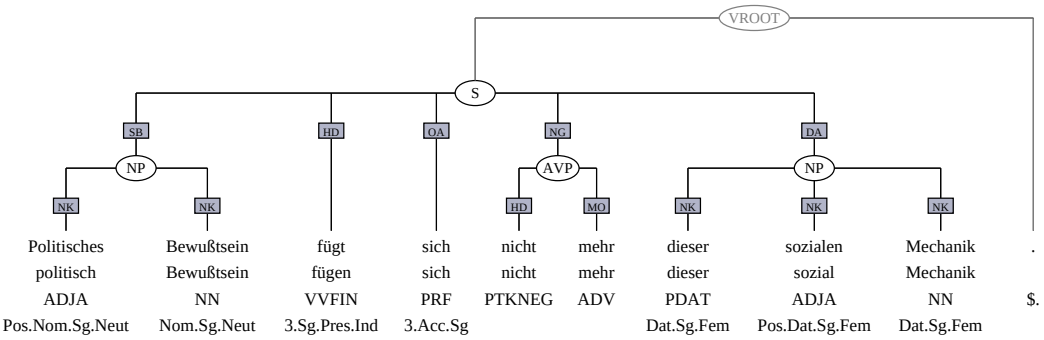


Figure 3.6: Structure for which binarization is required in order to adjoin the negation between the two arguments. The translation is provided in (3.6).

binarization should still be linguistically motivated, and would therefore require fine-grained rules. Regarding the amount of additional levels that would be necessary and the risk of introducing inconsistencies and questionable linguistic structures, in this work, the flat structures are preserved and the problem is solved via sister-adjunction. The investigation towards a linguistic binarization of the Tiger Treebank and the consequently straightforward extraction of a standard TAG lexicon is left for the future.

The decision for flat (sentence) structures is also encouraged by results by Dubey (2005). In his work, the Negra corpus is unflattened with the help of several rules. All of them improve the PCFG parsing performance. The only exception is the insertion of a VP to group finite verbs with their complements, which is exactly a transformation that is necessary to make adjunction possible. For data that has been converted by this rule, the parsing performance (F-measure) drops from 74.1 to 71.8.

While the sister-adjunction operation in (Chiang, 2000) is only constrained by the probability model of the parser, our sister-adjoining trees are constrained by the formalism itself, namely by the category of the node to which they sister-adjoin. Formally, sister-adjoining *modifier trees* are added to the TAG definition (Section 2.2.1) as an additional set M of elementary trees. Modifier trees adhere to the same constraints as all elementary trees. Additionally, the root node of a sister-adjoining elementary tree γ is required to have exactly one daughter. Figure 3.7 shows examples.

When γ sister-adjoins at position i to a node n that must have the same label as γ's root, n and γ's root are identified with each other[3], and the only daughter of γ's root, let's call it m, is added as a new daughter of n between the ith and $i+1$th child of n. If $i = 0$,

[3]It is not yet clear how the *identification* of n and γ's root would work in the context of feature-based TAG, since the concept of unifying half nodes cannot be applied to sister-adjunction in a straightforward way. This formalization is left for future work.

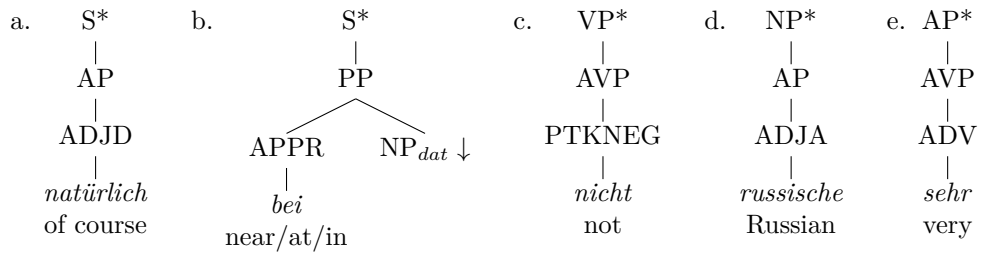


Figure 3.7: Sister-adjoining modifier trees

m with its subtree is added as the leftmost daughter of n; if i is equal to the number of children of n, m it is added as the rightmost daughter. To be able to distinguish initial trees from modifier trees, we mark the latter with an asterisk at the root node. Figure 3.8 gives an example of the sister-adjunction operation.

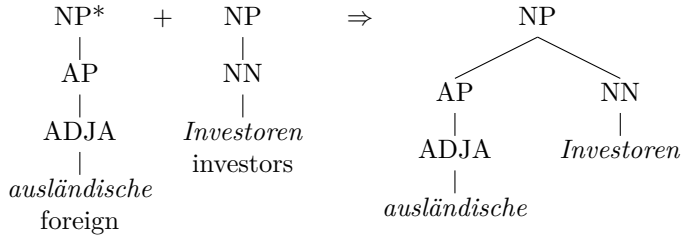


Figure 3.8: Sister-adjunction at position $i = 0$

The difference to Chiang's (2000) sister-adjoining trees is the root level, which we "add" to his kind of trees. It provides the information to which type of node the modifier tree can sister-adjoin and therein restricts its usage. This will make the probability model simpler, but at the same time, the lexicon will grow. For example, the modifier tree in Figure 3.7(b) is restricted to sister-adjoin to an S node, but we expect to also find the corresponding tree in the grammar which is compatible with VP, NP etc. However, the lexicon will not grow in a way which makes the grammar more ambiguous, as the modifier trees themselves are more restrictive. The worst case scenario, in which the grammar will specify modifier trees anchored in e.g. *bei* for all phrasal categories, is equivalent to using the modifier tree without the root level restriction and applying Chiang's (2000) definition.

Also consider for example the modifier tree in Figure 3.7(d). An adjective is expected to modify an NP, so we can as well directly encode this in the grammar, instead of leaving the modified category open and enlarging the search space of the parser (which might give a very small smoothed probability to sister-adjoining the modifier tree to other categories than NP). Overall, our new interpretation of sister-adjunction makes this operation more similar to regular adjunction, which also specifies the adjunction category in the auxiliary tree itself.

The system which extracts the grammar optionally uses Chiang's (2000) original defini-

tion of sister-adjunction and extracts the corresponding modifier trees if desired. See the program parameters (also presented in Appendix B) for more details.

3.2.3 Multi-Anchored Trees

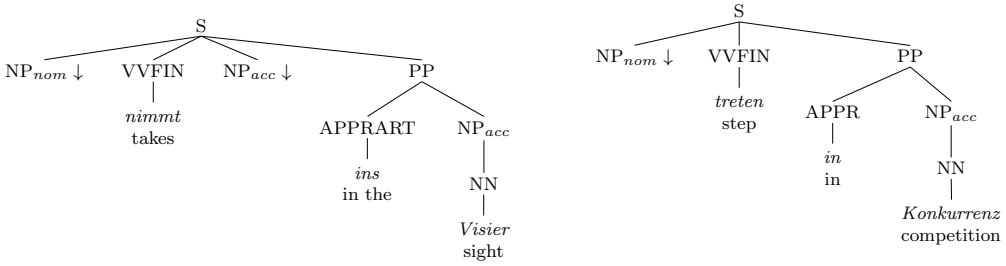
A *multi-anchored* (elementary) tree is an elementary tree which contains several overt lexical items. An example from English with two anchors is shown in Figure 2.4(c). Sometimes one of the anchors is considered the main anchor, whereas the others are *co-anchors*. The distinction can be based on semantics or syntactic dependency.

Multi-anchored trees in a natural language TAG are usually motivated by semantic considerations. Since an elementary tree should correspond to a single semantic unit according to the compositionality principle (Section 3.1), multi-anchored trees are used for expressions with a non-compositional meaning.[4] Chen (2001) for example extracts multi-anchored trees for verbs that co-occur with a particle or a preposition, e.g. *believe in*. The non-compositional meaning (*to have a firm conviction*) in this case is distinct from a compositional interpretation of *believe* + *in* (*to hold an opinion at a certain location*). As elaborated in (Abeillé and Schabes, 1996), TAGs are very well suited to represent idiomatic expressions, even if they are flexible (e.g. *Pat spilled the beans* vs. *The beans were spilled by Pat*) or contain gaps (e.g. *He took Mary's word into account*), because of the extended domain of locality.

It is, however, non-trivial to detect idioms automatically and extract the corresponding multi-anchored trees since there is often a literal meaning besides the figurative one. Unfortunately, neither the Tiger Treebank nor the PTB have annotation for idioms. Chen (2001) therefore uses heuristics to identify the aforementioned constructions. The Tiger Treebank contains functional labels which help to identify collocational verb constructions (*CVC*) and particle verbs from which the prefix has been separated (*SVP*), which also have non-compositional semantics. Circumpositions can be identified by their POS label (APPR + APZR) and grammatical function (*AC*). The extracted grammar contains those as multi-anchored elementary trees. Examples for collocational verb constructions are presented in Figure 3.9, for particle verbs in Figure 3.10 and also in Figure 3.4, and for a circumposition in Figure 3.11(a).

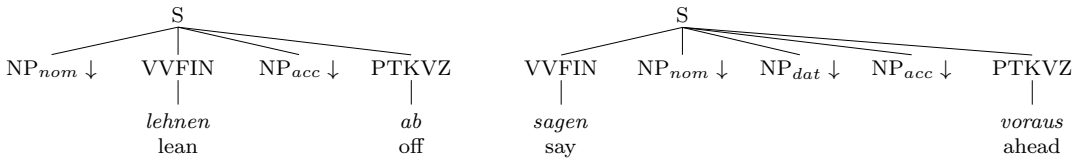
There are three other categories, which are rather idiosyncratic to Tiger, for which the semantics cannot be composed of the meanings of the individual parts. Their somewhat special status is already indicated in the Tiger annotation, which in turn allows us to extract them as multi-anchored trees (following the compositionality principle). Those are multi-token adjectives (MTA), which usually refer to a named entity, and multi-word expressions that have lost their original meaning. The individual components of the multi-word expressions have the edge label *AVC* (adverbial phrase component) in the case of

[4]Note that semantics in PLTAG is not yet defined, in particular it is not clear how the processing of non-compositional expressions is modeled.



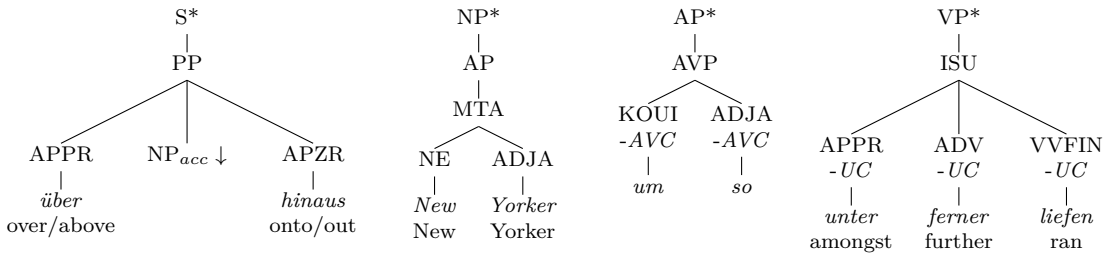
(a) "to target somebody" (b) "to compete"

Figure 3.9: Multi-anchored trees for collocational verb constructions (The literal meaning is indicated in the trees themselves while the idiomatic meaning is given in the caption.)



(a) "to disapprove of somebody/something" (b) "to predict something to somebody"

Figure 3.10: Multi-anchored trees for separated particle verbs (The literal meaning is indicated in the trees themselves while the idiomatic meaning is given in the caption.)



(a) "beyond something" (b) "a person from New York" (c) "all the more"/"even" (d) "also-run"

Figure 3.11: Multi-anchored trees for a circumposition (a), a multi-token adjective (b), a multi-word adverbial phrase (c) and an idiosyncratic unit (d) (The literal meaning is indicated in the trees themselves while the idiomatic meaning is given in the caption.)

adverbial phrases (AVP) and *UC* (unit component) in the case of idiosyncratic units (ISU). Figure 3.11 shows an example of a multi-token adjective in (b); (c) is a multi-word adverbial phrase[5] and (d) an idiosyncratic unit.

[5] *Um so* is spelled *umso* according to the current rules of orthography (`http://www.duden.de/rechtschreibung/umso`, January 2, 2012), which supports the choice to represent it as a multi-anchored elementary tree.

In the verbal multi-anchored trees (Figures 3.9 and 3.10), the verb is considered the main anchor since it stands in head relation with respect to the root node. For the trees in Figure 3.11, it is neither clear from the annotation nor linguistically which lexical item should be the head. The head finding procedures (Section 4.2.1) nevertheless must decide for one. This head it then considered the main anchor, whereas the others are co-anchors. In the lexicon entry, the main anchor is listed first, followed by the co-anchors separated by white-space.

Multi-anchored trees also serve another purpose in PLTAG. They activate predictions down to the lexical level, as was explained in Section 2.2.2. The PLTAG grammar should therefore contain multi-anchored trees for collocations in which one part is predictive of the other part(s), as e.g. for *either ... or* in English. Since German uses the exact same correlative coordination construction (*entweder ... oder*, *sowohl ... als* and others), the analysis from English was adapted, see Figure 3.12.[6]

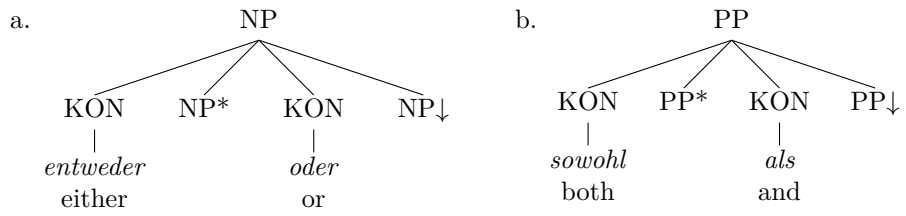


Figure 3.12: Multi-anchored trees for *entweder ... oder* and related constructions

Obviously an objective strategy is needed to decide consistently and automatically which lexical items to encode together in one elementary tree as a lexical prediction. Demberg-Winterfors (2010) suggests to follow a criterion used in data-oriented parsing, namely co-occurrence probabilities of words (or inner nodes). If two lexical items occur together more frequently than they occur individually, and, from an incremental perspective, if the first one is predictive of the second one, they should be encoded in one elementary tree. Thresholds are needed to define "more frequent" and "predictive". This strategy would also reveal whether the multi-anchored trees for non-compositional expressions are also useful in the sense of predictions in PLTAG. We will pick up this line of thought again in Section 3.4 when providing more details about prediction trees, but the corpus study that is necessary to implement the criterion will be left for future work.

3.3 Treatment of Specific Linguistic Phenomena

This section addresses specific, rather local linguistic phenomena, for which the encoding in the TAG lexicon is not straightforward.

[6]Note that technically the right anchor and the bottom half of its KON node are predicted, so they carry prediction markers. They will be validated when the corresponding word (e.g. *or*) is read. However, since those multi-anchored trees might be interesting for the TAG community in general, we do not make the markers explicit in this section.

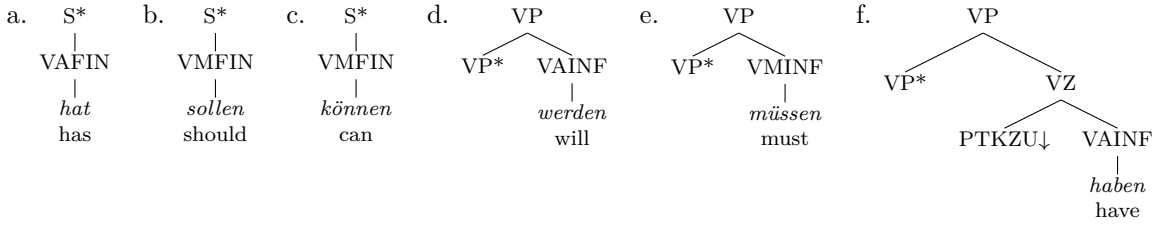


Figure 3.13: Elementary trees for modal and auxiliary verbs; (a)-(c) are finite, (d)-(f) are infinite.

3.3.1 Auxiliary and Modal Verbs

Auxiliary and modal verbs in compound tenses are usually encoded in English LTAGs as auxiliary trees that adjoin to a VP and that do not subcategorize for a subject. One consequence is that full verbs, even if they are not finite, can subcategorize for all their dependents, i.e. have corresponding nodes in their elementary trees, which is in agreement with the co-occurrence constraint.

We adapt this analysis for German. However, because of the flat sentence structure in the German grammar, finite auxiliaries and modals are not encoded as auxiliary trees, but as modifier trees that sister-adjoin to S. Non-finite auxiliaries and modals in contrast are VP auxiliary trees. They build the verb cluster at the right periphery of the sentence. Some example elementary trees for auxiliary and modal verbs are depicted in Figure 3.13. The interplay of trees for finite and non-finite verbs together with the elementary tree of a full verb when building a compound tense can be seen in Figure 3.14. The (non-incrementally) partially derived tree would be used to analyze the sentence in (3.7).

The advantage of this analysis in combination with not explicitly modeling the German sentence structure is that for non-finite full verbs one elementary tree is enough to generate all three German sentence types. Depending on the position i under the S node to which the elementary tree of the finite auxiliary or modal sister-adjoins, a verb-first ($i = 0$), verb-second (usually $i = 1$) or verb-final ($i = n$, where n is the number of daughters of S) sentence can be obtained. Figure 3.15 shows how a verb-final sentence could be parsed to its sentence structure.

3.3.2 Sentential Complements

Following the XTAG analysis, verbs that take a sentential complement (may it be indicative or infinitival) are often encoded as auxiliary trees in English LTAG lexica. The sentential complement is then represented as an S foot node of the predicative auxiliary tree. Long-distance dependencies, such as extraction out of the embedded clause, can be preserved thanks to this grammar design decision.

Our analysis of German verbs that subcategorize for a sentential complement follows

(3.7) *Er hätte es besser wissen müssen.*
He should have it better known must
'He should have known it better.'

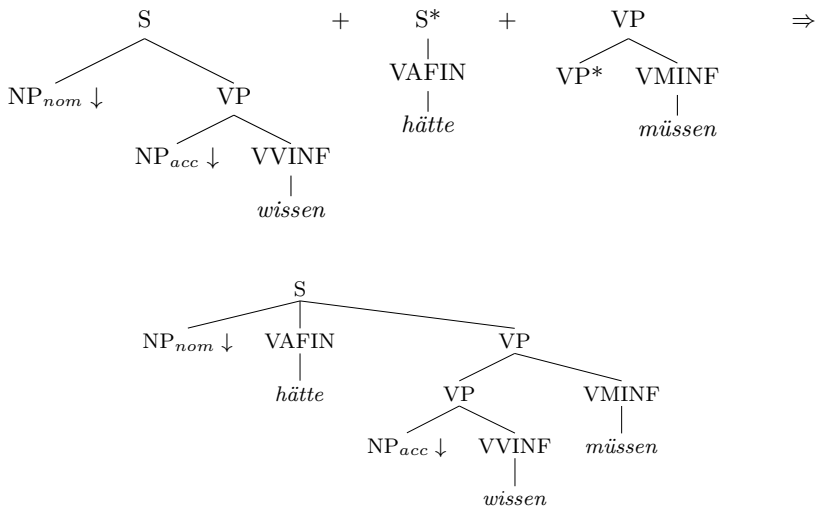


Figure 3.14: Elementary trees and partially derived tree for the sentence structure in (3.7)

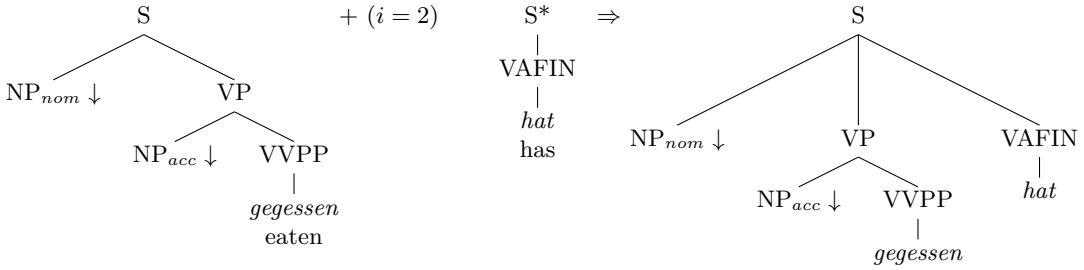


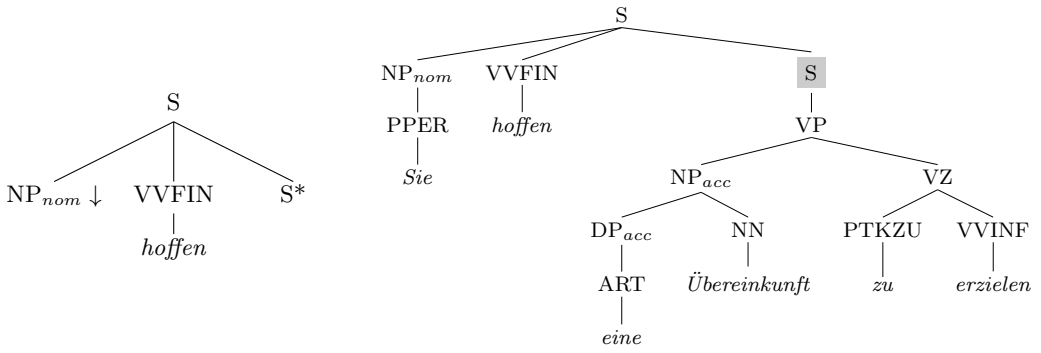
Figure 3.15: Generating a verb-final sentence, for example *dass Peter gestern einen Apfel gegessen hat*. The translation is provided in (2.3).

this analysis. As an example, Figure 3.16(a) shows the auxiliary tree which would be used to derive Example (3.8). Since non-finite clauses (infinitive or *zu*-infinitive) are of category VP in the Tiger Treebank, an S node governing the VP has to be introduced during the treebank conversion (see Section 4.1.2). This is illustrated by the shaded node in Figure 3.16(b). This conversion ensures that the matrix verb (raising or control verb) can indeed be encoded as an auxiliary tree of type S when inducing the lexicon. In fact, the tree in Figure 3.16(a) would also be used to derive 3.16(b). In Section 3.3.4, it will be shown how the provided analysis allows to localize long-distance dependencies.

Linguistic theories distinguish between raising verbs (*it seems to rain*, *he seems to sleep*) and control verbs (*he tries to sleep*, but **it tries to rain*), which both make the subject of the embedded clause their own argument. However, raising verbs do not assign a thematic

(3.8) *Sie hoffen, [dass ...]$_S$*
they hope that ...
'They hope that ...'

(3.9) *Sie hoffen [eine Übereinkunft zu erzielen]*
they hope an agreement PARTICLE strike
'They hope to strike an agreement'



(a) Predicative auxiliary tree for the matrix verb

(b) Converted Tiger graph for the sentence in Example (3.9). The shaded S node is inserted to ensure the recursion that is necessary to encode the matrix verb as an auxiliary tree.

Figure 3.16: Matrix verb *hoffen*

role to this argument. This difference has been accounted for in syntax (in theories following generative syntax) or in semantics. In the Tiger Treebank, raising and control verbs are annotated with the same syntactic structure, and there are no specific empty elements that would help to distinguish between the two. The extracted tree-adjoining grammar can only rely on the annotated verb-argument dependencies, and it therefore cannot account for constraints that the embedded verb might place on the subject. This means that, for example, both the elementary tree of *scheint* ('seems') and of *versucht* ('tries') provide an NP substitution slot for the subject besides the S foot node. A semantic component would have to find out about the dependency of this subject and the embedded verb. However, this work is beyond the scope of this thesis.

3.3.3 Nominal Case Marking

In Section 2.4.1 it has already been indicated that the case marking of German nouns and noun phrases is a factor that should be taken into account when writing or automatically extracting a grammar, since argument roles are determined by morphological case rather than by syntactic position. In Schiehlen's (2004) experiments for PCFG parsing of German, adding case annotation is the best performing strategy. The CCGBank and CCG lexicon for German extracted from Tiger by Hockenmaier (2006) also provides a case feature for NPs and nouns.

(3.10)

Konzernchefs	*lehnen*	*den*	*Milliardär*	*als*	*US-Präsidenten*	*ab*
nom		*acc*	*acc*		*acc*	
CEOs	reject	the	billionaire	as	US president	PTKVZ

'CEOs reject the billionaire as US president'

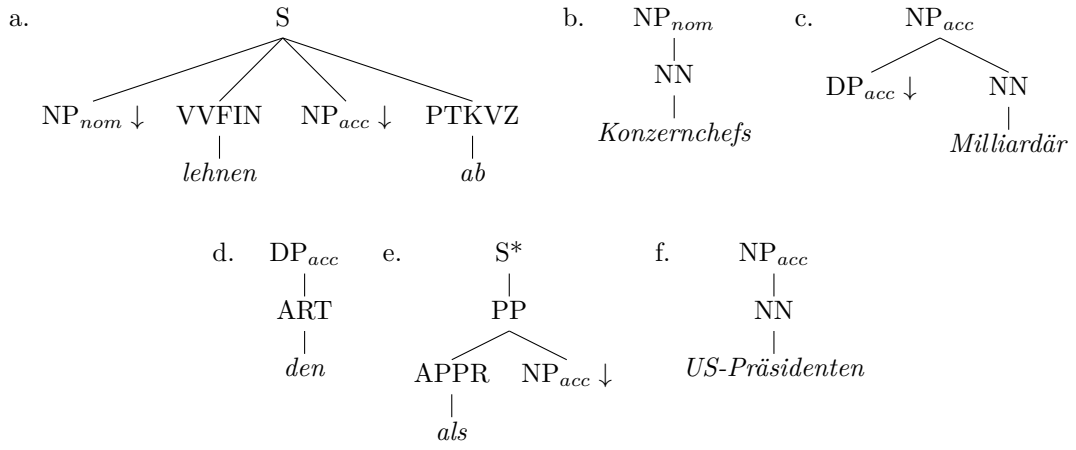


Figure 3.17: Elementary trees with nominal case marking for the sentence in (3.10)

We follow this work, and the converted treebank as well as the extracted lexicon includes case marking for NPs. It is obtained from the morphological layer of the Tiger Treebank and then projected upwards in the tree. Figure 3.17 illustrates this with some elementary trees. Since some nouns lack case annotation (e.g. measure nouns like *Prozent* 'percent'), the case marking of a corresponding substitution slot in a verbal elementary tree is inferred top-down via the grammatical function label. The *OA* label for 'accusative object', for example, triggers an annotation for accusative. Note that for prepositional complements the case cannot be determined top-down from the Tiger annotation, but only bottom-up from the morphological layer.

Since not only the head noun itself is inflected for case, but the complete noun phrase, determiners (DPs) are marked in the extracted grammar for case as well, see the trees in Figure 3.17(c) and (d). Determiners often disambiguate the case of a noun phrase: While the surface form *Milliardär* could be nominative as well as accusative or dative, *den Milliardär* unambiguously has accusative case.

At this point it is left open how the case annotation in the grammar will be used during parsing. It could be considered as a feature and treated like in feature-based TAG. Alternatively, a category with its case marking could be treated as a canonical phrasal category, which would increase the inventory of categories. In the latter case, the parser would also have to know that a category without case annotation (i.e. underspecified with respect to its case) is compatible with the corresponding category that is marked with a case, in correspondence to unification of features in feature-based TAG. In any case, the case annotation should restrict the possibilities of combining elementary trees.

Such considerations are of special importance in the framework of incremental parsing with PLTAG and modelling processing difficulties. If a sentence starts with *Den Milliardär* ..., predictions after the NP (in order to connect it with the following word) should only involve structures in which the NP is the object, i.e. in accusative case. Structures in which the NP has nominative case and serves as the subject should be ruled out in the first place in this example. This is achieved with the provided case annotation.

As the extracted German grammar with case annotations provides more details than its English counterpart, one could anticipate data sparsity issues, rather for the nouns than for the determines which are a closed word class. To alleviate the problem of unknown words, lexical rules can be used to generate the trees that are missing in the paradigm of a noun that has been seen during grammar extraction itself. The implementation of lexical rules is however beyond the scope of this thesis, and it will be left for future work to complement the extracted grammar with elementary trees induced via external linguistic knowledge. Attributive adjectives also agree in case with the noun they modify, so optimally they should also carry case marking in a grammar. However, in this case data-sparsity issues are anticipated to be even more severe.[7]

Including case marking in the grammar is only a first step in using morphology for parsing. Other, also more complicated, characteristics of a language relating to morphology could be implemented with features, as it is usually done within grammars for symbolic parsing (e.g. GerTT). Examples for German are agreement in number and gender within noun phrases and agreement in number between a verb and its subject.

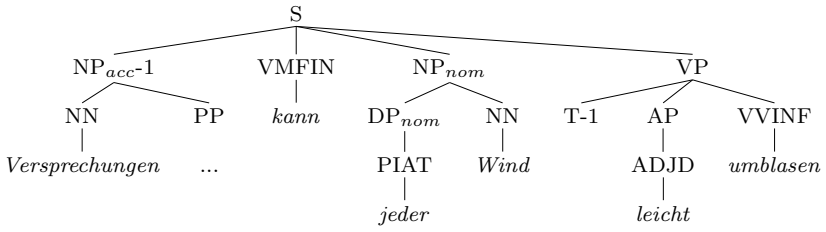
3.3.4 Discontinuous Constituents

The Tiger Treebank annotation contains discontinuous constituents, in the form of crossing branches in the phrase-structure graphs, to represent long-distance relationships, see Section 2.4.3, specifically the example in Figure 2.13. Since the crossing branches cannot be directly encoded in TAG, the Tiger graphs with crossing branches are converted to the classical constituent structure, namely trees with indexed traces (the PTB format), see Section 4.1. This is commonly done in the grammar extraction and parsing community,[8] also for the German LTAG induction approaches (Frank, 2001; Neumann, 2003). Figure 3.18(a) shows a converted treebank tree, in which, amongst others, the non-head daughter of the discontinuous constituent (VP) has been attached higher, leaving behind a trace (T-x) in the original position.

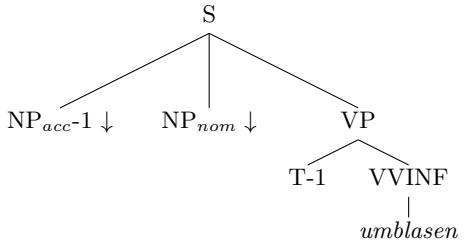
Traces and their filler counterparts are problematic insofar as they can only be interpreted correctly if trace and filler co-occur in the derived tree. If traces are generally removed from the grammar, as it is often done in PCFG parsing, the filler constituent cannot be

[7] Furthermore, attributive adjectives are encoded as sister-adjoining modifier trees. Currently, sister-adjunction is not formalized in the context of feature-based TAG.

[8] An exception are more powerful grammar formalisms, such as LCFRS (e.g. Maier (2010)), which are rarely used to date in NLP, mostly because of the prohibitive parsing complexity.



(a) Converted treebank tree. The original is shown in Figure 2.13.



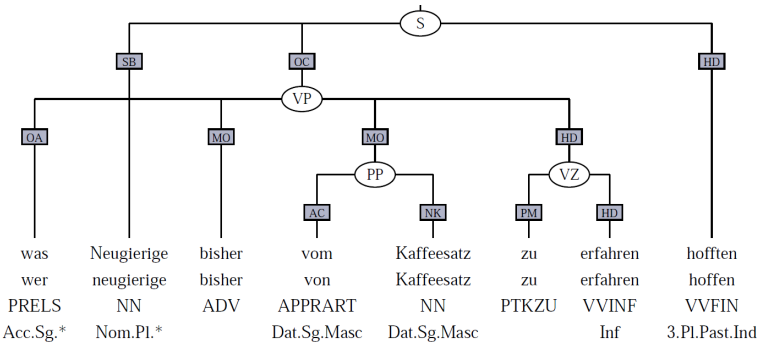
(b) Extracted elementary tree for *umblasen*

Figure 3.18: Localizing a trace-filler relation within one elementary tree

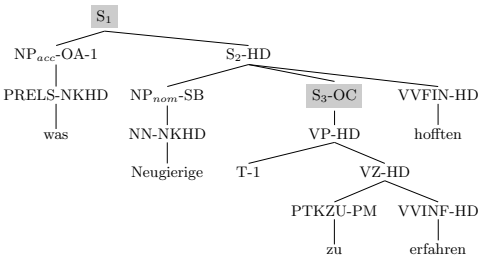
interpreted in its original position. TAG with its extended domain of locality can localize some of the trace-filler pairs within one elementary tree, thus making sure that trace and filler occur together in the final derived tree. Given our analysis for modal and auxiliary verbs (Section 3.3.1), this is for example the case for arguments of the non-finite verb in compound tenses. As shown in Figure 3.18(b), the fronted direct object of *umblasen* is realized within the verbal elementary tree, so the dependency between the predicate and its object is not lost.

To be able to describe scrambling of an argument of an embedded verb into a full verb matrix clause, an additional S level is inserted during the conversion procedure to which the argument filler is re-attached (see Section 4.1.2). The need for this transformation is illustrated in Figure 3.19. With the standard procedure to resolve crossing branches, the direct object (*OA*) of *erfahren* is attached to S_2, the S node governing the matrix control verb. This would thus lead to an elementary tree of *hofften* with a subject, a direct object (the one of *erfahren*) and a sentential complement, which does not correspond to the original predicate-argument dependency. However, by introducing an additional S projection (S_1 in Figure 3.19(b)) for the filler and given our analysis of control verbs as auxiliary trees, which requires the insertion of S_3 (see Section 3.3.2), the trace-filler relation is localized in the elementary tree of *erfahren*, as it is shown in Figure 3.19(c). Those elementary trees correspond directly to the schematic trees that were presented for scrambling phenomena in Figure 2.11. Note furthermore that the elementary tree of *hofften* is a wrapping auxiliary tree. The extracted German lexicon will thus indeed constitute an LTAG, in contrast to

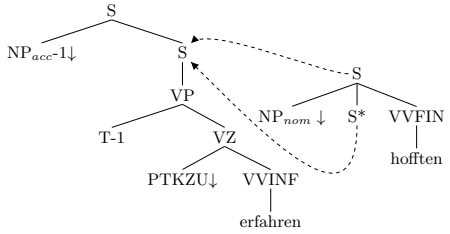
(3.11) *was* *Neugierige* *bisher* *vom* *Kaffeesatz* *zu* *erfahren* *hofften*
what rubbernecks so far from the coffee grounds PARTICLE found out hoped
'what rubbernecks hoped to find out from reading the tea leaves so far'



(a) Tiger graph with a discontinuous constituent because of an extracted object. The translation is provided in (3.11).



(b) Converted tree, omitting the two modifiers (The subscripts are used to be able to refer to the individual nodes in the text, they are not part of the alphabet.)



(c) Trace and filler are localized in one elementary tree, but will be further apart when the predicative auxiliary tree adjoins.

Figure 3.19: Localizing a long-distance dependency

the English lexicon from the PTB, which is an LTIG (Chiang, 2000; Demberg-Winterfors, 2010).

Thanks to the aforementioned design decisions and treebank conversions, the induced lexicon can cover 80% of all argument trace-filler pairs. This improves over any CFG extracted from the Tiger Treebank since context-free grammars only encode trees of depth one and cannot localize such predicate-argument dependencies. (Also see Section 2.4.3 for the treatment of crossing branches in PCFG parsing.) The TAG formalism is not powerful enough to describe all predicate-argument dependencies in a linguistically adequate way, as it has been shown in Section 2.4.2 (Becker et al., 1991). Note that elementary trees will still be extracted for the remaining cases, but trace and filler will be found in different elementary trees.

In order to cover the remaining scrambling phenomena, one could consider a more

powerful TAG alternative such as SN-MCTAG or TT-MCTAG (cf. Section 2.4.2). The topic of TT-MCTAG has already been picked up in Section 3.2.1. Generally, it would be possible to adapt the treebank conversion and lexicon induction in correspondence to a multi-component TAG. Using a different formalism, however, will have implications on notions of PLTAG, for example the definition of prediction trees. Additionally, it will be more difficult to adapt the existing PLTAG parser to work for German. Given the small number of concerned sentence ($\sim$ 1.5%), we decide to accept the shortcomings of standard TAG and leave the aforementioned, certainly worthwhile, investigations for future study.

The vast majority ($\sim$ 90%) of discontinuous constituents in the Tiger Treebank is caused by modifiers, not by arguments. The term modifier is used here in a very broad sense and does not only include VP modification and the typical NP modification such as extraposed relative clauses, but also, for example, antecedents (*RE*: repeated element) of resumptive pronouns (*PH*: place holders), parentheses and appositions. All of them would be encoded as auxiliary trees in a classic LTAG analysis and as sister-adjoining modifier trees in the German LTAG at hand. As local modifiers they would (sister-)adjoin to the node they modify, but if they occur non-locally (causing a discontinuous constituent) such an analysis is not possible. The sentence in Figure 3.20 contains two examples, namely an adverbial modifier causing a discontinuous VP and a repeated element in the Nachfeld causing a discontinuous NP.[9] When resolving the crossing branches, both modifiers become children of the S node. (This is depicted in Figure 4.1.) Since modifiers are factored into separate elementary trees and since they are optional, these trace-filler pairs cannot be captured in single elementary trees.

It should be noted here that many of the crossing modifier branches are due to an idiosyncrasy of the Tiger annotation. Modifiers are strictly attached according to the semantics, which means that verbal modifiers are mostly children of the VP governing the main verb, as the adverb in Figure 3.20. Re-attachment to a higher VP or S node is often possible without a change in meaning, and would probably be done (without the use of a trace) in other annotation schemes/linguistic theories, which do not dispose of crossing branches.

One general solution, that is also pursued in (Xia, 2001), is to extract a dislocated modifier and its trace as a multi-component tree set. The relationship is then localized within the set, and since all elements of the set have to attach (simultaneously) to the partially derived tree, the modifier could be interpreted in its original position. However, in the crucial cases such as extraposed relative clauses or repeated elements, the two components of the set would neither sister-adjoin into the same elementary tree nor into the same tree set. For example in an MCTAG derivation of the sentence in Figure 3.20, the repeated element (NP) sister-adjoins to the S node (i.e. to the elementary tree of *gespalten*), while its

[9] Punctuation also causes crossing branches, please see Section 3.3.6 for more information.

(3.12) *Da wird sie gespalten , die Moral .*
there becomes she/it divided , the morality/moral .
'Morality is being torn apart there.'

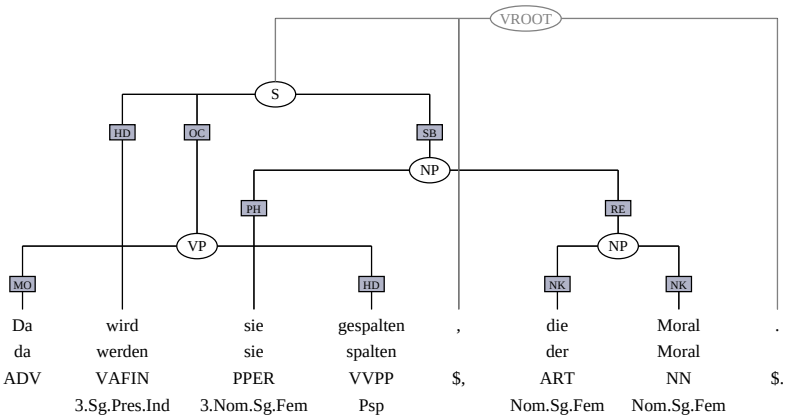


Figure 3.20: Tiger graph with a discontinuous VP due to a low-attached modifier and a discontinuous NP caused by a repeated element in the Nachfeld. The translation is given in (3.12).

trace sister-adjoins to the NP node of the initial tree anchored in *sie*. This non-local variant of MCTAG is much more expressive than TAG itself, which becomes problematic when it comes to parsing complexity.

For now we accept that the locality of standard TAG without extensions is not broad enough to cover non-local dependencies of modifiers. We settle for not describing them in syntax. In fact, the trace information is preserved throughout the complete transformation and extraction procedure, and only in a final step the modifier trace trees, which are not allowed to occur in an LTAG lexicon anyway, are discarded. Another (probably semantic) component will be needed to establish the relationship based on, amongst others, the information provided by the syntax tree. Should multi-component tree sets for modifiers be desired in the future, they can easily be created based on the "raw" lexicon extraction output.

3.3.5 Coordination

Coordinating constructions are (mostly) recursive structures. For an example, see the derived tree in Figure 3.21 in which two PPs are coordinated. Such coordination structures are usually modeled as auxiliary trees in LTAG, e.g. within XTAG for English (XTAG Research Group, 2001). For the German grammar, the same approach is pursued. As the tree in Figure 3.22(a) shows, the auxiliary tree is anchored in the coordinating conjunction.[10]

[10]Xia (2001) suggests an alternative approach in which the auxiliary tree is anchored in the head word of the second conjunct and the conjunction is substituted into this auxiliary tree. The advantage of such

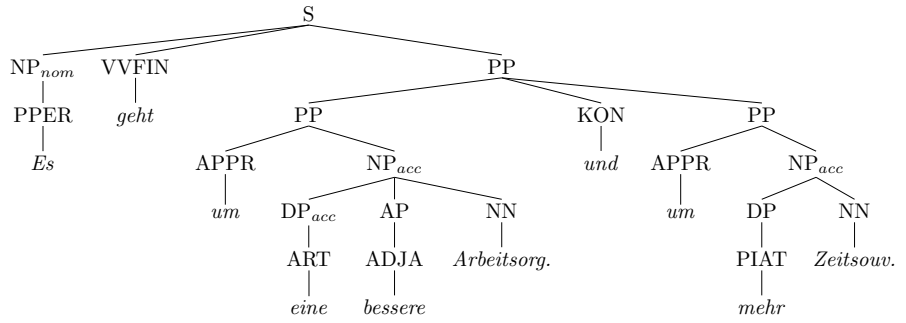


Figure 3.21: Derived tree (= converted Tiger tree) for the sentence in (3.13)

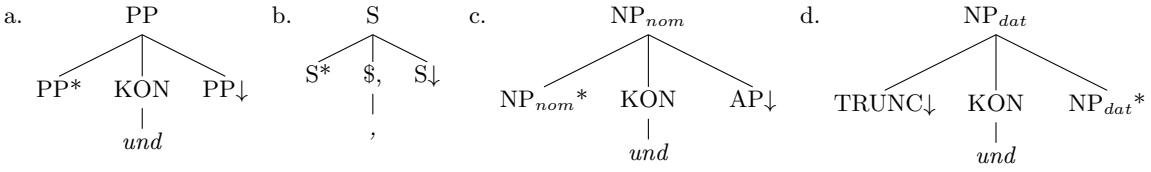


Figure 3.22: Elementary trees for coordination

It is supposed to adjoin to the first conjunct, while it contains a substitution slot for the second conjunct. Besides PPs, any kind of category can be coordinated in the same way. In the target grammar, not only designated conjunctions can anchor coordinating auxiliary trees, but also punctuation symbols which serve as the coordinating element. An example is shown in Figure 3.22(b), which would be used to analyze the coordinated sentences in (3.14).

(3.13) *Es geht um eine bessere Arbeitsorganisation und um mehr Zeitsouveränität*
it goes about a better work organization and about more time sovereignty
'It is about a better work organization and about more time sovereignty'

(3.14) *Die eine Form ist offenliegend, darauf reagieren wir*
the one type is exposed, thereon react we
'One of the types (of violence) can be observed, that is when we react'

In German (as well as in English) unlike categories can be conjoined. For example in sentence (3.15), a coordination of an NP and an AP forms the predicate. For such cases, the Tiger annotation provides the phrasal label CO as the parent category for the unlike conjuncts and the conjunction, see Figure 3.23. Given such an annotation, the coordination is not recursive, and can therefore not be encoded in an auxiliary tree. In case the coordinated phrases occur in an argument position, the status of the category CO is even more problematic. CO does not have an intrinsic syntactic meaning and it is therefore not desirable to have verbal elementary trees that subcategorize for CO. Our

an analysis is that the second conjunct, being an auxiliary tree, can be differentiated from its use in non-coordination constructions. This might be helpful for certain coordination phenomena as they are specified in the remainder of this section. However, for the non-problematic coordination cases, which clearly make up the majority, the analysis does not have an advantage, but rather leads to data sparsity.

(3.15) *Sie war [stellvertretende Ministerpräsidentin]$_{NP}$ und [verantwortlich für Gleichberechtigungsfragen]$_{AP}$*
she was deputy prime minister and responsible for equality issues
'She was Deputy Prime Minister and responsible for equality issues'

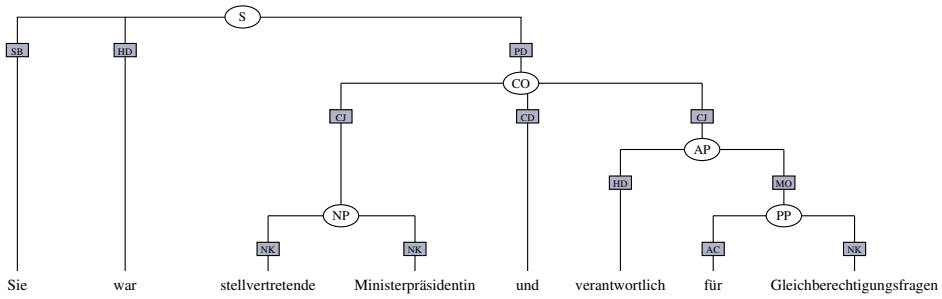


Figure 3.23: Coordination of unlike constituents. The translation is provided in (3.15).

strategy therefore is to replace each CO label with the label of the first conjunct (see also Section 4.1.2). The coordination of unlike categories can then be represented in an auxiliary tree, as in Figure 3.22(c). Accordingly, the elementary tree for *war* from sentence (3.15) has an NP substitution node for the predicate instead of a CO substitution node.

This analysis has the drawback that some local dependency information about when CO expands to which categories is not represented in the grammar. In fact, the coordination of, for example, an argument NP with an AP is only possible if both conjuncts are licensed by the subcategorizing verb (*war* in the example). One alternative would be to encode the verbal elementary tree together with the coordination structure (i.e. the tree in Figure 3.22(c)) into one, possibly multi-anchored, elementary tree. This would however increase the size of the lexicon considerably, and generalization to unseen coordination of unlike categories would not be given.

Truncated words can also be involved in a coordination, see Examples (3.16) and (3.17). In the first case, the coordination is not recursive with respect to the first conjunct, but with respect to the second. Our grammar thus contains the left auxiliary tree in 3.22(d). In the second case, a regular NP coordinating right auxiliary tree with a TRUNC substitution node is used. An alternative approach would be to introduce a maximal projection governing TRUNC such that even for the first case a regular right auxiliary tree can be employed. However, the special status of TRUNC, which cannot stand alone but must occur in a coordination to be completed, would be lost with such an analysis. A third alternative is to encode the complete coordinated NP as a multi-anchored tree.

(3.16) *an [[Bahn-]$_{TRUNC}$ und [Busstationen]$_{NP}$]$_{NP}$*
at train and bus stations
'at train and bus stations'

(3.17) *zwischen* *[[Arbeitgebern]$_{NP}$* *und* *[-nehmern]$_{TRUNC}$]$_{NP}$*
between employers and -employees
'between employers and employees'

Due to the flat annotation of Tiger, there are many cases where the recursion in a coordinating construction is not immediately present, as in Figure 3.24(a) for example. However, thanks to the conversion steps that are taken (cf. Section 4.1.1), such as inserting nodes that correspond to maximal projections, the recursion is made explicit (b) and can be factored into a coordinating auxiliary tree (c).

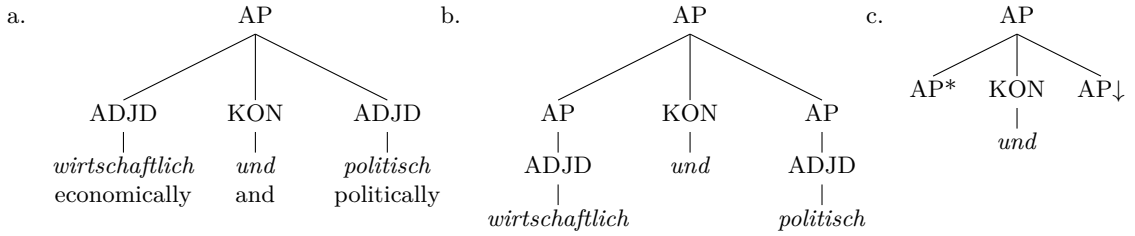
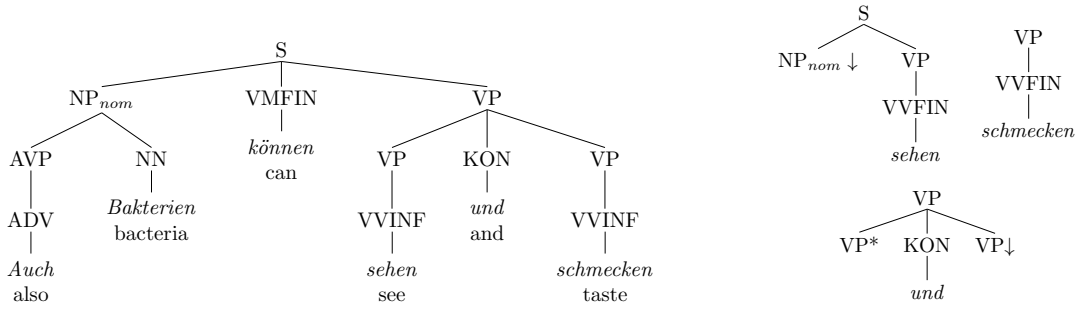


Figure 3.24: (a) Coordination without explicit recursion; (b) Coordination with explicit recursion thanks to maximal projections; (c) Coordinating auxiliary tree

The presented account for coordination works well for the straightforward cases, but for more complex coordinating constructions, LTAG fails to provide a satisfying analysis. Consider for example the tree in Figure 3.25(a), in which two VPs are conjoined. The elementary trees that are needed to derive the coordination are depicted in Figure 3.25(b). The elementary tree anchored in *schmecken* violates the co-occurrence principle (Section 3.1), because its dependent, the subject, is not localized within its elementary tree. Roughly speaking, the subject (as well as the S node) is shared by *sehen* and *schmecken*, but the notion of sharing cannot be implemented in a standard TAG analysis. Sarkar and Joshi (1996) suggest an account for coordination in TAG which involves a new operation *conjoin* and the notion of *contraction* of nodes (namely the nodes which we perceive as being shared in Figure 3.25(a)). However, not even the XTAG grammar, for which this account of coordination has been implemented, actually uses it because of the amount of ambiguities that it introduces for a parser (XTAG Research Group, 2001, p. 224).

Since node sharing in coordination constructions is not annotated explicitly in the PTB, it is difficult for grammars extracted from it to model the dependencies involved in predicative coordination. Elementary trees which do not conform with the co-occurrence principle and whose domain of locality is inappropriately small, like the one for *schmecken*, are therefore accepted (e.g. (Xia, 2001), (Chen, 2001) and also the PLTAG for English).

In contrast, in the Tiger Treebank, elided constituents in conjuncts are indicated via secondary edges (cf. Section 2.4.3). Figure 3.26 shows a graph with two secondary edges. With the presented account for coordination, the elementary tree for *liebe* is missing an NP substitution node for the object and could therefore be mistaken for an intransitive



(a) Derived tree (b) Some elementary trees

Figure 3.25: A derived tree and some elementary trees extracted from it

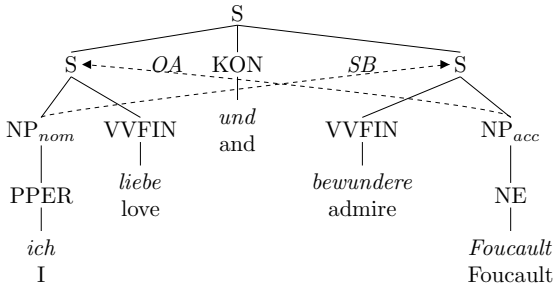


Figure 3.26: Parse tree with secondary edges for shared nodes

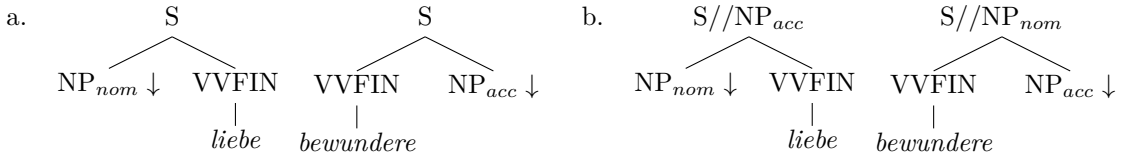


Figure 3.27: Verbal elementary trees extracted from the tree in Figure 3.26: (a) without taking the secondary edges into account; (b) with markings for the shared nodes

verb, while the elementary tree for *bewundere* does not have an NP substitution node for its subject (Figure 3.27(a)).

Without deciding on the strategy for coordination that a parser can handle, it is difficult to make use of the secondary edges in the grammar, since a variety of options seems to be possible. On the one hand, following Sarkar and Joshi (1996), the secondary edges should be used to infer the structure of the "original" elementary tree in which all dependents are specified. This would also improve the coverage of the grammar. The parser then must include a conjoin operation. On the other hand, assuming a parser that can only handle the standard TAG operations substitution and adjunction (and sister-adjunction), the secondary edges can be left unconsidered. In this case it would rather make sense to increase the

lexicon with the help of a mechanism (e.g. lexical rules) that systematically provides trees with elided arguments, even if the anchor has not been seen in a predicative coordination before.

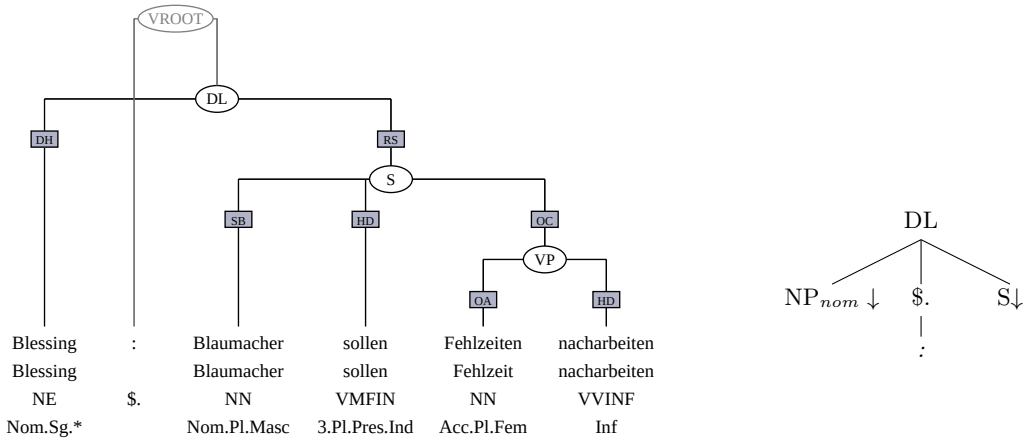
Since the extracted German grammar should be useful to a broad audience, none of the specific strategies is currently assumed and implemented. Instead, the secondary edges are preserved as links in the converted treebank, and nodes that have an incoming secondary edge for an argument or head carry this information in the extracted elementary tree (see the S nodes in Figure 3.27(b)). As a result, the elementary trees which are missing nodes due to their role in a coordination can be easily identified in the grammar and distinguished from elementary trees which obey the co-occurrence principle and have an appropriate domain of locality. With this information at hand, the marked elementary trees can be extended, changed or completely removed depending on the account of coordination supported by a specific parser, or only the additional information can be disregarded if not needed. (One could also imagine a parser that directly makes use of the information. Using such a tree in a derivation that is marked for missing a specific constituent would then be more probable if a corresponding constituent is present in the complementary conjunct.)

Even with this extension, some issues remain: As can be seen in Figure 3.25(a), not all nodes that would be shared/contracted in a TAG with a special account for coordination are also marked with secondary edges in the Tiger Treebank. The difference to the shared subject in Figure 3.26 is that in the first both verbs and the subject are dominated by a common S node, such that the verb-subject dependency is still represented in the tree. For an ideal analysis of coordination, those cases should be identified and considered during extraction (i.e. the tree template for *schmecken* should be the same as for *sehen* in case the parser can handle contractions). Furthermore, elided constituents other than arguments and heads are difficult to represent in our TAG lexicon in a theory-neutral way. Modification that relates to both conjuncts, for example, cannot be marked in the modified elementary tree, since modifiers are factored into their own elementary trees and we do not want to make modification obligatory. However, an additional (semantic) component is needed with a parser for this tree-adjoining grammar anyway if the exact scope of modifiers (as annotated in the Tiger Treebank with discontinuous constituents) is desired, see Section 3.3.4. The corresponding secondary links are furthermore not lost but preserved in the converted treebank, so appropriate models can be learned from there if required.

3.3.6 Punctuation

The treatment of punctuation is largely neglected in the parsing and grammar formalisms literature. Often punctuation symbols are simply removed. However, it is reasonable to include punctuation in a PLTAG lexicon since its goal is to model human processing of naturally occurring sentences, which generally contain punctuation marks. In the English PLTAG (Demberg-Winterfors, 2010), appositions are anchored by a comma. Otherwise

(3.18) *Blessing : Blaumacher sollen Fehlzeiten nacharbeiten*
Blessing : people who skip work should times absent rework
'Blessing: People who skip work should make up for their times absent'



(a) Tiger graph; the translation is provided in (3.18).

(b) Elementary tree anchored in a punctuation mark

Figure 3.28: Discourse level constituent

punctuation symbols are treated as modifiers, or removed if being sentence-final.

Thanks to the sister-adjunction operation in the German grammar, it is straightforward to include punctuation marks as modifier trees in the lexicon. If it turns out in the future that a corresponding parsing model cannot handle all punctuation marks adequately, they could still be disregarded partially or completely.

Some punctuation symbols are also treated as arguments, for example in a threepart coordination, where the first and second conjunct are separated by punctuation. Others also anchor larger elementary trees and subcategorize for constituents. This is for example the case in some coordinating constructions (see Figure 3.22(b)) and in 'discourse level constituents' (DL). The latter is a phrasal category which is annotated if there is no clear syntactic relation between a sentence of reported speech (*RS*) and the embedding clause (*DH* for discourse-level head). Figure 3.28(a) illustrates this with an example. After the conversion which, amongst others, attaches the colon to the DL node to resolve the crossing branch, the elementary tree in Figure 3.28(b) is extracted. It provides the connecting structure of the sentence and allows for the induction of standard NP and S trees, i.e. none of them projects to the DL category.

3.4 Prediction Trees

The extraction of prediction trees will be detailed in Section 4.2.3. Since prediction granularity is an open research question that goes beyond the scope of this work, the

(3.19) *Der texanische Milliardär hat das politische Establishment aufgeschreckt*
the Texan billionaire has the political establishment startled
'The Texan billionaire startled the political establishment'

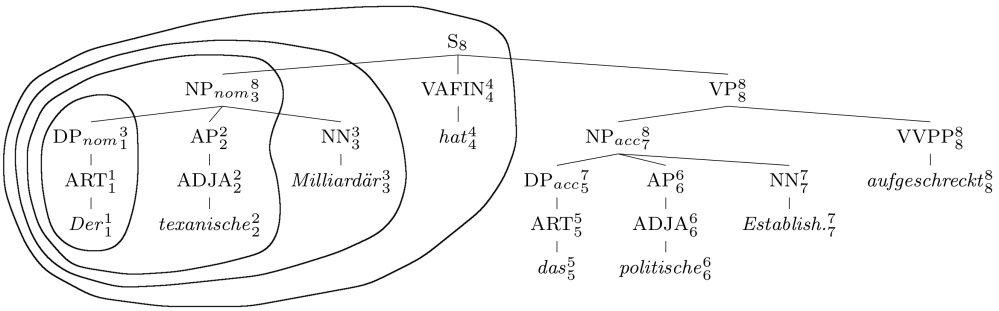


Figure 3.29: Structure that is required to connect each of the first four words with the previously seen words respectively. Nodes are numbered according to the canonical tree to which they belong. The translation is provided in Example (3.19).

definition of predictions trees as it is given for English by Demberg-Winterfors (2010) is adopted for the German prediction trees as well. This section focuses on the issues and interesting details that were encountered when transferring the definition to the German PLTAG lexicon.

3.4.1 Shape of the Prediction Trees

As it was done for the English PLTAG prediction lexicon, only those prediction trees are extracted which are required to derive the sentences in the Tiger Treebank strictly incrementally. Furthermore, prediction trees are always derived from canonical elementary trees. This configuration makes sure that a canonical tree is present in the lexicon which can verify the prediction during the derivation. Prediction trees do not have nodes to the right of the spine, and they do not have nodes at the bottom of the spine which are unary in the corresponding canonical tree. Given this configuration, prediction trees are not lexicalized.

Figure 3.29 illustrates the extraction and use of prediction trees for the discussion in this section. In order to combine the elementary trees of *der* and *texanische* into one syntactic structure, the NP node of the elementary tree of *Milliardär*, which provides the substitution node for the DP, is required. Since *Milliardär* has not been seen yet, the NP has to be predicted in form of the prediction tree in Figure 3.30(a). The predicted nodes with index j will be verified when *Milliardär* is read.

Even though the definition and extraction of prediction trees is the same for English and German, the granularity of the German predictions on the sentence level is different. This is related to the German sentence structure. Consider again the sentence in Figure 3.29. To

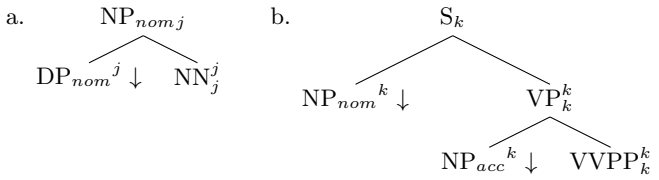


Figure 3.30: Prediction trees for a strictly incremental derivation of the first four words in Figure 3.29

integrate w_4, the auxiliary verb *hat*, into the partially derived tree, the root node S must be predicted. This node is provided by the elementary tree of the main verb of the sentence (*aufgeschreckt*) which is located in the right periphery. Following the specifications about prediction trees, the tree in Figure 3.30(b) is extracted during the lexicon induction and required when parsing the sentence incrementally. However, this tree signifies that when encountering a subject and an auxiliary verb, the grammar already predicts that an object is expected, i.e. that the main verb is transitive. This is questionable since, intuitively, it makes sense to say that 'something verbal' is expected, but not that it is a transitive verb. For English in contrast, the prediction would only go as far as the VP, specifying that a verb phrase is expected but leaving open the valency of the verb. The reason for this difference is the sentence structure: the NP daughter and the VVPP daughter of the VP node are reversed in a corresponding English parse tree. Consequently the object node is on the right side of the spine. It is therefore not included in the prediction tree.

This difference in the prediction granularity is formally not a problem. However, there is no psycholinguistic motivation for it, i.e. we do not know whether a speaker of German makes more fine-grained verbal predictions when processing a sentence than a speaker of English. This finding thus highlights the need for research to find the optimal prediction grain size. Furthermore, more fine-grained predictions increase the size of the prediction lexicon. We therefore expect to induce more prediction trees for German than Demberg-Winterfors (2010) extracted for English.

For Japanese it has been shown that the argument structure is predicted before encountering the sentence-final verb (Kamide et al., 2003). It is probably safe to assume the same for German, but the question is when exactly the argument structure is predicted. We do not know of psycholinguistic evidence that could shed more light on this issue. Intuitively, it seems more plausible to assume a two-step procedure: first predict the S node and the VP node for integrating *hat* (similar to the prediction in English, basically the upper half of the tree in Figure 3.30(b)) and then only predict more details once respective clues have been found, e.g. predict the object and the VVPP node (the bottom half of the tree in Figure 3.30(b)) when *das* is being integrated. In contrast, if the first NP is clearly marked for accusative, the subject and some participle could be predicted upon integrating an auxiliary verb. Such an approach is however not immediately realizable within the current PLTAG formalism. Some operation would have to be added that allows to combine the

(3.20) *Dies habe nicht zugesichert werden können*
this has not warranted been could
'This could not been warranted'

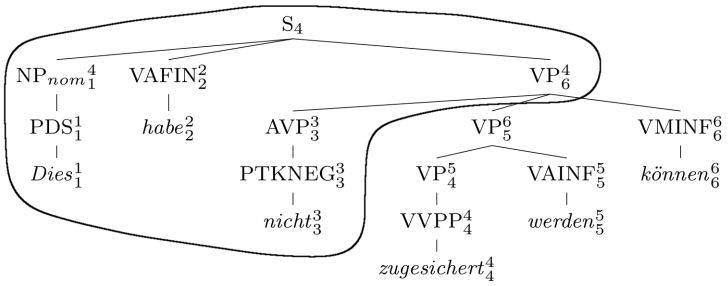


Figure 3.31: Structure that is required to connect the first three words. Nodes are numbered according to the canonical tree to which they belong. The translation is provided in Example (3.20).

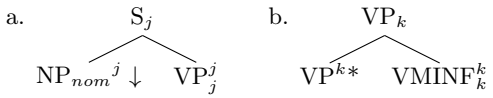


Figure 3.32: Prediction trees related to Figure 3.31

upper and lower fragments of the prediction tree at the VP node (for example some form of substitution). Additionally, one would have to make sure that they have the same prediction marker, since both the upper and the lower tree fragment would be validated together by the initial tree of *aufgeschreckt*.

Since no experimental evidence is available to support the preceding speculations, it is difficult to argue for one or another option. We therefore accept the change in prediction granularity for now which comes with adopting the conventions for English.

3.4.2 Sister-Adjunction and Prediction of Adjunction from the Right

Another point of discussion stems from the fact that the German grammar employs sister-adjunction for modifiers (and finite auxiliary and modal verbs, see Sections 3.2.2 and 3.3.1), while regular adjunction is used for those cases in the English (P)LTAG. For German this leads to configurations in which it would be necessary to predict adjunction of auxiliary trees from the right to be able to parse the sentence into the given treebank structure incrementally.

Consider the converted treebank tree in Figure 3.31 for illustration. The current point of interest is in integrating w_3 (*nicht*) into the partially derived tree structure of *Dies habe*. The nodes with index 4 in the circle have already been predicted by the prediction tree in Figure 3.32(a) prior to integrating w_2. If the modifier tree of *nicht* sister-adjoins to the

VP node without first predicting the adjunction of a modal auxiliary tree from the right (indicated by the bottom index 6 of the VP), the AVP ends up as the modifier of *zugesichert* instead of the modifier of *können*, as it can be seen in the derivation in Figure 3.33. To obtain the correct scope of the negation (namely as it is given in the treebank tree in Figure 3.31), the adjunction of the modal from the right has to be predicted (prediction tree in Figure 3.32(b)), see the derivation in Figure 3.34.

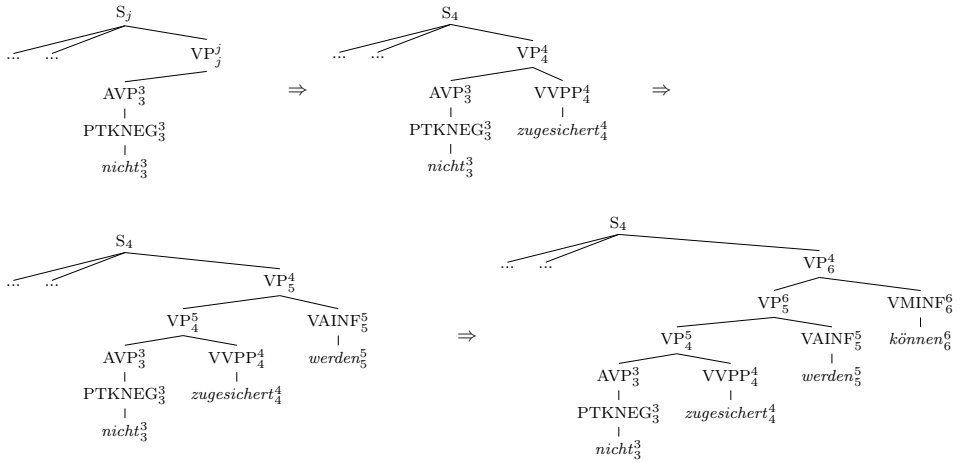


Figure 3.33: Derivation of the sentence in Figure 3.31 without predicting the adjunction of the modal from the right. The negation *nicht* has the wrong scope.

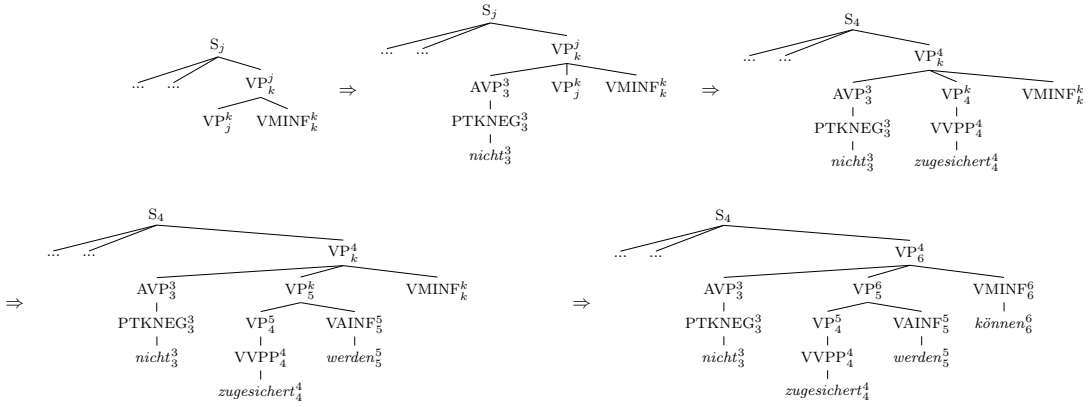


Figure 3.34: Derivation of the sentence in Figure 3.31 with prediction of the adjunction of the modal from the right (first depicted step). The negation *nicht* has the correct scope.

However, prediction of auxiliary trees that adjoin from the right is not desired, since it means determining the exact point of adjunction in advance. It also increases the size of the prediction lexicon, especially since the additional prediction trees might also be pre-combined with other prediction trees.

Furthermore, this prediction is not due to some special characteristics of the German language, but only to the choice of using sister-adjunction to be able to derive the flat treebank structures. In the English version, the treebank trees are binarized in order to enable regular adjunction of modifiers like *nicht*. It would be associated with a left VP auxiliary tree, leading to an additional VP layer in the derived tree (in comparison to Figure 3.31). Right auxiliary trees can then adjoin to one of the available levels without further complication, thereby disambiguating the structure depending on the node they adjoin to. For German in contrast, the alternatives have to be spelled out (in the form of the prediction tree) and kept available (in the form of competing parsing analyses) until the adjunction from the right actually happens.

So in fact, we are facing an attachment ambiguity here, which has to be resolved at some stage. In an English analysis, it is resolved with the point of adjunction when the right auxiliary tree adjoins. In the German PLTAG using sister-adjunction, the ambiguity has to be "anticipated" first, before it can be resolved in the same way as in English. We are not aware of literature or experiments which are concerned with the scope of modification/negation and the point at which such attachment ambiguities are resolved.

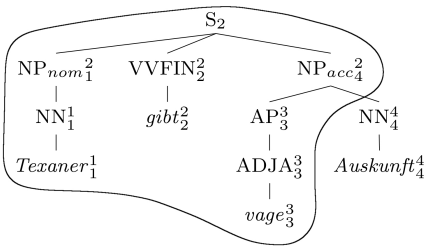
An alternative is to do without prediction of adjunction from the right, i.e. to accept the derivation in Figure 3.33 as the output of the core syntactic parser. A separate recursive mechanism would then operate on the derived tree and solve the ambiguity by possibly re-attaching a modifier tree at the foot node of some auxiliary tree to the former root node of the auxiliary tree. It could be trained with respect to the converted treebank trees from which the lexicon is extracted.

Since the re-attachment mechanism lies beyond the originally defined PLTAG formalism, it is not an ideal solution either. The lexicon extraction therefore implements two variants which can be chosen via a parameter: (a) predicting adjunction from the right if necessary to obtain the correct scope of a modifier tree or (b) no prediction of adjunction from the right. We found that strategy (a) yields only 3-4.5% more pre-combined prediction trees than strategy (b).

A closely related issue is whether sister-adjunction to an open substitution node is allowed in an incremental derivation. For illustration, consider the converted treebank tree in Figure 3.35(a). When *vage* is read, the object NP is an open substitution node. It has to be decided whether it be possible for the AP modifier tree to sister-adjoin to it, or whether the prediction tree in Figure 3.35(b) is required. It would first be substituted such that the NP is a full node before the AP modifier tree sister-adjoins.

One could argue that the NP has already been predicted and that the substitution node will have to be filled sooner or later for a valid PLTAG derivation. It should thus be possible to sister-adjoin the AP to it. On the other hand, sister-adjunction to an open substitution node would create a very unusual partially derived tree: the substitution node will still be open, but it will not be located at the frontier anymore. The substitution operation,

(3.21) *Texaner gibt vage Auskunft*
Texan gives vague information
'Texan provides vague information'



(a) Structure that is required to connect the first three words. Nodes are numbered according to the canonical tree to which they belong. The translation is provided in Example (3.21).

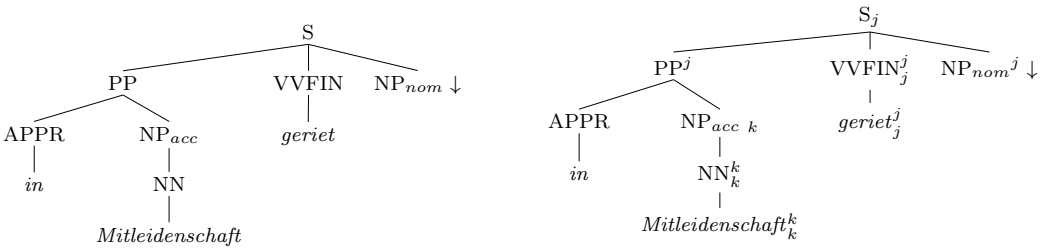
(b) Prediction tree that is required if sister-adjunction to an open substitution node is not allowed.

Figure 3.35: Sister-adjunction to a substitution node

or rather the substitution node itself, would consequently need a redefinition, such that substitution nodes can also be internal, non-frontier nodes. This seems unnatural for a TAG formalism. Therefore, we rather do not allow sister-adjunction to open substitution nodes during an incremental derivation and produce the necessary prediction trees with strategy (a) and (b). However, in the implementation an option (c) is available. It is based on strategy (b), but additionally allows sister-adjunction to open substitution nodes. Consequently, less prediction trees are extracted (∼12% less than with strategy (b)). Option (c) thus stands for a uniform interpretation of sister-adjunction within the PLTAG formalism as will be explained in the following.

Both issues - prediction of adjunction from the right because of a sister-adjoining modifier tree and prediction to fill a substitution slot before sister-adjunction - boil down to the same question: Looking at the final derived tree (which is the basis for the extraction), is sister-adjunction to a node n possible, if the bottom half (in relation to the final derived tree) has not been seen yet? The unseen bottom half corresponds to the index 6 of the VP in Figure 3.31 and index 4 of the NP in Figure 3.35(a). Those issues are of course not relevant for non-incremental derivations because the order of operations does not matter, so sister-adjunctions can always be the last actions performed on a node. It should therefore be emphasized here that even though sister-adjunction has been successfully used by Chiang (2000) for TAG parsing, one needs to be careful when adopting it in the incremental PLTAG formalism. Besides making the derivation of flat structures possible (and decreasing the size of the lexicon if Chiang's (2000) definition of sister-adjoining trees is used), it also raises questions that need to be answered, at the latest when a parser is implemented.

(3.22) *Besonders in Mitleidenschaft geriet das Amt für ...*
Especially into involvement got the department for ...
'The department for ... suffered especially'



(a) Tree without prediction markers

(b) Tree with prediction markers for the (invalid) assumption that *in* is the recognition point.

Figure 3.36: Elementary tree with three anchors. See (3.22) for the translation.

3.4.3 Predictions and Multi-Anchored Trees

As detailed in Section 3.2.3, multi-anchored trees are used in LTAG to represent expressions with non-compositional semantics. In PLTAG, multi-anchored trees are furthermore used for predictions on the lexical level.

So far, only trees with two anchors where the first one is predictive of the second have been considered in the context of PLTAG, see Figures 2.4(c) and 3.12. It is not clear how to treat multi-anchored trees with n anchors for $n > 2$ if the first i (with $1 < i < n$) anchors together activate the prediction for the subsequent anchor(s). For an example, consider Figure 3.36(a) and assume that *in Mitleidenschaft* is predictive for *geriet* and the recognition point of the collocation. We would therefore not want to predict the second and the third anchor after only having seen the first one. The tree that would be used for such a prediction is shown in Figure 3.36(b). However, if they are not predicted at that point, an additional operation is needed to check after the second anchor whether the first one has been seen in order to predict the third one.

In that context, it is also unclear how to compute a prediction tree based on such a multi-anchored canonical elementary tree. The notion of the spine of a canonical elementary tree is central to the computation of a corresponding prediction tree, but based on the aforementioned issue the notion of the spine also needs further discussion: it needs to be determined which of the n anchors defines the spine.

A further complication is that we currently do not know for the extracted German multi-anchored trees (cf. Section 3.2.3), in which cases the first i (with $1 \leq i < n$) anchors are actually predictive for the remaining ones. This will be tested in the future using co-occurrence statistics, as indicated in Section 3.2.3. So far we distinguish between the main anchor, which is the anchor of the elementary tree that provides the root of the multi-anchored tree, and the co-anchors. The main anchor is reached by following the head

path downwards starting from the root of the multi-anchored tree. Eventually, it needs to be shown how this distinction relates to predictivity.

If multi-anchored trees represent only non-compositional expressions and not predictions, predictivity and the status of the anchors is irrelevant. The main idea is to provide an idiomatic interpretation for the complete elementary tree which is not the same as the compositional interpretation of the individual anchors. Consequently, from the point of view of semantic interpretation, it does not make sense to exclude a multi-anchored tree only because the first anchor is not predictive for the following ones. Such issues can however only be fully worked out once the syntax-semantics interface for PLTAG is defined. Currently it seems that multi-anchored trees for expressions of non-compositional semantics have to be predicted from the first word on to be able to assign them their idiomatic interpretation. Otherwise, the interpretation would have to change from literal to idiomatic, which is not incremental.

In order to be able to extract a prediction lexicon at the current point of time, even though the interaction of syntax and semantics in PLTAG and the semantics of prediction trees are not yet specified, we need to make some assumptions and simplifications. We assume that the main anchor in multi-anchored trees constitutes the recognition point, i.e. the main anchor and all co-anchors left of it are predictive for the co-anchors on the right of the main anchor. We know that this assumption is problematic given how we currently determine the main anchor. However, once predictivity will be determined in an objective way, other anchors can easily become the main anchor. Furthermore, since so far there are no techniques to handle cases in which the first i anchors, with $i > 1$, are predictive for the following ones, we simply ignore multi-anchored trees in which the main anchor is not the leftmost one. Consequently, the multi-anchored tree in Figure 3.36 is not extracted as such, but as three individual elementary trees. Besides this strategy (ii) and the option of extracting no multi-anchored trees at all (strategy (i)), the extraction program also offers the option of inducing all multi-anchored trees, also those where the main anchor is not the leftmost one (strategy (iii)). This is targeted at people who are not interested in the prediction lexicon, but rather in pure LTAG and multi-anchored trees for idiomatic expressions. For reference, Appendix B provides all program options.

3.5 Summary

The most important and controversial points of a PLTAG lexicon for German have been discussed. Crucially, the German topological structure will not be modeled to keep the lexicon small, and sister-adjunction will be employed instead of binarizing the treebank trees. Some issues involving predictions in PLTAG arose, but could not be satisfactorily answered at the current point in time. The implementation will therefore allow the extraction of alternative lexica with respect to multi-anchored trees and prediction of adjunction from

the right. Otherwise the shape of the elementary trees in the target grammar has been laid out if not yet clear given the standard assumptions concerning natural language TAG.

Chapter 4

Treebank Conversion and Grammar Induction

The crucial points of the target (P)LTAG lexicon for German have been specified in the previous chapter. The first part of this chapter describes how the Tiger graphs are enriched to constitute (P)LTAG derived trees. They are the prerequisite for the lexicon induction, which can be regarded as the inverse process of parsing: the derived tree is segmented into its elementary trees. This process as well as the computation of prediction trees based on the converted treebank and the induced canonical elementary trees is provided in the second part.

The individual steps are implemented as methods in Java, that are called in sequence to manipulate and work on tree data structures representing the Tiger trees. The program runs on a typical personal computer (with at least 2 GB RAM). The output is the converted treebank and the PLTAG lexicon consisting of canonical elementary trees as well as prediction trees.

4.1 Treebank Conversion

The goal of converting the Tiger Treebank is to obtain derived LTAG trees from which elementary trees can directly be extracted in a second step by dividing the derived tree into its lexical components. The main points concerning the shape of elementary trees and thus also of the derived trees have been detailed in the description of the target grammar in Chapter 3. Due to the decisions to not model sentence structure following generative syntax and to use sister-adjunction, the overall backbone structure of the treebank trees does not have to be changed considerably. The main purpose of the conversion is thus to provide derived trees from which a lexicon with good generalization capacity can be extracted. Furthermore, as many argument traces as possible should be bound to their filler in one elementary tree. The transformations increase the size of the treebank in terms of nodes by almost 25%.

As the very first step, the Tiger graphs with crossing branches of the original treebank are transformed to trees without crossing branches in accordance with the standard practices in the literature (Kübler and Penn, 2008). In the most popular approach, the head of

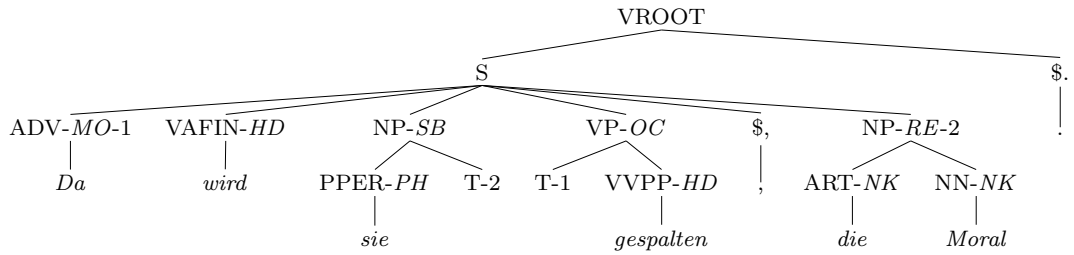


Figure 4.1: Tiger graph from Figure 3.20 with resolved crossing branches. Note the traces T-1 and T-2 and the corresponding fillers.

discontinuous constituent nodes is identified using the grammatical functions. The non-head daughters which cause the crossing branches are then attached to the minimally higher node for which the crossing branch is resolved. We slightly modified a script by Michael Schiehlen for this task.[1] It was originally written for the Negra Treebank, whose graphs follow almost the same annotation principles as those in the Tiger Treebank (cf. Brants and Hansen (2002)). The treebank is converted from Negra export format (Brants, 1997) to Penn Treebank format, i.e. to trees with indexed traces. Punctuation, which is attached to VROOT, also leads to many crossing branches. It is re-attached as high as possible without causing discontinuous constituents. In the tree in Figure 4.1, all crossing branches have been resolved.

Since secondary edges, lemmata and morphology are not part of the PTB format, this information is added from the export format to the tree data structures. Thus, all the information within the original treebank is made available to the conversion and the subsequent extraction procedures.

Reversibility Since the purpose of treebank grammars is parsing, we have to bear in mind the comparability of parsing results. For German, we do not know of a probabilistic TAG parser, let alone an incremental one, with which a future (P)LTAG parser would have to compare. Other parsers trained on the Tiger Treebank (e.g. PCFG parsers as mentioned in Section 2.4.3) are based on the original Tiger graphs (with prior conversion to context-free trees for CFG parsing, of course). We therefore aim at being able to revert the transformations that are presented in the following sections. To this end, a short discussion concerning reversibility is included for each transformation. Finally, if (P)LTAG parse trees can be reverted to the Tiger format, they can be compared easily to the gold treebank trees.

The treebank conversion is heavily based on the function labels, so the reversion would be simplified a lot if the syntactic functions were available in the parser output. They are, however, very unlikely to be part of a scalable PLTAG lexicon. (They are informative, but the treebank is probably too small to learn such fine-grained distinctions.) Considerations about reversibility in the following sections are therefore based on the assumption that

[1] Thanks to Aoife Cahill, who provided us with the script.

in the derived tree returned by the parser, no syntactic functions will be available. It is, however, assumed that the information about which node (more specifically, which node half) originates from which elementary tree is still present in the derived tree. Without this information, most transformations are reversible nevertheless, but fine-grained rules would be required for the reversion.

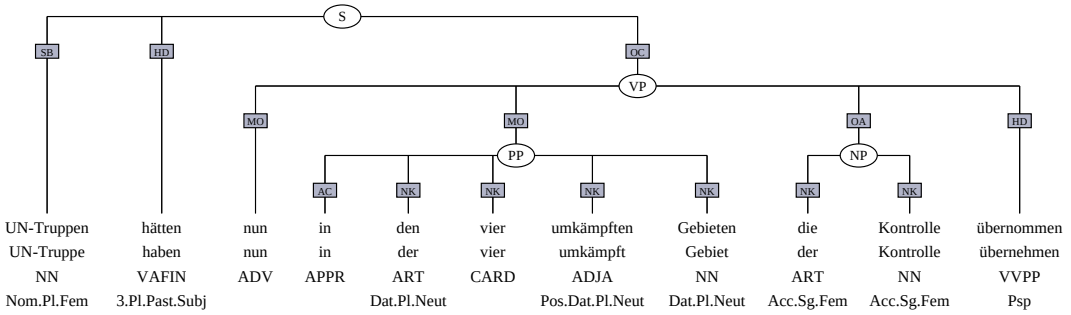
4.1.1 Linguistic Generalizations

In order to obtain a modular grammar with good coverage on unseen data, linguistic generalizations have to be added to the Tiger annotation. They are often missing because the annotation scheme minimizes the use of unary branching nodes. Consider the tree in Figure 4.2(a) for illustration. If it is taken as the basis for grammar induction without prior conversion, the elementary tree of the main verb would have an NN substitution node for the subject (*SB*). The elementary tree would thus not apply to nominal subjects in general, such as proper names or complex noun phrases like a noun with a determiner and/or a modifier. This desired generalization is only possible if all nouns project to a common category, for example NP. As a second example, consider the adverb *nun*. Like all categories, ADV does not have a maximal projection (AVP) unless it has its own dependents, e.g. in *spätestens nun* ('at the latest now') where *spätestens* modifies *nun*. However, in a modular grammar, we would not want two distinct trees for adverbs: one if they are modified and one if they are not. Each category should instead be supplied with the phrasal projection which is necessary for having dependents. The generalization is then that theoretically all adverbs can be modified, even if specific ones have not been seen with a modifier in the training data. This allows us to cover these adverbs in case they occur with a modifier during testing. As a last example, the flat prepositional phrase (PP) in Figure 4.2(a) does not explicitly state the generalization that prepositions subcategorize for a nominal group (of a specific case). All those desired generalizations can be made explicit with the help of the phrasal and functional categories as described in the following sections.

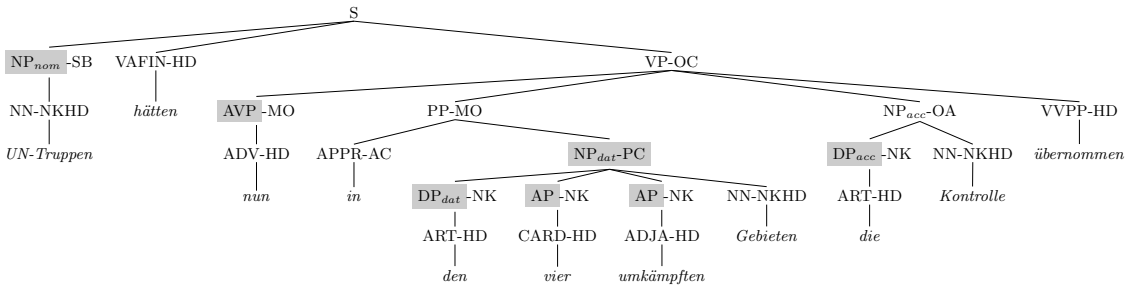
NP Complement for Adpositions

For adpositions, an NP node is inserted for the nominal complement. This is achieved by identifying the pre- and/or postposition of a prepositional phrase by the functional label *AC*. For all daughters right of the preposition, left of the postposition or in between the pre- and the postposition (i.e. a circumposition), a new mother node with the phrasal label NP and the functional category *PC* (prepositional complement) is inserted if at least one of these daughters has the function *NK* (noun kernel). *NK* implicitly marks the missing NP. See the PP in Figure 4.2(b) for an example. If there is only one daughter that would be grouped under NP, and it is of category NP (or CNP) or of one of the adverbial or adjectival categories (ADV, AVP, ADJD, AP etc.), no new NP node is inserted. An example for the

(4.1) *UN-Truppen* *hätten* *nun* *in* *den* *vier* *umkämpften* *Gebieten* *die* *Kontrolle* *übernommen*
UN troops would have now in the four contested areas the control taken over
'UN troops have now taken control of the four contested areas'



(a) Tiger graph. The translation is provided in (4.1).



(b) Converted (= derived) tree. The inserted nodes are shaded.

Figure 4.2: Adding linguistic generalizations and case annotation for noun phrases

latter is *von gestern* ('from yesterday'), which is excluded to avoid a nominal projection for an adverb. Instead, the elementary tree of *von* will subcategorize for an AVP.

With regard to reversibility, the NP complement of an adposition can easily be identified structurally. Furthermore, it will be encoded as a substitution node. The transformation can thus be reverted. However, if the corresponding NP node is the root of a coordination auxiliary tree, it will not be removed. Note that with this reversion, NP complements of adpositions that have already been annotated as such in the treebank will also be reverted. However, this is very rare, and the instances seem to be annotation errors anyway.

Phrasal Projections

For a coherent treebank and TAG lexicon, the main phrasal projections AP, AVP, VP, NP and DP are inserted. First, a treebank investigation using TIGERSearch (Lezius, 2002) reveals which categories should project to which phrasal category. For example, we find

New category p_2	dominated category P_1	function F_1
AVP	ADV PTKNEG	*HD PH AVC UC*
AP	ADJA ADJD MTA AA	*HD PH ADC*
VP	VMPP VAPP VVPP VAINF VVINF	*HD*
VP	VVIZU VZ CVZ	*HD CJ*
DP	ART PDAT PPOSAT PRELAT PWAT PIAT PDS-*AG*	*HD*
NP	NN NE PIS PPER PPOSS PRELS PRF PWS PN PDS[2]	*PH NKHD PNC ...* ... *NMC HD NK ADC*

Table 4.1: Inserting phrasal categories. The table is to be read in the following way: a unary node n_2 of phrasal category p_2 (shown in column 1) is inserted directly dominating a node n_1 if n_1 has the phrasal category $p_1 \in P_1$ (column 2) and not the syntactic function $f_1 \in F_1$ (column 3).

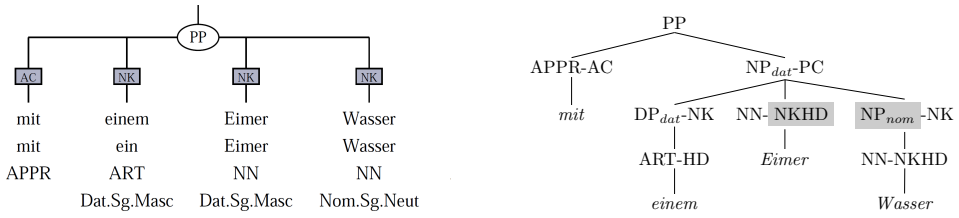
that, if ADV is the head (*HD*) of a phrasal category, the phrasal category is AVP in 99% of the cases, otherwise AP (probably due to annotation errors). From this, we conclude that a node n_1 with label ADV that is not already the head of some other node should be headed by an AVP node. We therefore introduce a unary node n_2 with label AVP directly dominating n_1, with n_2 having the former syntactic function of n_1 and n_1 carrying the function *HD*. The shaded node dominating *nun* in Figure 4.2(b) is an example.

However, we will not introduce n_2 if n_1 has the syntactic function *PH* (place holder), *AVP* (adverbial phrase component) or *UC* (unit component). We find that place holders in general and adverbial phrase components already project to their phrasal category. Examples can be found in Figures 3.20 and 3.11(c). Unit components form flat phrasal categories with the label ISU (idiosyncratic unit) or CH (chunk). As this indicates that they are somewhat linguistically special and should rather be treated as one unit, we abstain from introducing internal hierarchies. See Figure 3.11(d) for an example.

A very similar reasoning as for ADV/AVP holds for the other categories as well. Table 4.1 lists the conditions for introducing phrasal projections. See in particular lines 1 to 4.

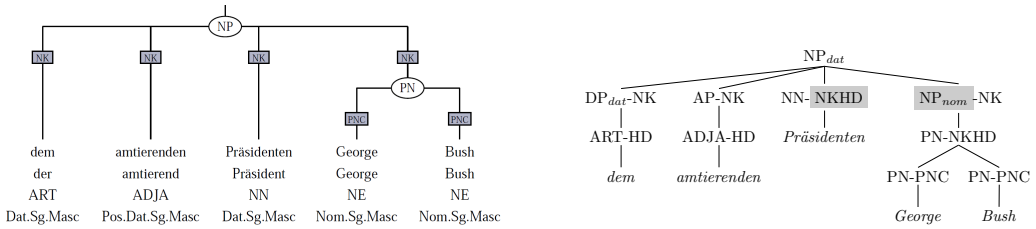
Determiners do not have a designated phrasal category in the Tiger annotation. In the few cases that they have a dependent, they head an AP node. However, adjectives and determiners are different categories in terms of distribution, closeness and encoding in a TAG lexicon. While adjectives are clearly modifiers, determiners are often considered obligatory for certain types of nouns. Since our grammar provides substitution nodes for determiners within noun phrases, we would like to distinguish between adjectives and determiners on a phrasal level. We therefore introduce a new phrasal category DP. See line 5 in Table 4.1 and the shaded DPs in Figure 4.2(b). The aforementioned AP nodes that are headed by a determiner category are relabeled.

[2]Excluding PDS nodes with function *AG*, as we treat those as determiners.



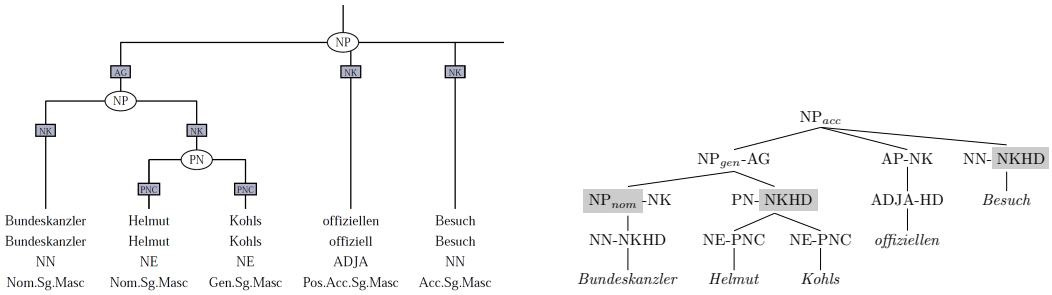
(a) Tiger graph (b) Converted tree

Figure 4.3: Disambiguation of the NP: *with [a bucket of water]*



(a) Tiger graph (b) Converted tree

Figure 4.4: Disambiguation of the NP: *the acting President George Bush*



(a) Tiger graph (b) Converted tree

Figure 4.5: Disambiguation of the NP: *[Chancellor Helmut Kohls] official visit*

Disambiguating Noun Phrases As mentioned in Section 2.4.3, NPs in the Tiger Treebank do not have an explicitly marked head, see the functional labels *NK* in Figure 4.2(a). Identifying the head of each NP is not only necessary for the lexicon induction step, but also for introducing linguistic generalizations: the non-head nominal material will be completed with phrasal projections and factored out of the NP during elementary tree extraction.

We follow a note in (Albert et al., 2003) saying that NPs can be structurally disambiguated given the POS information and the phrasal categories. Assuming that adpositions already have an NP complement and that the phrasal categories DP, AP and AVP have already been inserted, all *NK*-daughters of an NP that do not have the category DP, AP or AVP are considered potential heads. If the number of potential heads equals one, which is mostly the case, the head of the NP is found. It is given the functional label *NKHD*. The

sentence in Figure 4.2(b) contains three examples.

If there are two potential heads, our judgment is based on the case annotation principles as given in the annotation guidelines (Crysmann et al., 2005, Section 3). For NN+NN and NN+NP combinations (measurements, accusatives of time etc.), the first part is inflected for case and therefore considered the head. It thus receives the functional label *NKHD*. If not already present, an NP projection is inserted for the second part. Figure 4.3 shows an example. Note that the second part *Wasser* is annotated with the default case nominative while the first part *Eimer* has dative case. Dative is also the case of the complete NP as selected by the preposition *mit*.

If the second part is NE or PN, the NP is a close apposition. If it includes a determiner (DP or APPRART if the NP is the complement of an adposition), again, the first part is inflected for case and therefore constitutes the head. See Figure 4.4 for an example. If there is no determiner, the second part is inflected for case and thus the head, as it can be seen in Figure 4.5 where the NP-*AG* as well as *Kohls* is in genitive case.

A very small percentage of NPs is not covered by our informed disambiguation, e.g. NPs with three or more potential heads. More investigation about the underlying regularities and corresponding fine-grained rules are needed for the disambiguation to happen in a sensible way. We leave this for future work. Note that for elementary tree extraction, these NPs will still be assigned a head by a less informed head-finding procedure later (cf. Section 4.2.1).

After this NP-internal disambiguation, the treebank is completed with NP projections in the general manner, subject to the condition in line 6 in Table 4.1.

In Coordinations We know from Section 3.3.5 that recursion in the tree is required in order to extract coordinating elementary trees. This recursion is often not immediately present in the treebank, but revealed thanks to inserting the maximal projections. (An example was shown in Figure 3.24.) Since the Tiger Treebank has special categories for coordinated adpositions (CAC) and *zu*-marked infinitives (CVZ), some nodes in addition to the main phrasal categories need to be introduced in order to provide the required recursion. These conditions are presented in Table 4.2, entries 1 to 3.[3] Figure 4.6 illustrates why *zu*-marked infinitives require a special treatment in coordinations. A node of the category VVIZU can either be a conjunct in (a) a coordination of pure *zu*-marked infinitives or in (b) a coordination of VPs, where the *zu*-marked infinitive is an intransitive verb without modifiers.

There are some categories in the Tiger annotation for which it is not straightforward to determine the phrasal projection, mostly because they are used in different contexts. One example is the POS category PROAV (pronominal adverb), e.g. *dafür* (for it), *wovon*

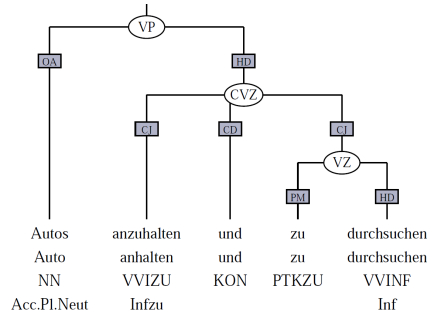
[3]Clearly, this can be interpreted as calling for the general insertion of AC and VZ phrasal category. They are, however, different from the inserted maximal projections in that they do never provide an attachment point for dependents: adpositional modifiers attach to PP and modifiers of a *zu*-marked infinitive attach to the corresponding VP node. Additional AC and VZ nodes are only required in coordinations. Inserting them in a more general way might be investigated in future work.

New node category p_2	dominated category P_1	function f_1	parent category p_3
AC	APPR APPRART APPO APZR	*CJ*	CAC
VP	VVIZU VZ	*CJ*	CVP
VZ	VVIZU	*CJ*	CVZ
AP	NM CARD	*CJ*	CAP
NP	NM CARD	*CJ*	CNP

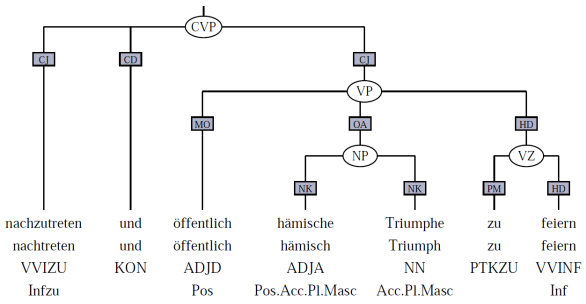
Table 4.2: Inserting phrasal categories for specific conjuncts. The table is to be read in the following way: a unary node n_2 of phrasal category p_2 (shown in column 1) is inserted directly dominating a node n_1 if n_1 has the phrasal category $p_1 \in P_1$ (column 2), n_1 has the syntactic function f_1 (column 3) and the current parent node of n_1 has the phrasal category p_3 (column 4). n_2 receives the functional category of n_1, while n_1 receives the functional label *HD*.

(4.2) *Autos* *anzuhalten* *und* *zu* *durchsuchen*
cars stop+PARTICLE and PARTICLE search through
'. . . to stop and search through cars'

(4.3) *nachzutreten* *und* *öffentlich* *hämische* *Triumphe* *zu* *feiern*
put the boot in +PARTICLE and publicly malicious triumphs PARTICLE celebrate
'. . . to put the boot in and to celebrate malicious triumphs publicly'



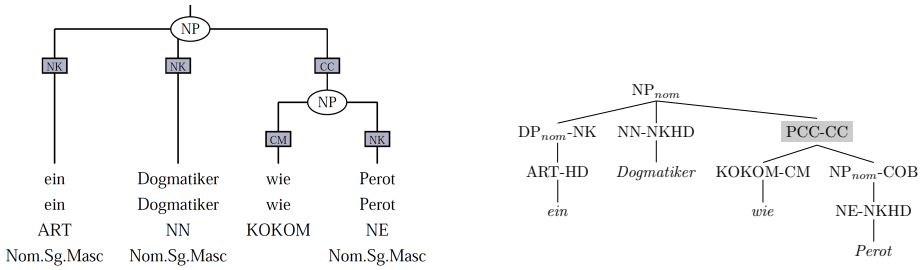
(a) Coordination of VZs. A VZ node dominating *anzuhalten* will be inserted. The translation is provided in (4.2).



(b) Coordination of VPs. A VP node dominating *nachzutreten* will be inserted. The translation is provided in (4.3).

Figure 4.6: *zu*-marked infinitives in coordination

(whereof). Out of 4949 occurrences, it is only marked as a head 63 times, thereof 60 times as the head of an AVP node, suggesting it should project to the AVP category. However, in 1093 cases it has the function 'place holder', thereof 1090 times dominated by a PP. From this, we would rather infer that PROAV is used with the distribution of a PP. In coordination constructions, it occurs with the dominating category CPP as well as CAVP, thus also not providing further insight. Another example is the POS category CARD (cardinal number) and the related phrasal category NM (multi-token number). Both can occur in the same contexts as adjectives but also in the same contexts as noun phrases. We



(a) Tiger graph (b) Converted tree

Figure 4.7: Comparative complement structure: *a dogmatist [like Perot]*

conclude from these investigations that, with rules in the style of Table 4.1, inserting phrasal projections for such heterogeneous categories would not make the treebank more coherent. More fine-grained and context-dependent rules are needed, but this detail is left for future work. However, the extraction procedure fails for a number of sentences because of non-recursive coordination structures that involve categories for which no maximal projection was inserted. To minimize this number, we insert the corresponding phrasal nodes that are required for recursion for the most frequent cases that would otherwise fail. Those are exactly the aforementioned number categories, see entries 4 and 5 of Table 4.2.

Reversibility The insertion of phrasal projections always results in a unary node. Unary nodes are extremely rare in the Tiger Treebank, so these configurations are easily identifiable for reversion. For the few cases where such configurations are already present in the original treebank, there is the risk to revert them as well. However, they are mostly annotation errors, or structures which involve a proper noun (PN), for which it seems that more fine-grained reversion rules could help. DPs under which a modifier tree is sister-adjoined in the parser output have to be relabeled as AP.

Comparative Complement Phrases

Comparative complements are marked by the syntactic function *CC* in the Tiger Treebank that is introduced by a comparative conjunction (*CM*). They can virtually occur with any phrasal category. Figure 4.7(a) shows an NP as an example. To be able to distinguish a category that acts as a comparative complement (e.g. NP-*CC*) from the corresponding "ordinary" category (e.g. NP) in a general way without being dependent on the usage of the syntactic functions, we unflatten the phrases that are introduced by a comparative conjunction. A node of the new phrasal category PCC (comparative complement phrase) is inserted. This node has two daughters: the comparative conjunction and the node that was originally marked with *CC* minus its comparative conjunction daughter. This is illustrated in Figure 4.7(b). By encoding *wie* as a modifier tree with an NP substitution node and *Perot* as an NP initial tree, the obtained tree structures are more uniform.

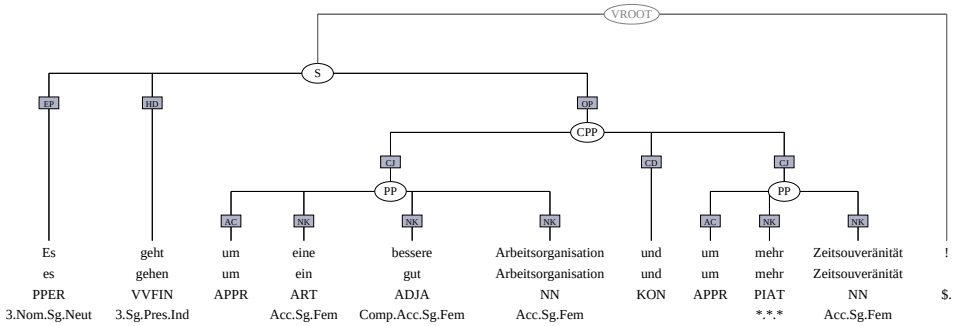


Figure 4.8: Tiger tree in which the CPP label will be replaced by PP during the conversion. Furthermore, the exclamation mark will become a child of S, such that the VROOT can be removed.

For reversion, PCC nodes and their *CM* daughters are identifiable in a straightforward way: PCC does not belong to the original Tiger category inventory, and the comparative conjunction builds the lexical anchor of the elementary tree whose maximal projection is the PCC node.

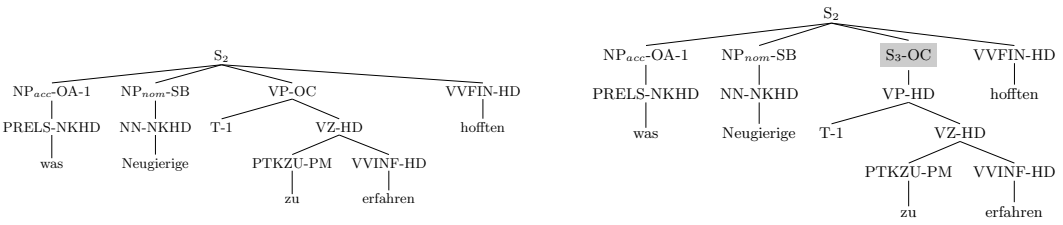
4.1.2 Miscellaneous

In the transformation steps described in the following, S nodes in relation to predicative auxiliary trees are inserted, and the VROOT node is removed if possible. Furthermore coordination nodes are relabeled and morphological information is added to inner nodes of the tree. Some of these steps can be accomplished in a more accurate way if information about head (and argument) children is already present in the derived tree. The head-argument-modifier classification, which will be presented as part of the extraction procedure (Section 4.2.1), is therefore actually performed after the introduction of linguistic generalizations. The following transformations make sure to not break this distinction when inserting new nodes.

Coordinated Categories

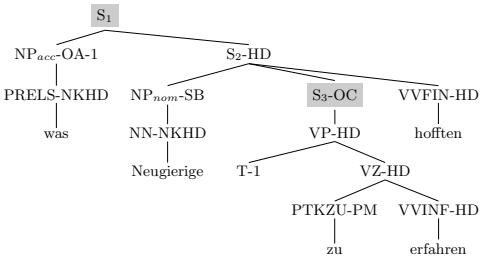
In Section 3.3.5, coordination constructions were shown to be recursive and therefore implemented as auxiliary trees in the German (P)LTAG. The Tiger annotation makes it easy to identify coordination thanks to the coordinated categories (Section 2.4.3). They however need to be replaced by their non-coordinated counterpart (e.g. CNP by NP) for the recursion. In Figure 4.8, the Tiger tree which corresponds to the converted tree in Figure 3.21 is depicted for illustration. The CPP node is relabeled as PP during the conversion. Information identifying a coordination structure is stored as a property of the node which will be exploited for lexicon induction.

The CO label is replaced with the label of its leftmost conjunct (*CJ*). Note that for a perfect relabeling, the nodes of the tree are traversed bottom-up.



(a) Partially converted tree. The matrix verb *hofften* embeds a verbal object VP-*OC*. Linguistic generalizations have already been introduced.

(b) The S node which will later provide the foot node of the auxiliary tree of *hofften* is inserted.



(c) Additional S level to describe scrambling of the object of the embedded verb is inserted.

Figure 4.9: Insertion of S nodes for predicative auxiliary trees. This conversion procedure relates to Figure 3.19. The translation is provided in (3.11). (The subscripts are not part of the alphabet.)

These transformations can be reverted by adding the 'C' again to the category of nodes of which the bottom half constitutes the root of a coordination auxiliary tree. In case unlike phrasal categories are conjoined in a coordination auxiliary tree, the root node label has to be changed back to CO. A few constituents in the Tiger Treebank are annotated as being a coordination, but they neither dominate a conjunction nor a punctuation symbol which could provide the anchor of the coordination auxiliary tree. For those (anomalous) constructions, no auxiliary tree is extracted. If a parse tree contained such an instance, it would not be possible to revert the labeling.

Additional S Nodes for Predicative Auxiliary Trees

Two types of S nodes are inserted into the Tiger trees in relation to predicative auxiliary trees: S nodes that are required for recursion and later provide the foot node in the auxiliary tree of the matrix verb, and S nodes that help to localize trace-filler pairs within the elementary tree of the embedded verb.

S nodes for recursion are required since raising and control verbs in the Tiger annotation embed a VP, but project to an S node (cf. Section 3.3.2). An example is shown in Figure 4.9(a). If a clausal object VP (VP-*OC*), let's call the node x, is an argument (according to the classification in Section 4.2.1), it is decided whether the S node should be

inserted by checking whether the embedding construction will be encoded as an auxiliary tree later. Roughly, this means that a node n of category S has to be found which will be the root node of the predicative auxiliary tree. n must either directly dominate x or n must be reached by following the path of head children upwards, starting from the node that directly dominates x. If the embedding construction is identified as a predicative auxiliary tree, a node n' of category S-OC is inserted that directly dominates x. x becomes the head (HD) of the new node n'. In Figure 4.9(b), Node S_2 corresponds to n and S_3 corresponds to n'. The details of how predicative auxiliary trees are identified structurally are presented in Algorithm 1, which follows ideas by Chen (2001). The node r represents the head path upwards. The configuration we are interested in is described in lines 9 to 13.

Algorithm 1 Predicative Auxiliary Tree Identification

Require: T (derived tree)

```
1:  ▷ mapping from foot nodes to root nodes
2:  initialize ψ_T(x)
3:  for all nodes x of T in depth-first, top-down order do
4:      ▷ Search for potential foot nodes
5:      if x is arg and x.functCat = OC and (x.phrasalCat = S or VP) and x is not a
        filler then
6:          ▷ Search for the potential root r of the pred. auxiliary tree along the head path
7:          initialize node r ← x.parent
8:          while r ≠ null do
9:              ▷ S predicative auxiliary tree
10:             if r.phrasalCat = S then
11:                 if x.phrasalCat = VP then
12:                     insert new S node n' dominating x as specified in the text
13:                     x ← n'
14:                     ▷ Additionally insert an S node for the scrambling config. (see text)
15:                 r.pred ← true
16:                 ψ_T(x) ← r
17:                 break
18:             ▷ VP predicative auxiliary tree for modal and auxiliary verbs
19:             if r.phrasalCat = VP and x.phrasalCat = VP then
20:                 if head of r is modal or auxiliary verb then
21:                     r.pred ← true
22:                     ψ_T(x) ← r
23:                     break
24:             if r is head child then
25:                 ▷ Skip nested predicative auxiliary trees
26:                 if ψ_T(r) ≠ null then
27:                     r ← ψ_T(r)
                    r ← r.parent
28:             else
29:                 r ← null
30: return ψ_T
```

To be able to adequately handle scrambling in the grammar (cf. Section 3.3.4), it is furthermore checked whether the leftmost daughter d of n is an argument (again according to the classification in Section 4.2.1) and a filler, i.e. it is an argument, but not of n's head. Furthermore, if the trace which corresponds to the filler d is found in the subtree that is dominated by n', another new node n'' is inserted dominating n. It has the phrasal category S and the functional category of n, while n becomes the head (*HD*) of n''. Node d (and its subtree) is furthermore moved from its location as a daughter of n to be the left daughter of n''. Figure 4.9(c) illustrates this with an example: S_1 corresponds to n''. This description refers to line 14 in Algorithm 1.

We have seen that it is necessary to already identify future predicative auxiliary trees during the conversion, in order to insert the S nodes in the correct places. To avoid doing the same work again, and since the extraction of predicative auxiliary trees stands apart from the standard extraction procedure anyway (cf. Section 4.2.2), the roots of predicative auxiliary trees are already marked as such during the conversion procedure in the implementation ($r.pred \leftarrow true$ in Algorithm 1).

This solution also helps to cover an issue which we have not considered so far. When trying to identify the future root node n, the procedure might first find the foot node of another auxiliary tree, since auxiliary trees can be nested. On the search for potential foot nodes, we therefore go through the tree in a top-down manner, while looking for the future root n works bottom-up. This guarantees to always find the innermost future auxiliary tree first. A 1:1 mapping ψ_T from foot nodes to root nodes is then build up for each derived tree T. As a result, potential nested auxiliary trees can easily be skipped during the bottom up search for n along the head path (lines 25 to 27).

With a view to reversibility, the inserted S nodes produce configurations that originally do not exist in the Tiger Treebank. Thus, it is straightforward to identify and revert them.

Case Marking

The morphological annotation in the leaf layer of Tiger is projected upwards in the derived tree. Currently, this is restricted to the annotation of NPs and DPs with a case feature, but the implementation allows for extensions.

Thus, for NP and DP nodes, the corresponding case is obtained bottom-up from the head child. If it is underspecified there, the procedure attempts to infer it in a top-down fashion: if an NP has the functional label *SB*, *PD* or *EP*, it is annotated for nominative case, *OA* and *OA2* trigger accusative case, while *OG* and *DA* lead to genitive and dative case respectively.

Removal of the Virtual Root

The virtual root (VROOT) is a technical artifact of the Tiger treebank to by able to provide a single root node for each sentence. It does not have a linguistic motivation. We therefore

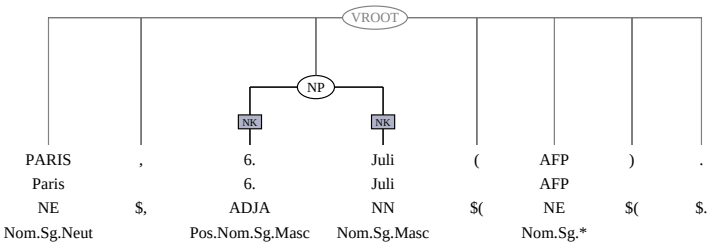


Figure 4.10: Example sentence where the VROOT cannot be removed: *PARIS, July 6 (AFP).*

remove it if possible as described in the following.

Since many nodes that were originally children of VROOT, mostly punctuation, have been attached lower as part of the conversion to PTB format, the VROOT node is technically not required anymore in many cases. However, the specific re-attachment strategy we used only lowered those nodes that caused crossing branches. As a consequence, sentence-initial and sentence-final punctuation marks are still attached to VROOT. We check whether VROOT has exactly one phrasal daughter node n when disregarding punctuation. If this is the case, the remaining punctuation daughters of VROOT are attached to n.

Finally, VROOT is removed from a tree if it has exactly one daughter, which becomes the new root of the tree. Otherwise, VROOT cannot be removed without destroying the tree structure, so it is preserved. The latter holds for expressions like the one depicted in Figure 4.10. In Figure 4.8 in contrast, the sentence-final exclamation mark will be made a daughter of S in this step and the VROOT node will be removed.

To revert these transformations, a VROOT node has to be inserted as the new root in case such a node is not already present in the parse tree. Sentence-initial and sentence-final punctuation which is attached to the root of the parse tree can then be re-attached to VROOT.

4.2 Extraction procedure

After converting the trees in the Tiger Treebank to LTAG derived trees, the (P)LTAG lexicon is induced from them. The transformed structures are annotated with head information and a distinction between modifiers and arguments which is required for the lexicon induction. Then the derived trees are segmented into canonical LTAG elementary trees. Finally, those lexicon entries are taken as the basis for inducing the PLTAG prediction lexicon.

4.2.1 Head-Argument-Modifier Distinction

One of the essential parts of the extraction procedure described in (Xia et al., 2000) and in related approaches is to identify the head child of each non-terminal node in the tree and to

Node category p	child functions F	search direction
AA	*HD*	-
AC	*HD*	-
AP	*HD PH*	-
AVP	*HD PH AVC*	from left
CH	*UC*	from right
DL	$\star$	
DP	*HD*	-
ISU	*UC*	from left
MTA	*ADC*	from left
NM	*NMC*	from right
NP	*NKHD PH HD*	-
PCC	*CM*	-
PN	*PNC*	from right
PP	*AC PH*	-
VP S	$\star$	
VZ	*HD*	-
VROOT	$\star$	
coord	*CD* $\star$	

Table 4.3: Head identification. The table is to be read in the following way: For a node with the phrasal category p (shown in column 1), do a one-pass search through its children as given in the search direction (column 3). The first child with function $f \in F$ (column 2) is the head. The star $\star$ means that an explanation is given in the text (Section 4.2.1).

flag each of the remaining children as either an argument or a modifier of the head. For the PTB, this distinction is based on heuristics, such as a head percolation table in the style of Magerman (1994). The lexicon induction in (Demberg-Winterfors, 2010) partly resorts from these heuristics by additionally disambiguating noun phrases and adding PropBank annotation (Palmer et al., 2005) to help the argument-modifier distinction.

In the Tiger Treebank, syntactic functions are annotated as presented in Section 2.4.3. For almost all categories, they specify the syntactic head, and they provide indicators for whether nodes should be considered as arguments or modifiers. Thus, apart from few exceptions, no heuristics are needed for the head-argument-modifier distinction. In the following, the details of the classification will be provided.

Head Identification

For each non-terminal node in the tree, exactly one of the children has to be marked has the head child. For most categories, the grammatical functions annotated in the Tiger Treebank determine a unique syntactic head. It is marked as such in one pass through the children considering their functional categories. Table 4.3 lists the details. If the head is unique, the search direction should not matter (-). In the implementation, the search starts from the left in those cases. In Figure 4.11, the identified head children are boxed.

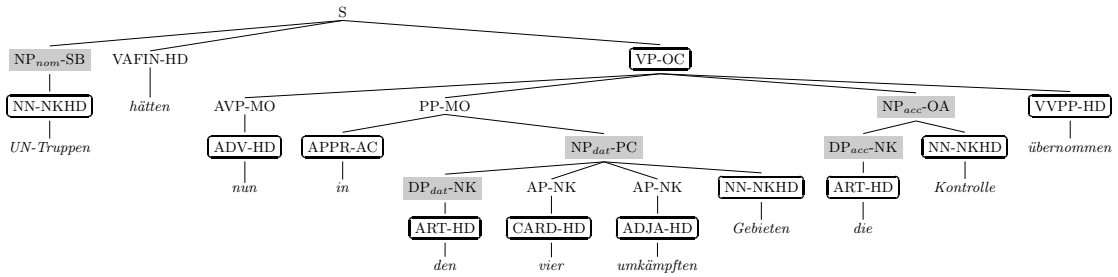


Figure 4.11: The same converted tree as in Figure 4.2 but with heads (boxed) and arguments (shaded) marked. The other nodes (except root and terminals) are modifiers.

For some constituents, the head is not clear from a linguistic point of view. In those cases, the Tiger annotation leaves the decision open and gives the same function to each "potential" head. Those functional labels are *AVC*, *UC*, *NMC*, *ADC* and *PNC*. Optimally, some of them are later encoded as multi-anchored trees as specified in Section 3.2.3, but for the algorithm to work a unique head has to be identified. We therefore decide for the leftmost of the potential heads without a specific reason, with the exception of multi-token numbers (NM), proper nouns (PN) and chunks (CH). In multi-token numbers, the rightmost component can be a noun (e.g. *Millionen*), so it makes sense to consider it as the head. In proper nouns, the rightmost component sometimes is marked for case, which is a feature that should project upwards in the tree, so the rightmost PNC is chosen as the head. Chunks group all kinds of things, for example foreign language material. Since they sometimes correspond to noun phrases, we consider the rightmost component to be the head as well. This information is also presented in Table 4.3.

For the remaining categories (marked with $\star$ in Table 4.3), the head is either not specified in terms of functional labels in the treebank or a different notion of syntactic head is required in order to be able to ultimately extract the target grammar as specified in the previous chapter. The remainder of this section details the procedures that are used in the implementation.

For DL nodes, the annotated "discourse-level head" is not chosen as the head for reasons explained in Section 3.3.6. Instead, the first comma, sentence-final punctuation symbol, colon or dash from the left is marked as the head.

The virtual root (VROOT) will eventually be removed from most trees, but a head child needs to be identified anyway for the case that the removal is not possible (Section 4.1.2). The leftmost child of category S, DL or CS is marked as the head. If there is no such child, the leftmost child which is not a punctuation symbol becomes the head.

For nodes n of category VP or S, a common head finding procedure is implemented since both categories are verbal projections. The annotated head child (*HD*) is considered the head, except if it is a modal or auxiliary verb, one of its siblings is a VP clausal object (VP-*OC*) and n has the category S. In this case, it is a compound tense and the VP is

tagged as the head. Figure 4.11 contains such an instance (see VAFIN-*HD* and VP-*OC*). Besides being a POS category, the *HD* child can also be a *zu*-marked infinitive (VZ) or a coordinated VP in which case the algorithm descends further to find out whether the verb at the end of the path is an auxiliary, modal or main verb. In combination with the head-driven extraction procedure and the induction of predicative auxiliary trees (Section 4.2.2), this will provide the desired verbal elementary trees in correspondence to the target grammar. If there is no verbal *HD* child at all, but a child of category VP-*OC* or with the functional category *SVP* (separable verb prefix), it becomes the head.

Nodes of a coordinated category (see Section 2.4.3) are treated in one common procedure and therefore not explicitly listed in Table 4.3. Generally, the leftmost coordinating conjunction (*CD*) is tagged as the head as it should later become the anchor of the coordinating structure. If the node in question does not have a *CD* child, the rightmost comma (and if there is no comma, then any other punctuation symbol) is considered the head. This will lead to coordinating elementary trees anchored in punctuation, as described in Section 3.3.5 about coordination.

The specified head finding strategy is as accurate as possible, but also very optimistic in that it assumes perfect annotation and also that the head identification does not miss any cases. However, neither can be guaranteed. For example, it was mentioned earlier that a few NPs are not disambiguated (i.e. they have neither a child with the functional label *NKHD* nor *PH* or *HD*). To improve coverage of the non-standard cases, a fall-back strategy is implemented which resorts to the classical head percolation table in case no head child could be identified. A table developed for the Tiger Treebank by Rehbein (2009, p.127) with additional entries for the new phrasal categories is used. It is solely based on phrasal categories. It is, however, only applied for $< 1\%$ of all nodes. Thus, the head identification step makes use of the functional information annotated in the Tiger Treebank, and is accurate, but, it is still able to identify a head child for each node, even in anomalous cases.

Argument-Modifier Distinction

After having identified the head child of a node n, the non-head children of n are tagged as either arguments or modifiers. This classification is based on the phrasal category of n, the functional category of the child and the category of the head child of n. The conditions are provided in Table 4.4. In Figure 4.11, the identified argument nodes are shaded.

For NPs, the phrasal category of the child informs the argument-modifier distinction: a DP child is considered an argument, while all other non-head children are modifiers. In case there are two DP children, the POS category when following the head path downwards is deciding. If one of the DPs is headed by a PIAT (attributive indefinite pronoun), it is tagged as a modifier while the other DP becomes the argument. Examples are all_{PIAT} die_{ART} *Jahre* ('all the years') or die_{ART} $meisten_{PIAT}$ *Topmanager* ('most top managers').

Node category p	child functions F	head child category p_{hd}
AA	*PM*	
AC	-	
AP	*OA DA OG AMS*	
AVP	*AVC**	
CH	*UC*	
DL	*DH RS*	
DP	-	
ISU	*UC**	
MTA	*ADC**	
NM	*NMC*	NN
NP	⋆	
PCC	*COB*	
PN	*PNC*	
PP	*PC AC**	
VP S	*OC SB OA OA2 OG OP PD DA SBP EP SVP* CVC**	
VZ	*PM*	
VROOT	-	
coord	⋆	

Table 4.4: Argument-modifier distinction. The table is to be read in the following way: For a node n with the phrasal category p (shown in column 1), a child c is an argument if c is not the head child of n, c has the function $f \in F$ (column 2) and, if specified, the head child of n has the phrasal category p_{hd} (column 3). Else, c is a modifier. The asterisk * means that c is additionally flagged as providing a co-anchor in case multi-anchored trees are extracted. For categories with a star ⋆, the details are provided in the text (Section 4.2.1).

In all other cases, only the leftmost DP child is marked as an argument.

Non-head children of a coordination node are tagged as arguments if their functional label is *CJ* (conjunct) or *CD* (coordinating conjunction). In case the leftmost *CD* child (identified as the head in the head finding step) is to the left of the leftmost conjunct, the subsequent *CD* child is additionally flagged as providing a co-anchor for multi-anchored trees. This will cover the *entweder ... oder* ('either ... or') cases, specified in Section 3.2.3, Figure 3.12. Punctuation marks which are not located at the right periphery of the sentence are also tagged as arguments. Everything else is considered a modifier.

4.2.2 Canonical Elementary Trees

Each node in the derived tree is now annotated for whether it is the head child of its parent, a modifier or an argument. The next step is to segment the derived trees into elementary trees. Since it was decided to not explicitly model the German sentence structure as explained in Section 3.2.1, no specific extraction procedure needs to be designed, but the universal LTAG extraction procedure proposed by Xia et al. (2000) can be used. It was already presented in Section 2.3.1, but more details of our adapted version will be provided

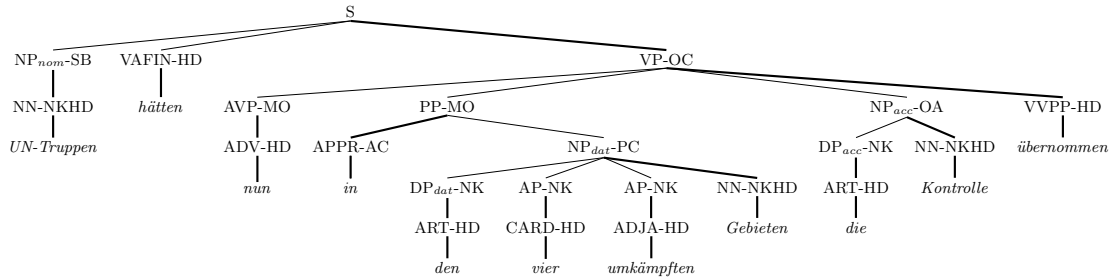


Figure 4.12: Spines (= path of head children) are marked (tree from Figure 4.2)

in this section.

Algorithm 2 specifies the extraction procedure. For each derived tree T of the treebank extractElemTree($root_T$, t, *true*) is called, where $root_T$ is the root node of T and t is an "empty" initial elementary tree ($t.type$ = init) that will be "filled" during the procedure. The overall idea is to start from the root of the derived tree and follow the path of head children downwards until reaching a terminal node (*while*-loop lines 3 to 39).[4] This path constitutes the spine of the elementary tree. In Figure 4.12, the spines are indicated by thicker edges. The siblings of the corresponding head nodes are considered in the *for*-loop starting in line 17. If a sibling is an argument, it is excised as an initial tree, for which the extraction procedure is called recursively. It leaves behind a substitution node in its host tree. If a sibling is a modifier, it is excised as a modifier tree (see Section 3.2.2 about sister-adjunction), in contrast to Xia's grammar where modifiers are encoded as auxiliary trees. Of course, these characteristics have already been considered when converting the treebank trees to derived trees. The procedure is also called recursively for the new modifier tree.

Formally, the procedure is best described by splitting each node n in the derived tree into two halves, as suggested by Xia (see Section 2.3.1). Foot nodes as well as substitution nodes are thought of top halves (*top*), while root nodes are considered as bottom halves (*bot*). As a consequence, when creating, for example, a substitution site, the top half of the node stays in the host tree (line 19), while the bottom half will be part of the new initial tree (line 14 when extractElemTree is called for t_{new}). Figure 4.13 illustrates how the nodes in the derived tree are split. As an example, consider the leftmost NP (NP$_{nom}$-SB_1^{11}). The top half belongs to t_{11} anchored in *übernommen* and the bottom half to t_1 anchored in *UN-Truppen*. The corresponding trees are depicted in Figure 4.15(a) and (d).

Sister-adjunction, which is not a standard TAG operation, does not fit directly into this picture. We conceptualize the node c which is added as a new daughter to the sister-adjunction site n, as being a complete note. This is in turn reflected during the extraction:

[4] Demberg-Winterfors (2010) also conceptually follows Xia et al. (2000), but builds the elementary trees in a bottom-up manner starting from the terminals. Given that each non-terminal node has exactly one head child, both methodologies will lead to the same elementary trees.

Algorithm 2 Elementary Tree Extraction

Require: n (node in the derived tree from which to start); t (an "empty" elementary tree); *first* (a boolean stating whether the procedure is called for the first time for n)

procedure EXTRACTELEMTREE(n, t, *first*)
  ▷ Starting from n, go down the head path to the lexical item → spine
  while n is not a leaf **do**
    ▷ Factor out coordination and predicative auxiliary trees
    while n is root of coordination **or** $n.pred = true$ **do**
      initialize $foot \leftarrow null$
      if n is root of coordination **then**
        $foot \leftarrow$ EXTRACTCOORDTREE(n)
      else if $n.pred = true$ **and** *first* **then**
        $foot \leftarrow$ EXTRACTPREDAUXTREE(n)
      if $foot = null$ **then**
        break
      $n \leftarrow foot$
    $t \leftarrow t \cup n.bot$
    initialize $hd \leftarrow$ head child of n
    $t \leftarrow t \cup hd.top$
    for all non-head children c of n **do**
      if c is argument **then**
        $t \leftarrow t \cup c.top$
        if c is a trace **then**
          ▷ Add the argument trace to t
          $t \leftarrow t \cup c.bot$
        else if $t.type =$ AUX **and** $\psi_T(c) = t.root$ **then**
          ▷ We found the foot note of the predicative auxiliary tree
          $t.foot \leftarrow c$
        else
          ▷ Create a substitution node and start a new initial tree
          mark $c.top \in t$ as substitution node
          initialize t_{new}
          $t_{new}.type \leftarrow$ INIT
          EXTRACTELEMTREES(c, t_{new}, *true*)
      else if c is modifier **then**
        ▷ Start a new modifier tree
        initialize t_{new}
        $t_{new}.type \leftarrow$ MOD
        $t_{new}.mother \leftarrow n$
        $t_{new} \leftarrow t_{new} \cup c.top$
        EXTRACTELEMTREES(c, t_{new}, *true*)
    $n \leftarrow hd$
  add t to global list of elementary trees

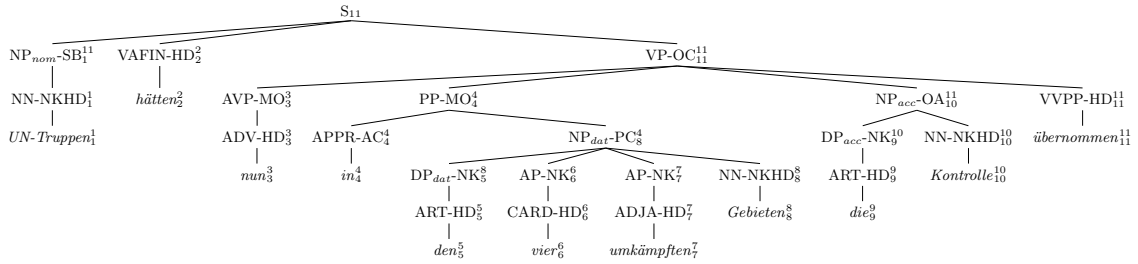


Figure 4.13: The nodes are indexed (top and bottom halves) according to which elementary tree they belong to (tree from Figure 4.2)

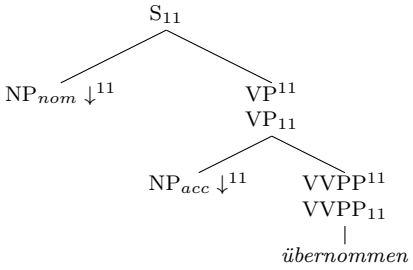


Figure 4.14: Illustration of how the nodes halves of t_{11} are combined to form the elementary tree in Figure 4.15(a).

in contrast to the extraction of an initial tree, the top of c is added to the new modifier tree t_{new} (line 37). See for example the node AVP-*MO* (index 3) in Figure 4.13. The unary root node of the corresponding modifier tree (which is identified with n during parsing) is stored as a property (*mother*) of the modifier tree during the extraction procedure, while c constitutes the actual root of the extracted tree t_{new}. See lines 34 to 38.

When adding a node half h to the current elementary tree t (operation $\cup$), t is composed in such a way that all dominance relations of the derived tree T still hold. *top*s and *bottom*s are then combined pairwise to form inner nodes. Figure 4.14 shows the node halves that are added to t_{11} in the dominance relation of T (Figure 4.13). Note that, strictly speaking, the node halves are copied before they are added to t. Figure 4.15 shows some of the elementary trees that are extracted by Algorithm 2 from our running example.

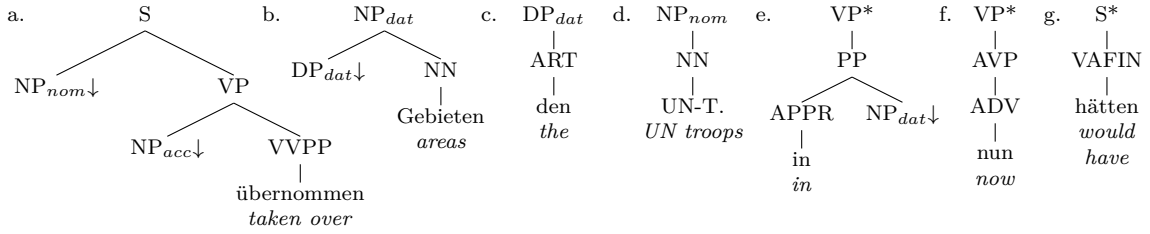


Figure 4.15: Some canonical initial (a - d) and modifier (e - f) trees extracted from the sentence in Figure 4.12.

Secondary Edges The secondary edges of the Tiger Treebank are preserved during the conversion. As discussed in Section 3.3.5, we currently do not fully exploit them because the usage depends on the treatment of coordination in a future parser, but we still represent shared arguments in the elementary trees. For secondary edges whose source node n is an argument or head, the bottom half h of the corresponding target node is marked for the category of the elided node n. This information is then also represented in the elementary tree to which h belongs. An example has already been provided in Figure 3.27(b).

Auxiliary Trees

Xia's algorithm does not cover the extraction of predicative auxiliary trees. In the German (P)LTAG, they are, however, required for non-finite auxiliary and modal verbs, and especially for verbs that embed sentential complements since some long-distance relations can consequently be covered. (Please find the details about the target trees in Section 3.3.) Therefore ideas by Chen (2001) are followed to extract predicative auxiliary trees.

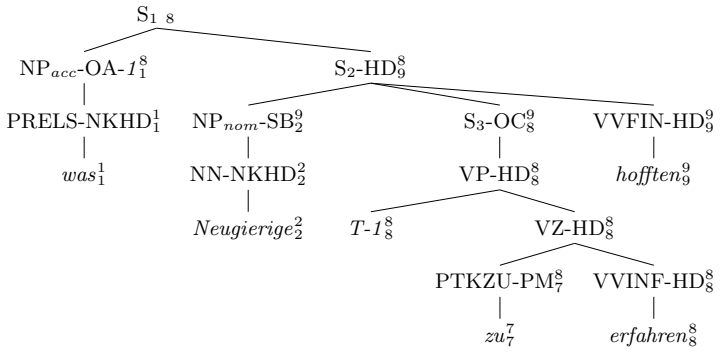
In Section 4.1.2, we have shown that future predicative auxiliary trees are already identified during the conversion procedure in order to insert additional S nodes that are required for the recursion in the auxiliary tree. As the result of the identification step, the root nodes of all future predicative auxiliary trees in the derived tree T are marked as such ($n.pred = true$). Furthermore, T is equipped with a 1:1 mapping ψ_T from foot nodes n' to root nodes n of the predicative auxiliary trees in T. Based on this information, the extraction procedure extracts a predicative auxiliary tree whenever encountering a node marked as $n.pred = true$, see line 9 in Algorithm 2.

The procedure for extraction of predicative auxiliary trees (Algorithm 3) is almost the same as for initial (and modifier) trees. This is motivated by the fact that the predicative auxiliary trees that we aim at are very similar to initial trees, with the exception that one of the arguments is sentential and provides the foot node. When this argument is encountered, it is marked in t as the foot (line 25 in Algorithm 2). After finishing the extraction of an auxiliary tree t, the hosting procedure continues with the extraction of the elementary tree from which t has been excised (see line 13 in Algorithm 2).

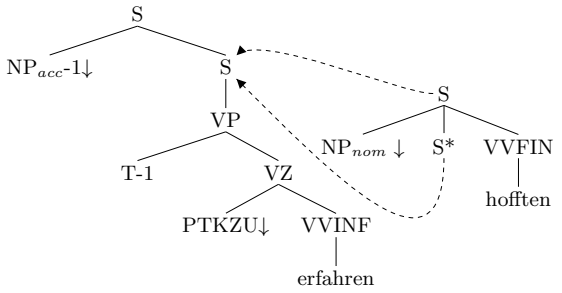
Figure 4.16(a) demonstrates the extraction of the predicative auxiliary tree of *hofften* by the means of indices. During the extraction of t_8, S_2, which is marked as $n.pred = true$, is encountered. Consequently, the procedure extractPredAuxTree is called to extract

Algorithm 3 Predicative Auxiliary Tree Extraction

Require: n (node in the derived tree from which to start)
1: **procedure** extractPredAuxTree(n)
2: initialize t
3: $t.type \leftarrow$ aux
4: extractElemTree(n, t, *false*)
5: **return** $t.foot$



(a) Derived tree: the nodes are indexed (top and bottom halves) according to which elementary tree they belong to.



(b) Predicative auxiliary tree t_9 (right) and the host tree t_8 (left); repeated from Figure 3.19(c).

Figure 4.16: Extraction of a predicative auxiliary tree. The translation is provided in (3.11).

t_9. In the course of its extraction, S_3, which is an argument and also mapped to S_2 by ψ_T, is encountered and becomes the foot of t_9. When the extraction of t_9 has finished, the extraction of t_8 continues with S_3. t_8 and t_9 have already been depicted in Figure 3.19(c), but are repeated in Figure 4.16(b). Note how the trace T-1 and its filler are localized in one elementary tree.

The extraction of auxiliary trees for coordinating structures as specified in the target grammar (Section 3.3.5) is similar to the extraction of predicative auxiliary trees. It is also the root node of the future auxiliary tree that triggers the procedure extractCoordTree due to the annotation of coordinated categories in the Tiger Treebank. See line 7 in Algorithm 2. However, the foot node of the coordination auxiliary tree remains to be identified within the extraction procedure. As Algorithm 4 (lines 14 to 16) shows, it is the leftmost conjunct (CJ) that does not correspond to a truncated word.[5] Otherwise, siblings c,

[5]Note that in this case, it is not verified whether the category of the foot node corresponds to the category of the root node when extracting the auxiliary tree. A verification step will reveal later that the extraction failed in such a case.

Algorithm 4 Coordination Auxiliary Tree Extraction

Require: n (node in the derived tree from which to start)

procedure EXTRACTCOORDTREE(n)
  initialize t
  $t.type \leftarrow$ AUX
  if head child of n is not a leaf **then**
    ▷ Anomalous coordination
    return *null*
  $t \leftarrow t \cup n.bot$
  for all children c of n from left to right **do**
    if c is head child of n **then**
      $t \leftarrow t \cup c.top$
      $t \leftarrow t \cup c.bot$
    else if c is argument **then**
      $t \leftarrow t \cup c.top$
      if $t.foot = null$ **and** $c.functCat = CJ$ **and** $c.phrasalCat \neq$ TRUNC **then**
        ▷ Foot node
        $t.foot \leftarrow c$
      else
        ▷ Create a substitution node and start a new initial tree
        mark $c.top \in t$ as substitution node
        initialize t_{new}
        $t_{new}.type \leftarrow$ INIT
        EXTRACTELEMTREES(c, t_{new}, *true*)
    else if c is modifier **then**
      ▷ Start a new modifier tree
      initialize t_{new}
      $t_{new}.type \leftarrow$ MOD
      $t_{new}.mother \leftarrow n$
      $t_{new} \leftarrow t_{new} \cup c.top$
      EXTRACTELEMTREES(c, t_{new}, *true*)
  return $t.foot$

classified as arguments or modifiers, are treated in the same way as in extractElemTree.

In the Tiger Treebank, some questionable constituents are annotated as coordinations. They lack a conjunction or punctuation symbol that could form the anchor of the elementary tree. Instead one of the conjuncts is classified as the head child by the back-up head identification strategy. Instead of trying to extract an auxiliary tree, extractCoordTree returns null (line 6). It basically gives back the node n to the standard extraction procedure, see line 11 in Algorithm 2.

Assembling Multi-Anchored Trees

During the argument-modifier classification, some nodes are additionally marked for being part of a multi-anchored tree (instead of initializing a new elementary tree t_{new}) in case multi-anchored trees are extracted (strategy (iii)), see Section 4.2.1. The lexical item that is reached when following the head path downwards from such a node will provide a co-anchor.

To induce multi-anchored trees, the extraction procedure stays exactly the same, but whenever a sibling c that provides a co-anchor is considered, the corresponding elementary tree t_{new} is marked for being part of a multi-anchored tree, and added to the list L_t, which remembers the co-anchor elementary trees of the host tree t. After the core extraction procedure, elements on L_t are recursively combined with t via substitution (or sister-adjunction if a co-anchor modifier tree was defined). For strategy (ii) (see Section 3.4.3), only those multi-anchored trees are assembled in which the host tree t provides the leftmost anchor. With strategy (i), no multi-anchored trees are assembled at all.

If in the future, other criteria for multi-anchored trees will be defined (for example, based on co-occurrence statistics), they can be integrated easily by adding the corresponding trees to L_t.

Since multi-anchored trees in PLTAG encode lexical predictions, the nodes that correspond to the individual elementary trees on L_t are marked with a unique prediction marker for each co-anchor. Note that this only makes sense for strategy (ii) as discussed in Section 3.4.3.

4.2.3 Prediction Trees

As already announced in Section 3.4, the prediction trees of the German PLTAG lexicon follow the definition that Demberg-Winterfors (2010) uses for the English prediction lexicon, see Section 2.2.2. Consequently, a similar extraction process as presented in Section 2.3.2 will be followed to extract the German prediction lexicon. The idea is to only extract prediction structures that are needed to parse the treebank fully incrementally using the canonical lexicon induced in Section 4.2.2. Furthermore, all prediction structures are derived from canonical elementary trees, but they have neither nodes to the right of the spine nor unary nodes at the bottom of the spine.

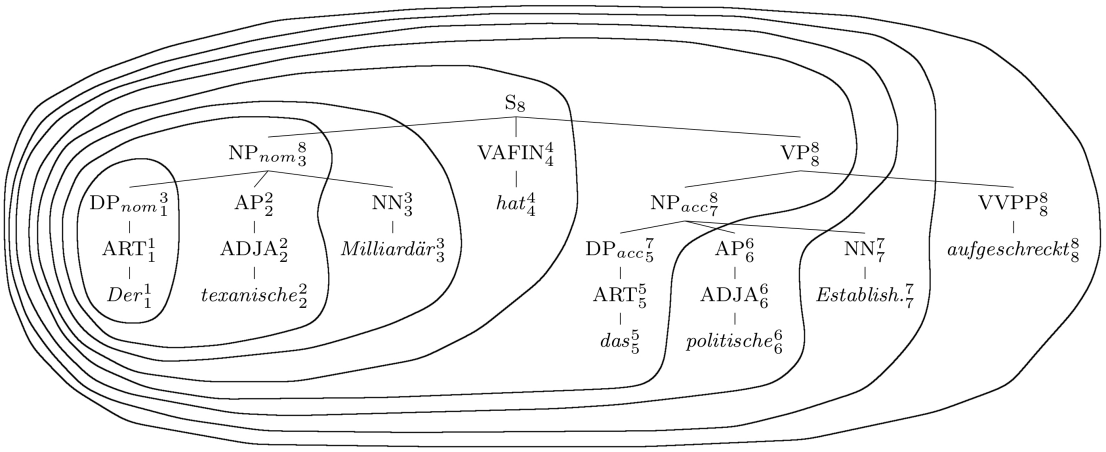


Figure 4.17: Connection paths for the words in the derived tree. Nodes are numbered according to the canonical elementary tree to which they belong. (The translation is provided with Figure 3.29.)

Each derived tree T has been segmented into canonical elementary trees by extractElemTree($root_T$, n, *first*). In Figure 4.17, the elementary trees are indicated by the indices. For each of the non-empty words w_i in T (with $1 \leq i \leq n$), the connection path $\theta_{1...i}$ for words $w_1 \ldots w_i$ is calculated.[6] It is the minimal amount of structure that is required to connect the words $w_1 \ldots w_i$ into the same syntactic tree. This amount of structures is indicated by the circles in Figure 4.17. In the implementation, it is obtained by a depth-first traversal of T.

To find out whether a prediction tree is required to integrate each w_i with the previously seen words $w_1 \ldots w_{i-1}$ into one syntactic structure, the nodes in $\theta_{1...i}$ are considered. For each top and bottom half h of the nodes in $\theta_{1...i}$ (except for the top of the current root of $\theta_{1...i}$), the procedure checks whether h belongs (a) to a canonical elementary tree anchored in words $w_1 \ldots w_i$ or (b) to a prediction tree that has been generated for words $w_1 \ldots w_{i-1}$. If neither (a) nor (b) is the case, h is added to the list of required nodes R_i, and the elementary tree t to which h belongs becomes a candidate for being predicted.

A prediction tree p is built on the basis of an elementary tree t by copying all nodes left of the spine of t, including the spine itself, to p. For a left-auxiliary tree t, also the foot node of t and all nodes between the root and the foot are copied to p. Then, starting from the bottom of the spine of p, all unary nodes which are not on R_i are removed recursively. As per definition, all nodes of p are marked as predicted. Note that if t is a multi-anchored tree, this procedure is only reasonable if we assume that the spine of t is the path from the root to the main anchor, which is the leftmost anchor (strategy (ii)). This has been discussed in Section 3.4.3.

As an example, consider w_4 and the connection path indicated by the corresponding

[6] Note that no prediction trees are calculated in order to integrate traces. For parsing this means that empty elements will not necessarily be integrated in the incremental order.

circle in Figure 4.17. All node halves with index $j \leq 4$ have already been derived. What remains is the top of the NP node and the bottom of the S node, both belonging to t_8. In t_8 (which has the same tree structure as for example the initial tree in Figure 4.15(a)), there are no nodes to the right of the spine and the only unary production is the lexical item, which is consequently removed from the corresponding p_8. Figure 4.18 shows p_8 (b) and the two other prediction trees that are extracted from Figure 4.17.

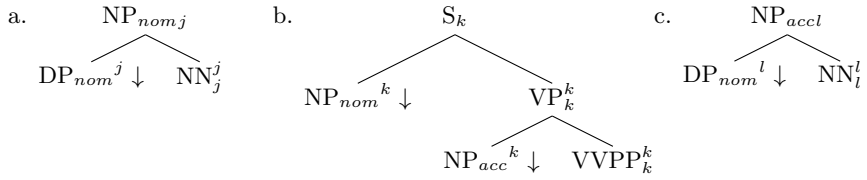


Figure 4.18: Prediction trees for a strictly incremental derivation of the tree in Figure 4.17

As discussed in Section 3.4.2, adjunction from the right is not predicted in the English PLTAG. For the German prediction lexicon, we therefore turn down a candidate t if t is an auxiliary tree, if both its root node and its foot node are on R_i, and if there is no other node which belongs to t on R_i (strategy (b)). However, we know from Section 3.4.2 that predicting adjunction from the right is sometimes necessary for the German grammar (using sister-adjunction) in order to obtain the correct scope of a modifier tree. For strategy (a) (see Section 3.4.2), it is therefore additionally checked whether t_i is a modifier child of the root node of t in the derived tree T. In this case, t would be predicted.

Strategy (c) is based on strategy (b). In addition, a bottom half h does not trigger its elementary tree t as a candidate prediction tree if the following hold: t_i is a modifier tree that sister-adjoins to the node of which h is the bottom half and if the corresponding top half is not on R_i (i.e. it is already predicted or derived).

If more than one prediction tree is generated for a connection path $\theta_{1\ldots i}$, they are combined into a pre-combined prediction tree with one of the TAG operations. The pre-combined prediction tree has unique prediction markers for node halves that originate from different elementary trees t.

4.3 Summary

This chapter started out by describing how linguistic generalizations are introduced to the Tiger Treebank trees to make them more uniform. It also explained further conversion steps that are necessary to turn the Tiger graphs into derived (P)LTAG trees. Along with the transformations, it was also indicated that, given the output of a TAG parser, they can be reverted almost completely. In the second part, the induction of canonical trees as well as prediction trees was detailed. They constitute the PLTAG lexicon. The converted treebank together with the (P)LTAG lexicon are the required resources for probabilistic TAG parsing.

Chapter 5

Evaluation

Treebank conversion and lexicon extraction as described in Chapter 4 is applied to the Tiger Treebank. Following the methodology of Dubey (2005), the corpus is split into training, development and test sets in the following way. The $i-1$th sentence (for all $i \geq 1$) is placed into bucket number i mod 20. Buckets number 1 to 18 make up the training data, the test set consists of bucket number 20 and bucket number 19 can be used for development (or testing as well). This puts 45,428 sentences in the training set, from which the (P)LTAG lexicon is extracted, and 2,523 sentences in the development and test set each.

The treebank conversion and extraction procedures generate complete, correct tree structures for 99.8% of all trees in the Tiger Treebank. This is tested by re-combining the extracted canonical elementary trees with substitution, adjunction and sister-adjunction, and comparing the derived tree with the converted treebank tree. The failed sentences all contain instances of non-recursive coordination, for which no valid auxiliary tree could be extracted. Many of them involve conjuncts of type 'chunk' (CH) and 'foreign language material' (FM). The statistics in this chapter are based on the sentences for which conversion and extraction were successful.

The extracted (P)LTAG lexicon is examined in this chapter with respect to its size and coverage. A manual investigation reveals the nature of uncovered tree structures in unseen data.

5.1 Lexicon size

During treebank conversion, the node inventory of phrasal categories in Tiger is extended to also include case annotation for NPs and DPs (Section 3.3.3) and secondary edges (Section 3.3.5). Furthermore, the functional edge labels of the Tiger annotation can be seen as a refinement of the phrasal categories, such that they provide *complex* node categories of the form X-Y together. For example S-RC labels a sentence that is a relative clause, where S is the phrasal and RC the functional category. All of those node extensions can be easily removed from the lexicon and the treebank due to their distinct annotation. Thus several kinds of grammars are obtained that differ in size and the specificity of information that is

	Node label inventory				**Templates**				Lexicalized
	Sec. Edge		Case	Compl. Cat.	Initial	Auxiliary	Mod.	Sum	trees
A					1,860	880	2,619	**5,359**	144,886
B	x				2,400	904	2,672	5,976	145,518
C			x		2,742	1,251	3,424	**7,417**	164,598
D				x	4,441	1,945	10,077	16,463	201,150
E	x		x	x	5,848	2,424	11,442	19,714	210,304

Table 5.1: Number of canonical tree types extracted with respect to grammars of different node label inventories. None of the lexica includes multi-anchored trees (strategy (i)).

contained.

Table 5.1 shows a comparison. Lexicon A, without annotation for case and secondary edges and without functional labels, is the most comparable to the English TAG lexica extracted from the PTB. Indeed, with less than 5,400 tree templates (i.e. unlexicalized canonical elementary trees), the German LTAG lexicon has about the same size as those extracted for English by Xia et al. (2000), Chen and Vijay-Shanker (2004) and Demberg-Winterfors (2010). All of them have a size of very roughly 6,000 templates. Note that even though the German LTAG uses sister-adjunction for modification as proposed in (Chiang, 2000), it cannot compete in size with Chiang's grammar because we use a special type of tree (modifier trees) for sister-adjunction whereas he uses initial trees (Section 3.2.2). With respect to the type of different templates needed, the German lexicon is therefore closer to one which uses regular adjunction of auxiliary trees for modification. A meaningful comparison to previous work about German TAG extraction (Section 2.3.3) is not possible because of the small treebank used by Neumann (2003) and because of the different data set and no indication about the overall number of extracted templates in (Frank, 2001).[1]

Clearly, the more information is encoded in the node labels, the larger the size of the label inventory and thus the larger the extracted lexicon (e.g. lexicon E). However, lexicon C, which includes the case annotation for which we have especially argued in Section 3.3.3, still has a manageable size of about 7,400 canonical tree templates and is not much larger than Demberg-Winterfors's (2010) LTAG lexicon for English. If not explicitly stated otherwise, the results in the following are obtained with lexicon C.

Concerning the complex categories (lexicon D), which considerably blow up the extracted lexicon (16,463 vs. 5,359 templates), it seems that the benefit of using them needs to be determined in a practical setting. One might follow the approach taken by Schiehlen (2004) and include only some of the functional labels to form complex categories in order to obtain a compromise between grammar size and additional information contained in the grammar.

[1] Furthermore, for both approaches it is not clear whether the lexicon for which numbers are reported includes the functional information or not.

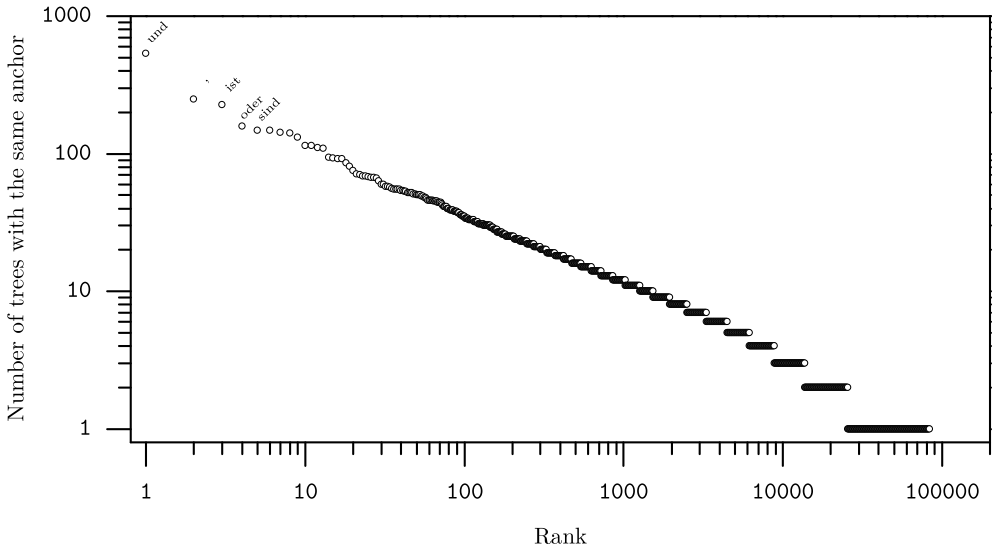


Figure 5.1: Distribution of ambiguity in the LTAG lexicon (log-log scale)

Lexica A to E do not contain multi-anchored trees, so that the numbers can be better compared to the English lexica. If extraction strategy (iii) is used instead in order to obtain the multi-anchored trees as proposed in Section 3.2.3, 3,895 lexicalized multi-anchored tree types are generated for lexicon A (4,035 for lexicon C). Excluding those multi-anchored trees in which the main anchor is not the leftmost one (strategy (ii), see Section 3.4.3) still yields 3,413 lexicalized multi-anchored tree types (3,545 for lexicon C).

The average ambiguity per lexical item is 1.96 trees per word form, which is less than what Demberg-Winterfors (2010) reports for the English LTAG (2.45 trees per word). This is plausible because of the richer morphology of German. If lemmata instead of surface forms are considered, the number approaches the English one. For parsing or supertagging, the richer morphology might lead to data sparsity. One could therefore consider to also take the lemma information into account, which we store for each lexical entry. For example, in addition to selecting the elementary trees with the same surface word form, also the ones with a matching lemma could be considered by the parser.

Figure 5.1 shows the distribution of ambiguity per word form of the extracted lexicon. It can be seen that the data follows Zipf's law as it is usually the case with linguistic data. Very few words are associated with many different trees, whereas the majority of words is associated with only one tree (see the long tail in the graph). The most ambiguous words in terms of trees are certain function words like *und* ('and', 533 trees), *oder* ('or', 158 trees) and *wie* ('as/how/like', 147 trees) and verbs that occur in different contexts and with different functions such as *sein* ('to be') and its forms (*ist*: 226 trees, *sind*: 148 trees). The comma

is also very ambiguous (249 trees). The reason is that it anchors coordination structures just like *und* and *oder*, but it is also encoded as a modifier tree in many circumstances and as the anchor for 'discourse level constituents' (cf. Section 3.3.6).

Prediction Trees When generating prediction trees as described in Section 4.2.3, strategy (b) yields 4,407 combined prediction trees for lexicon A and 6,004 for lexicon C. They correspond to 2,397 and 3,624 uncombined trees respectively. With strategy (a) (adjunction from the right is predicted if necessary to yield the correct scope of a modifier tree), there are 4,605 pre-combined prediction trees for lexicon A and 6,205 for lexicon C. Strategy (c) leads to 3,858 and 5,309 combined trees respectively.

No matter which strategy is chosen, considerably more prediction trees are induced for German than for English (2800 in (Demberg-Winterfors, 2010)). This was expected because of the different prediction granularities (cf. Section 3.4.1), but of course increases the ambiguity for a German PLTAG parser.

However, just as in the PTB, no more than 5 trees have to be pre-combined in order to achieve full connectivity. Among the instances where a prediction tree is needed in the Tiger Treebank, 92.67% of the cases use only one prediction tree (Lexicon C, strategy (b)), in 7% of the cases two prediction trees have to be combined, and in less than 0.35% of the cases nodes from three lexical anchors are required.

5.2 Growth and Coverage

One way to evaluate a grammar is to find out how complete it is, i.e. to measure its coverage on unseen data. We therefore also convert the Tiger test set and extract canonical elementary trees and prediction trees from it.

Lexicon C achieves a coverage of 99.6% of the canonical templates and 98.2% of the prediction trees that were extracted from the Tiger test data (overall 99.3%, and 99.5% for lexicon A). On lexicalized elementary trees we yield a coverage of 85.8%. Even though these numbers sound satisfactory, they indicate that the grammar does not converge, i.e. that even after having seen all the training data, new templates still occur in unseen data.

This observation is also confirmed when having a look at the growth of the lexicon during training. Figure 5.2 shows a plot of the number of templates as a function of the portion of the treebank that was used to generate the templates. We observe that the number of templates does not converge with successively more data. Xia (2001) as well as Chiang (2004) report very similar results for their English LTAG lexica.

It can, however, be assumed that the obtained coverage is not problematic for parsing. This is firstly motivated by the good parsing results of Chiang (2004), and furthermore by the template frequency distribution: a few templates occur very often in the corpus while many others are very rare. This is shown in Figure 5.3 which plots the frequency of the templates versus their rank, again roughly following a Zipf distribution. Out of the 7,417

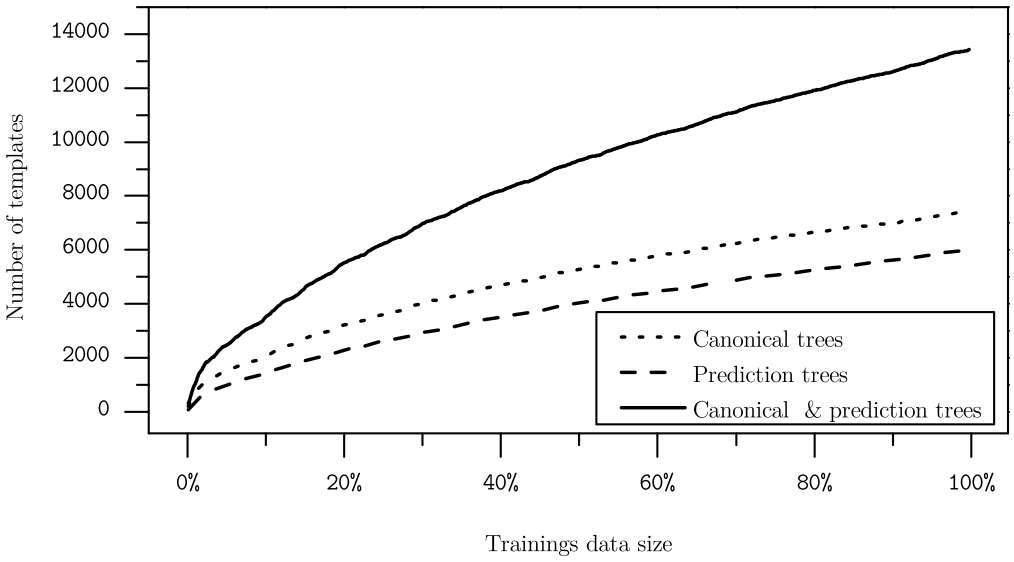


Figure 5.2: The growth of the grammar during training

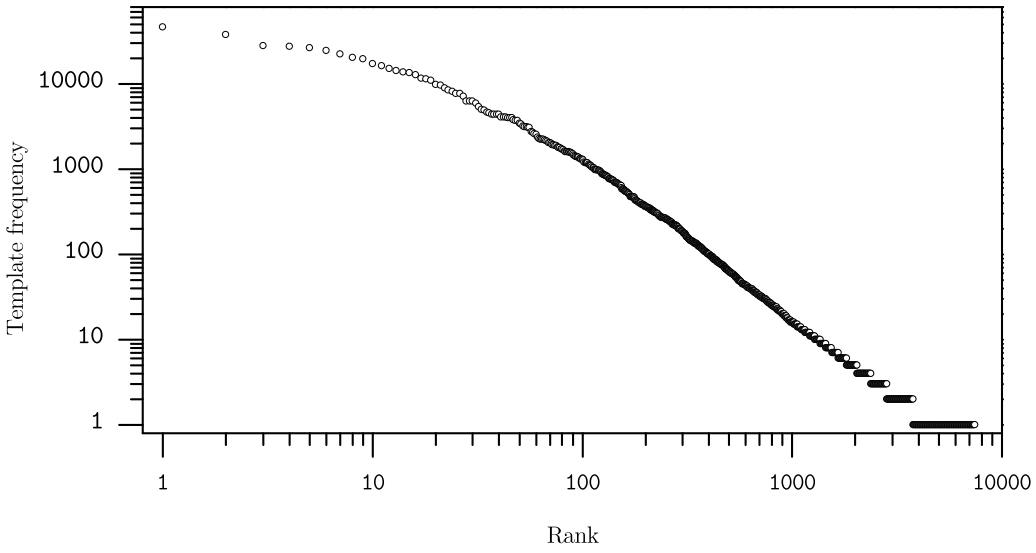


Figure 5.3: Distribution of canonical template frequencies (log-log scale)

canonical templates, 3,645 occur only once in the complete training data, accounting for 0.45% of all template tokens. In contrast, there are only 114 templates with a frequency of 1,000 or higher, but they cover 83.5% of all template occurrences. If templates and prediction trees which occur only once in the complete training corpus are disregarded, the grammar actually converges. The growth of this 'core' lexicon is shown in Figure 5.4. In

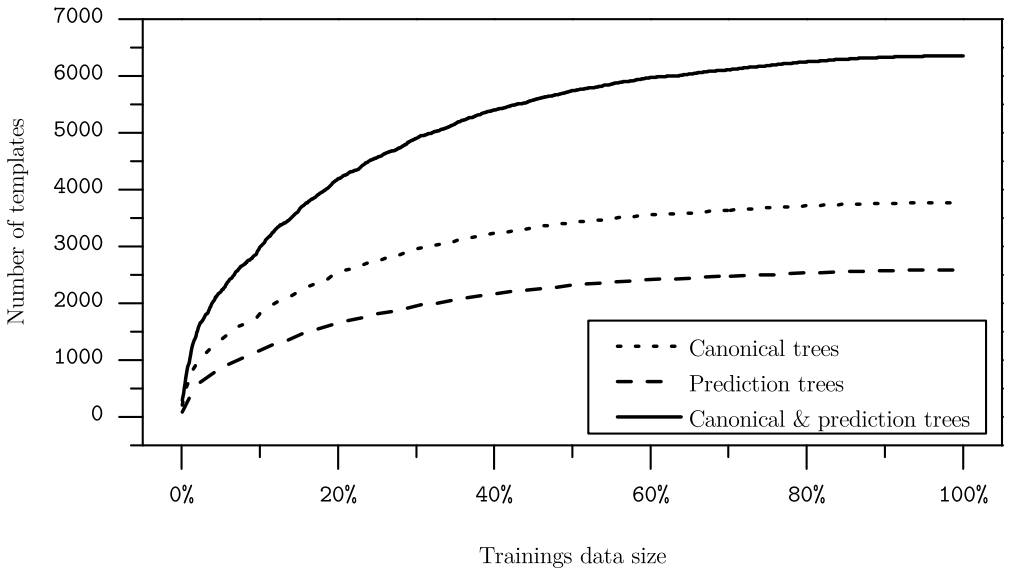


Figure 5.4: The growth of the core grammar (templates of frequency > 1 in the training corpus) during training

practice, such cut-off grammars are used for parsing (e.g. Chiang (2000)).

It can consequently be assumed that the most 'important' templates (i.e. those that cover the major part of the data) have been extracted during training. The templates which remain uncovered in unseen data are infrequent ones. The coverage of 99.6% and 98.2% on canonical templates and prediction trees respectively is good enough to parse unseen data with state-of-the-art results.

5.3 Manual Inspection

It turns out that the most frequent templates are rather simple. Figure 5.5 shows the five most frequent ones according to the ranking in Figure 5.3: two modifier trees and three initial trees. The most frequent prediction trees are shown in Figure 5.6.

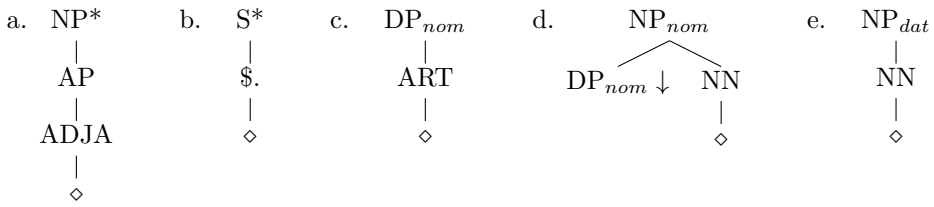


Figure 5.5: The five most frequent canonical templates in lexicon C

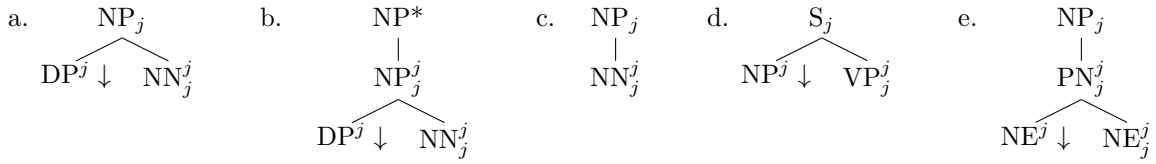


Figure 5.6: The five most frequent prediction trees in lexicon A (strategy (b))

After having observed that the generated grammar does not converge, an explanation needs to be found. A random sample of 100 templates that occur in the test data, but not in the training data is manually inspected and categorized.[2] The sample that is investigated contains 60 canonical tree templates and 40 prediction trees.

Canonical templates Following previous work (Xia, 2001; Chiang, 2004), the canonical templates are divided into three categories:

1. There is an error/inconsistency in the annotation of the treebank which causes an erroneous template.
2. A new, linguistically plausible construction appears.
3. The conversion and/or extraction heuristics fail to describe the phenomenon.

We classify 18.3% of the canonical templates of the sample into category 1. The remaining templates should be tagged with either category 2 or 3, but it is not obvious how to interpret them in the context of the Tiger Treebank. The reason is that the conversion and extraction rules are almost exclusively based on the linguistic annotation, so heuristics can fail only very rarely. We assume that a tree produces linguistically plausible templates if it seems to be correctly annotated and if the conversion and extraction procedures operate on it in the expected way. We assign category 3 to a template if we think that more conversion rules would have helped to cover it. An example is shown in Figure 5.7. With fine-grained rules that insert appropriate projections for the CARD category (an NP in this case), the template would certainly have been encountered during training. Accordingly, 23.3% of the canonical templates in the sample are classified as category 3.

The remaining 58.3% templates correspond to new constructions. They include, amongst others, templates which correspond to previously unseen orderings of arguments. Due to our interpretation of the categories, category 2 also contains linguistically questionable templates that originate from discontinuous constituents which TAG cannot cover adequately. We furthermore find linguistically problematic templates that stem from incomplete sentences. Such sentences appear in our data from the newspaper domain, for instance in headlines. An example is shown in Figure 5.8(a). Since there is no obvious linguistic head of the S

[2]The investigation is performed by the author of this thesis by casual inspection. For a more reliable evaluation, more annotators would have to categorize more templates.

(5.1) *193 bekamen in dieser Zeit Brustkrebs*
193 contracted in this time breast cancer
'During this period, 193 contracted breast cancer'

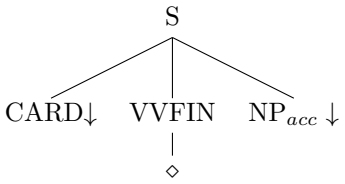
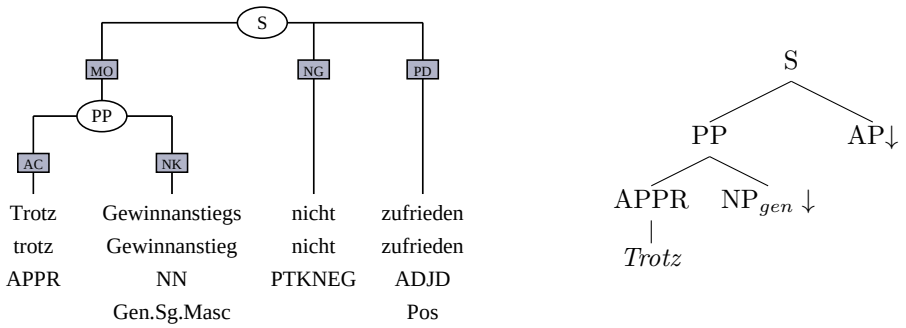


Figure 5.7: Template of *bekamen*, corresponding to the sentence in (5.1)

(5.2) *Trotz Gewinnanstiegs nicht zufrieden*
despite profit growth not satisfied
'Not satisfied despite profit growth'



(a) Tiger graph. The translation is provided in (5.2). (b) Elementary tree extracted from the graph on the left.

Figure 5.8: An incomplete sentence leading to a linguistically odd and uncovered template

node due to ellipsis of the verb, choosing any of the three children will lead to templates that do not comply with the co-occurrence principle. They are linguistically odd, and they are too rare and diverse to be generally covered in the training data. Figure 5.8(b) shows the structure-giving template if the leftmost child is chosen as the head (which is the current strategy, cf. Section 4.2.1).

Prediction trees For previously unseen prediction trees, we devise the following four categories:

1. An uncombined prediction tree: the corresponding canonical template did not occur before.

2. An uncombined prediction tree: the corresponding canonical template occurred before, but not the prediction tree.

3. A pre-combined prediction tree: one of its individual prediction trees has not been encountered before.

4. A pre-combined prediction tree: the individual prediction trees have been observed before, but not in this combination.

In the sample, 14 prediction trees (35%) are not pre-combined. All of them are new because the corresponding canonical template has not been observed before (category 1). Among the pre-combined prediction trees, 8 (20%) include a new individual prediction tree. For the remaining ones, the specific combination has not been encountered before (category 4).

This experiment shows that slightly more templates could be covered with more conversion rules. However, even though the Tiger Treebank is pretty large, the lexicon is not likely to include all plausible and necessary templates. As expected, new templates include unseen phenomena and argument orders, but also slightly odd templates due to artifacts of the Tiger data. New templates directly lead to new prediction trees, but also previously observed prediction trees can be combined in new ways.

5.4 Summary

We have shown that the German LTAG lexicon is of comparable size and coverage to English LTAG lexica extracted from the PTB. We therefore expect that parsing with it will be feasible and achieve similar results. The prediction lexicon for PLTAG is larger than for English, which we had already anticipated. It remains to investigate how a PLTAG parser will handle this additional ambiguity. A small manual investigation revealed that the conversion procedures could still be improved in order to slightly increase the coverage. However, genuine, but rare templates and prediction trees will still emerge in unseen data.

Chapter 6

Conclusion and Future Work

A broad-coverage grammar and a corpus of corresponding derived trees are the core requirements for training the parameters of a statistical broad-coverage parser. This thesis presented the first German resources of this kind that are available for a psycholinguistically motivated, incremental version of TAG: a German PLTAG treebank of derived trees and a linguistically motivated lexicon, converted and extracted from the Tiger treebank. Since standard LTAG trees are a subset of the PLTAG lexicon and both tree-adjoining formalisms generate the same derived trees, the presented resources are also a long overdue contribution to parsing with tree-adjoining grammars in general. For German no LTAG treebank grammar has been available for parsing to date.

The grammar induction procedure has made extensive use of the linguistic annotation contained in the Tiger treebank, especially of the functional edge labels. It therefore only needed to resort to general heuristics, as used for the induction of English TAGs from the PTB, in very few cases. In contrast to previous approaches for LTAG extraction from German treebanks in the literature, the extracted grammar is comparable in size and coverage to the English lexica for tree-adjoining grammar.

The next step is to build on the resources, for example to extend the English PLTAG parser with the notion of sister-adjunction and wrapping adjunction for automatic incremental TAG derivations of German sentences. Such a parser would then also contribute to the validation of the PLTAG sentence processing theory. For example, it could succeed at capturing certain phenomena like anti-locality effects, which appear specifically in German.

Apart from that, the induced lexicon itself could be improved and refined in the ways indicated throughout this work. Very fine-grained rules can be implemented to introduce further linguistic generalizations and to improve the head finding strategy within noun phrases. The same holds for the small set of sentences containing non-recursive coordination which are currently not covered. The trace-filler pairs which TAG cannot describe in a linguistically adequate way could be covered with a more powerful variant of TAG, such as MCTAG. Trace and filler would then, for example, form one tree set.

This thesis has also identified starting points for research directions in the field of formal grammar. Augmenting the TAG formalism with sister-adjunction is a worthwhile exploration

since, first, it makes inducing tree-adjoining grammars from treebanks more straightforward and, second, it allows to separate modification (which is usually encoded in auxiliary trees) from the group of other phenomena that are modeled via adjunction (in predicative and coordinating auxiliary trees). To the best of our knowledge, sister-adjunction has not been considered in the context of feature-based TAG so far. Its exact role when splitting nodes into halves, i.e. top and bottom feature structures, still needs to be determined. A corresponding formalization is also relevant for defining (P)LTAG semantics and for the computation of prediction trees since it is based on the notion of node halves.

To avoid the open questions that have been raised in relation to sister-adjunction, one could also revert to using the standard TAG operations substitution and adjunction exclusively. The main research would then target a linguistically motivated binarization of the Tiger Treebank. Employing a corresponding grammar with already existing standard (P)LTAG parsers would be facilitated since no new operation must be implemented.

In the context of PLTAG and prediction of upcoming structures and lexemes during sentence processing, the need for more work and corresponding psycholinguistic experiments, that has already been pointed out by Demberg-Winterfors (2010), was reinforced by points of discussion within the work at hand. This involves the prediction granularity, especially in head-final structures, and the prediction of modifiers and negation. Furthermore, as indicated in Section 3.2.3, lexical predictions should in the future be consistently encoded as multi-anchored trees. The computation could be based on co-occurrence statistics from a large corpus.

References

Abeillé, A., Candito, M., and Kinyon, A. (1999). FTAG: Current Status and Parsing Scheme. In *Proc. Vextal*, volume 99.

Abeillé, A. and Rambow, O. (2000). Tree Adjoining Grammar: An Overview. In *Tree Adjoining Grammars: Formalisms, Linguistic Analysis and Processing*. CSLI Publications.

Abeillé, A. and Schabes, Y. (1996). Non-compositional discontinuous constituents in Tree Adjoining Grammar. In Blunt, H. and van Horck, A., editors, *Discontiuous Constituency*, volume 6 of *Natural Language Processing*, pages 113–140. Mouton de Gruyter.

Albert, S., Anderssen, J., Bader, R., Becker, S., Bracht, T., Brants, S., Brants, T., Demberg, V., Dipper, S., Eisenberg, P., et al. (2003). *TIGER Annotationsschema*. Universität des Saarlandes and Universität Stuttgart and Universität Potsdam.

Becker, T., Joshi, A., and Rambow, O. (1991). Long-Distance Scrambling and Tree Adjoining Grammars. In *Proceedings of the fifth Conference of the EACL*, pages 21–26.

Brants, S., Dipper, S., Hansen, S., Lezius, W., and Smith, G. (2002). The TIGER treebank. In *Proceedings of the Workshop on Treebanks and Linguistic Theories*.

Brants, S. and Hansen, S. (2002). Developments in the TIGER Annotation Scheme and their Realization in the Corpus. In *Proceedings of the Third International Conference on Language Resources and Evaluation*, Las Palmas, Spain. European Language Resources Association (ELRA).

Cahill, A. (2004). *Parsing with Automatically Acquired, Wide-Coverage, Robust, Probabilistic LFG Approximations*. PhD thesis, Dublin City University.

Charniak, E. (1996). Tree-bank Grammars. In *Proceedings of the AAAI-96*.

Chen, J. (2001). *Towards Efficient Statistical Parsing Using Lexicalized Grammatical Information*. PhD thesis, University of Delaware.

Chen, J. and Vijay-Shanker, K. (2004). Automated extraction of TAGs from the Penn Treebank. *New developments in parsing technology*.

Chiang, D. (2000). Statistical Parsing with an Automatically-Extracted Tree Adjoining Grammar. In *Proceedings ACL*, volume 38.

Chiang, D. (2004). *Evaluating Grammar Formalisms for Applications to Natural Language Processing and Biological Sequence Analysis*. PhD thesis, University of Pennsylvania.

Crysmann, B., Hansen-Schirra, S., Smith, G., and Ziegler-Eisele, D. (2005). TIGER Morphologie-Annotationsschema. Technical report, Universität Potsdam, Universität Saarbrücken.

Demberg, V. and Keller, F. (2008). A Psycholinguistically Motivated Version of TAG. In *Proceedings of TAG+9*.

Demberg-Winterfors, V. (2010). *A Broad-Coverage Model of Prediction in Human Sentence Processing*. PhD thesis, University of Edinburgh.

Dubey, A. (2005). *Statistical Parsing for German: Modeling syntactic properties and annotation differences*. PhD thesis, Universität des Saarlandes.

Dubey, A. and Keller, F. (2003). Probabilistic Parsing for German Using Sister-Head Dependencies. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 96–103. Association for Computational Linguistics.

Frank, A. (2001). Treebank Conversion. Converting the NEGRA treebank to an LTAG grammar. In *Proceedings of the Workshop on Multi-layer Corpus-based Analysis*.

Gamon, M., Ringger, E., Zhang, Z., Moore, R., and Corston-Oliver, S. (2002). Extraposition: A Case Study in German Sentence Realization. In *Proceedings of the 19th international conference on Computational linguistics - Volume 1*, COLING '02, pages 1–7.

Gerdes, K. (2002). DTAG. Attempt to Generate a Useful TAG for German Using a Metagrammar. In *Proceedings TAG+6*, Venice.

Grewendorf, G., Hamm, F., and Sternefeld, W. (1989). *Sprachliches Wissen: Eine Einführung in moderne Theorien der grammatischen Beschreibung*. Suhrkamp, Frankfurt am Main, 3rd edition.

Hockenmaier, J. (2006). Creating a CCGbank and a wide-coverage CCG lexicon for German. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, ACL-44, pages 505–512. Association for Computational Linguistics.

Höhle, T. (1986). Der Begriff "Mittelfeld", Anmerkungen über die Theorie der topologischen Felder. *Akten des Siebten Internationalen Germanistenkongresses 1985*, pages 329–340.

Joshi, A., Levy, L., and Takahashi, M. (1975). Tree adjunct grammars. *Journal of computer and system sciences*, 10(1):136–163.

Joshi, A. and Schabes, Y. (1997). Tree-adjoining grammars. *Handbook of formal languages*, 3:69–124.

Kaeshammer, M. and Demberg, V. (2012). German and English Treebanks and Lexica for Tree-Adjoining Grammars. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2012)*.

Kallmeyer, L. (2010). *Parsing Beyond Context-Free Grammars*. Springer Verlag.

Kallmeyer, L. and Parmentier, Y. (2008). On the relation between Multicomponent Tree Adjoining Grammars with Tree Tuples (TT-MCTAG) and Range Concatenation Grammars (RCG). *Language and Automata Theory and Applications*, pages 263–274.

Kallmeyer, L. and Yoon, S. (2004). Tree-local MCTAG with Shared Nodes: An Analysis of Word Order Variation in German and Korean. *Traitement automatique des langues TAL*, 45(3):49–69.

Kamide, Y., Scheepers, C., and Altmann, G. T. M. (2003). Integration of syntactic and semantic information in predictive processing: Cross-linguistic evidence from German and English. *Journal of Psycholinguistic Research*, 32.

Konieczny, L. (2000). Locality and parsing complexity. *Journal of Psycholinguistic Research*, 29(6).

Konieczny, L. and Döring, P. (2003). Anticipation of Clause-Final Heads: Evidence from Eye-tracking and SRNs. In *Proceedings of ICCS/ASCS*.

Kübler, S. and Penn, G., editors (2008). *Proceedings of the Workshop on Parsing German*. Association for Computational Linguistics, Columbus, Ohio.

Lezius, W. (2002). TIGERSearch - Ein Suchwerkzeug für Baumbanken. In *Tagungsband zur Konvens*.

Lichte, T. (2007). An MCTAG with Tuples for Coherent Constructions in German. In *FG 2007: The 12th conference on Formal Grammar*.

Lichte, T. and Kallmeyer, L. (2008). Factorizing Complementation in a TT-MCTAG for German. In *Proceedings of the The Ninth International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+ 9)*.

Lombardo, V. and Sturt, P. (2002). Incrementality and Lexicalism. In *Lexical Representations in Sentence Processing*.

Magerman, D. M. (1994). *Natural Language Parsing as Statistical Pattern Recognition*. PhD thesis, Stanford University.

Maier, W. (2010). Direct Parsing of Discontinuous Constituents in German. In *Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 58–66. Association for Computational Linguistics.

Marcus, M., Marcinkiewicz, M., and Santorini, B. (1993). Building a large annotated corpus of English: The Penn Treebank. *Computational linguistics*, 19(2).

Mazzei, A. and Lombardo, V. (2004). Building a Large Grammar for Italian. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2004)*.

Neumann, G. (2003). A Uniform Method for Automatically Extracting Stochastic Lexicalized Tree Grammars from Treebanks and HPSG. *Treebanks: Building and Using Parsed Corpora*.

Palmer, M., Gildea, D., and Kingsbury, P. (2005). The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1).

Rambow, O. (1994). *Formal and Computational Aspects of Natural Language Syntax*. PhD thesis, University of Pennsylvania.

Rambow, O., Vijay-Shanker, K., and Weir, D. (1995). D-Tree Grammars. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, pages 151–158.

Rehbein, I. (2009). Treebank-Based Grammar Acquisition for German.

Sarkar, A. and Joshi, A. (1996). Coordination in Tree Adjoining Grammars: Formalization and Implementation. In *Proceedings of the 16th conference on Computational Linguistics*, pages 610–615. Association for Computational Linguistics.

Sarkar, A., Xia, F., and Joshi, A. (2000). Some Experiments on Indicators of Parsing Complexity for Lexicalized Grammars. In *Proceedings of COLING 2000*, pages 37–42.

Schiehlen, M. (2004). Annotation Strategies for Probabilistic Parsing in German. In *Proceedings of the 20th international conference on Computational Linguistics*.

Schlangen, D. and Skantze, G. (2009). A General, Abstract Model of Incremental Dialogue Processing. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 710–718. Association for Computational Linguistics.

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*, pages 88–95. Association for Computational Linguistics.

Staub, A. and Clifton, C. (2006). Syntactic prediction in language comprehension: Evidence from either ... or. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 32.

Sturt, P. and Lombardo, V. (2005). Processing coordinate structures: Incrementality and connectedness. *Cognitive Science*, 29.

Tanenhaus, M. K., Spivey-Knowlton, M. J., Eberhard, K. M., and Sedivy, J. C. (1995). Integration of visual and linguistic information in spoken language comprehension. *Science*, 268.

Vadas, D. and Curran, J. (2007). Adding Noun Phrase Structure to the Penn Treebank. In *Proceedings ACL*, volume 45.

Vijay-Shanker, K. and Joshi, A. K. (1988). Feature Structures Based Tree Adjoining Grammars. In *Proceedings of the 12th conference on Computational linguistics - Volume 2*, COLING '88, pages 714–719.

Xia, F. (2001). *Automatic Grammar Generation From Two Different Perspectives*. PhD thesis, University of Pennsylvania.

Xia, F., Palmer, M., and Joshi, A. (2000). A Uniform Method of Grammar Extraction and its Applications. In *Proceedings of the 2000 Joint SIGDAT conference on EMNLP/VLC*.

XTAG Research Group (2001). A Lexicalized Tree Adjoining Grammar for English. Technical report, Technical Report IRCS-01-03, IRCS, University of Pennsylvania.

Appendix A

Tiger Category Inventory

Table A.1: Phrasal category inventory of Tiger, plus the newly introduced categories (marked with *)

Category	Description
AA	superlative phrase with *am*
AC*	adposition
AP	adjectival phrase
AVP	adverbial phrase
CAC	coordinated adposition
CAP	coordinated adjectival phrase
CAVP	coordinated adverbial phrase
CCP	coordinated complementizer
CH	chunk
CNP	coordinated noun phrase
CO	coordination
CPP	coordinated adpositional phrase
CS	coordinated sentence
CVP	coordinated verb phrase (non-finite)
CVZ	coordinated *zu*-marked infinitive
DL	discourse level constituent
DP*	determiner phrase
ISU	idiosyncratic unit
MTA	multi-token adjective
NM	multi-token number
NP	noun phrase
PCC*	comparative complement phrase
PN	proper noun
PP	adpositional phrase
...	

Category	Description
(QL	quasi-language)
S	sentence
VP	verb phrase (non-finite)
VZ	*zu*-marked infinitive
VROOT	virtual root

Table A.2: Grammatical function inventory of Tiger, plus the newly introduced categories (marked with *)

Category	Description
AC	adpositional case marker
ADC	adjective component
AG	genitive attribute
AMS	measure argument of adjective
APP	apposition
AVC	adverbial phrase component
CC	comparative complement
CD	coordinating conjunction
CJ	conjunct
CM	comparative conjunction
COB*	comparative object
CP	complementizer
CVC	collocational verb construction
DA	dative
DH	discourse-level head
DM	discourse marker
EP	expletive *es*
HD	head
JU	junctor
(MC	comitative)
(MI	instrumental)
(ML	locative)
MNR	postnominal modifier
MO	modifier
(MR	rhetorical modifier)
(MW	way, directional modifier)
...	

Category	Description
NG	negation
NK	noun kernel modifier
NKHD*	head of noun kernel
NMC	numerical component
OA	accusative object
OA2	second accusative object
OC	clausal object
OG	genitive object
OP	prepositional object
PAR	parenthesis
PC*	prepositional complement
PD	predicate
PG	phrasal genitive
PH	placeholder
PM	morphological particle
PNC	proper noun component
RC	relative clause
RE	repeated element
RS	reported speech
SB	subject
SBP	passivized subject
SP	subject or predicate
SVP	separable verb prefix
UC	unit component
VO	vocative

Table A.3: Stuttgart-Tübingen-Tagset STTS (Albert et al., 2003)

Tag	Description
ADJA	attributive adjective
ADJD	adverbially or predicatively used adjective
ADV	adverb
APPR	preposition or left part of circumposition
APPRART	preposition with article
APPO	postposition
...	

Tag	Description
APZR	right part of circumposition
ART	definite or indefinite article
CARD	cardinal number
FM	foreign language material
ITJ	interjection
KOUI	subordinating conjunction with *zu*-infinitive
KOUS	subordinating conjunction (with a sentence)
KON	coordinating conjunction
KOKOM	comparative conjunction
NN	common noun
NE	proper noun
PDAT	attributive demonstrative pronoun
PDS	substituting demonstrative pronoun
PIAT	attributive indefinite pronoun
PIS	substituting indefinite pronoun
PPER	irreflexive personal pronoun
PPOSAT	attributive possessive pronoun
PPOSS	substituting possessive pronoun
PRELAT	attributive relative pronoun
PRELS	substituting relative pronoun
PRF	reflexive personal pronoun
PWS	substituting interrogative pronoun
PWAT	attribute interrogative pronoun
PWAV	adverbial interrogative or relative pronoun
PROAV	pronominal adverb
PTKZU	*zu* before an infinitive
PTKNEG	negation particle
PTKVZ	separated verbal particle
PTKANT	answer particle
PTKA	particle with adjective or adverb
TRUNC	first (separated) part of composition
VVFIN	finite content verb
VAFIN	finite verb, primary auxiliary
VMFIN	finite verb, modal auxiliary
VVINF	infinitive of content verb
VAINF	infinitive, primary auxiliary
VMINF	infinitive, modal auxiliary
...	

Tag	Description
VVIMP	imperative content verb
VAIMP	imperative, primary auxiliary
VVPP	past participle of content verb
VAPP	past participle, primary auxiliary
VMPP	past participle, modal auxiliary
VVIZU	infinitive of content verb with *zu*
XY	non-word, containing special characters
$,	comma
$.	sentence-final punctuation
$(	other punctuation, sentence-internal

Appendix B

Program Options

The conversion and extraction have been implemented in Java 6. Besides the treebank and the lexicon, the system as well as the implementation will be made available on request. The most important points are specified here for reference.

Input

- the original Tiger Treebank in Negra export format (`.export`)
- the Tiger Treebank in PTB format i.e. without crossing branches (`.penn`)
- a head percolation table for the Tiger Treebank (`.txt`)

The head percolation table (an adapted version of the one presented in (Rehbein, 2009, p.127)) as well as a Perl script that generates PTB format from Negra export format (a modified version of a script by Michael Schiehlen) are part of the system.

Parameters

1. `<source_path>`
   Path to the folder with the input resources
2. `<output_path>`
   Path to the folder where the output files will be created
3. `<file_affix>`
   A common affix for all output files
4. `-predStrat <X>`
   Strategy on how to treat sister-adjunction and adjunction from the right when computing prediction trees. `X` can be `a`, `b` or `c`. Also see Section 3.4.2 for details.

   `a` During derivation, sister-adjunction is only allowed on complete nodes. During training, both top and bottom halves of the node to which a modifier tree sister-adjoins have to be seen or predicted. For the extraction of prediction trees, this means that

regular adjunction from the right is predicted if a modifier tree sister-adjoins to the future adjunction site.

`b` In contrast to `a` regular adjunction from the right is not predicted if a tree sister-adjoins to the future adjunction site. In the derivation, the sister-adjoining tree will therefore end up at the foot node of the auxiliary tree instead of at the root node.

`c` Extension of `b`: during derivation, sister-adjunction is allowed to incomplete nodes, e.g. open substitution nodes. During training, the bottom half of the node to which a modifier tree sister-adjoins does not have to be seen, thus it is not predicted.

5. `-matStrat <X>`
   Strategy concerning the extraction of multi-anchored trees (MATs). `X` can be `i` (no MATs), `ii` (only MATs where the main anchor is the leftmost one) or `iii` (all MATs). See Section 3.4.3 for details.

6. `-ourSisAdj <X>`
   Parameter for setting the definition of sister-adjunction. `X` can be `true` (our definition with additional root node for restricting the adjunction category) or `false` (Chiang's definition)

Running it

The main class and entry point is `pltag.german.TAGCorpus`. Since the complete Tiger Treebank is read in, give the Java VM enough memory: e.g. `-Xmx1536m`.

Output

1. the Tiger Treebank converted to (P)LTAG format, i.e. derived (P)LTAG trees

2. the lexicon specifying prediction trees and canonical elementary trees, sentence by sentence

The format of the treebank and the lexicon should be self-explanatory if you are familiar with tree-adjoining grammars. They use the standard bracketing format. (Left and right round brackets in the treebank sentences and the POS tags a substituted by `*LRB*` and `*RRB*` respectively.) In the following some more details are provided and illustrated with the help of sentence 256.

```
%%% 256
Daraus kann *T1* gefolgert werden :
PROAV Daraus   VMFIN kann   VVPP gefolgert   VAINF werden   $. :
(S (PROAV Daraus)(VMFIN kann)(VP (VP (*T1* -)(VVPP gefolgert))(VAINF werden))
    ($. :))
```

In the treebank file, each sentence is represented in four lines. The sentence ID is followed by the words of the sentence including traces separated by white-space. Those are the terminals of the treebank tree. In the third line, the words of the sentence excluding traces are shown, each of them preceded by the gold POS tag. This is most likely the input for a parser. The fourth line presents the gold parse tree of the sentence. A secondary edge is represented in the gold parse tree by an edgemarker. This is an index i and the functional label of the edge between dollar symbols (e.g. `$$1-HD$$`) following the phrasal label of the source node of the secondary edge. The phrasal label of the target node of the secondary edge is followed by the corresponding index i between dollar symbols as well (e.g. `$$1$$`). Morphological information is encoded between brackets (e.g. `[acc]`) following the phrasal label, but preceding the edgemarker in case there is one. If not needed, secondary edge markers and morphological information can be removed easily from this file format.

```
%%% 256
prediction:   ARG   (S^null_1 (PROAV-*T1*-OP^1_null! )(VP-OC^1_1 (*T1*-OP^1_1 -)
    (VVPP-HD^1_1 )))
Daraus daraus   ARG   (PROAV-OP^null_x Daraus<>)   --
kann können   MOD   (S^null_x* (VMFIN-HD^x_x kann<>))   3.sg.pres.ind
gefolgert folgern   ARG   (S^null_x (PROAV-*T1*-OP^x_null! )(VP-OC^x_x
    (*T1*-OP^x_x -)(VVPP-HD^x_x gefolgert<>)))   psp
werden werden   ADJ   (VP-OC^null_x (VP-OC^x_null* )(VAINF-HD^x_x werden<>))   inf
: :   MOD   (S^null_x* ($.^x_x :<>))   --
```

In the lexicon file, each extracted elementary tree is represented on one line. Each line has five columns, separated by tabs. The first columns either marks a prediction tree as such (`prediction:`) or contains the lexical anchor of the canonical elementary tree followed by its lemma. For multi-anchored trees, the main anchor precedes the co-anchors. The second column specifies the type of elementary tree: `ARG` for initial trees, `MOD` for modifier trees and `ADJ` for auxiliary trees.

The third column contains the elementary tree itself in bracketing format. Substitution nodes are indicated by an exclamation mark `!`. The asterisk `*` marks the foot node in auxiliary trees as well as the root node of modifier trees. Non-terminals are represented by the phrase label and the following additional information in the given order, if present: (1) a dash and the functional label e.g. `-HD`, (2) morphological information between brackets e.g. `[acc]`, (3) secondary edge markers. In the lexicon, secondary edge markers are carried by the target node, providing information about the source node, as specified in Section 3.3.5, e.g. `$$VAFIN-HD$$`. Nodes furthermore carry information about the status of their top (`^X`) and bottom (`_X`) halves. `X` > 0 are prediction markers. `X=x` indicates that the corresponding half is part of the current elementary tree. The main lexical anchor is indicated by `<>`, the co-anchors carry prediction markers. The forth column contains the morphological information from the Tiger Treebank. It is most likely not immediately useful.

The lexicon contains as much information as possible, but it is likely that users will discard some of it. The information about syntactic functions, morphology and secondary edges can be be removed from the lexicon in a straightforward way. People interested in LTAG can ignore the prediction trees. The information about top and bottom node halves is only relevant if a corresponding parser uses the notion of node halves.

All traces that were introduced by converting the Tiger Treebank trees with crossing edges to classical phrase structure trees with indexed traces are part of the lexicon. However, since TAG cannot adequately represent such discontinuous information, trace and filler are not always bound together in one elementary tree (see Section 3.3.4). This leads (a) to argument traces that are bound in the "wrong" elementary tree and (b) to modifier traces that anchor elementary trees. The latter are indicated by `EMPTY-ANCHOR` in the fifth column. Since users most likely want a lexicalized TAG, those lexicon entries should be discarded. (Alternatively, a tree-set for each unbound trace-filler pair could be built.)