Parsing Discontinuous Constituents in English

Kilian Evang

Tübingen, January 5, 2011

Master's thesis in Computational Linguistics submitted to the
Seminar für Sprachwissenschaft, Universität Tübingen

Supervisor and first examiner: Prof. Dr. Laura Kallmeyer
Second examiner: PD Dr. Frank Richter

Erklärung

Hiermit versichere ich, dass ich die vorgelegte Arbeit selbstständig und nur mit den angegebenen Quellen und Hilfsmitteln einschließlich des WWW und anderer elektronischer Quellen angefertigt habe. Alle Stellen der Arbeit, die ich anderen Werken dem Wortlaut oder dem Sinne nach entnommen habe, sind kenntlich gemacht.

Kilian Evang

Abstract

Recent work by Maier and Kallmeyer (2010) on data-driven constituent parsing of German text has used Probabilistic Linear Context-Free Rewriting Systems (PLCFRS), a generalization of Probabilistic Context-Free Grammars (PCFG) that allows for discontinuous constituents in the resulting syntactic representations. This makes it possible to recognize certain types of non-local dependencies directly in parsing.

This master's thesis transfers their approach from parsing German to parsing English. Its main contributions are 1) a method for creating a discontinuous constituent treebank of English by automatic conversion from the syntactic annotation of the Penn Treebank, 2) presenting what are to the author's best knowledge the first experimental results on large-scale discontinuous constituent parsing of English.

The existing techniques used for extracting PLCFRSs from treebanks and parsing with them are described in detail. Special attention is given to methods for tuning PLCFRSs for speed and accuracy and the effect of these methods on the English data. Finally, a detailed analysis of the resulting parses with respect to discontinuous constituents reveals strengths and weaknesses of the current setup and points to possible routes for further work on the annotation scheme, the parsing model and the tuning methods.

Zusammenfassung

Für das datengetriebene konstituentenbasierte syntaktische Parsing von deutschsprachigem Text ist von Maier und Kallmeyer (2010) die Benutzung von *Probabilistic Context-Free Rewriting Systems* (PLCFRS) erprobt worden. Dieser Formalismus stellt eine Generalisierung probabilistischer kontextfreier Grammatiken (PCFG) dar. Die syntaktischen Repräsentationen von PLCFRS lassen diskontinuierliche Konstituenten zu, was es ermöglicht, bestimmte Typen nicht lokaler Abhängigkeiten direkt im Parsing-Vorgang zu erkennen.

Die vorliegende Arbeit überträgt diesen Ansatz vom Deutschen auf das Englische. Ihre Hauptbeiträge sind 1) eine Methode zur Erstellung einer diskontinuierlichen Konstituentenbaumbank des Englischen durch automatische Konversion der syntaktischen Annotation der Penn Treebank und 2) die nach dem Wissen des Autors ersten Ergebnisse größer angelegter Experimente zum Parsing englischen Textes mit diskontinuierlichen Konstituenten.

Die bestehenden Verfahren zur Extraktion von PLCFRS-Grammatiken aus Baumbanken und zum Parsing mit PLCFRS werden im Detail beschrieben. Ein Schwerpunkt der Darstellung liegt auf Methoden, Grammatiken mit Blick auf Parsing-Effizienz und -Performanz zu optimieren, sowie auf der Wirksamkeit dieser Methoden auf den englischsprachigen Daten. Eine detailierte Analyse der Parsing-Ergebnisse hinsichtlich diskontinuierlicher Konstituenten zeigt schließlich Stärken und Schwächen der verwendeten Parameter auf und deutet auf Möglichkeiten weiterer Verbesserung durch Veränderungen am Annotationsschema, am Parsing-Modell und an den Optimierungsmethoden hin.

Acknowledgements

I am happy to have come to the point where I can submit this master's thesis. Of the many people who have helped me get there, some are particularly prominent.

First and foremost, I am very grateful to Laura Kallmeyer for her time, patience and enthusiasm in supervising the thesis. I count myself truly lucky to have received such committed and valuable guidance and support. I am also very grateful to Wolfgang Maier for his involvement in many of the discussions surrounding this work, and for his invaluable support in conducting the experiments.

I wish to thank all of my teachers and fellow students at the Seminar für Sprachwissenschaft in Tübingen for creating the pleasant and inspiring atmosphere that has defined my years of study here, and for all that I have learned. I am especially grateful to Frank Richter and Johannes Dellert for years of enjoyable and fruitful collaboration.

Finally and in many ways most importantly, I wish to thank my friends and family for their friendship, love and support, and for being the wonderful persons they are.

Contents

Chapter 1.

Introduction

1.1. Motivation

Syntactic parsing is a fundamental component in many natural-language processing applications. Its input is text in the form of a sequence of tokens, i.e. words and punctuation marks. Its output is a *syntactic representation* containing information about the relations in which tokens in the sentence stand to each other and what syntactic functions they fulfill. Syntactic representations serve as the input to further processing, e.g. for information extraction or for machine translation.

One of the most widely used kinds of syntactic representation is phrase structure trees, one per sentence, describing a derivation of the sentence in a Context-Free Grammar (CFG). This kind of representation embodies the notion that a sentence is recursively built up from smaller units called *constituents* or *phrases* which can also be regarded in isolation and have syntactic structure and meaning on their own. Consider sentence 1.1:

(1.1) Mr. Vinken is chairman of Elsevier N.V., the Dutch publishing group.

Here, the tokens *of Elsevier N.V., the Dutch publishing group* would widely be considered to form a constituent, more specifically a noun phrase (NP). This is expressed by the tree in Figure 1.1[1] in that these words are the leaves of a subtree, rooted in the PP (prepositional phrase) node. Note that if the tree is supposed to represent a derivation

[1] All tree diagrams with natural language examples in this thesis have been generated from the original syntactic annotation, or a transformed version of the syntactic annotation, or parse trees produced automatically from sentences in, the WSJ part of the Penn Treebank, using the TIGERSearch software (Lezius, 2002).

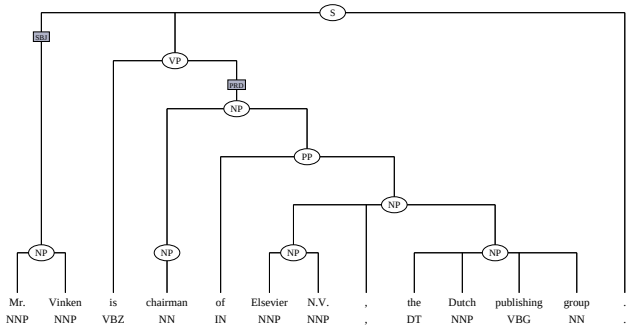


Figure 1.1.: A phrase structure tree from the WSJ treebank

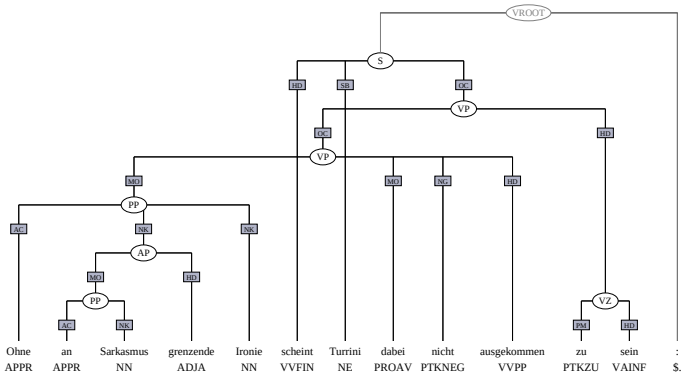


Figure 1.2.: A phrase structure tree from the German treebank NEGRA

tree of the sentence in a CFG, the tokens in each constituent are required to form a continuous subsequence of the sentence. However, this restriction is not shared by all constituent-based theories of syntax. Consider the German sentence 1.2:

(1.2) *Ohne an Sarkasmus grenzende Ironie scheint Turrini dabei*
without on sarcasm bordering irony seems Turrini in doing so
nicht ausgekommen zu sein :
not managed to have :
"Turrini seems not to have managed to do so without irony bordering on sarcasm:"

Here, the discontinuous subsequence *ohne an Sarkasmus grenzende Ironie ... nicht ausgekommen* ("not managed without irony bordering on sarcasm") can be said to form a discontinuous constituent, more specifically a verb phrase (VP), and the complement *ohne an Sarkasmus grenzende Ironie* occupies a position detached from the rest of the

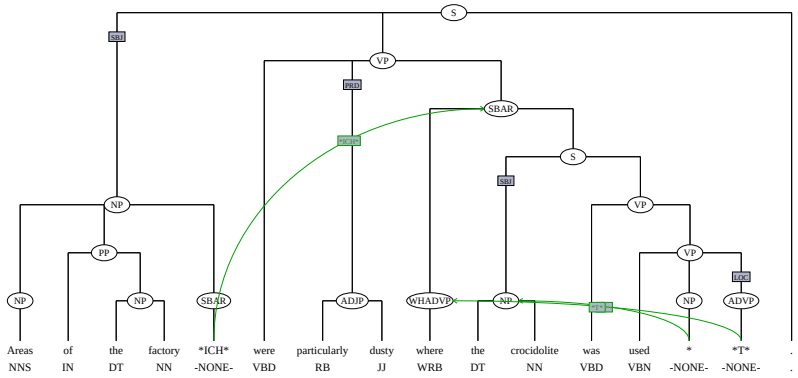


Figure 1.3.: Penn Treebank annotation of an extraposed relative clause

phrase. The tree in Figure 1.2 accordingly groups them under a common node. This creates crossing branches in the tree and makes it impossible to interpret it as a context-free derivation of the sentence.

Phenomena where two non-adjacent parts of a sentence logically belong together are known as *non-local dependencies*, or *long-distance dependencies*, or *long-range dependencies*. Although they are especially frequent in languages with relatively free word order – such as German – English has its share of them too. Consider sentence 1.3:

(1.3) Areas of the factory were particularly dusty where the crocidolite was used.

Here, the extraposed relative clause *where the crocidolite was used* modifies the noun phrase *Areas of the factory* and could therefore be directly grouped with it under the same node. Similar discontinuous analyses of English syntax have long been argued for, see e.g. McCawley (1982). Still, a majority of syntactic analyses of English opt for context-free trees, i.e. the absence of crossing branches. In such cases, non-local dependencies are either not represented or represented using additional structure on top of the trees.

This concerns, for instance, treebanks. A treebank is a large collection of text where the sentences have been manually annotated with the corresponding syntactic representation. One of the largest and most widely used English treebanks is the Wall Street Journal part of the English Penn Treebank (Marcus et al., 1993), henceforth called the WSJ treebank. It consists of context-free trees which, to represent non-local dependencies, contain additional leaves called *null elements* which do not correspond to any token

occurring in the text, and are decorated with *secondary edges*. For example, consider the analysis for sentence 1.3, shown in Figure 1.3. Here, a secondary edge connects the SBAR node rooting the extraposed relative clause to a null element which marks a position inside the noun phrase that the relative clause modifies.

Fostered by and fostering the wide availability of context-free English treebanks, context-free trees are also a standard syntactic representation of choice for *data-driven parsing* of English. In data-driven parsing, treebanks are used as input to a process of *training*, i.e. automatically extracting rules which can then be used to parse previously unseen text. A standard technique is to extract Probabilistic Context-Free Grammars (PCFG) and use these for parsing. This approach cannot deal with discontinuous constituents and it does not take into account secondary edges, thus the output of the parser does not contain information on non-local dependencies. Such information can, however, be vital for correctly interpreting sentences in further processing steps in an application.

Only recently has data-driven parsing with discontinuous constituents been attempted. Plaehn (2004) describes a probabilistic parsing algorithm for Discontinuous Phrase Structure Grammar (DPSG), a grammar formalism in which the derivation trees may contain crossing branches. Another such grammar formalism, with lower parsing complexity, is Linear Context-Free Rewriting Systems (LCFRS), a generalization of CFG. Maier and Kallmeyer (2010) adopt the PCFG approach to it and use it for data-driven parsing of German. They achieve accuracy comparable with previous results for parsing German while producing syntactic representations that are more informative than context-free trees. This suggests that it may be possible to overcome the limitations of current standard approaches to parsing English, concerning non-local dependencies, in a similar manner. Another approach to data-driven LCFRS parsing is presented, but not experimentally explored, in Levy (2005, Chapter 4).

1.2. Thesis aims

The primary aim of this thesis is to present a methodology and first results for parsing English text using a probabilistic treebank LCFRS which encodes non-local dependencies using discontinuous constituents. To this end, parsing experiments are carried out using the parser of Maier and Kallmeyer (2010) with probabilistic LCFRSs (PLCFRSs) extracted from the WSJ treebank using the extraction algorithm of Maier and Søgaard

(2008), after transforming the treebank to a format where non-local dependencies are encoded using discontinuous constituents, as far as possible. A secondary aim is to provide a detailed discussion of how such a transformation can be carried out automatically, and how problematic cases can be treated. A final aim is to evaluate the approach with respect to the additional information that the parse trees provide compared with context-free parses, i.e. which types of non-local dependencies are correctly captured and which ones still pose problems.

1.3. Related work

While the above-mentioned body of work on data-driven discontinuous constituent parsing is still small, there are other means to automatically obtain syntactic representations with both local and non-local dependencies. Most work to date has to this end combined context-free parsers such as those of Collins (1999) or Charniak (2000) with preprocessing and/or postprocessing steps and carried out experiments on the WSJ treebank. Following Nivre (2006) and using the terminology of Cahill et al. (2004), these approaches can be classified into two broad categories: pipeline approaches and integrated approaches. A further possibility to obtain both local and non-local dependencies, besides discontinuous constituent parsing, is (non-projective) dependency parsing. The three classes of approaches are briefly presented in the following sections.

1.3.1. Pipeline approaches

Pipeline approaches take the output of a context-free parser trained on a normal context-free treebank and then try to recover non-local dependencies from the parse trees. Non-local dependencies are frequently encoded by adding indices and null elements, using the same format as in the original treebank and thus facilitating evaluation against the gold standard.

The method of Johnson (2002) finds pairs of nodes likely to be in a non-local dependency relationship by searching in the parse trees for tree fragments found to frequently connect non-locally dependent nodes in the gold standard trees.

Jijkoun and de Rijke (2004) convert parse trees to dependency graphs (cf. Section 1.3.3), then use memory-based learning with a variety of features extracted from

gold-standard trees subjected to the same dependency conversion to determine where nodes and edges should be inserted, deleted or relabeled.

The method presented in Levy and Manning (2004) and Levy (2005, Chapter 3) similarly uses machine learning with various contextual features, but operates directly on the constituent trees and is more focused on the task of inserting specific types of null elements and indices in the Penn Treebank annotation scheme. A parsing system which uses a similar postprocessing method after recovering function tags which help in recovering non-local dependencies is presented in Gabbard et al. (2006).

Campbell (2004) presents a method that abstains from probabilistic inference and uses a completely hand-crafted algorithm for recovering non-local dependencies based on linguistic principles.

1.3.2. Integrated approaches

Integrated approaches are integrated in the sense that the training treebank is already enriched with additional information about non-local dependencies, in the form of null elements and/or enhanced node labels. Thus, the probabilistic context-free parser is at least implicitly aware of non-local dependencies and their distributional properties. Sometimes the probabilistic models of parsers are modified to explicitly take non-local dependencies into account. In integrated approaches, parsers produce parse trees from whose null elements and/or enhanced node labels non-local dependencies can be read off more or less directly. The term "integrated" should not imply that producing the syntactic representations from text invariably consists of a single (parsing) step, some preprocessing and/or postprocessing may be needed.

Model 3 of Collins' parser (Collins, 1999) and the method of Dienes and Dubey (2003) use null elements (traces) to mark the phrases which are in a non-local dependency relationship with a distant element. Collins makes this possible by extending his parsing model to allow a limited treatment of empty elements, Dienes and Dubey use a shallow trace tagger to insert traces into the text before parsing. Both methods use enhanced non-terminals to propagate information about traces and extracted elements in the tree; both methods also need postprocessing to assign traces to the constituent at the other end of the non-local dependency.

Hockenmaier (2003) converts the WSJ treebank to normal-form derivation trees for Combinatory Categorial Grammar (CCG), whose complex categories and combinatory

rules, together with indices manually inserted into some lexical categories, represent non-local dependencies explicitly so they can be found by a context-free parser without a need for null elements.

Cahill et al. (2004) present an approach similar to that of Johnson (2002) in that dependencies are found by probabilistic inference from parts of the tree (paths in this case) connecting the two dependent nodes. Since their format for representing non-local dependencies is based on enhanced node labels and does not use null elements, they can not only use their algorithm as a postprocessing step but also apply it to the training corpus and extract a "non-local-dependency-aware" grammar to use in an integrated approach, which they find to yield superior results.

1.3.3. Dependency parsing

Dependency-based methods for syntactic parsing produce syntactic structures that do not employ the notion of constituents or phrases but consist only of dependencies between individual words in the sentence. These structures are called *dependency graphs*. While phrase structure trees represent local dependencies as sister relations and non-local dependencies through additional devices such as secondary edges, dependency graphs represent both types of dependencies in the same format, making it a natural choice of representation for further processing even when parsing was done with a constituent-based parser. Various methods for converting constituent trees to dependency graphs have been designed for facilitating semantic interpretation, including that used as part of Jijkoun and de Rijke (2004)'s dependency recovery method and that included with recent versions of the Stanford parser (de Marneffe et al., 2006).

As Levy (2005) shows, the distinction between context-free and discontinuous trees has a direct equivalent in dependency graphs under standard conversion methods, viz. the distinction between projective and non-projective dependency graphs. The relation between both kinds of syntactic structures is further formalized and explored in Maier and Lichte (2009).

Dependency parsing is the process of directly producing dependency graphs from text. Some dependency parsing algorithms, such as that of Nivre (2003), can only produce projective graphs and thus cannot recognize non-local dependencies. However, state-of-the-art data-driven dependency parsers (McDonald et al., 2005; Nivre et al., 2007) can handle non-projective dependencies. These parsers are not grammar-based,

and dependency parsing in general is less strongly influenced by formal grammar than constituent parsing, where CFG as a very well-understood grammar formalism is still very dominating. This may be an important reason why non-local dependencies seem better integrated in the dependency parsing world today than in constituent parsing.

1.4. Structure of the thesis

The remainder of the thesis is structured as follows. Chapter 2 gives an introduction to Linear Context-Free Rewriting Systems (LCFRSs) and probabilistic LCFRSs (PLCFRSs) and presents the algorithms for extracting a PLCFRS from a treebank as well as for parsing text with a PLCFRS. Chapter 3 discusses the way non-linear dependencies are annotated in the WSJ treebank and develops principles and an algorithm for automatically incorporating this information into the tree structures by admitting crossing branches. Chapter 4 presents various techniques for tuning treebank PLCFRSs and surveys their effects on parsing accuracy for a limited test set of test data. In Chapter 5, a parsing experiment on a larger test set is described, and the parser's ability to recover specific types of non-local dependencies is evaluated in detail. Chapter 6 summarizes and discusses the findings.

Chapter 2.

Linear Context-Free Rewriting Systems (LCFRS)

This chapter presents the formalism and the essential techniques used for the present work: Linear Context-Free Rewriting Systems (LCFRS), their probabilistic extension PLCFRS, the extraction of PLCFRSs from treebanks with discontinuous constituents, and a CYK parsing algorithm for PLCFRS.

2.1. Introduction to LCFRS

Since the 1980s, it has been known that context-free grammars cannot describe all natural languages (Shieber, 1985), much less with derivation structures that contain all the relevant dependencies, since some of them are non-local (Bresnan et al., 1982; Joshi, 1985). There has been considerable interest in grammar formalisms which can describe non-local dependencies, yet are still restricted enough to not only permit efficient parsing, but also reflect restrictions that may be found to apply to the syntax of natural language itself. This led to the definition of *mildly context-sensitive* formalisms (Joshi, 1985), here defined following Kallmeyer (2010):

Definition 2.1 (Mild context-sensitivity):

1. A set of languages $\mathcal{L}$ over an alphabet T is *mildly context-sensitive* iff

   a) $\mathcal{L}$ contains all context-free languages.

$$\begin{aligned} S(XYZ) &\rightarrow A(X,Y,Z) \\ A(aX,aY,aZ) &\rightarrow A(X,Y,Z) \\ A(bX,bY,bZ) &\rightarrow A(X,Y,Z) \\ A(a,a,a) &\rightarrow \varepsilon \\ A(b,b,b) &\rightarrow \varepsilon \end{aligned}$$

Figure 2.1.: Rules of an LCFRS for the language $\{www|w \in \{a,b\}^+\}$

b) $\mathcal{L}$ can describe cross-serial dependencies: there is an $n \geq 2$ such that $\{w^k|w \in T^*\} \in \mathcal{L}$ for all $k \leq n$.

c) The languages in $\mathcal{L}$ can be parsed in polynomial time.

d) The languages in $\mathcal{L}$ have the constant growth property.

2. A formalism is mildly context-sensitive iff the set of languages it can describe is mildly context-sensitive.

Informally, a language has the constant growth property if the length of its words, when ordered by length, grows in a linear way. It is defined as follows in Weir (1988):

Definition 2.2 (Constant growth property): Let X be an alphabet and $L \subseteq X^*$. L has the *constant growth property* iff there is a constant $c_0 > 0$ and a finite set of constants $C \subset \mathbb{N}\backslash\{0\}$ such that for all $w \in L$ with $|w| > c_0$, there is a $w' \in L$ with $|w| = |w'| + c$ for some $c \in C$.

Linear Context-Free Rewriting Systems (LCFRS) (Vijay-Shanker et al., 1987; Weir, 1988) is one of the most powerful known mildly context-sensitive grammar formalisms. It is equivalent to a number of other formalisms, including Multiple Context-Free Grammars (MCFG) (Seki et al., 1991) and Simple Range Concatenation Grammars (SRCG) (Boullier, 1998).

In an LCFRS, similarly to a CFG, there are rules with left-hand sides (LHSs) and right-hand sides (RHSs), and words are derived by recursively applying rules. In contrast to a CFG, a non-terminal does not yield a single string of terminals, but rather a vector of strings. The strings in one vector can be interleaved with the strings in other vectors, thereby creating discontinuities. Every non-terminal A yields vectors with a

fixed number of dimensions $dim(A)$, called the *fan-out* of A. The start symbol always has fan-out 1 since it must yield the whole word.

Rather than the standard notation for LCFRS rules, I will use the SRCG notation from Boullier (1998) because it is easier to read. Figure 2.1 shows the rules of an example LCFRS which generates the language $\{www|w \in \{a,b\}^+\}$, i.e. the language of all words consisting of three times the same non-empty string of a's and b's. Apart from the start symbol S, there is just one non-terminal, A, with $dim(A) = 3$. Looking at it in a bottom-up fashion, the grammar can be thought of as generating all words in the language by building up triples of identical strings and finally concatenating them. The fourth and fifth rule provide the starting point for this process; they specify that the tuples $\langle a,a,a\rangle$ and $\langle b,b,b\rangle$ are in the yield of A. The second and third rule prepend to all three parts the same symbol (a resp. b), i.e. they specify that for any triple $\langle X,Y,Z\rangle$ in the yield of A, the triples $\langle aX,aY,aZ\rangle$ and $\langle bX,bY,bZ\rangle$ are also in the yield of A. Finally, the first rule concatenates the three parts, specifying that the 1-tuple containing the concatenation of any tuple in the yield of A is in the yield of S.

Definition 2.3 (Linear Context-Free Rewriting System): A linear context-free rewriting system (LCFRS) is a tuple $\langle N,T,V,P,S\rangle$ where

- N,T,V are pairwise disjoint finite sets of symbols. The elements of N are called *non-terminals*, those of T *terminals* and those of V *variables*.
- Each non-terminal $A \in N$ has an associated *fan-out* $dim(A) \geq 1$.
- P is a finite set of *rules* of the form $A(\chi_{01},\ldots\chi_{0dim(A)}) \rightarrow A_1(\chi_{11},\ldots\chi_{1dim(A_1)})\ldots A_m(\chi_{m1},\ldots\chi_{mdim(A_m)})$ where
  - the part left of the arrow is called *left-hand side (LHS)* and the part right of the arrow is called *right-hand side (RHS)*,
  - $m \geq 0$ (called the *rank* of the rule),
  - $A, A_1,\ldots A_m \in N$,
  - $\chi_{01},\ldots\chi_{0dim(A)} \in (T \cup V)^*$, are strings of terminals and variables called the *LHS arguments*,
  - $\chi_{11},\ldots\chi_{1dim(A_1)},\ldots\chi_{m1},\ldots\chi_{mdim(A_m)} \in V$, i.e. each argument of a RHS element consists of exactly one variable,

  - every variable occurring in a rule occurs exactly once on the LHS and exactly once on the RHS.

  If $m = 0$, the RHS is written as ε. The *argument vector* of a LHS or RHS element is defined as the vector whose components are the arguments of that LHS or RHS element, in the same order.

- $S \in N$ with $dim(S) = 1$ is called the *start symbol*.

An LCFRS G where for all $A \in N$, $dim(A) \leq k$ is said to have fan-out k and is called a k-LCFRS. An LCFRS whose rules have a maximal rank m is said to have rank m.

Definition 2.4 (String language of an LCFRS): Let $G = \langle N, T, V, P, S \rangle$ be an LCFRS.

1. The function *yield* maps non-terminal symbols to sets of vectors of strings. It is defined as follows. Suppose that $A \in N$ and that $r = A(\boldsymbol{\alpha}) \rightarrow A_1(\boldsymbol{\alpha_1}) \ldots A_m(\boldsymbol{\alpha_m}) \in P$ with $m \geq 0$ and that $\vec{\rho_1} \in yield(A_1), \ldots \vec{\rho_m} \in yield(A_m)$.[1] Let $\vec{\alpha}$ be the argument vector of the LHS of r. Then $\vec{\rho} \in yield(A)$ where $\vec{\rho}$ is obtained from $\vec{\alpha}$ by replacing each variable X with $\vec{\rho_i}(j)$ such that X is the jth argument in $\boldsymbol{\alpha_i}$. Nothing else is in $yield(A)$.

2. The string language of G is $L(G) = \{w | \langle w \rangle \in yield(S)\}$.

For extracting LCFRSs from discontinuous treebanks, the notion of an LCFRS *derivation tree* as introduced in Kallmeyer (2010, Chapter 6) is central. An LCFRS derivation tree describes the derivation of a word from the start symbol of the grammar, much like a classical phrase-structure tree (a CFG derivation tree) describes the derivation of a word using a context-free grammar. LCFRS are conventionally drawn so that the visual order of the leaves corresponds to the order of terminals in the word, but unlike CFG derivation trees, they are not ordered, which leads to the possibility of crossing branches. In order to unambiguously identify *occurrences* of terminals in the word, the leaves of an LCFRS derivation tree are thus labeled with *ranges* corresponding to the positions of terminals and empty elements in the word. The internal nodes are labeled with non-terminals. Each local tree (an internal node together with its children) corresponds to the application of a rule where the parent is labeled with the LHS non-terminal and the children consist of a) internal nodes labeled with the RHS

[1] Boldface Greek letters represent sequences of arguments.

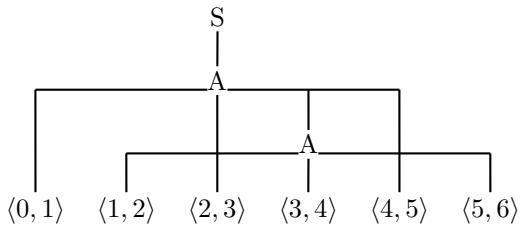


Figure 2.2.: Derivation tree for the word *ababab* in the grammar from Figure 2.1

non-terminals and b) leaves corresponding to terminals and empty arguments occurring on the LHS. An example derivation tree is given in Figure 2.2.

To define derivation trees precisely, we first need to define the notions of ranges, range concatenations and rule instances, adapted from Range Concatenation Grammars (Boullier, 1998). I assume the definition of trees as triples $\langle V, E, r\rangle$ where $\langle V, E\rangle$ is a directed graph and r is the root of the tree, as given e.g. in Kallmeyer (2010, Chapter 1).

Definition 2.5 (Range): Let $G = \langle N, T, V, P, S\rangle$ be an LCFRS and w a word with $w = w_1 \dots w_n$ where $w_i \in T$ for $1 \leq i \leq n$. A *range* in w is a pair $\langle l, r\rangle$ with $0 \leq l \leq r \leq n$. Its *yield* $\langle l, r\rangle(w)$ is the substring $w_{l+1} \dots w_r$. Two ranges $\langle l_1, r_1\rangle$ and $\langle l_2, r_2\rangle$ are *non-overlapping* iff $r_1 \leq l_2 \vee r_2 \leq l_1$.

Definition 2.6 (Range concatenation):

1. The concatenation of two ranges $\langle l_1, r_1\rangle$ and $\langle l_2, r_2\rangle$ is defined as $\langle l_1, r_2\rangle$ iff $r_1 = l_2$, otherwise it is undefined.
2. The concatenation of a string of ranges $\langle l, r\rangle\boldsymbol{\alpha}$ where $\boldsymbol{\alpha}$ is a possibly empty string of ranges is defined as $\langle l, r\rangle$ if $\boldsymbol{\alpha}$ is empty, and as $\langle l, r\rangle$ concatenated with the concatenation of $\boldsymbol{\alpha}$ otherwise.

Definition 2.7 (Rule instance): Let $G = \langle N, T, V, P, S\rangle$ be an LCFRS, $w \in T^*$ and $r = A(\boldsymbol{\alpha}) \rightarrow A_1(\boldsymbol{\alpha_1}) \dots A_m(\boldsymbol{\alpha_m}) \in P$. Then r' is an instance of r with respect to w if it is obtained from r by

1. replacing every occurring variable with a range in w, on both sides,
2. replacing every occurrence of a terminal t on the LHS with a range $\langle l, l+1\rangle$ such that $\langle l, l+1\rangle(w) = t$,

3. replacing every empty LHS argument with a range $\langle i, i\rangle$ in w and
4. replacing every LHS argument with its concatenation

such that the resulting ranges on the LHS are pairwise non-overlapping.

Definition 2.8 (Derivation tree): Let $G = \langle N, T, V_G, P, S\rangle$ be an LCFRS and $w = a_1 \dots a_n$ with $a_i \in T, 1 \leq i \leq n$.

1. Let $D = \langle V, E, r\rangle$ be a tree such that there are n pairwise different leaves $u_1, \dots u_n$ in D with $l(u_i) = \langle i-1, i\rangle$ $(1 \leq i \leq n)$ and every other leaf is labeled $\langle i, i\rangle$ for some i with $0 \leq i \leq n$.
2. The function *n-yield*, assigning vectors of ranges to nodes in the tree, is defined as follows:
   a) For every leaf $u \in V$, $\textit{n-yield}(u) = \{l(u)\}$.
   b) For every internal node $v_0 \in V$, for every order $v_1, \dots v_k$ of the pairwise different daughters of v_0 that are internal nodes such that $l(v_i) = A_i$ for $0 \leq i \leq k$, for every rule $r = A_0(\boldsymbol{\alpha_0}) \rightarrow A_1(\boldsymbol{\alpha_1}) \dots A_k(\boldsymbol{\alpha_k})$, for every instance $A_0(\vec{\rho_0}) \rightarrow A_1(\vec{\rho_1}) \dots A_k(\vec{\rho_k})$ of r wrt. w, $\rho_0 \in \textit{n-yield}(v_0)$ if
      i. $\vec{\rho_i} \in \textit{n-yield}(v_i)$, $1 \leq i \leq k$,
      ii. for every terminal t occurring in $\boldsymbol{\alpha_0}$, v_0 has a child u where u is a leaf and $l(u)$ is the range with which t was replaced to obtain $\vec{\rho_0}$ in instantiating r,
      iii. for all $1 \leq i \leq \textit{dim}(A_0)$ such that the ith argument in $\boldsymbol{\alpha}$ is empty, v_0 has a child u where u is a leaf and $l(u) = \vec{\rho_0}(i)$ and
      iv. v_0 has no other children which are leaves.

      Nothing else is in $\textit{n-yield}(v_0)$.
3. D is a derivation tree of w in G iff $l(r) = S \wedge \langle\langle 0, n\rangle\rangle \in \textit{n-yield}(r)$.

2.2. PLCFRS extraction from treebanks

Maier and Søgaard (2008) present a technique for extracting LCFRSs from treebanks with crossing branches by interpreting the trees as derivation trees[2] of the sentences in a grammar that is not known. This grammar is then "recovered" from the trees. The following presentation assumes that the treebank does not contain null elements.

First, we define the function *rule* to extract rules from internal nodes in treebank trees as follows: For every tree $t = \langle V, E, r_t \rangle$ in the treebank whose leaves are $w_1, \ldots w_n$ (in the order of the sentence which t spans), use the variables $X_1, \ldots X_n$. For every internal node $v_0 \in V$ whose children that are internal nodes are $v_1, \ldots v_m$, conventionally ordered by the leftmost dominated leaf, let $r = A_0(\boldsymbol{\alpha_0}) \rightarrow A_1(\boldsymbol{\alpha_1}) \ldots A_m(\boldsymbol{\alpha_m})$ such that for $0 \leq i \leq m$,

- for every leaf w_j dominated by v_i, $\boldsymbol{\alpha_i}$ contains the variable X_j such that X_j precedes X_k in $\boldsymbol{\alpha_i}$ if $j < k$,
- $\boldsymbol{\alpha_i}$ contains no other components,
- X_j is the last component of an argument in $\boldsymbol{\alpha_i}$ iff X_j occurs in $\boldsymbol{\alpha_i}$ and v_i does not dominate w_{j+1}, i.e. an argument boundary is introduced at each discontinuity,
- there are no empty arguments in $\boldsymbol{\alpha_i}$ and
- $A_i = l(v_i)_f$ where f is the number of arguments in $\boldsymbol{\alpha_i}$, i.e. the symbols are equipped with an index indicating the number of spans in the yield of the corresponding node. This ensures that each non-terminal in the grammar will have a fixed fan-out.

Then $\mathit{rule}(v_0) := r'$ where r' is obtained from r by

1. replacing each X_i in $\boldsymbol{\alpha_0}$ by the terminal $l(w_i)$ if w_i is a child of v_0 and
2. replacing each span of variables $X_i \ldots X_j$ that occurs as a RHS argument by a new variable on both sides. This ensures that each RHS argument consists of exactly one variable. The new variables are chosen according to some canonical naming scheme which, together with the ordering convention mentioned above, guarantees that *rule* is a function and that equivalent rules extracted from different nodes are equal.

[2] Almost; the leaves of treebank trees are assumed to be labeled with terminals rather than ranges. The information about the ranges of leaves is still present because each leaf is assigned to a specific token in the sentence.

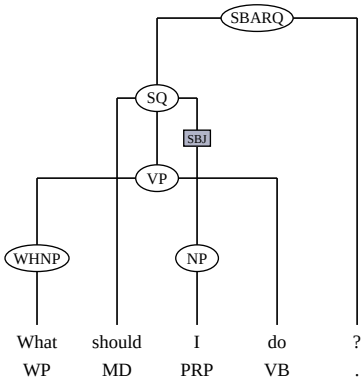


Figure 2.3.: A discontinuous tree

$$
\begin{aligned}
\mathrm{S}_1(X) &\rightarrow \mathrm{SBARQ}_1(X) \\
\mathrm{SBARQ}_1(XY) &\rightarrow \mathrm{SQ}_1(X)._1(Y) \\
\mathrm{SQ}_1(XYZU) &\rightarrow \mathrm{VP}_2(XU)\mathrm{MD}_1(Y)\mathrm{NP}_1(Z) \\
\mathrm{VP}_2(X,Y) &\rightarrow \mathrm{WHNP}_1(X)\mathrm{VB}_1(Y) \\
\mathrm{WHNP}_1(X) &\rightarrow \mathrm{WP}_1(X) \\
\mathrm{NP}_1(X) &\rightarrow \mathrm{PRP}_1(X)
\end{aligned}
$$

Figure 2.4.: Rules extracted from the tree in Figure 2.3

One new non-terminal S_1 is introduced as the start symbol since trees in a treebank may have differently labeled roots and all sentences in the treebank should be words in the language described by the extracted grammar. Then the grammar to extract from the treebank is the LCFRS $\langle M \cup \{S_1\}, T, V, P, S_1\rangle$ where

- T is the set of all leaf labels in the treebank,
- $P = \{rule(v) | v \text{ is an internal node in the treebank}\} \cup \{S_1(X) \rightarrow A_1(X) | v \text{ is a root in the treebank}, A = l(v)\}$,
- V is the set of variables occurring in the rules in P and
- $M = \{A | \text{ some } A(\boldsymbol{\alpha}) \rightarrow \Phi \in P\}$.

As an example, consider the treebank consisting of the single discontinuous tree given in Figure 2.3 and the set of rules extracted from it, given in (Figure 2.4).

For $r \in P$, $count(r)$ is defined as the number of nodes v in the treebank for which $rule(v) = r$. Based on the function $count$, a standard estimation procedure such as maximum likelihood estimation can be used to calculate probabilities of RHSs given a specific LHS non-terminal and obtain a probabilistic LCFRS (PLCFRS):

Definition 2.9 (Probabilistic LCFRS): A probabilistic LCFRS (PLCFRS) is a tuple $\langle N, T, V, P, S, p\rangle$ such that $\langle N, T, V, P, S\rangle$ is an LCFRS and $p : P \rightarrow [0, 1]$ such that for all $A \in N$, $\sum_{A(\boldsymbol{\alpha}) \rightarrow \Phi \in P} p(A(\boldsymbol{\alpha}) \rightarrow \Phi) = 1$.

2.3. PLCFRS parsing

PLCFRSs produced by the above extraction procedure have a number of special properties:

- They are *ordered* (Villemonte de la Clergerie, 2002), i.e. within each RHS element, the variables occur in the same order in which these variables occur on the LHS.
- They are *ε-free* (Boullier, 1998), i.e. no LHS argument is empty.
- Assuming that in the treebank, every word is represented as a leaf with no siblings whose parent node is labeled with the part-of-speech tag of the word, terminals occur only in *terminating rules* of the form $A_1(t) \rightarrow \varepsilon$ where t is a word and A is its part-of-speech tag. The LHS arguments of all other rules contain only variables.

$$\textsc{Scan:}\quad \frac{}{0 : [A, \langle\langle i, i+1\rangle\rangle]}\ A \text{ is the part-of-speech tag of } w_{i+1}$$

$$\textsc{Unary:}\quad \frac{in : [B, \vec{\rho}]}{in + \log(p) : [A, \vec{\rho}]}\ p : A(\vec{\rho}) \to B(\vec{\rho}) \in P$$

$$\textsc{Binary:}\quad \frac{in_B : [B, \vec{\rho_B}], in_C : [C, \vec{\rho_C}]}{in_B + in_C + \log(p) : [A, \vec{\rho_A}]}\ \begin{array}{l} p : A(\vec{\rho_A}) \to B(\vec{\rho_B})C(\vec{\rho_C}) \text{ an instance} \\ \text{of a rule in the grammar wrt. } w \end{array}$$

Goal items are of the form $m : [S, \langle\langle 0, |w|\rangle\rangle]$.

Figure 2.5.: Weighted CYK deduction system for parsing w with a PLCFRS $\langle N, T, V, P, S, p\rangle$

- In rules of rank 1, the LHS non-terminal and the RHS non-terminal have the same number of arguments.

Maier and Kallmeyer (2010) define and implement a CYK parsing algorithm for PLCFRS which relies upon all of these special properties for simplicity and efficiency. Additionally, the definition of the parsing algorithm assumes that the grammar is *binary*, i.e. has at most rank 2. Every LCFRS can be binarized; algorithms for doing so are discussed in Section 4.1. Finally, it is assumed that every terminal in the parser input has already been assigned a part-of-speech tag.

For a PLCFRS $\langle N, T, V, P, S, p\rangle$ and an input w, intermediate parsing results are represented as *weighted items* of the form $m : [A, \vec{\rho}]$ where

- $A \in N$,
- $\vec{\rho}$ is a vector of ranges (range vector) in w with $|\vec{\rho}| = dim(A)$ and
- m is a real number called the *weight* of the item, more specifically a logarithmically encoded probability, used to determine the relative probability of different parses according to the PLCFRS.

The meaning of an item $m : [A, \vec{\rho}]$ is that $\vec{\rho}(w) \in yield(A)$ (with associated weight m). The set of items that exist for the given grammar and the given input can be specified via the CYK-style deduction system (set of deduction rules) given in Figure 2.5. A deduction rule specifies that if all of the *antecedent items* (above the bar) are derivable and the *side condition* (right of the bar) holds, then the *consequent item* (below the bar)

add SCAN results to $\mathcal{A}$;
while $\mathcal{A} \neq \emptyset$ **do**
  remove best item $x : I$ from $\mathcal{A}$;
  add $x : I$ to $\mathcal{C}$;
  if I *goal item* **then**
    stop and output true;
  end
  else
    forall the $y : I'$ *deduced from* $x : I$ *and items in* $\mathcal{C}$ **do**
      if *there is no* z *with* $z : I' \in \mathcal{C} \cup \mathcal{A}$ **then**
        add $y : I'$ to $\mathcal{A}$;
      end
      else
        if $z : I' \in \mathcal{A}$ *for some* z **then**
          update weight of I' in $\mathcal{A}$ to $\max(y, z)$
        end
      end
    end
  end
end

Algorithm 2.1: Weighted deductive parsing

is also derivable. The idea of the deduction system is that when for a given rule range vectors for all of the RHS elements have been found, then this gives us a range vector for the LHS. The SCAN rule starts by finding items for the terminating rules, then the rules UNARY and BINARY find ranges for rules of rank 1 and 2, respectively, based on existing items. w is in the language of the PLCFRS iff a goal item can be derived using the deduction system.

While the deduction system defines the search space of all possible items for the given grammar and input, an efficient search strategy is needed to quickly find the "best" goal item – i.e. the one with the highest weight – in order to find the most probable parse. The search strategy used, an instance of weighted deductive parsing (Nederhof, 2003), is given as Algorithm 2.1. It uses two structures to store derived items, an *agenda* $\mathcal{A}$ and a *chart* $\mathcal{C}$. A newly derived item is first stored on the agenda until it has been combined with all items already on the chart to derive new items, then it is also moved to the chart. This ensures that all possible combinations of items are tried exactly once. By

always retrieving the best item on the agenda first, the algorithm ensures that the first goal item found will be the best goal item.

Every time an item is derived, the actual implementation of the algorithm also stores backpointers to the items it was derived from. When the weight of an item on the agenda is updated, i.e. a more probable way to derive the same item has been found, the backpointers are also updated. Using the backpointers, the best derivation tree for w can be constructed from the items on the chart after finding the best goal item.

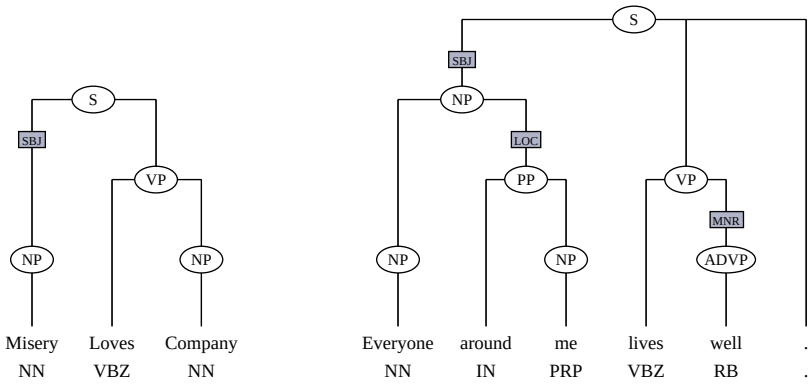
Chapter 3.

Transforming Penn Treebank trees into a discontinuous format

The phrase structure trees in the Penn Treebank are context-free. They contain additional information about a number of types of non-local dependencies. This information is represented using null elements and coindexation. Context-free parsers cannot make adequate use of coindexation, and many parsers are limited in their ability to handle null elements. Before training a probabilistic parser, indices and null elements are therefore usually removed, losing the information about non-local dependencies completely.

This chapter presents a method for transforming the trees in such a way that certain types of non-local dependencies are encoded as part of the tree structure. This is possible when discontinuous constituents are allowed. When a probabilistic parser that is able to deal with discontinuous constituents is then trained on the transformed trees, as presented in the previous chapter, it will also learn information about non-local dependencies.

The general considerations which have guided the development of the transformation method are explained in Section 3.1. Section 3.2 presents the basic transformation algorithm. Section 3.3 presents the different types of non-local dependency annotation in the WSJ treebank that the algorithm can be applied to. It also motivates some enhancements to the algorithm which are then summarized in Section 3.4. Section 3.5 discusses those types of non-local dependency annotation which are not included in the transformed treebank. Section 3.6 gives a statistical account of characteristics of the transformed treebank, e.g. frequency of discontinuity and of certain configurations.



(a) Subject and direct object of verb

(b) PP modifier of noun

Figure 3.1.: Examples of dependents in their interpretation locations

3.1. Guiding principles

There may be many sensible ways to incorporate non-local dependency information into (discontinuous) phrase structure trees. The method presented here was inspired by the treebank annotation scheme laid out in Skut et al. (1997) as well as its implementations in the German treebanks NEGRA (Skut et al., 1998) and TIGER (Brants et al., 2002), particularly by the principle to aim for a "uniform representation of local and non-local dependencies" (Skut et al., 1997). The local and non-local dependencies it is concerned with are primarily those dependencies that make up *argument structure*, meaning dependencies between verb, noun, adjective, adverb and preposition *heads* and their *arguments/adjuncts* such as subjects, objects, or modifiers. In the following, I use the term *dependent* to mean *argument or adjunct*, not to be confused with the more general use of the term in Dependency Grammar.

Bies et al. (1995, Section 1.1) give specific rules governing where and how dependents are attached in the Penn Treebank with respect to their heads. For example, subjects are attached at the clause level S node while most other dependents of the predicate are attached to the same VP node as the predicate. This can be seen in Figure 3.1(a). To give another example, the attachment site for PP dependents of nouns is within an extra NP into which the NP containing the noun is embedded (this is known as Chomsky-adjunction, see Figure 3.1(b)). I will call the attachment site of a dependent

according to these rules its *interpretation location* because from the perspective of an application processing phrase structure trees, it identifies the head with respect to which the dependent is to be interpreted.

Where word-order-related phenomena like *wh*-movement, topicalization or extraposition occur, it is often impossible to attach a dependent at its interpretation location in a context-free tree because this would create discontinuous constituents. It is instead attached at another location in the tree that preserves both word order and context-freeness. Such cases constitute exceptions to the attachment rules described above and are typical examples of non-local dependencies. The basic idea behind the method presented here is to eliminate these exceptions by allowing discontinuous constituents and reattaching the dependents to their interpretation locations.

The method presented is designed to be simple and automatic. It consists in a tree transformation algorithm that uses only non-local dependency annotation already present in the Penn Treebank, in the form of null elements and coindexation. No manual reannotation has been performed, except for correcting some annotation errors and inconsistencies before carrying out the transformation. The corrections are listed in Appendix A.

3.2. The basic algorithm

The format used for annotating non-local dependencies in the Penn Treebank is exemplified in the first tree in Figure 3.2. The general case is that some dependent c (in this case WHNP) is not in its interpretation location (in this case the interpretation location is in the VP because *what* is the object of the verb *do*); word order makes it impossible to attach it there in a context-free tree because other words (*should*, *I*) intervene. Instead, c is attached in a phrase higher up in the tree, and a placeholder node p (NP) is attached at the interpretation location. p has one child, a null element (*T*) that is coindexed with c, i.e. the null element and c bear the same numerical index in their labels. Coindexations represent directed secondary edges from the null element to the coindexed internal node, and as such they are shown in the diagrams.

To incorporate the information about non-local dependencies into the tree structure, the tree transformation algorithm given as Algorithm 3.1 can be used. It replaces *T*, *ICH* and *EXP* null elements and their placeholder parent nodes with the coindexed

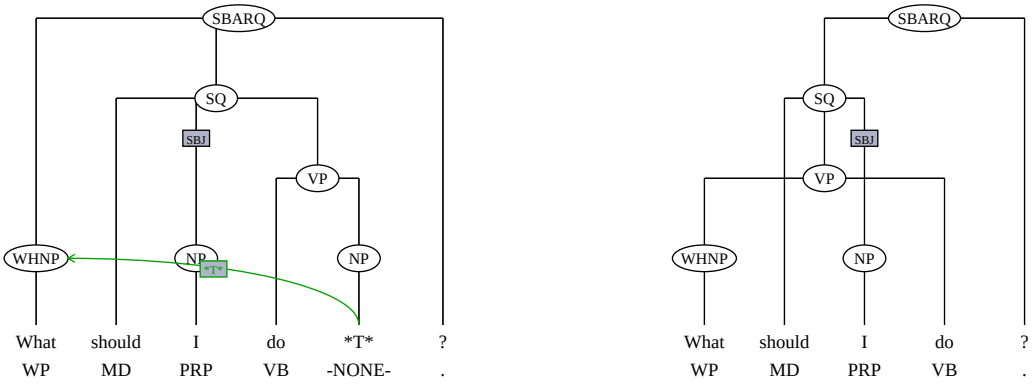


Figure 3.2.: A *wh*-question and its transformed version

constituents, removing these from their previous attachment sites. (This choice of null element types to process is justified and extended in the following sections.) All other null elements are also removed, as well as resulting empty constituents and remaining indices on node labels, as is common practice for preprocessing the Penn Treebank for training a parser.

foreach *sentence s* **do**
  foreach *node c in s coindexed with a *T*, *ICH* or *EXP* null element n* **do**
    select the grandparent of n as the destination node d;
    remove c from its current parent and reattach it to d;
  end
end
delete all null elements;
delete non-terminal nodes without children until none exist;
remove all remaining indices from node labels;

Algorithm 3.1: The basic transformation algorithm

The result of the algorithm for the example is given as the second tree in Figure 3.2. The WHNP is now in its interpretation location within the VP. Note that the VP has become a discontinuous constituent, i.e. it covers non-adjacent spans of terminals.

Note also that the original attachment of the WHNP at a higher node than its interpretation location is not directly motivated by the intervening material, but rather by the theoretical generalization that in *wh*-questions, movement of *wh*-phrases to the front of the sentence always takes place. The original annotation reflects this generalization

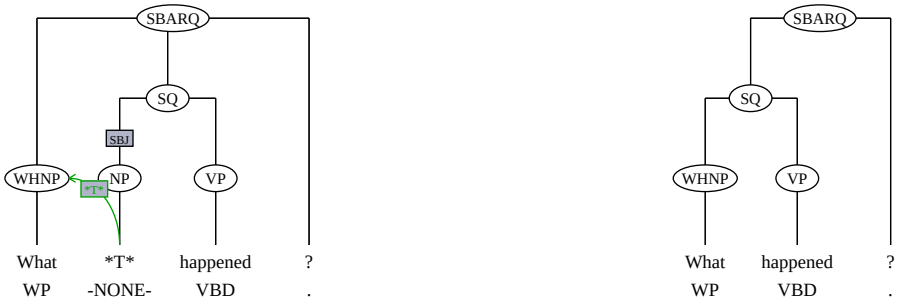


Figure 3.3.: A sentence where the transformation does not create a discontinuous constituent

even when there is no intervening material, as can be seen in Figure 3.3. In such cases, the application of the algorithm simplifies the tree but does not create a discontinuous constituent.

3.3. Types of non-local dependencies included in the transformed treebank

This section looks in detail at the types of grammatical phenomena that are annotated using *T*, *ICH* and *EXP* null elements in the WSJ treebank. It justifies why it makes sense to apply the algorithm to them and discusses two problematic cases which motivate an extension of the algorithm.

3.3.1. wh-questions, relative clauses and tough movement

wh-questions and relative clauses are similar in English: both start with what I will call a *wh*-element that is to be interpreted as a dependent of some phrase within the sentence, i.e. which can be seen has having moved out of its interpretation location. A *wh*-element can be an interrogative word or a relative pronoun or a phrase containing one. In the WSJ treebank, a *wh*-element is represented as a phrase of category WHNP, WHADVP, WHPP or WHADJP that is adjoined to the clause-level node using an SBARQ resp. SBAR parent node.

This location can be seen as reflecting only word order. There is no head with respect to which the SBAR/SBARQ phrase would provide an interpretation location – rather, the *wh*-element is still dependent on the inner predicate, not only as a semantic argument, but also morphosyntactically. This is shown by the fact that the pronoun *who* can take the form *whom* if it fills an object slot of the inner predicate. In fact, the interpretation location of the *wh*-element with respect to the inner predicate is marked by a corresponding placeholder phrase containing a null element of type *T*, as already seen in the example given in the previous section. Thus, it makes sense to apply the algorithm, remove the *wh*-element from the SBAR/SBARQ phrase and attach it at the interpretation location instead. Examples for *wh*-questions have already been given in Figure 3.2 and Figure 3.3; a relative clause and the result of the transformation can be seen in Figure 3.4.

In infinitival (Figure 3.5) and some object relative (Figure 3.6) clauses, there is no overt *wh*-element, but the annotation assumes a *null complementizer*, represented by a null element of type 0 embedded in a WHNP. This WHNP is coindexed with a *T* null element in the relative clause. The same is true for the *tough movement* construction (Figure 3.7). The algorithm applies to such cases as well, reattaching the empty WHNP to the VP, replacing the empty NP. However, this has no effect in the end since all remaining empty elements are deleted. In the transformed trees, no trace of the null complementizers remains.

3.3.2. Fronted and circumpositioned dependents

In the WSJ treebank, fronted dependents are attached directly to the clause-level phrase and not to an additional SBAR/SBARQ phrase. However, just like *wh*-elements, their interpretation location (usually in the VP) is marked using a coindexed *T* null element, provided that the fronted dependent is an *argument*. The annotation guidelines are not completely clear on what notion of "argument" is meant (cf. Bies et al., 1995, p. 65 ff.), but inspection of the treebank suggests it is the notion of "complement" used ibid., section 1.1.4. This includes NP objects, clauses, quotations, the passive logical-subject *by*-phrase, VPs and constituents with the function tag -BNF, -CLR, -DTV, -PRD or -PUT (cf. Bies et al., 1995, Section 2.2). An example of a fronted argument, along with the result of the transformation, is given in Figure 3.8.

Other fronted dependents ("adjuncts", ibid., p. 67) like the ADVP in Figure 3.9 are treated differently in that their interpretation locations (in the VP) are normally not

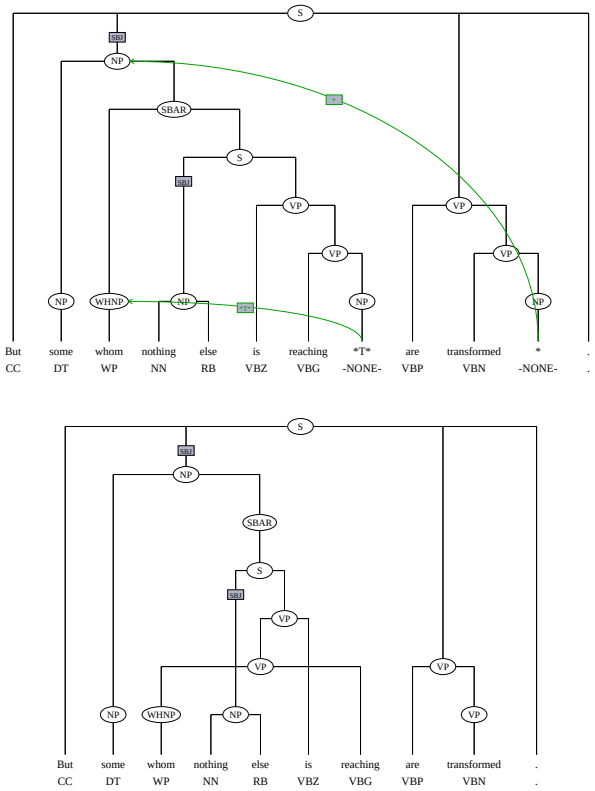


Figure 3.4.: Sentence with a relative clause

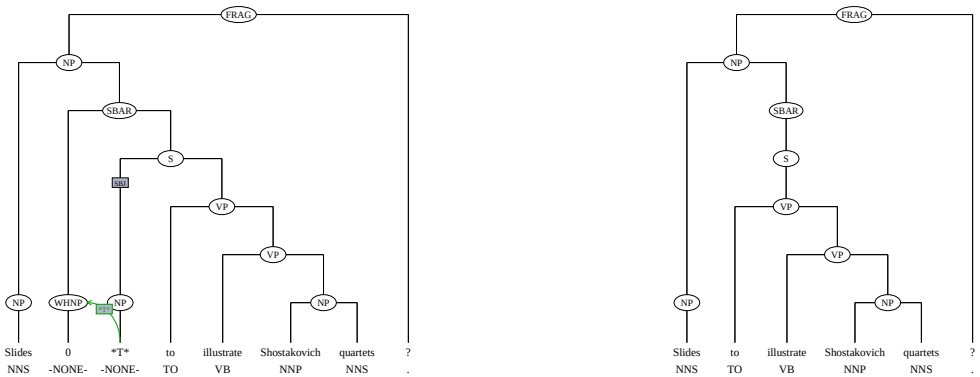


Figure 3.5.: Null complementizer in an infinitival relative clause

Figure 3.6.: Null complementizer in an object relative clause

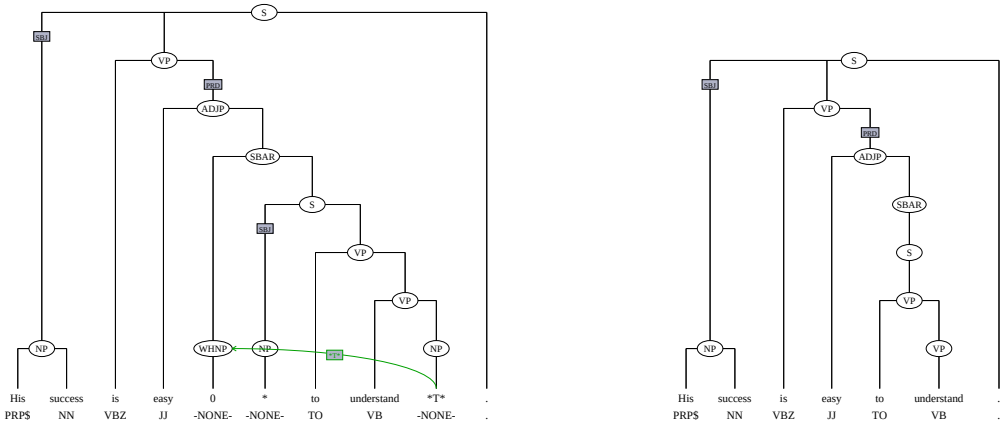


Figure 3.7.: Tough movement

Figure 3.8.: Sentence with a fronted dependent

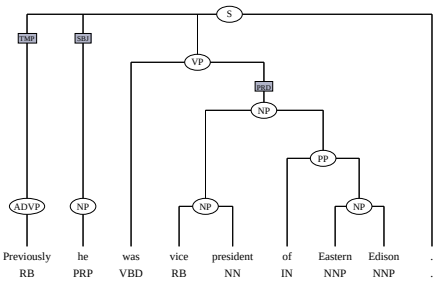


Figure 3.9.: Sentence with a fronted non-complement dependent, original and transformed version

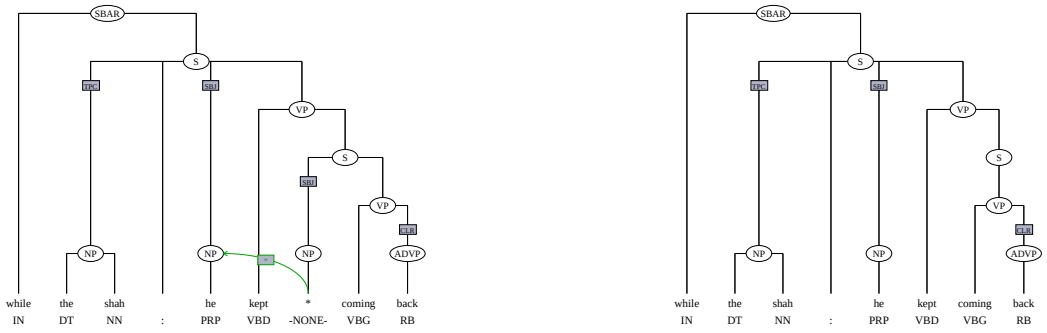


Figure 3.10.: Left dislocation: a resumptive pronoun appears in the interpretation location of the fronted argument. There is no *T* here for the algorithm to apply to.

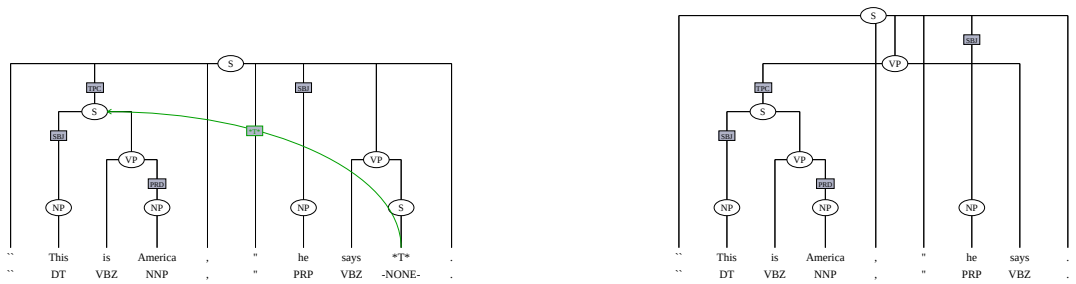


Figure 3.11.: Sentence with a fronted quotation

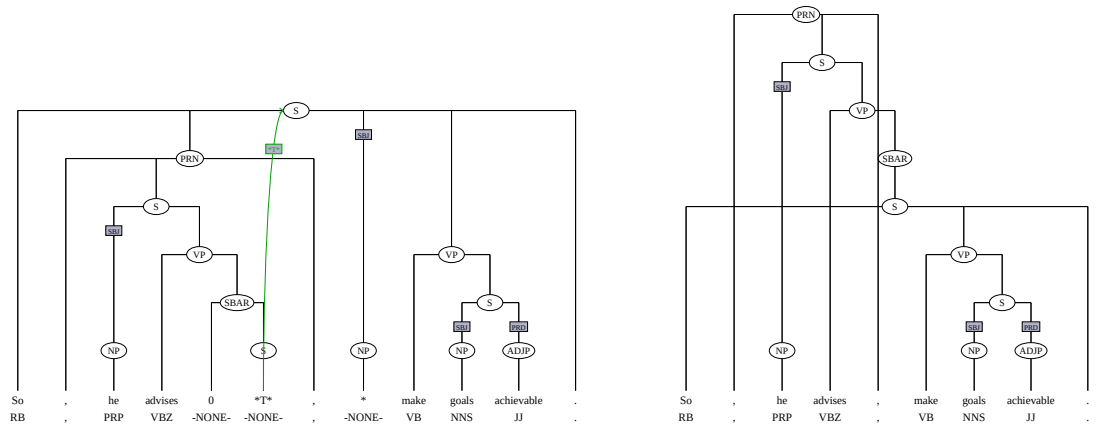


Figure 3.12.: Sentence consisting of a discontinuous quotation with interrupting matrix sentence

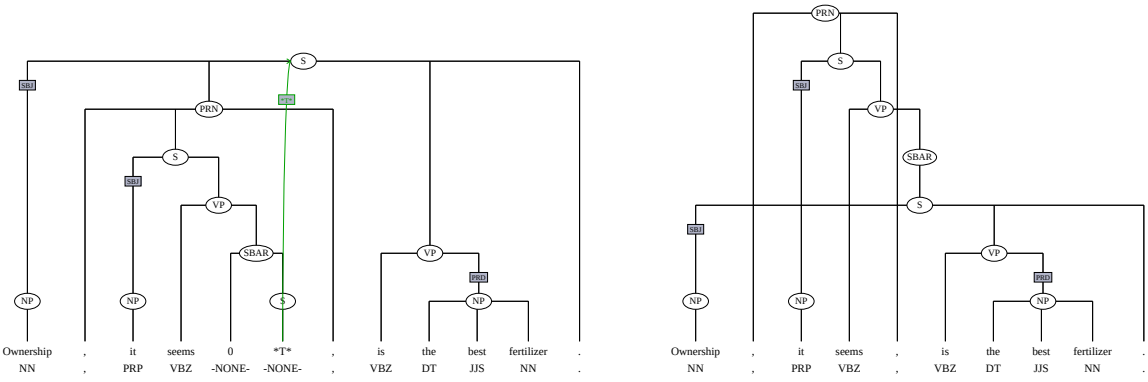


Figure 3.13.: Sentence consisting of a discontinuous sentential argument with interrupting matrix sentence

Figure 3.14.: A modifier to the subject appearing after the verb, causing a discontinuous NP

marked, thus the transformation does not apply to them. Not marked and thus not considered in the transformation either are fronted arguments in whose interpretation location a resumptive pronoun appears (*left dislocation*), as in Figure 3.10.

Quotations are treated as arguments of the verb of saying (ibid., p. 31), whether enclosed in quotation marks or not. They are often fronted, in which case the algorithm applies in the usual way (Figure 3.11), or circumpositioned, with one portion of the quotation appearing before and one after the verb (Figure 3.12). Like fronting, circumpositioning also occurs with arguments other than quotations, albeit rarely (Figure 3.13). The circumpositioned argument, usually a sentence, is grouped under one root node, and the interruptive material which contains the matrix VP is embedded into it using a PRN (*parenthetical*) node. The interpretation location of the circumpositioned argument within the matrix VP is marked with a coindexed *T* null element as usual.

Applying the basic algorithm in such cases would destroy the tree structure: the PRN node dominates the matrix VP node and is itself dominated by the root of the circumpositioned constituent; introducing an edge from the VP node to the root would create a cycle. Since a PRN node does not have a specific interpretation location, one solution is to remove the parenthetical from within the circumpositioned constituent and attach it as a sibling of the circumpositioned constituent instead (if the root of the circumpositioned constituent is not the root of the whole tree) before carrying out the usual reattachment, which attaches the root of the circumpositioned constituent within the PRN constituent – turning the tree structure inside out, so to speak. The results for the examples are shown on the right sides of Figures 3.12 and 3.13. The precise enhanced algorithm is given below in Section 3.4, together with one more refinement.

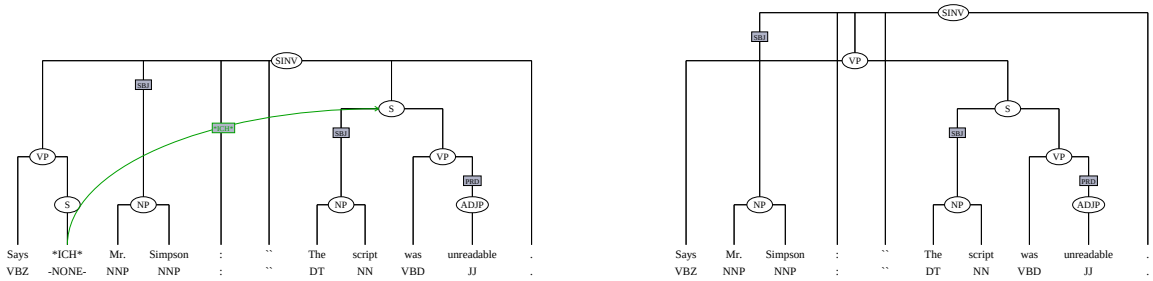


Figure 3.15.: Subject-verb inversion causing a discontinuous VP

3.3.3. Discontinuous dependency

The *ICH* null element type is very similar to *T*: it marks the interpretation location of constituents that are not attached there. While *T* is mainly used for clause-initial and circumpositioned constituents, *ICH* is used almost exclusively for constituents that appear to the right of their interpretation location. Typical uses include modifiers such as the PP in Figure 3.14 which are to be interpreted as modifiers to a distant noun rather than at their attachment site in the context-free tree, and cases of subject-verb inversion as in Figure 3.15 where the subject appears between the verb and some of its non-subject dependents, making it impossible to construct a continuous VP following the usual attachment rules. In short, *ICH* is used "to indicate a relationship of constituency between elements separated by intervening material" (Bies et al., 1995, p. 107), and applying the transformation algorithm realizes this relationship by grouping the separated elements under the same node.

3.3.4. it-extraposition

Clausal arguments can be extraposed in English while leaving an expletive *it* in the interpretation location. In the case of clausal subjects, this is annotated in the Penn Treebank by Chomksy-adjoining to the expletive *it* an *EXP* null element which is coindexed with the extraposed subject, as in Figure 3.16. Applying the algorithm to this configuration creates a constituent that groups the expletive *it* and the extraposed subject together. Even though this constituent may seem somewhat artificial, it has the advantage of removing the extraposed subject from a location where it is attached only for reasons of word order and attaching it to its interpretation location, if indirectly

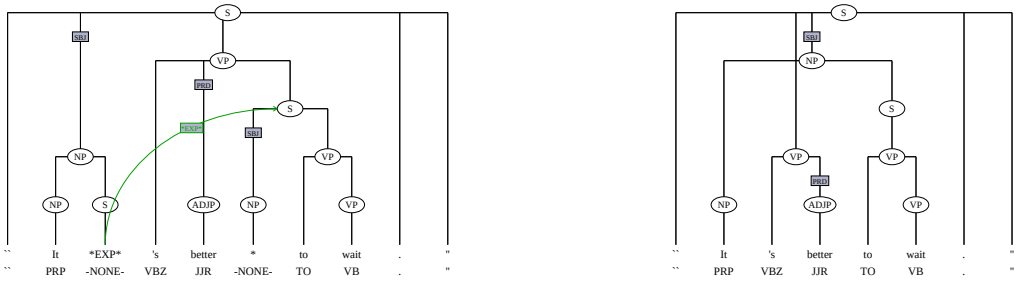


Figure 3.16.: Extraposition with expletive *it*

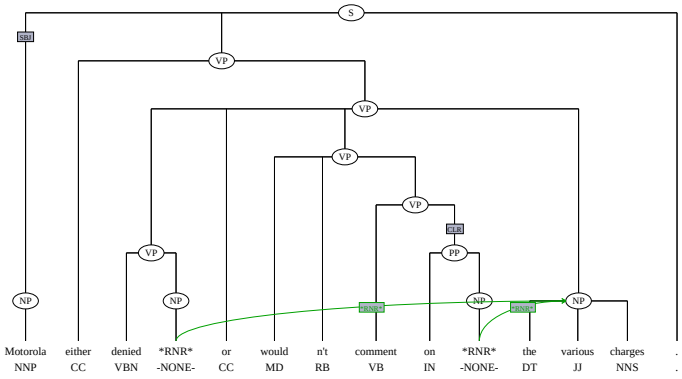


Figure 3.17.: Right node raising

via an additional NP node. It is also consistent with the way the German treebanks NEGRA and TIGER treat the analogous expletive *es*.

3.3.5. Dealing with shared dependents

As the examples discussed so far show, discontinuous constituents make it possible to attach dependents in a more consistent way than with a system of coindexing on top of context-free trees. What discontinuous constituents do not offer, unfortunately, is a natural way to represent situations in which a dependent is shared between more than one head and thus has two or more interpretation locations. This is the case in coordination when a non-subject dependent must be interpreted in both coordinated elements, but is realized only once.

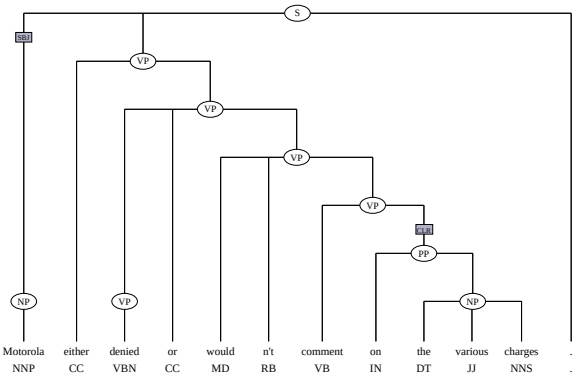


Figure 3.18.: Transformed version of the tree in Figure 3.17, attaching the shared dependent at the closest interpretation location

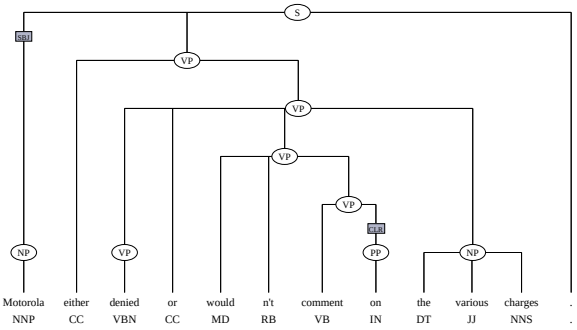


Figure 3.19.: Transformed version of the tree in Figure 3.17, attaching the shared dependent at the coordination level

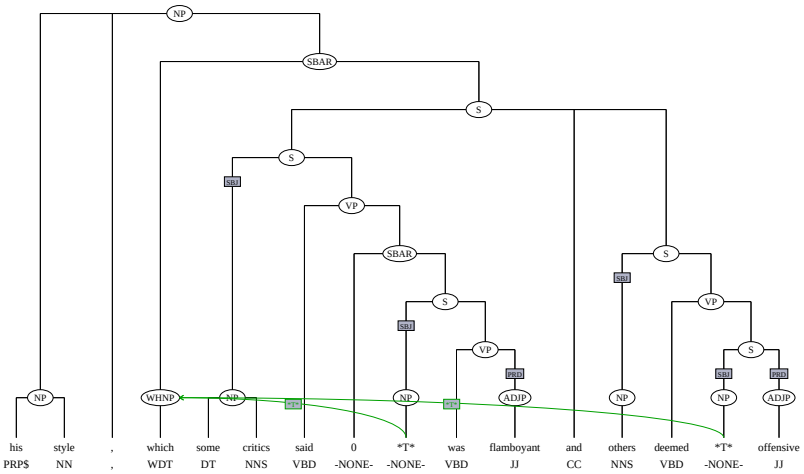


Figure 3.20.: A relative pronoun shared between two coordinated phrases

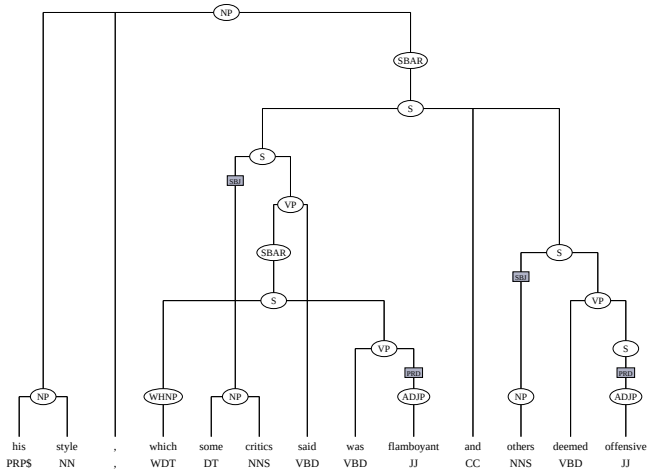


Figure 3.21.: Transformed version of the tree in Figure 3.20, attaching the shared dependent at the closest interpretation location

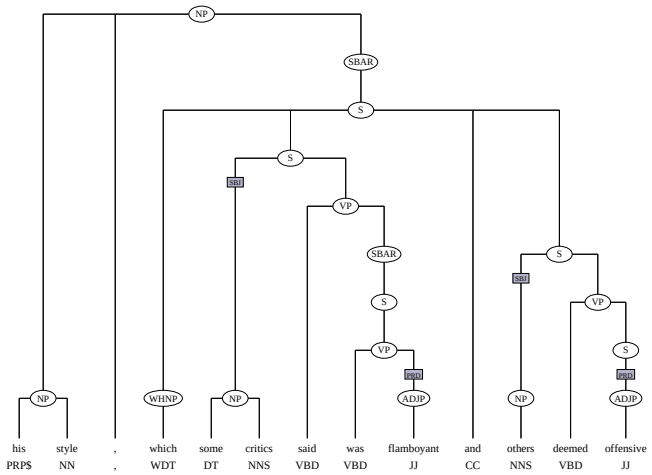


Figure 3.22.: Transformed version of the tree in Figure 3.20, attaching the shared dependent at the lowest common ancestor of all interpretation locations

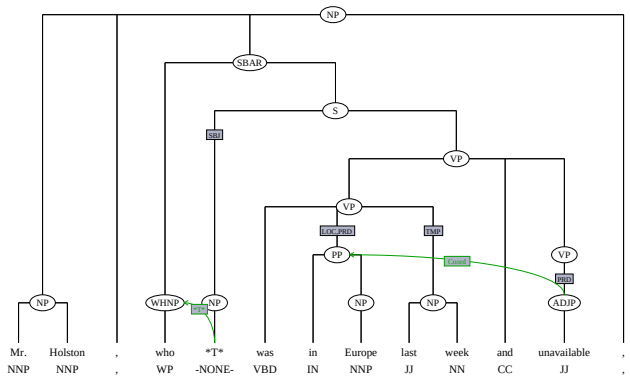


Figure 3.23.: The NP *last week* may be seen as a shared dependent that is neither right-node raised nor *wh*-moved or fronted.

When this dependent appears at the right periphery of the coordination, the Penn Treebank annotation attaches it at the coordination level, and placeholder constituents with *RNR* (*right node raising*) null elements mark the interpretation locations, as shown in Figure 3.17. When the shared dependent is *wh*-moved or fronted, it attaches at the usual non-interpretation-location, and *T* rather than *RNR* null elements are used to mark the interpretation locations, as shown in Figure 3.20. In the rare cases where the shared dependent is neither *wh*-moved nor fronted nor right-node raised, it is attached in that coordinated constituent where it is realized, and no explicit annotation marks its relation to the other coordinated constituent. The NP *last week* in Figure 3.23 may be seen as such a case.

In a tree, every node can have only one parent, so it is not possible to attach the shared dependent at all of its plausible interpretation locations. In the case of right node raising constructions, it is the rightmost interpretation location that suggests itself, for two reasons: first, the phonological surface form of right-node-raising constructions indeed allows to hear the rightmost coordinated constituent as containing the shared dependent. Attaching it there thus reflects a linguistically plausible analysis where the shared dependent is realized in the last coordinated constituent and elided in the others. Where the shared dependent is *wh*-moved or fronted, it can analogously be attached at the interpretation location in the leftmost coordinated constituent, treating it as elided in the other coordinated constituents. This strategy amounts to always attaching a shared dependent at the closest interpretation location, and this strategy has been chosen for the transformation. The resulting trees are shown in Figure 3.18

and Figure 3.21, respectively. In cases like that of Figure 3.23, no transformation is required to be consistent with this strategy, as shared dependents are already attached within the coordinated constituent where they are realized.

A possible alternative strategy would be to always attach shared dependents at coordination level. This would require no action in the case of right node raising (except for removing the *RNR* null elements), see e.g. Figure 3.19. In the case of *wh*-moved and fronted elements, it would be undesirable to leave the shared and moved dependent in its "artificial" SBAR location since this would be inconsistent with the new annotation of non-shared dependents, where the algorithm abolishes attachment of moved dependents to SBAR altogether. A "coordination level" attachment site for the shared dependent can be found by choosing the lowest common ancestor of all interpretation locations. The result of such a reattachment is shown in Figure 3.22. However, this alternative strategy was not explored further.

3.4. The enhanced algorithm

The transformation algorithm with the enhancements for dealing with parentheticals (Section 3.3.2) and shared dependents (Section 3.3.5) is given as algorithm 3.2. The body of the main loop consists of three blocks: selecting a destination location d for the constituent c that is to be moved, preventing cycles by detaching or reattaching parenthesis (PRN) nodes, and actually reattaching c to its destination d.

3.5. Types of non-local dependencies which are not included

The types of non-local dependencies considered so far, annotated using the null elements *T*, *ICH*, *EXP* and *RNR*, can all be interpreted as a dependent appearing in a position removed from its interpretation location. Through allowing discontinuous constituents and reattaching these dependents, it has been more or less unproblematic to incorporate these dependencies into the tree structure. There are other types of non-local dependencies annotated in the Penn Treebank where this recipe is not applicable, and which are therefore not included in the transformed treebank. This section briefly

foreach *sentence* s **do**
  foreach *node* c *in* s *coindexed with *T*, *ICH*, *EXP* or *RNR* null elements* $n_1, \ldots, n_k$ **do**
    $d \leftarrow$ grandparent of that null element among $n_1, \ldots, n_k$ which is closest to c;
    if c *properly dominates* d **then**
      find the node labeled PRN on the path between c and d, call it r (if not present, abort and continue with next c);
      remove r from its current parent;
      if c *is not the root of* s **then**
        reattach r to the parent of c;
      end
    end
    remove c from its current parent and reattach it to d;
  end
end
delete all null elements;
delete non-terminal nodes without children until none exist;
remove all remaining indices from node labels;

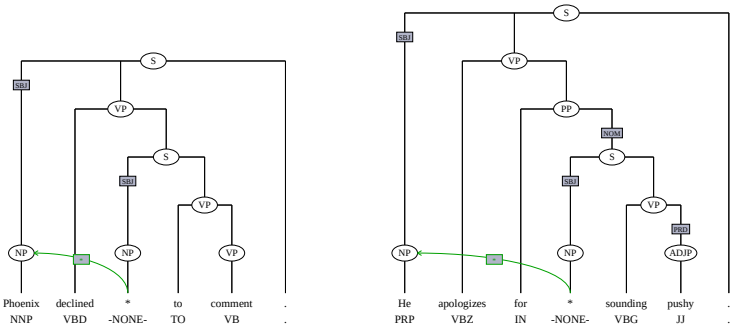
Algorithm 3.2: The enhanced transformation algorithm

presents the remaining types of null elements used in the Penn Treebank annotation, and explains why they are discarded in the transformation.

3.5.1. PRO and passive trace

In subordinate clauses without an overt subject, the Penn Treebank annotation marks the empty subject position with a placeholder NP containing a * null element, corresponding to the phonologically empty PRO subject postulated in Government and Binding theory. It is often coindexed with another NP in the same sentence. The coindexation may express the control relation in subject control, object control and raising constructions, but it is used more generally to connect the * element to an NP "whenever there is an appropriate referent elsewhere in the same sentence" (Bies et al., 1995, p. 97).

These referents could be reattached to replace the * subject in the subordinate clause, reflecting the fact that they fill a semantic role of the subordinate verb. But unlike



(a) Controlled subject in infinitival clause

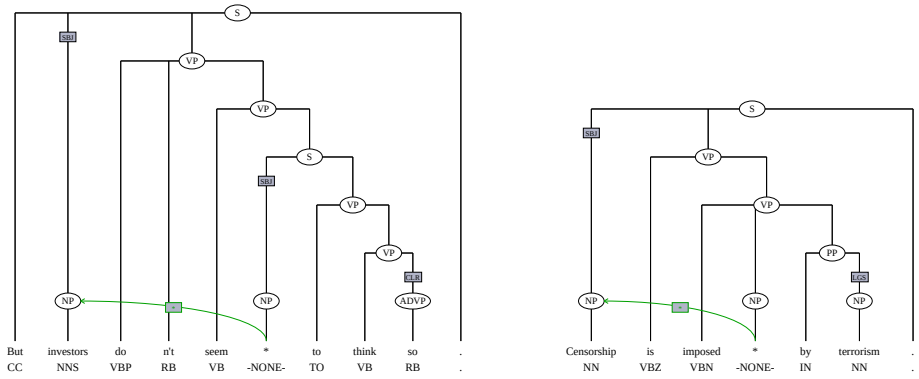
(b) Controlled subject in gerund clause

Figure 3.24.: Constructions where a "raised" dependent fills a semantic role of both the matrix verb and the subordinate verb

constituents coindexed with *T* null elements, these referents generally depend syntactically on a matrix verb. This can be tested using pronoun case: *They persuaded him to leave* is grammatical while **They persuaded he to leave* is not – the pronoun is assigned case by the object position of the matrix verb *expect* rather than the subject position of the subordinate verb *leave*. This syntactic dependency is a strong argument against reattaching referents of * null elements, independently of whether the referent additionally fills a semantic role of the matrix verb, as in the examples in Figure 3.24, or not, as in raising constructions (Figure 3.25(a)). The same is true for the surface subject in passive constructions as in Figure 3.25(b), which is co-indexed with a * null element in the position within the subordinate VP, marking the position where the surface subject would be realized as the object in a corresponding active clause.

3.5.2. Permanent predictable ambiguity

PPA null elements are used to indicate attachment ambiguities (*permanent predictable ambiguity*) that annotators could not resolve using context. As Bies et al. (1995, p. 102) write, the default is "to attach the constituent at the more likely site (or if that is impossible to determine, at the higher site) and then to pseudo-attach it at all other plausible sites." An example is given in Figure 3.26, where it is unclear if the *from*-phrase modifies *finagled* or *loan*. As in the case of shared dependents (Section 3.3.5), it is impossible to attach the constituent at more than one location in a tree, even though



(a) Raising (b) Passive

Figure 3.25.: Constructions where a "raised" dependent fills a semantic role only of the subordinate verb, but still syntactically depends on the matrix verb

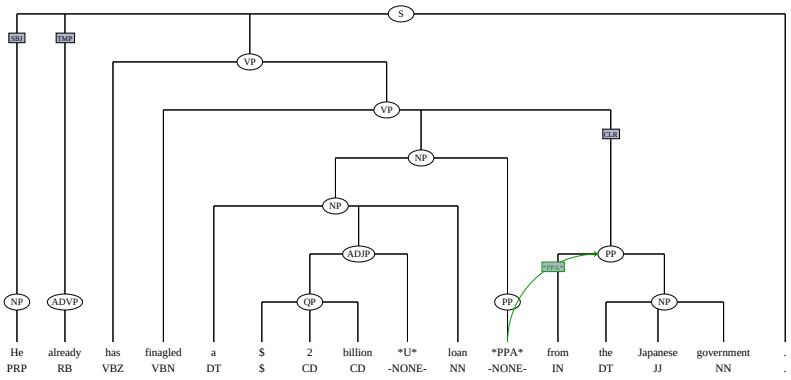


Figure 3.26.: Permanent predictable ambiguity

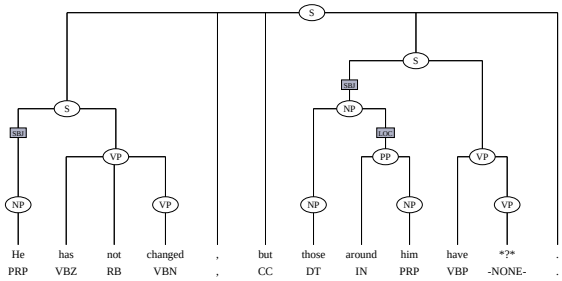


Figure 3.27.: Ellipsis

it would make sense at more than one. However, in this case, a default attachment choice is already made by the annotators. The algorithm sticks with this choice and discards *PPA* elements.

3.5.3. Ellipses

The location of elided phrases is marked with a placeholder phrase containing a *?* null element (cf. Bies et al., 1995, p. 81). A phrase understood in such a location is not always identical to another phrase in the same sentence, and even if it is, as is the case with the lower VPs in Figure 3.27, it would be unwise to reattach that phrase at the location of the ellipsis because it must also be interpreted where it is realized. *?* null elements are therefore discarded.

3.5.4. Null elements not indicating non-local dependencies

The remaining three types of null elements that are used in the Penn Treebank are not used to indicate dependencies, thus they are just removed by the algorithm:

0 represents the null complementizer (cf. Section 3.3.1).

U serves as the head noun of NPs denoting monetary amounts or percentages, as in `(NP $ 27 *U*)` or `(NP (QP between $ 5 and $ 15) *U*)`, as a placeholder for the unit symbol which is not in the normal location for a head noun. In cases like the latter example, discontinuous constituents would allow to attach the dollar symbols directly

			trees with gap-degree		
type	instances	trees	0	1	2
T	18759	15452	7292	7924	236
T-PRN	843	843	0	71	772
ICH	1268	1240	7	1200	33
EXP	658	651	1	630	20
RNR	210	208	131	67	10
any reattachment	21738	17187	7397	8996	794
no reattachment	n/a	32021	32021	0	0
total	n/a	49208	39418	8996	794

Table 3.1.: Reattachment types and gap-degrees of resulting trees

within the NP instead of the *U*. However, it seems unlikely that this would offer an advantage to parsing or to semantic interpretation.

The "anti-placeholder" *NOT* is a very rare element used only in *template gapping* (Bies et al., 1995, Section 7.4.6). It appears in a coordinated element and is coindexed with a constituent within another coordinated element, indicating that the latter should *not* be interpreted in the former, thus forbidding the introduction of an edge through reattachment.

3.6. Characteristics of the transformed treebank

Having transformed the WSJ treebank to a discontinuous format (let us call the result WSJ-D with a D for "discontinuous"), questions arise about the frequency of individual types of non-local dependencies as well as about the degree of discontinuity in the transformed trees. The latter is important for parsing complexity. As a measure for the discontinuity of trees, Maier and Lichte (2009) define the *gap-degree* of a node in a tree as the number of gaps in the yield of the node. For example, in the transformed tree in Figure 3.14, the higher NP node dominates five terminal nodes in two contiguous spans.

		gap-degree				ill-degree		
treebank	trees	0	1	2	3	0	1	2
NEGRA	20597	72.44%	24.23%	3.30%	0.04%	98.75%	1.25%	0.000%
TIGER	40013	71.01%	25.77%	3.18%	0.04%	98.90%	1.10%	0.000%
WSJ-D	49208	80.10%	18.28%	1.61%	0.00%	99.75%	0.25%	0.002%

Table 3.2.: Comparison of the characteristics of the transformed WSJ treebank with the data of Maier and Lichte (2009) for two discontinuous German treebanks

That is, the sequence of dominated terminal nodes is interrupted by one gap, containing the words *fell* and *marginally* which are not dominated by the NP node. Therefore, the gap-degree of this node is 1. The gap-degree of all other nodes in this tree is 0, including the top S node – the terminal nodes it dominates are all contiguous. The gap-degree of a tree is defined as the maximum of the gap-degrees of all of its nodes.

The upper part of Table 3.1 gives an overview of the reattachment operations made in the transformation, broken down by the type of the null element involved (*T*, *ICH*, *EXP* or *RNR*). Reattachments of discontinuous sentential arguments that necessitated reattaching a PRN node as described in Section 3.3.2 are listed separately as **T*-PRN*. The second column shows the total number of times each type of reattachment was made, summing to a total of 21378 operations. One type of reattachment may occur more than once in the same tree, so the third column contains the size of the set of trees affected by one or more instances of each reattachment type. The last three columns show the distribution of gap-degrees within each such set of trees.

More than one type of reattachment may occur in the same tree, so the gap-degrees in each row cannot directly be attributed to the corresponding types of reattachment. However, the figures can give a rough idea of each type's tendency to introduce gaps. **T**, with the vast majority of instances, results in discontinuous trees (i.e. trees with gap-degree > 0) in only about half of the cases, plausibly explained by the fact that *wh*-movement is frequently from the subject position, as shown in Figure 3.3. By contrast, *ICH* and *EXP* null elements are almost only used when in fact necessitated by intervening material, resulting in very few trees that remain context-free in the transformation. The opposite is true for *RNR* elements: the strategy to attach "right-node-

raised" elements at their rightmost interpretation location (cf. Section 3.3.5) precludes the introduction of additional discontinuity except in a few cases where punctuation intervenes between the rightmost interpretation location and the right-node raised element. Finally, the category **T*-PRN* is very prone to discontinuity – unsurprisingly, as in these cases the yield of a parenthetical is removed from the yield of a discontinuous sentential argument, leading to a gap-degree of at least 1 in the root node of the sentential argument. Additionally, punctuation surrounding the parenthetical is often attached at the PRN node and thus not dominated by the parenthetical S node, which then even has 2 gaps.

Looking at the lower part of the table, one notes that roughly a third of all trees have been affected by at least one reattachment while the rest of the treebank is unchanged except for removed null elements and coindexation. Of the trees affected by reattachment, about 60% have become discontinuous, that is roughly a fifth of all trees. Most of these have gap-degree 1, only some have gap-degree 2, no higher gap-degrees occur. Table 3.2 compare these figures to those found by Maier and Lichte (2009) for the German treebanks NEGRA (Skut et al., 1998) and TIGER (Brants et al., 2002). It also shows the number of *ill-nested* (cf. Maier and Lichte, 2009) trees in the three treebanks.

The annotation schemes of NEGRA and TIGER are very similar; they do not use coindexation or null elements and they allow for discontinuous constituents in order to encode head-dependent relations consistently, similar to WSJ-D. There are relatively more discontinuous trees in the two German treebanks than in WSJ-D. This seems to confirm the intuition that German is "more discontinuous", i.e. "more context-sensitive" than English due to its freer word order. The difference in number would be even higher if the WSJ(-D) treebank and NEGRA/TIGER used more similar annotation schemes: NEGRA and TIGER attach an inflected verb and all of its dependents directly at the clause level with no intervening VP node, thus avoiding discontinuity when a dependent of the inflected verb is *wh*-moved or fronted. It can thus be said that annotating non-local head-dependent relations in English using discontinuous constituents introduces only a relatively limited degree of discontinuity into the trees.

Chapter 4.

Parameters for grammar annotation and parsing

Extracting a "vanilla" probabilistic grammar from a treebank and using a parsing algorithm to find the most probable parse according to this grammar is only the basic technique in data-driven parsing and leaves much room for optimization in terms of both accuracy and speed. In the case of PCFG, several techniques have emerged and successfully been employed over the years. They may take the shape of modifications to the extracted grammar (grammar annotation) or to the parsing algorithm. Some of these techniques have already been adapted to PLCFRS parsing. In this chapter, I describe the techniques of binarization (with Markovization), category splitting, and briefly also outside estimates and cutoff. In the concluding section, I present a series of parsing experiments applying the techniques to a subset of the data produced in Chapter 3 in order to find a good combination of parameters for larger parsing experiments on the full transformed WSJ treebank.

4.1. Binarization

The parsing algorithm described in Section 2.3 assumes that the grammar is (at most) of rank 2, i.e. that the right-hand side of each rule has at most two elements. This allows for efficient parsing with only passive items, which are simpler than active items and therefore can be stored and retrieved more quickly. Gómez-Rodríguez et al. (2009) show that it is possible to transform any LCFRS into a strongly equivalent LCFRS of rank 2 and give algorithms for doing so. The process is called *binarizing* the grammar and

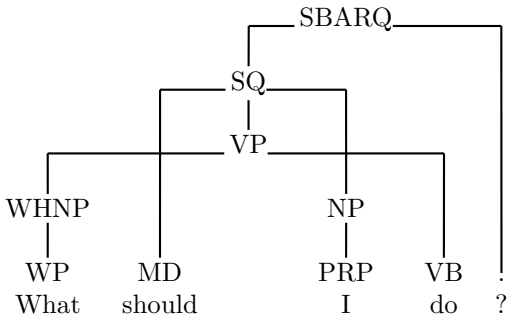


Figure 4.1.: Example of a non-binary tree

$$
\begin{aligned}
\text{SBARQ}(XY) &\rightarrow \text{SQ}(X).(Y) \\
\text{SQ}(XYZU) &\rightarrow \text{VP}(X,U)\text{MD}(Y)\text{NP}(Z) \\
\text{VP}(X,Y) &\rightarrow \text{WHNP}(X)\text{VB}(Y) \\
\text{WHNP}(X) &\rightarrow \text{WP}(X) \\
\text{NP}(X) &\rightarrow \text{PRP}(X)
\end{aligned}
$$

Figure 4.2.: Rules extracted from the tree in Figure 4.1

similar to the transformation of a context-free grammar into Chomsky Normal Form. "Strongly equivalent" here means that the derivations of the original grammar for any input w can be reconstructed from the derivations of the binarized grammar for w (the parse trees can be *debinarized*), using some simple homomorphism. Note that it is *not* generally possible to transform a k-LCFRS into an equivalent k-LCFRS of rank 2; the fan-out may increase in binarization. This issue is revisited in Section 4.1.2.

The basic binarization algorithm used in Maier and Kallmeyer (2010) is here given as Algorithm 4.1, extended to update the *count* function (cf. Section 2.2). It converts the rules in the grammar with more than two RHS elements into a number of binary rules. For each of these, the same count is recorded as for the original rule. The algorithm proceeds by going through the RHS of the original rule, in each iteration "splitting of" the leftmost RHS element and grouping the remaining RHS elements under a RHS element with a new predicate. The arguments $\boldsymbol{\gamma}$ of the new RHS element are the arguments $\boldsymbol{\alpha}$ of the LHS where terminals and the variables occurring in the arguments $\boldsymbol{\chi}$ of the split-off element have been removed, and new argument boundaries have been introduced such that non-adjacent variables in $\boldsymbol{\alpha}$ are also non-adjacent in $\boldsymbol{\gamma}$. For example, if $\boldsymbol{\alpha} = XYtZ, U$ and $\boldsymbol{\chi} = Y, U$, then $\boldsymbol{\gamma} = X, Z$. Formally, this is defined following Maier and Kallmeyer (2010) as the *reduction* of $\boldsymbol{\alpha}$ with $\boldsymbol{\chi}$ as follows:

Definition 4.1 (Reduction): Let $\langle N, T, V, P, S\rangle$ be an LCFRS, $\boldsymbol{\alpha}$ a sequence of arguments $\alpha_1, \ldots \alpha_i \in (V \cup T)^*$ and $\boldsymbol{\chi}$ a sequence of variables $x_1, \ldots x_j \in V$ for some $i, j \in \mathbb{N}$. Let $w = \alpha_1\$ \ldots \$\alpha_i$ be the string obtained from concatenating the arguments in $\boldsymbol{\alpha}$, separated by a new symbol $\$ \notin (V \cup T)$. Let w' be the image of w under a homomorphism h defined as follows: $h(a) = \$$ for all $a \in T$, $h(X) = \$$ for all $X \in \{x_1, \ldots x_j\}$ and $h(y) = y$ in all other cases. Let $\boldsymbol{\gamma} = y_1, \ldots y_m \in V^+$ such that $w' \in \$^* y_1 \$^+ y_2 \$^+ \ldots \$^+ y_m \$^*$. Then $\boldsymbol{\gamma}$ is the *reduction* of $\boldsymbol{\alpha}$ by $\boldsymbol{\chi}$.

Data: an LCFRS $G = \langle N, T, V, P, S\rangle$ with a count function *count*
Result: G and *count* updated to represent a binarized LCFRS with counts
foreach *rule* $r = A(\boldsymbol{\alpha}) \to A_1(\boldsymbol{\alpha_1}) \ldots A_m(\boldsymbol{\alpha_m}) \in P$ *with* $m > 2$ **do**
  remove r from P;
  $R := \emptyset$;
  pick new non-terminals $C_2, \ldots, C_{m-1}$;
  add the rule $A(\boldsymbol{\alpha}) \to A_1(\boldsymbol{\alpha_1}) C_2(\boldsymbol{\gamma_2})$ to R where $\boldsymbol{\gamma_2}$ is obtained by reducing $\boldsymbol{\alpha}$ with $\boldsymbol{\alpha_1}$;
  foreach $i, 2 \leq i \leq m-2$ **do**
    add the rule $C_i(\boldsymbol{\gamma_i}) \to A_i(\boldsymbol{\alpha_i}) C_{i+1}(\boldsymbol{\gamma_{i+1}})$ to R where $\boldsymbol{\gamma_{i+1}}$ is obtained by reducing $\boldsymbol{\gamma_i}$ with $\boldsymbol{\alpha_i}$;
  end
  add the rule $C_{m-1}(\boldsymbol{\gamma_{m-1}}) \to A_{m-1}(\boldsymbol{\alpha_{m-1}}) A_m(\boldsymbol{\alpha_m})$ to R;
  foreach *rule* $r' \in R$ **do**
    replace RHS arguments of length > 1 with new variables (in both sides) and add the result to P;
    set $count(r') := count(r)$;
  end
end

Algorithm 4.1: Deterministic algorithm for binarizing an LCFRS with counts, adapted from Maier and Kallmeyer (2010)

As an example, consider the grammar from Figure 2.4, repeated here as Figure 4.2 with fan-out indices and the top production from the "artificial" start symbol dropped for simplicity. It has one rule with rank 3, which is binarized to two rules with rank 2. The binarized grammar is shown in Figure 4.3.

4.1.1. Markovization

In the above binarization algorithm, each binary rule newly introduced while binarizing an original rule gets its own unique LHS non-terminal. Each of these non-terminals

$$
\begin{aligned}
\text{SBARQ}(XY) &\rightarrow \text{SQ}(X).(Y) \\
\text{SQ}(XVU) &\rightarrow \text{VP}(X,U)C_2(V) \\
C_2(YZ) &\rightarrow \text{MD}(Y)\text{NP}(Z) \\
\text{VP}(X,Y) &\rightarrow \text{WHNP}(X)\text{VB}(Y) \\
\text{WHNP}(X) &\rightarrow \text{WP}(X) \\
\text{NP}(X) &\rightarrow \text{PRP}(X)
\end{aligned}
$$

Figure 4.3.: The grammar from Figure 4.2, binarized

occurs on the RHS of only one rule and of the LHS of only one rule. This has the effect that the rules stemming from the binarization remain "chained together" – they can only be used together in a derivation, simulating the use of the original unbinarized rule. If one of them is used, the rule to be used for expanding the newly introduced non-terminal on the RHS is already determined because there is only one with that non-terminal on the LHS. In this sense, the binarization is *deterministic*.

Alternatively, the LHS non-terminals of the newly introduced binary rules can be chosen to be non-unique, thus allowing recombinations of binary rules in derivations. One way to choose non-unique non-terminals is by horizontal and vertical context.

Definition 4.2 (Horizontal context): Given some $h \geq 0$ and a rule $r = A(\boldsymbol{\alpha}) \rightarrow A_1(\boldsymbol{\alpha_1}) \ldots A_m(\boldsymbol{\alpha_m})$, $hc(r, i, h)$ with $1 \leq i \leq m$ is called the *horizontal context* of the ith RHS element and defined as follows: if $h = 0$, then $hc(r, i, h) = \langle\rangle$. Otherwise, $hc(r, i, h) = \langle A_i, \ldots A_{max(i-h+1,1)}\rangle$.

Definition 4.3 (Vertical context): Given some $v \geq 1$ and a node N in a treebank where p is the shortest path from N to the root of the tree, the *vertical context* $vc(N, v)$ is defined as $\langle l(p(1)), \ldots l(p(\max(v, |p|)))\rangle$. If $r = rule(N)$ (cf. Section 2.2), then r is said to occur with vertical context $vc(N, v)$.

For example, consider the tree in Figure 4.1 and the (unbinarized) grammar extracted from it given in Figure 4.2. With $v = 1$, the vertical context of the SQ node is $\langle SQ\rangle$. The second rule in the grammar – let us call it r – has been extracted from that node, it is therefore said to occur with vertical context $\langle SQ\rangle$. With $v = 2$, the vertical context would be $\langle SQ, SBARQ\rangle$. Then, with a larger treebank, the same rule could also appear

in different vertical contexts. With $h = 2$, the horizontal context of the second RHS element of r, MD(Y), is $hc(r, 2, 2) = \langle \text{MD}, \text{VP} \rangle$.

The idea of binarization with Markovization is to not create a new LHS predicate for each new binary rule but instead choose predicates $C^{\vec{V}}_{\vec{H}}$ where $\vec{V}$ is a vertical context in which the rule to binarize occurs and $\vec{H}$ is the horizontal context of the RHS element that is split off by the new binary rule. This way, the binary rules introduced can be shared among different original unbinarized rules, as long as these have some RHS subsequences of length h as well as some vertical contexts in which they occur in common. Binarization then leads to horizontal Markovization of the grammar, a kind of factoring which has successfully been used to address the problem of sparse data: with several smaller rules, chances of finding a sufficient number of occurrences for reliably calculating the probabilities increase. Also, factoring rules makes the number of possible rules finite, making it possible in principle to apply standard smoothing techniques to further reduce problems stemming from sparse data.

The downside of factoring is that it introduces new independence assumptions which may be false. To prevent this from being detrimental, h should be chosen large enough. Without binarization or with deterministic binarization, h is effectively ∞ since RHSs are not factored at all. Similarly, choosing a large v counters the false independence assumption that expansions of RHSs are independent from vertical context. In the vanilla treebank model, v is 1 because the expansion of a rule is only conditioned on its LHS non-terminal. In choosing the parameters, there is a trade-off between false independence assumptions and the sparse data problem. Klein and Manning (2003b) discuss these issues extensively and present a type of context-free treebank model which uses horizontal and vertical context for the full grammar, not just for binarized rules as in the present work.

The algorithm for binarization with Markovization is given as Algorithm 4.2. Since we no longer choose a fresh LHS symbol for each new rule, the same rule (up to renamed variables) may be added to the grammar more than once during binarization. The algorithm takes care of this by adding the new count to the old count each time an equivalent rule is added. Also, while traversing an original rule, we no longer add a rule r to the set R, but rather a pair $\langle r, i \rangle$ where i is a unique index. This way, if the same rule is created twice while binarizing one rule, it is also counted twice. To accommodate vertical context, each original rule is now processed not only once, but once for each distinct vertical context it occurs with.

Data: an LCFRS $G = \langle N, T, V, P, S\rangle$ with a count function *count*, $h \geq 0$, $v \geq 1$
Result: G and *count* updated to represent a Markovized and binarized LCFRS with counts
foreach *rule* $r = A(\boldsymbol{\alpha}) \to A_1(\boldsymbol{\alpha_1}) \dots A_m(\boldsymbol{\alpha_m}) \in P$ *with* $m > 2$ **do**
  remove r from P;
  foreach *vertical context* $\vec{V}$ *with which* r *occurs in the treebank* **do**
    $R := \emptyset$;
    add the pair $\langle A(\boldsymbol{\alpha}) \to C^{\vec{V}}_{hc(r,1,h)}(\boldsymbol{\gamma_1}), 1\rangle$ to R where $\boldsymbol{\gamma_1} = \boldsymbol{\alpha}$;
    foreach $i, 1 \leq i \leq m-1$ **do**
      add the pair $\langle C^{\vec{V}}_{hc(r,i,h)}(\boldsymbol{\gamma_i}) \to A_i(\boldsymbol{\alpha_i}) C^{\vec{V}}_{hc(r,i+1,h)}(\boldsymbol{\gamma_{i+1}}), i+1\rangle$ to R where $\boldsymbol{\gamma_{i+1}}$ is obtained by reducing $\boldsymbol{\gamma_i}$ with $\boldsymbol{\alpha_i}$;
    end
    add the pair $\langle C^{\boldsymbol{V}}_{hc(r,m,h)}(\boldsymbol{\gamma_m}) \to A_m(\boldsymbol{\alpha_m}), m+1\rangle$ to R
    $c :=$ the number of times r occurs with $\vec{V}$ in the treebank;
    foreach *pair* $\langle r', i\rangle \in R$ **do**
      replace RHS arguments of length $> 1 \in r'$ with new variables (in both sides);
      if *some* $r'' \in P$ *is identical to* r' *via some bijection of the variables* **then**
        update $count(r'') := count(r'') + c$;
      end
      else
        add r' to P;
        set $count(r') := c$;
      end
    end
  end
end

Algorithm 4.2: Binarization algorithm with Markovization

$$
\begin{aligned}
\mathrm{SBARQ}(XY) &\rightarrow \mathrm{SQ}(X).(Y) \\
\mathrm{SQ}(V) &\rightarrow C^{\langle \mathrm{SQ} \rangle}_{\langle \mathrm{VP} \rangle}(V) \\
C^{\langle \mathrm{SQ} \rangle}_{\langle \mathrm{VP} \rangle}(XVU) &\rightarrow \mathrm{VP}(X,U)C^{\langle \mathrm{SQ} \rangle}_{\langle \mathrm{MD,VP} \rangle}(V) \\
C^{\langle \mathrm{SQ} \rangle}_{\langle \mathrm{MD,VP} \rangle}(YZ) &\rightarrow \mathrm{MD}(Y)C^{\langle \mathrm{SQ} \rangle}_{\langle \mathrm{NP,MD} \rangle}(Z) \\
C^{\langle \mathrm{SQ} \rangle}_{\langle \mathrm{NP,MD} \rangle}(Z) &\rightarrow \mathrm{NP}(Z) \\
\mathrm{VP}(X,Y) &\rightarrow \mathrm{WHNP}(X)\mathrm{VB}(Y) \\
\mathrm{WHNP}(X) &\rightarrow \mathrm{WP}(X) \\
\mathrm{NP}(X) &\rightarrow \mathrm{PRP}(X)
\end{aligned}
$$

Figure 4.4.: The grammar from Figure 4.2, left-to-right binarized with Markovization ($v = 1, h = 2$)

A final change is that, following Klein and Manning (2003b), a unary "top" rule is introduced before the leftmost RHS element of the original rule is split off, and a unary "bottom" rule is used to split off the rightmost RHS element. This provides additional factorization.

A binarized version of the example grammar in Figure 4.2 with Markovization is shown in Figure 4.4. The only non-binary rule is binarized to four rules: the first one is the "top" rule. In each of the subsequent three rules, one of the original RHS elements is "split off" while the remaining elements (if any) are grouped under a new non-terminal and factored out to the remaining rules. The new non-terminal symbols contain the vertical and horizontal context. In this case, $v = 1$, so the vertical context consists of only the LHS non-terminal of the original rule, SQ. $h = 2$, so the vertical context of each original RHS element consists of the non-terminal of this RHS element, followed by the non-terminal of the preceding RHS element (if any). The horizontal context of each RHS element is recorded on the new non-terminal that appears as the LHS non-terminal of that rule which splits off this RHS element.

4.1.2. Order of binarization

Unlike CFG, the order of the RHS elements in an LCFRS is immaterial. Therefore, they can be reordered before binarizing. In the case of binarization with Markovization of a PLCFRS, the order can influence parsing accuracy since it determines for RHS predicates

$$
\begin{aligned}
\text{SBARQ}(XY) &\rightarrow \text{SQ}(X).(Y) \\
\text{SQ}(XYZU) &\rightarrow \text{VP}(X,U)\text{NP}(Z)\text{MD}(Y) \\
\text{VP}(X,Y) &\rightarrow \text{WHNP}(X)\text{VB}(Y) \\
\text{WHNP}(X) &\rightarrow \text{WP}(X) \\
\text{NP}(X) &\rightarrow \text{PRP}(X)
\end{aligned}
$$

Figure 4.5.: The grammar from Figure 4.2, head-outward reordered

which other RHS predicates they are conditioned on. Put differently, it influences which independence assumptions one makes by using horizontal Markovization.

Klein and Manning (2003b) use *head-outward binarization* for binarizing CFGs, motivated by "the traditional linguistic insight that phrases are organized around a head". I.e., for each rule, that RHS element is first identified which represents the head of the corresponding phrase. Then, elements are split off starting from the periphery, making the head the "lowest" element. In the case of LCFRS, this can be achieved simply by reordering the RHS of each rule such that the head appears last. Before that, the elements right of the head appear, ordered from rightmost to leftmost in the original rule, followed by the elements left of the head, ordered from leftmost to rightmost. (It is also possible to put the elements left of the head before those right of the head, see below.) Figure 4.5 shows the example grammar after being reordered in this way. Then applying the binarization algorithm results in a grammar binarized with a different order, as shown in Figure 4.6. Figure 4.7 shows the parse tree resulting from parsing the example sentence with this binarized grammar directly. This illustrates that the modal verb tagged MD, which is the head of the SQ phrase, has become the most deeply embedded one of the original RHS elements.

For identifying the head of a rule, I use the head-finding rules of Collins (1999). These naturally refer to rules in a context-free grammar and rely on the RHS elements being in a particular order. In the extraction algorithm of Section 2.2, however, the order of RHS elements is undefined (since it is immaterial from the perspective of LCFRS) and in practice follows the position of the leftmost dominated terminal, which may not match the context-free order in the case of discontinuities. In the implementation, this problem is solved by retaining the original order of the children of each node in the context-free trees and basing the head-finding process on that order. A constituent that is reattached in transformation is treated as assuming the position of the empty placeholder

$$
\begin{aligned}
\text{SBARQ}(XY) &\rightarrow \text{SQ}(X).(Y) \\
\text{SQ}(V) &\rightarrow C^{\langle \text{SQ} \rangle}_{\langle \text{NP} \rangle}(V) \\
C^{\langle \text{SQ} \rangle}_{\langle \text{NP} \rangle}(VZU) &\rightarrow C^{\langle \text{SQ} \rangle}_{\langle \text{VP,NP} \rangle}(V,U)\text{NP}(Z) \\
C^{\langle \text{SQ} \rangle}_{\langle \text{VP,NP} \rangle}(XY,U) &\rightarrow \text{VP}(X,U)C^{\langle \text{SQ} \rangle}_{\langle \text{MD,VP} \rangle}(Y) \\
C^{\langle \text{SQ} \rangle}_{\langle \text{MD,VP} \rangle}(Y) &\rightarrow \text{MD}(Y) \\
\text{VP}(X,Y) &\rightarrow \text{WHNP}(X)\text{VB}(Y) \\
\text{WHNP}(X) &\rightarrow \text{WP}(X) \\
\text{NP}(X) &\rightarrow \text{PRP}(X)
\end{aligned}
$$

Figure 4.6.: The grammar from Figure 4.2, head-outward binarized with Markovization ($v = 1, h = 2$)

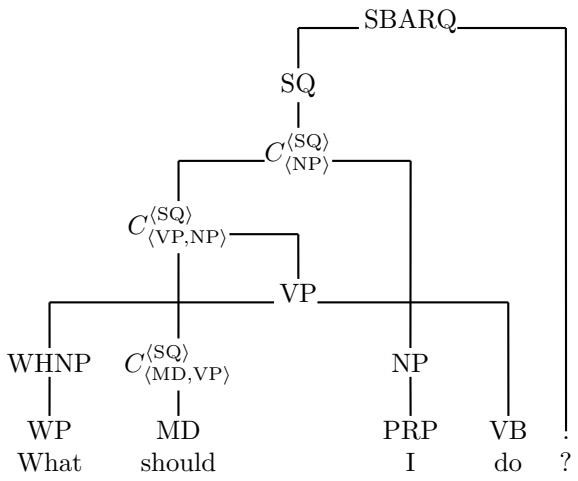


Figure 4.7.: Parse of the sentence using the grammar in Figure 4.6 before debinarizing

constituent that is removed in exchange. This also has the effect that (extracted) WHNP, WHPP, WHADVP and WHADJP nodes may appear in positions where the head-finding rules assume an NP, PP, ADVP or ADJP node. The rules have thus been modified to treat categories with and without the WH prefix alike.

Another consideration for binarization is that as a result of reducing argument sequences, the newly introduced rules in binarized grammars may have a higher fan-out than the predicates of the original rule, possibly leading to a higher fan-out of the grammar. The parsing complexity of an LCFRS is exponential in both its rank and its fan-out, so binarization potentially means trading one evil for another. Gómez-Rodríguez et al. (2009) report that typically, in the context of syntactic structures, binarization does not lead to a (strong) increase in fan-out. Still, it may be beneficial to binarize in an order such that a grammar with minimal fan-out is obtained.

The following are the binarization strategies I have tried out. Some of the RHS reorderings are given as sequences of the original RHS position numbers, p being the head position and m the rank of the original rule:

det-lr Binarization without Markovization, no reordering.

ho-lr RHS reordered with $m, \ldots, p{+}1, 1, \ldots, p$, the result is a head-outward binarization, i.e. the head is the lowest subtree. It is extended by first adding all its sisters to the left, then all its sisters to the right. This is the order used in Maier and Kallmeyer (2010) for discontinuous constituent parsing of German.

ho-rl RHS reordered with $1, \ldots, p{-}1, m, \ldots, p$, the result is a head-outward binarization, i.e. the head is the lowest subtree. It is extended by first adding all its sisters to the right, then all its sisters to the left. This is the order used in Klein and Manning (2003b) for context-free parsing of English.

lr RHS not reordered, the result is a right-branching binarization.

rl RHS reversed, the result is a left-branching binarization.

min Strategy for minimizing fan-out and the number of variables. In each iteration, an RHS element is split off such that the resulting fan-out is minimal. Ties are broken by minimizing the number of variables in the binary rule to introduce. The exact algorithm is given in Kallmeyer (2010, Chapter 7).

4.2. Category splitting

Category splitting is a technique with a purpose similar to Markovization, viz. cutting back on certain independence assumptions in order to speed up parsing and increase accuracy. It consists in *splitting* certain constituent categories, i.e. relabeling internal nodes in the treebanks before extracting the grammar, depending on context. This can improve the model in cases where certain expansions of a non-terminal symbol correlate with certain contexts in which that non-terminal symbol occurs. Maier and Kallmeyer (2010) report improved parsing accuracy in their model for German after manually introducing two splits: relabeling S nodes to SRC where the node is the root of a relative clause and relabeling VP to VP, VP-PP, VP-INF or VP-ZU depending on the form of the head verb. The following two splits for the Penn Treebank annotation can be defined analogously:

S$_{\text{WH}}$ S nodes are relabeled to SWH if they root a question or a relative clause. This is the case if, before the transformation to the discontinuous format, the parent of the node is an SBAR node and has a child node whose label starts with WH (cf. Section 3.3.1). In the discontinuous constituent parsing experiments where this split is used, SBAR nodes that have only one child after the transformation are removed (both in the training and gold standard test corpus) so the category split has an effect on the rule extracted from the phrase in which the clause appears. This split aims to reflect the fact that clauses with extracted *wh*-elements appear in different contexts from other sentence-level constituents, e.g. relative clauses frequently occur as post-nominal modifiers.

VP$_{\text{PART/INF}}$ VP nodes are relabeled to VPHINF if their head is an infinitive (tagged VB) or the *to* introducing a *to*-infinitive (tagged TO) and to VPHPART if the head verb is a participle or gerund (tagged VBN or VBG). The head verb is found by looking for the leftmost daughter node of the VP node whose tag starts with VB. This split aims to reflect the fact that infinitival and participle/gerund verb phrases tend to occur in different contexts from finite ones, e.g. finite verb phrases occur less frequently as a dependent of another verb or a preposition than infinitives, participles or gerunds.

In addition, three further splits were defined for the Penn Treebank annotation:

S$_{\text{INF/PRO}}$ S nodes rooting infinitival clauses are relabeled to SPRO if the subject is the empty subject * (cf. Section 3.5.1), and to SINF otherwise. In order for the empty

subject information to be available, this split is carried out before the transformation to the discontinuous format. This split is designed to be used in combination with $\text{VP}_{\text{PART/INF}}$, percolating the information about infinitive head verbs one level higher up in the (not yet binarized) tree and allowing the grammar to distinguish the contexts of infinitival vs. non-infinitival clauses, and in addition to distinguish the contexts of clauses where the subject is empty, such as in raising and control constructions.

S_{INF} A variant of $\text{S}_{\text{INF/PRO}}$ that does not distinguish between infinitival clauses with empty and non-empty subjects and relabels both to SINF.

$\text{VP}_{\text{PART/INF/TO}}$ A variant of $\text{VP}_{\text{PART/INF}}$ where *to*-infinitives are distinguished as a category VPHTO distinct from VPHINF. This split is intended to capture distributional characteristics of bare infinitives vs. *to*-infinitives, e.g. the former are selected by modal verbs (tagged MD) while the latter are not.

When using splits, the split categories in the parser output are mapped back to their original categories before evaluation, and evaluated against the unsplit gold standard.

4.3. Estimates and Cutoff

The parsing algorithm presented in Section 2.3 is guaranteed to find the most probable parse first, and thus to output it as the result. A disadvantage is that very many items may have to be processed before arriving at a parse. It is possible to equip items with *outside estimates*, i.e. roughly the estimated probabilities of them leading to a successful parse. In retrieving items from the agenda, the parser can then prioritize items with good estimates and thus find a parse more quickly. Klein and Manning (2003a) use outside estimates for parsing CFG. Their method of calculating estimates is *monotonic*, a condition under which the first parse found is still guaranteed to be the best one (A* parsing). Monotonic estimate types for PLCFRS are presented in Kallmeyer and Maier (2010), but found to be too expensive to compute for practical parsing. Maier (2010) therefore uses a non-monotonic type of estimate for PLCFRS parsing, *SX simple LR estimate*, which is found to be relatively effective for reducing parsing time without an excessive loss in accuracy despite not always finding the best parse.

Another way to reduce the number of items produced is *cutoff*, i.e. discarding very infrequent rules. For a cutoff value n, every rule r with $count(r) \leq n$ is removed from the binarized grammar before computing the probabilities.

4.4. Experimental choice of parameter settings

To find a good set of parameter settings, a series of parsing experiments on the same data but with different settings was performed. The goal was not to determine an absolute optimal combination of parameter settings, which would be hardly feasible seeing that such optima can be highly data-dependent, but to gain an overview of the effects of individual parameters on the data at hand and to find a reasonable combination of settings for the main experiments in the next chapter. Therefore, not all possible combinations were tried out – instead, in most cases one parameter was varied while the other parameters remained fixed.

4.4.1. Data set

Due to time constraints, a relatively small data set with 3600 sentences for training and 400 sentences for testing was chosen from the transformed treebank for this series of experiments, and only sentences with at most 25 tokens after removing null elements were included.

The training and test set were each chosen to be representative of the whole corpus in the sense that each of the five types of reattachment operation distinguished in Section 3.6 should occur with the same relative frequency as in the whole corpus. Since 31% of all trees (15452 of 49208) contain the result of a reattachment operation of type **T**, 1116 (31% of 3600) were chosen at random and added to the training set, and 124 (31% of 400) were chosen at random and added to the test set. Similar for the other four types. The process was repeated a few times until a training set and test set were found that did not overlap and each had the desired number of sentences with reattachment operations (36%, i.e. 1296 for training, 146 for testing), thereby implicitly excluding sentences which contain more than one type of reattachment. This is a somewhat artificial constraint, but one that does not make a great difference given the already limited sentence length and the infrequency of different types of reattachment in one sentence. Finally, the remaining 64%, 2558 sentences without reattachment operations, were cho-

sen at random, 2304 of them added to the training set and the remaining 254 to the test set.

4.4.2. Evaluation

For comparative evaluation of the parser outputs given different parameter settings, the EVALB-style metric from Maier and Kallmeyer (2010) was used: each constituent in a tree over a string w is represented by a pair $\langle A, \vec{\rho} \rangle$ where A is the label of the constituent root and $\vec{\rho} \in (Pos(w) \times Pos(w))^{dim(A)}$ contains the (ordered) ranges of leaves dominated by the constituent root. Precision, recall and f-measure are computed in the usual way by comparing the pairs appearing in the gold standard vs. the pairs appearing in the parser output.

4.4.3. Experiments and results

The combinations of parameter settings tried out are given as the rows of Table 4.1. For each combination, one experiment was conducted on the transformed version of the small corpus. For comparison, the experiments were later repeated on the untransformed, context-free versions of the trees, after only making corrections (see Appendix A) and taking the usual preprocessing steps for parsing the Penn Treebank (removing null elements, removing non-terminals without children until none are left, and removing all indices from node labels). However, the focus for choosing the "best" settings was on the discontinuous, transformed trees. As a measure of the relative complexity of and hence parsing time taken by each experiment, the number of items produced during the parsing processes for all 400 sentence in the small test corpus is given, in millions (mil. it.) As a measure of accuracy, labeled (lf) and unlabeled f-measure (ulf) are given.

A first series of experiments explored different Markovization settings, using ho-lr binarization, no estimates, no cutoff and no category splits. The varying parameters were vertical Markovization (1 or 2) and horizontal Markovization (0, 1 or 2). All combinations were tried out. The different degrees of vertical Markovization made only a small difference for accuracy, whereas accuracy increased considerably with the degree of horizontal Markovization, making a vertical degree of 1 and a horizontal degree of 2 the winning combination. This is the combination used for parsing German in Maier

						transformed small corpus			context-free small corpus		
v	h	bin	est	cut	split	mil. it.	lf (%)	ulf (%)	mil. it.	lf (%)	ulf (%)
1	0	ho-lr	-	-	-	98	60.09	65.10	2	63.18	67.85
1	1	ho-lr	-	-	-	66	66.20	71.57	6	72.25	77.33
1	2	ho-lr	-	-	-	75	**69.59**	**74.96**	17	72.02	77.24
1	3	ho-lr	-	-	-	99	68.25	73.91	32	72.18	67.97
2	0	ho-lr	-	-	-	155	59.94	65.15	30	61.99	66.69
2	1	ho-lr	-	-	-	150	64.31	69.56	34	68.34	73.11
2	2	ho-lr	-	-	-	166	68.27	73.68	56	71.62	76.92
1	2	ho-rl	-	-	-	65	69.62	74.85	16	71.30	76.37
1	2	lr	-	-	-	107	69.93	75.30	18	71.13	76.42
1	2	rl	-	-	-	63	68.38	73.91	24	72.56	77.57
1	2	min	-	-	-	99	70.32	75.87	-	-	-
-	-	det-lr	-	-	-	**56**	**71.69**	**76.79**	33	74.56	79.22
-	-	det-lr	sxslr	-	-	**43**	68.62	74.18	28	68.76	74.24
-	-	det-lr	-	1	-	**12**	68.49	73.45	8	71.17	75.92
1	2	ho-lr	-	1	-	**18**	65.06	70.48	7	67.29	72.19
-	-	det-lr	-	-	S_{WH}	**52**	72.73	76.85	33	73.65	78.39
-	-	det-lr	-	-	$VP_{PART/INF}$	54	73.34	77.27	31	75.39	79.31
-	-	det-lr	-	-	$VP_{PART/INF/TO}$	54	**73.39**	**77.18**	31	74.74	78.42
1	2	ho-lr	-	-	$VP_{PART/INF/TO}$	64	70.86	75.40	18	73.11	77.35
-	-	det-lr	-	-	$VP_{PART/INF/TO} \circ S_{INF}$	**50**	**74.07**	**77.45**	28	76.20	79.54
-	-	det-lr	-	-	$VP_{PART/INF/TO} \circ S_{INF/PRO}$	**50**	73.22	76.75	28	76.07	79.52
-	-	det-lr	-	-	$S_{WH} \circ VP_{PART/INF/TO}$	49	**75.76**	**78.75**	31	74.88	78.20
-	-	det-lr	-	-	$S_{WH} \circ VP_{PART/INF/TO} \circ S_{INF}$	**42**	75.34	78.11	28	75.85	78.93

Table 4.1.: Experimental choice of parameter settings, results. In each series on the transformed corpus, bold figures indicate the strongest improvement (if any) over the previous comparable combination of parameter settings.

and Kallmeyer (2010) as well. A further experiment raising horizontal Markovization to 3 saw accuracy decline again.

With the Markovization degrees fixed, a second series explored the five alternative binarization strategies. Somewhat surprisingly, the most primitive strategy without Markovization, det-lr, performed best. The second best results were obtained with the min strategy. For the context-free case, where the unbinarized extracted grammar is necessarily a 1-LCFRS with exactly one argument per non-terminal and one variable per RHS element, the min strategy was left out as there is no room for optimization and it would default to the same binarization as produced by the lr strategy.

Using the outside-probability estimate or a cutoff with $n = 1$ reduced parsing time, drastically in the case of the cutoff, but only at the price of a considerable loss in accuracy. Both techniques were therefore abandoned for the remaining experiments.

The next series of experiments explored the three most basic category splits: S_{WH}, $\mathrm{VP}_{\mathrm{PART/INF}}$ and its finer-grained variant $\mathrm{VP}_{\mathrm{PART/INF/TO}}$. Each of the three splits by itself reduced the total number of items slightly and increased labeled f-measure by about 1%. The last of these splits was also tried out with the more sophisticated binarization strategy ho-lr to see if this strategy can profit significantly more from it than det-lr, but this was not the case. There was almost no difference in the gains afforded by $\mathrm{VP}_{\mathrm{PART/INF}}$ and its finer-grained variant $\mathrm{VP}_{\mathrm{PART/INF/TO}}$. The remaining experiments focused on the latter, first combining it with $\mathrm{S}_{\mathrm{INF}}$ (and its finer-grained variant $\mathrm{S}_{\mathrm{INF/PRO}}$, which performed worse), then adding S_{WH}. In each step, accuracy could again be boosted by about 1%, showing that the splits do not only work in isolation, but their positive effects can – to some degree – be combined. Excessive use of more splits, however, would increase the number of categories further and could thus lead to a new data sparseness problem. This perhaps already shows in the fact that the winning combination of splits in terms of accuracy is not $\mathrm{S}_{\mathrm{WH}} \circ \mathrm{VP}_{\mathrm{PART/INF/TO}} \circ \mathrm{S}_{\mathrm{INF}}$, but, by a small amount, the one obtained when leaving out $\mathrm{S}_{\mathrm{INF}}$. However, seeing that data sparseness problems would presumably be mitigated when using the full training corpus, and that the three-split combination was a clear winner in terms of efficiency, it was ultimately chosen for the main parsing experiments in Chapter 5.

It becomes clear from these figures that category splits are a promising way to improve PLCFRS parsing. It is also clear that they warrant a much more systematic investigation. For example, Maier and Kallmeyer (2010) envision coming up with useful splits automatically rather than manually, by adapting a technique to PLCFRS that was

presented and successfully used by Ule (2003) for PCFG. Another such technique, previously used for successfully for parsing English, is split-state grammars (Petrov et al., 2006).

The fan-out of the grammar extracted from the small training corpus is necessarily 1 in the context-free case, and 3 in the discontinuous case, corresponding to the maximal gap degree of 2 found in Section 3.6. In no experiment did the fan-out increase as a result of binarization.

In the context-free experiments, much fewer items are produced due to the lower parsing complexity in the case of 1-LCFRS as opposed to 3-LCFRS and also due to a smaller number of extracted rules (by about 10%). Absolute parsing accuracy is also considerably higher in the experiments where no splits are used. The relative results are not much different, though: while the combinations of Markovization degrees rank differently here (but not by much), the preference for deterministic left-to-right binarization is confirmed. So are the effects of the outside-probability estimate and of cutoff 1. The effect of splits is less significant than in the discontinuous case, especially where the S_{WH} split is used. In the discontinuous case, this split makes the removal of SBAR nodes possible which presumably block relevant contextual information from being propagated in the trees (further splits could be used to try to propagate this information through SBAR nodes in the context-free case). It should be kept in mind that higher figures for the context-free experiments do not necessarily indicate better parses as they are evaluated against a less informative gold standard (no non-local dependencies) than in the discontinuous experiments. This point will be taken up again in evaluating the results of the main parsing experiments in the next chapter.

Chapter 5.

Main experiments and evaluation

This chapter presents and evaluates two parsing experiments, one discontinuous and one context-free. They both use sections 1-22 of the WSJ treebank for training and section 23-24 for testing. Due to limited time and the complexity of LCFRS parsing, both training and testing was restricted to those sentences from the respective sections that have no more than 25 tokens after removing null elements. This amounts to a training set of 25801 sentences and a test set of 2233 sentences. Parsing is carried out with the parameter settings chosen in the previous chapter, i.e. deterministic binarization, no Markovization, no estimates, no cutoff and with the category split $S_{WH} \circ VP_{PART/INF/TO} \circ S_{INF}$. As in the previous chapter, the context-free experiment uses the WSJ treebank with the corrections presented in Appendix A, null elements, empty constituents and indices removed, and the discontinuous experiment uses the version of the corrected treebank transformed using Algorithm 3.2. Table 5.1 shows the results of evaluating the parse trees against the gold standard annotation of the test corpus, using the EVALB-style metric already used in the last chapter, now not only showing f-measure but also recall and precision.

As already noted at the end of the previous chapter, a meaningful comparison between the EVALB-style evaluation results of discontinuous and context-free experiments is difficult, as the evaluation is against different gold standards. The transformed treebank differs from the untransformed treebank in that it 1) contains non-local attachment that induces additional semantically relevant dependencies and 2) leaves out instances of local attachment of dependents removed from their interpretation locations, i.e. attachment that does not systematically induce semantically relevant dependencies. Thus, if the goal is to assess how much closer discontinuous parsing comes to the ideal of a

	discontinuous	context-free
million items produced	1056	580
labeled recall	77.61%	76.94%
labeled precision	80.36%	80.37%
labeled f-measure	78.96%	78.61%
unlabeled recall	80.88%	80.33%
unlabeled precision	83.74%	83.91%
unlabeled f-measure	82.29%	82.08%

Table 5.1.: EVALB-style evaluation of the big parsing experiments

syntactic representation inducing all the semantically relevant dependencies, both parser outputs should be evaluated against the dependencies induced by the transformed gold standard annotation of the test set. Such an evaluation methodology is offered by *dependency evaluation* (Lin, 1995). This evaluation method has gained in popularity after the deficiencies of EVALB-style evaluation became more and more apparent (see e.g. Rehbein and van Genabith, 2007) and is now a standard method for parser evaluation, also because it directly reflects what is arguably the most important quality of a syntactic representation for subsequent semantic interpretation: how well word-word dependencies are recognized. Dependency evaluation has been used before for evaluating PLCFRS parser output by Maier (2010).

A method is needed for converting the gold standard and both parser outputs from constituent trees to sets of directed word-word dependencies $\langle d, h \rangle$ where d is the dependent word and h is the head word. Like Maier, I use the standard method already described in (Lin, 1995): among the children of every internal node in the tree, one is marked as the head child. For this, the modified head finding rules from Collins (1999) are used, (cf. Section 4.1.2). The *lexical head* of a node n is defined as n if n is a leaf (i.e. a word), otherwise as the lexical head of its head child. Then for every internal node m in the tree, for each non-head child c of m, a dependency $\langle d, h \rangle$ is recorded where d is the lexical head of c and h is the lexical head of m. For each tree, a dependency $\langle r, v \rangle$ is recorded where r is the lexical head of the root and v is a virtual word, meaning that r does not depend on any real other word. Together these dependencies make up the set of word-word dependencies induced by the constituent tree. For evaluation, both the

	discontinuous		context-free	
T	$\frac{203}{439}$	46%	$\frac{136}{439}$	31%
T-PRN	$\frac{25}{32}$	78%	$\frac{8}{32}$	25%
ICH	$\frac{5}{31}$	16%	$\frac{4}{31}$	13%
EXP	$\frac{1}{18}$	6%	$\frac{0}{18}$	0%
RNR	$\frac{3}{4}$	75%	$\frac{2}{4}$	50%
other	$\frac{29983}{35915}$	83%	$\frac{29637}{35915}$	82%
total	$\frac{28202}{36439}$	82%	$\frac{27786}{36439}$	81%

Table 5.2.: Unlabeled attachment score broken down by dependency type

parser output trees and the gold standard trees are converted to sets of dependencies P (parser output) and G (gold standard). The *unlabeled attachment score* is given by $\frac{|P \cap G|}{|G|}$, i.e. the recall of unlabeled directed dependencies in the gold standard that were "recognized" by the parser, or, equivalently, the percentage of tokens that are recognized as depending on the correct head word.

A further advantage of dependency evaluation is that it allows for scoring individual types of dependencies separately. To this end, the program implementing the transformation algorithm of Chapter 3 was extended so that on each reattachment operation performed, it records the type of operation (cf. Section 3.6) and the dependencies thereby created. The program computing unlabeled attachment score can then put dependencies into different bins depending on their type and output separate recall scores. The unlabeled attachment scores for the results of the two experiments are shown in Table 5.2.

As is to be expected, the scores for the five types of dependencies created by reattachment operations that potentially introduce discontinuities are much higher for the discontinuous experiment than for the context-free one, though the latter are not zero as one might initially expect. The reason is that even in context-free trees, the dependency conversion method described above sometimes incidentally introduces correct non-local dependencies. This can be seen in a tree as simple as the first example in Figure 3.2 (left): the non-local dependency relevant here is that of the pronoun *What* on the verb *do*. Since *do* is the lexical head of the SBARQ root, which also has a child whose lexical head is *What*, this dependency is produced. If it is an advantage that in the discontinu-

ous analysis (Figure 3.2, right) the dependent directly enters into a relation with its verb head even on the level of constituents, then this advantage is not taken into account by dependency evaluation.

Another slight problem is that the head-finding algorithm needed for dependency conversion does not work quite reliably on the (unordered) discontinuous parse trees since it partly relies on a particular order of the children of each node, which can only be approximated by ordering nodes by leftmost dominated token.[1] Finally, the numbers in the table give a rough idea of which types of non-local dependencies are problematic (especially *ICH* and *EXP*), but do not say much about potential reasons and types of errors made by the parser. The rest of this chapter will therefore focus on a manual analysis of the parse trees and of the non-local dependencies recognized (or not recognized) in the discontinuous parsing experiment.

5.1. Manual evaluation of discontinuities

Not all transformation operations introduce discontinuities. Recall, for example, that right-node raised constituents are almost always reattached so that no discontinuity is introduced, and *wh*-movement from the subject position does not normally introduce a discontinuity either. To gain an insight into how well the parser deals specifically with recognizing discontinuities, a manual evaluation of all sentences with discontinuities in the test set was performed. The grammatical phenomena that cause discontinuities have been categorized into six classes: *wh*-movement, fronted quotations, other fronted arguments (all three corresponding to the *T* transformation operation from Section 3.6), circumpositioned dependents (corresponding to *T*-PRN), *it*-extraposition (corresponding to *EXP*) and discontinuous dependency (corresponding to *ICH*). Table 5.3 shows for each class how many instances appear in the test set and how many of them were recognized by the parser fully, partially, or not at all. The following six subsections discuss the six classes, possible reasons for success and failure, and types of errors. 77 discontinuities found by the parser do not have corresponding discontinuities in the gold standard, thus were classified as false positives and roughly assigned

[1] A head-finding algorithm adapted to discontinuous trees that could be used to solve this problem is presented in Levy (2005, p. 128).

wh-movement	gold-standard	129	
	fully recognized	108	84%
	moved phrase too small	1	1%
	moved phrase too big	1	1%
	moved phrase misattached	5	4%
	not recognized	14	11%
fronted quotations	gold standard	142	
	fully recognized	129	91%
	fronted phrase too small	7	5%
	not recognized	6	4%
other fronted arguments	gold standard	6	
	fully recognized	0	0%
	fronted phrase too big	1	17%
	not recognized	5	83%
circumpositioned dependents	gold standard	31	
	fully recognized	22	71%
	left part too small	1	3%
	not recognized	8	26%
it-extraposition	gold standard	14	
	recognized	2	14%
	not recognized	12	86%
discontinuous dependency	gold standard	30	
	recognized	2	7%
	not recognized	28	93%

Table 5.3.: Causes of discontinuities in the transformed treebank and their rates of recognition by the parser

wh-movement	2
fronted quotations and other fronted arguments	25
discontinuous dependency and *it*-extraposition	44
circumpositioned dependents	6

Table 5.4.: Incorrectly recognized discontinuities (false positives), roughly classified by the grammatical phenomenon the parser seems to have recognized

.

to specific causes of discontinuity incorrectly suggested by the parse trees, as shown in Table 5.4. These cases are discussed in Section 5.1.7.

The node labels of gold standard and parse trees shown in this section reflect the category splits, they have not yet been conflated back to the original categories for evaluation. This way, some decisions of the parser become more transparent.

5.1.1. wh-movement

The test set contains 129 discontinuities that arise through *wh*-movement. Credit for recognizing an instance is given to the parser if there is a node in the parse tree whose lexical head is the lexical head of the phrase containing the interpretation location of the moved *wh*-phrase in the gold standard, which also dominates the moved *wh*-word, and whose root does not dominate some tokens occurring between the *wh*-word and the head. In order for the instance to count as "fully recognized", this node must dominate all the tokens of the the whole moved *wh*-phrase, as defined by the gold standard, and must not dominate the token immediately to the left or immediately to the right of the moved phrase. If one of the last two criteria is not fulfilled, partial credit is given as "moved phrase too small" resp. "moved phrase too big". Partial credit is also given if the correct phrase is recognized as moved, but attached at a wrong level such that it depends on the wrong lexical head.

wh-movement is fully recognized in most cases, even when the interpretation location of the moved *wh*-phrase is quite deeply embedded into the clause and thus a relatively large number of "stacked" discontinuous constituents is needed, as in Figure 5.1. The occurrence of this type of discontinuity strongly correlates with the occurrence of a *wh*-

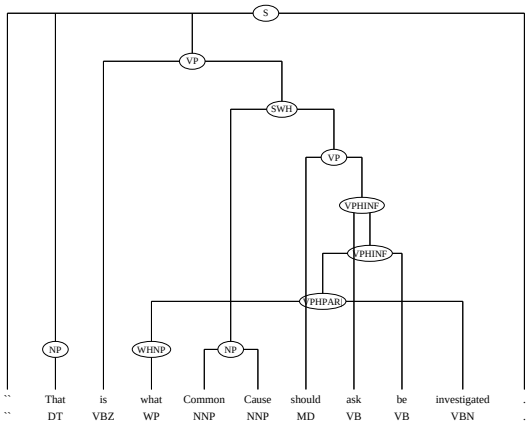


Figure 5.1.: Correct parse of a deeply embedded moved *wh*-phrase

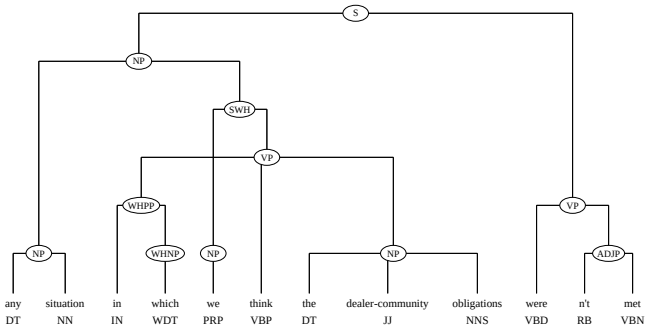


Figure 5.2.: Incorrect parse: among other errors, the moved *wh*-phrase is attached to the wrong VP.

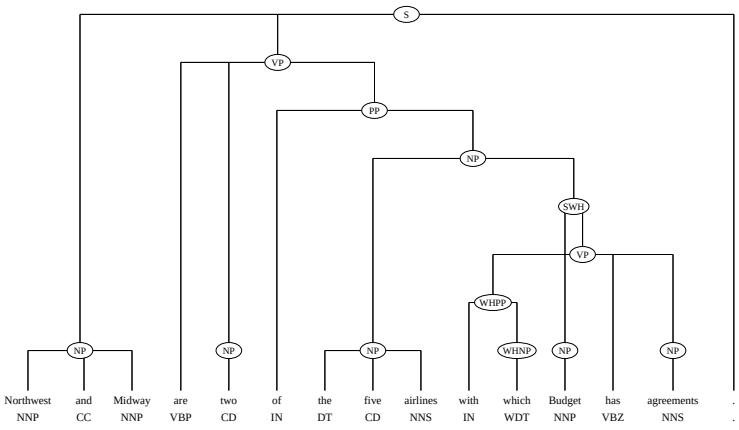


Figure 5.3.: Parse with incorrectly resolved attachment ambiguity: *with which* depends on *agreements* according to the gold standard, not *has*.

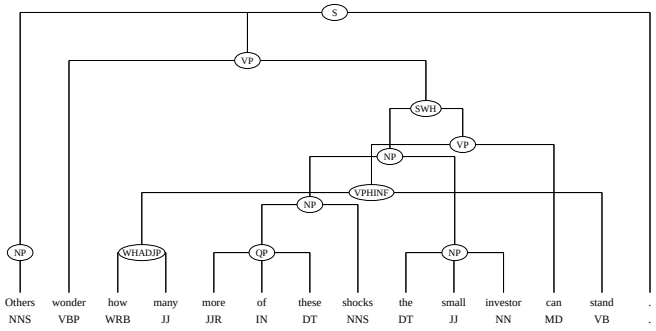


Figure 5.4.: Incorrect parse: a too small *wh*-phrase is recognized as moved. It should be a whole WHNP *how many more of these shocks*. Instead, the remaining part of the *wh*-phrase is incorrectly attached to the subject NP.

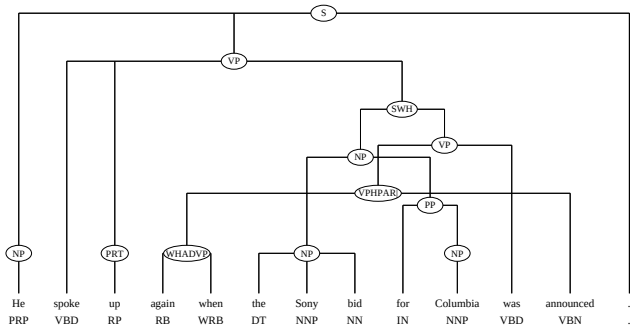


Figure 5.5.: Incorrect parse: an adverb from outside a clause is incorrectly attached to its moved *wh*-phrase.

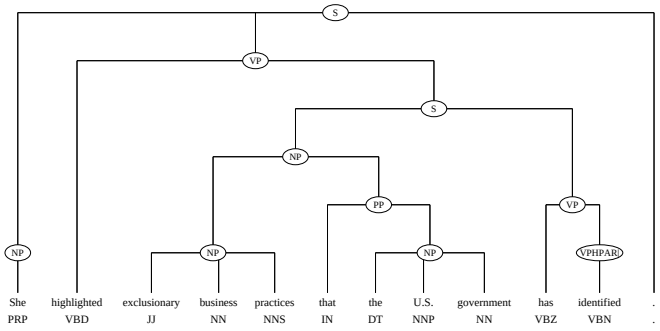


Figure 5.6.: Incorrect parse: *wh*-movement is not recognized due to the relative complementizer *that* being mistagged.

word (tagged WDT, WP, W$ or WRB) followed by a noun phrase (the subject of the clause) before the next verb. This is presumably what makes recognition easy for the parser. It also correctly attaches recognized *wh*-phrases in all but 5 instances, in 4 of which the correct attachment site is unavailable due to other parsing errors (see e.g. Figure 5.2) and one of which (Figure 5.3) has a genuine attachment ambiguity. This small number of attachment errors reflects another regularity in the data: *wh*-phrases usually attach at the lowest phrase in a stack of verbal phrases. Errors of the types "moved phrase too small" (Figure 5.4) or "moved phrase too big" (Figure 5.5) are even rarer, with one instance each.

Of the 14 instances of *wh*-movement not recognized by the parser, eight involve a plain tagging error in the original Penn Treebank annotation: the word *that* is tagged IN although it introduces a relative clause and should thus be tagged WDT (cf. Santorini, 1990). Such relative clauses are then usually not recognized as a constituent at all by the parser, see e.g. Figure 5.6.

The remaining six instances of missed *wh*-movement seem to be due to the same kinds of reasons that context-free parsers fail to recognize some continuous constituents: the correct analysis would require the use of rules that occur rarely or not at all in the training corpus while there are other rules that can be used to construct a valid but potentially nonsensical parse with a high probabilistic score. One might think that in the present setup where no horizontal Markovization is used, VP nodes with many children might be a main reason for such problems due to sparse training data: beyond various argument structures, the grammar rules are further diversified by S

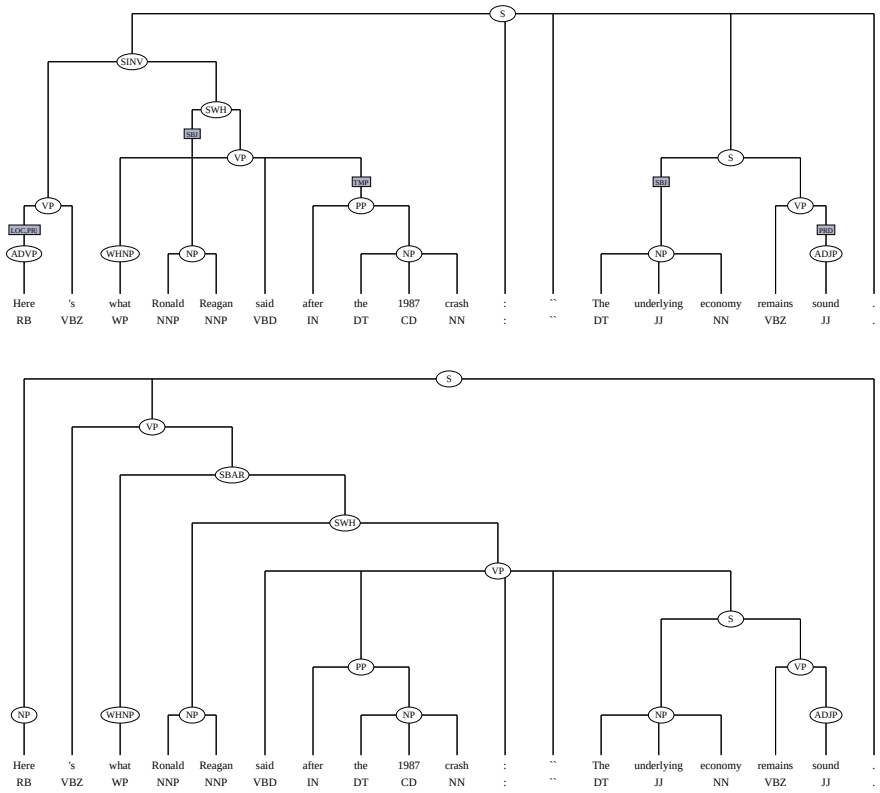


Figure 5.7.: Gold standard and parse: an unusual construction involving *wh*-movement, not recognized.

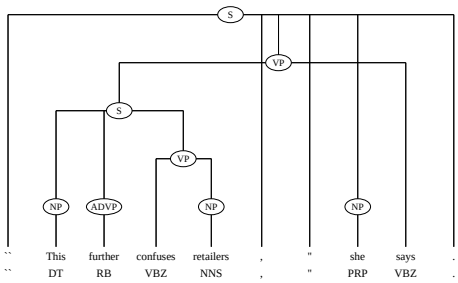


Figure 5.8.: Correct parse: fronted quotation in a sentence with regular word order

and VP category splits and by *wh*-moved vs. non-*wh*-moved arguments. However, in the instances of unrecognized *wh*-movement found, the SWH and VP rules required for the correct analysis are fairly frequent, the problem is more likely with recognizing the context in which an SWH constituent can occur. One example is shown in Figure 5.7. The unusual construction involved is a fused relative clause that serves as the subject of a sentence with subject-verb inversion (SINV). Here, both of the rules directly involved with the discontinuity, $\mathrm{SWH}_1(XYZ) \rightarrow \mathrm{VP}_2(X,Z)\mathrm{NP}_1(Y)$ and $\mathrm{VP}_2(X,YZ) \rightarrow \mathrm{WHNP}_1(X)\mathrm{VBD}_1(Y)\mathrm{PP}_1(Z)$, are frequent, and the rare rule the parser fails to apply is $\mathrm{SINV}_1(XY) \rightarrow \mathrm{VP}_1(X)\mathrm{SWH}_1(Y)$. The other false negatives are similar in this respect: the problem does not seem to be with the discontinuous rules but with other rules.

In the presence of *wh*-words, wrong analyses involving SBAR such as that shown in Figure 5.7 are made possible by rules of the form $\mathrm{SBAR}_1(XY) \rightarrow \mathrm{W...}_1(X)\mathrm{S}_1(Y)$. These still occur in the transformed treebank mainly because of incompletely annotated *wh*-movement in the original treebank where coindexed null elements were omitted and thus reattachment to the interpretation location did not take place in the transformation.

5.1.2. Fronted quotations

As a characteristic of newspaper text, fronted quotations (cf. Section 3.3.2) are a particularly frequent construction introducing discontinuities in the WSJ treebank. They come in two broad flavors, characterized by rules of the forms $\mathrm{S}_1(\ldots X \ldots YZ \ldots) \rightarrow \ldots \mathrm{VP}_2(X,Z) \ldots \mathrm{NP}_1(Y) \ldots$ for standard word order as shown in Figure 5.8 and $\mathrm{SINV}_1(\ldots X \ldots YZ \ldots) \rightarrow \ldots \mathrm{VP}_2(X,Y) \ldots \mathrm{NP}_1(Z) \ldots$ for subject-verb inverted word order

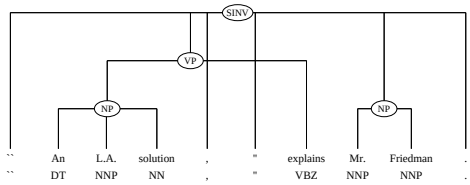


Figure 5.9.: Correct parse: fronted quotation in a sentence with inverted word order

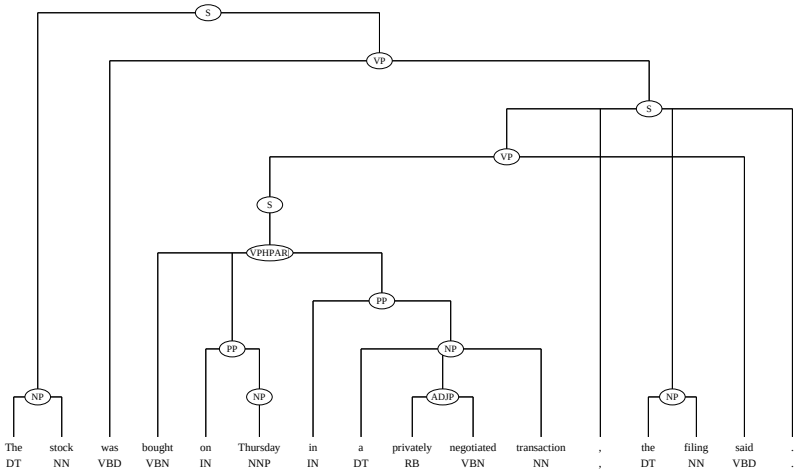


Figure 5.10.: Incorrect parse: a fronted quotation is recognized, but too small.

as shown in Figure 5.9. The dots stand for various patterns of punctuation; conjunctions and adjuncts attached at sentence level can also occur. With inverted word order, the discontinuity is only caused by punctuation which is attached at sentence level and intervenes between the quotation and the verb of saying on which it depends. With standard word order, the subject of the sentence (i.e. the NP denoting the person or organization quoted) also intervenes. There is a third, much less frequent discontinuous construction used for quotations, where the sentence starts with the verb of saying. However, such sentences are treated as instances of discontinuous dependency (*ICH*) in the Penn Treebank annotation and are thus dealt with in Section 5.1.5.

A fronted quotation is scored as fully recognized if it is parsed as one constituent and attached to the same VP node as the verb of saying. Examples of fully recognized fronted quotations are given in Figures 5.8 and 5.9. Partial credit is given if a fronted phrase is recognized and correctly attached but contains only part of the actual quotation ("fronted phrase too small") as exemplified in Figure 5.10. As an exception, attaching

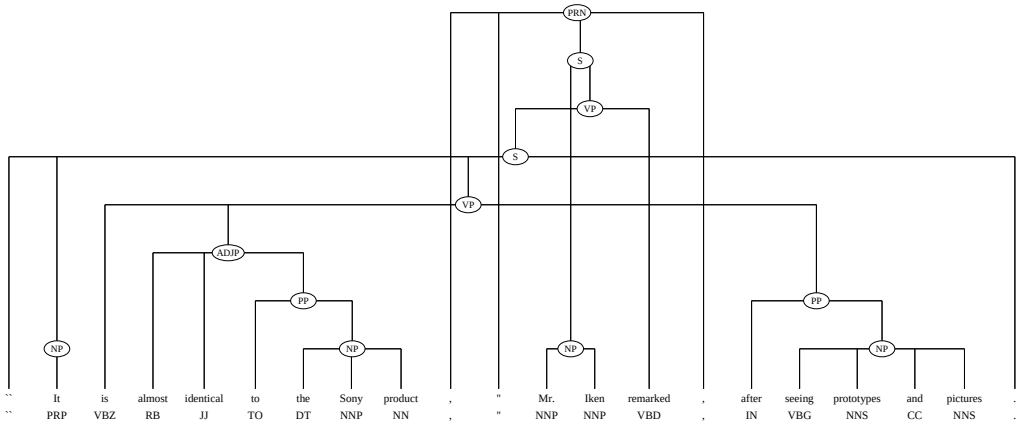


Figure 5.11.: Incorrect parse: a circumpositioned dependent is recognized instead of a fronted one.

a sentence-initial conjunction to the fronted quotation when the gold standard attaches it to the matrix sentence, or vice versa, is not penalized. Like *wh*-movement, fronted quotations turn out to pose no major problem to the parser, presumably due to their high frequency and characteristic structure. 129 out of 142 instances are fully recognized, another seven with some material missing from the fronted quotation. The six unrecognized instances can be said to involve relatively complex and rare constructions such as coordination within the fronted quotation or comma-separated enumerations, but mostly resist attempts to find a single obvious source of error. Worth highlighting are two cases, one of which is shown in Figure 5.11, where a postverbal adjunct is analyzed as belonging to the quotation, which is thus incorrectly parsed as a circumpositioned dependent (cf. Section 5.1.4 below).

In the last example, two paired quotation marks are parsed each as an opening quotation mark. This illustrates that with the possibility for quotation marks to attach either inside or outside the constituent they surround, there is no mechanism by which the PLCFRS model could reliably match pairs of quotation marks since both may well be introduced by separate rules in the correct parse. As a result, the parser profits from the quotation marks only to a limited degree to recognize the structure of a sentence and may even be confused as in the present example. The asymmetric attachment of quotation marks is to a great extent due to the decision of the Penn Treebank annotation guidelines to put them "at the very bottom of the pecking order" (Bies et al., 1995, Section 3.1.1) among punctuation marks, in combination with the requirement for context-freeness.

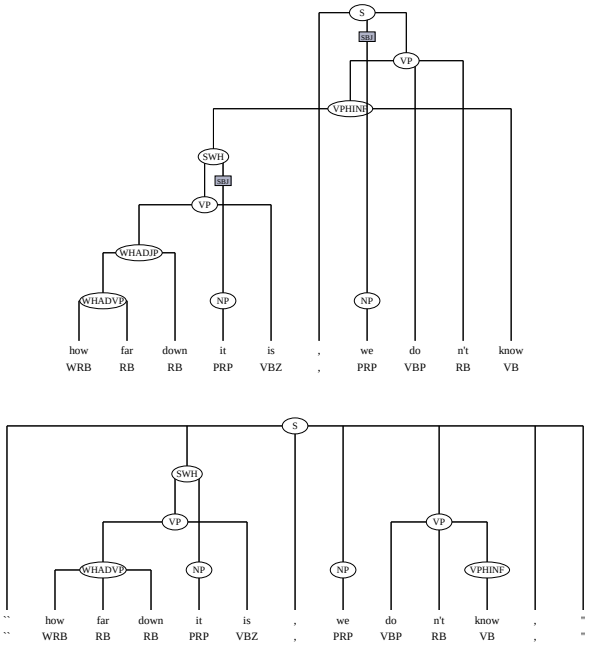


Figure 5.12.: Gold standard and parse: attachment of a fronted constituent at the VP level in the gold standard, but at the sentence level by the parser (1 of 5 instances)

In particular, commas and periods are attached according to the same rules whether or not they occur within a quotation, then quotation marks are attached wherever context-freeness permits. A different annotation scheme that always makes pairs of quotation marks siblings, possibly even using discontinuity for doing so, might be beneficial for parsing accuracy, particularly for texts like in the WSJ treebank where quotations are rather frequent.

5.1.3. Other fronted dependents

Outside of quotations, discontinuities arising through fronting are so rare that there are only six instances in the test set, none of which is analyzed as mandated by the gold standard. A problem here is that every fronted dependent (roughly, an argument or adjunct that appears before the subject and before the inflected verb of a sentence) carries a potential attachment ambiguity, viz. whether to attach it at the sentence level or at the VP level. In the gold standard, the latter is done only for arguments and for adjuncts whose interpretation location is in an embedded clause, other adjuncts are attached at

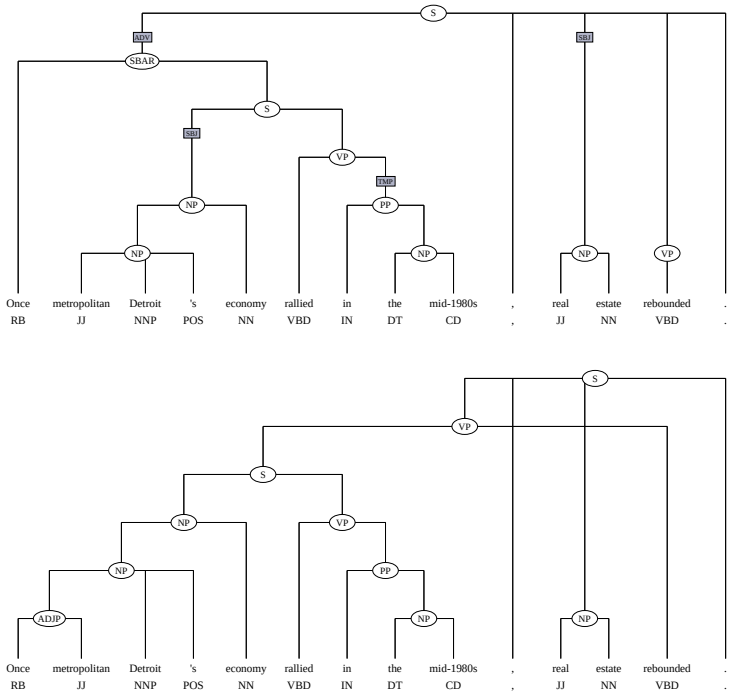


Figure 5.13.: Gold standard and parse: attachment of a fronted constituent at the sentence level in the gold standard, but at the VP level by the parser (the only instance)

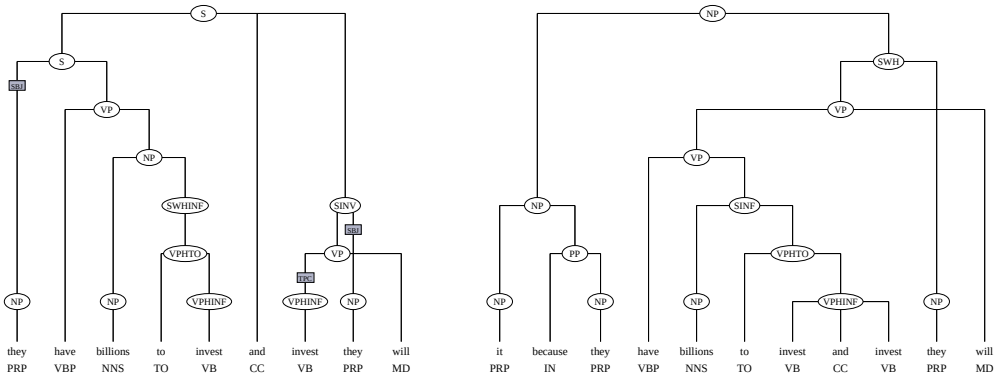


Figure 5.14.: Gold standard and parse: the fronted VP is recognized, but too big in the parse.

sentence-level (cf. Bies et al., 1995, Section 1.3.3). In the original context-free annotation, this criterion only decides whether a *T* null element is introduced to specify an interpretation location for the fronted dependent, its attachment is always at sentence level to maintain context-freeness. Thus, in context-free parsing, the attachment ambiguity does not arise. However, in the transformed treebank, the distinction, which is very hard to make from a sequence of part-of-speech tags alone, decides between a context-free analysis and a discontinuous analysis. It is therefore not surprising that the parser often fails to make the distinction correctly and chooses continuous attachment at sentence level in five of the six cases (see e.g. Figure 5.12) where the gold standard specifies discontinuous fronting.

The choice between sentence level and VP level is presumably heavily influenced by the category of the fronted dependent through the distribution of occurrences of rules which produce fronted dependents in the training data: fronted phrases of category S are almost always quotations and arguments of the inflected verb, and correctly recognized as such in most cases, as the previous section shows. Moreover, in the one case where the parser attaches a fronted dependent at the VP level where the gold standard mandates the S level (Figure 5.13), the fronted dependent is also (incorrectly) parsed as of category S. (This is a false positive, see Section 5.1.7 below.) Other categories such as SWH, ADVP or NP are much less frequent as fronted dependents attached at VP level, giving the parser a preference for attaching them at sentence level, insofar as five instances can count as evidence. Figure 5.14 shows the only instance of a fronted constituent correctly attached at VP level by the parser – here, the category is VP. However, what the parser recognizes as the fronted VP is a too big phrase, so this instance receives only partial credit.

5.1.4. Circumpositioned dependents

Circumpositioned dependents (cf. Section 3.3.2) are mostly quotations. These are like fronted quotations, except that only part of the quotation appears at the left periphery of the matrix clause, and the rest appears at the right periphery. As a legacy of the original context-free annotation, their annotation in the gold standard also involves a PRN node directly dominating the root of the matrix clause. However, whether this PRN node is present or not is disregarded in the manual evaluation. A circumpositioned dependent is counted as fully recognized whenever there is a discontinuous constituent made up of exactly the tokens of the circumpositioned dependent, both the left and the right part,

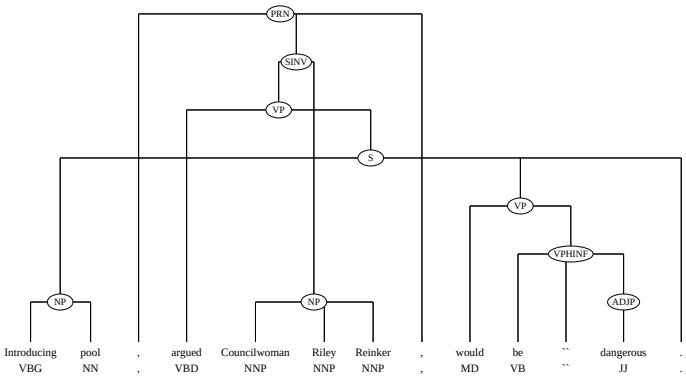


Figure 5.15.: Correct parse: fully recognized circumpositioned quotation

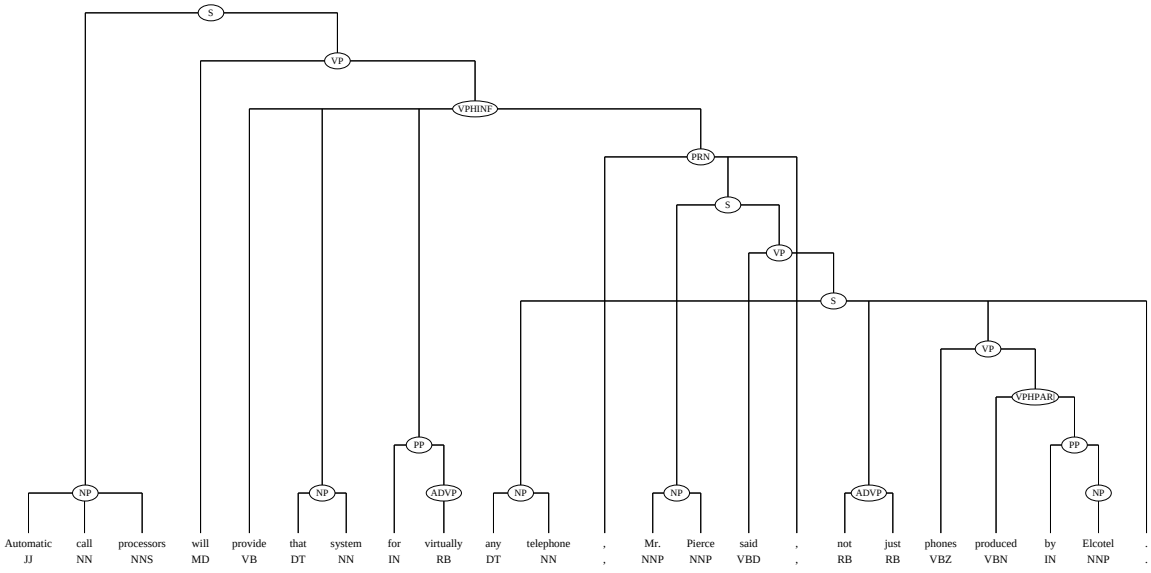


Figure 5.16.: Incorrect parse: a circumpositioned quotation is recognized, but too small.

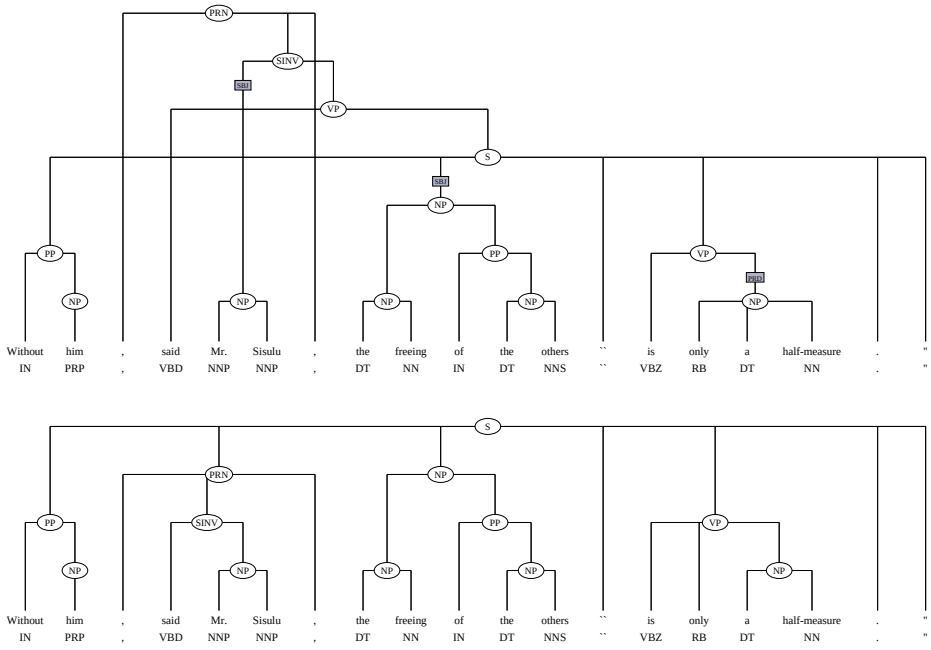


Figure 5.17.: Gold standard and parse: the circumpositioned quotation is not recognized, presumably due to the rare S_2 rule.

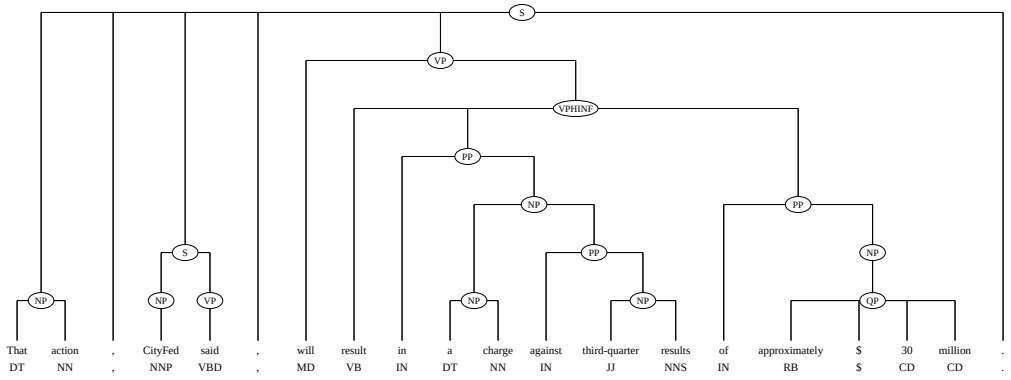


Figure 5.18.: Incorrect parse: circumpositioned quotation where the matrix clause is incorrectly recognized as a sentential adjunct

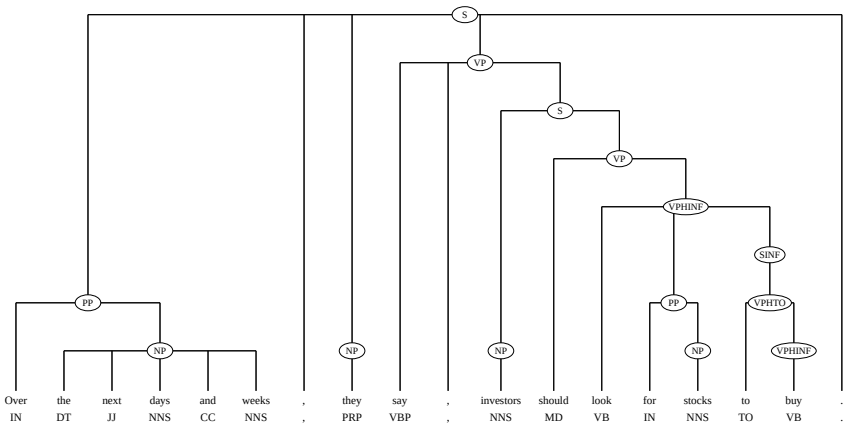


Figure 5.19.: Incorrect parse: unrecognized circumpositioned quotation whose left part consists of only a PP, interpreted by the parser as an adjunct to the matrix verb

and it is attached in the same VP as the matrix verb. For a relatively rare phenomenon (31 instances in the test set), a fair share (22 instances) was fully recognized, an example is given in Figure 5.15. Partial credit is given in the one instance where peripheral material is missing from the discontinuous constituent (Figure 5.16).

The two commas that almost invariably separate the central part of the matrix clause from the circumpositioned dependent seem to give a good clue to the parser as to when this construction occurs. In terms of LCFRS rules, clauses with a circumpositioned dependent can be analyzed rather uniformly as being of category PRN with the top rule $\mathrm{PRN}(XUYVZ)_1 \rightarrow {,}_1(U)\mathrm{S}_3(X,Y,Z){,}_1(V)$, with a few variations such as SINV instead of S, or with quotation marks added. Further rules typically involved are $\mathrm{S}_3(X,UY,Z) \rightarrow \mathrm{NP}_1(U)\mathrm{VP}_3(X,Y,Z)$, $\mathrm{VP}_3(X,Y,Z) \rightarrow \mathrm{VBD}_1(Y)\mathrm{S}_2(X,Z)$ and a few variations concerning, for example, the verb form or the category of the embedded sentence. With constructions with circumpositioned dependents almost always following this simple structure and non-terminals with fan-out 3[2] appearing almost exclusively in such constructions, there is not much here that could confuse the parser.

What is problematic and might have caused some of the eight cases of unrecognized circumpositioned dependents is the expansion of S_2 to the circumpositioned sentence itself. Due to punctuation and adjuncts attached at sentence level, there are many possible expansions for root nodes of sentences. Since the frequent S_1 and the rare S_2

[2]corresponding to nodes with gap degree 2, cf. Section 3.6

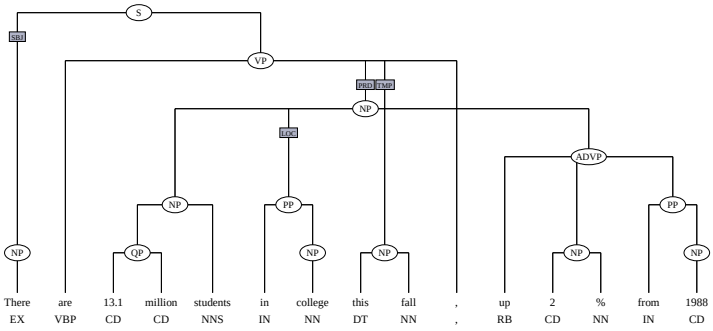


Figure 5.20.: Gold standard: discontinuous dependency within a VP

(and likewise, the moderately frequent SINV_1 and the rare SINV_2) are different non-terminals, training data for the expansions of the former cannot help with expansions for the latter. The diversification and resultant sparseness of S_2 rules is increased further by the fact that circumpositioned sentences can be separated rather freely, e.g. between initial conjunctions/adjuncts and the subject, after the subject, within the VP and so on, each possible separation requiring a different rule. Thus, it is not surprising that some circumpositioned quotations whose correct parse would require a more unusual S_2 rule, such as the one shown in Figure 5.17, are missed. Other shapes that missed circumpositioning can take are the interpretation of the matrix sentence as an adjunct in the actually embedded sentence (Figure 5.18) and the interpretation of the left part as an adjunct in the matrix sentence, leaving only the right part as an embedded sentence (Figure 5.19).

The sparseness problem for S_2 rules is an instance of what is arguably a general weakness of the PLCFRS model: what is the same category in the syntactic analysis is separated into different categories depending on how many gaps there are in the yield. One possibility to address this problem is to refine the probabilistic model so that it distinguishes between *expansions* (RHS non-terminals and their order) and *separations* (number of gaps and where they occur) as two different degrees of freedom for one and the same category. A model in this spirit is proposed in Levy (2005, Section 4.8) and awaits further research.

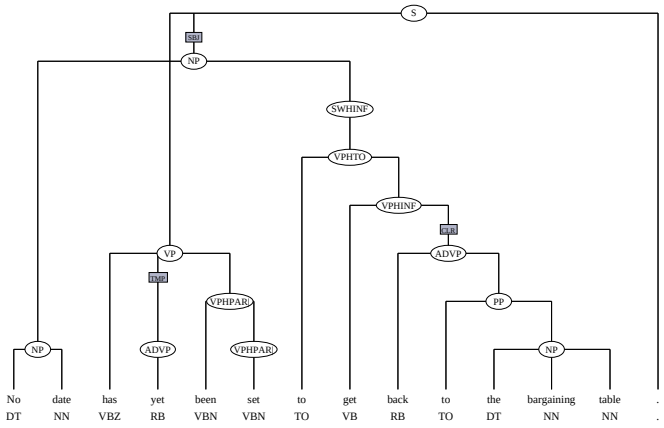


Figure 5.21.: Gold standard: right-dislocated clause depending on the subject

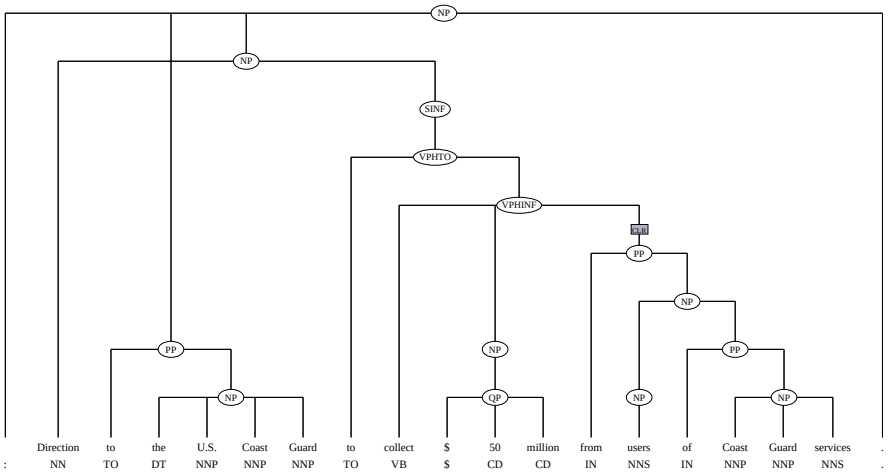


Figure 5.22.: Correct parse: two dependents on the same head; the closer one is adjoined at a higher level.

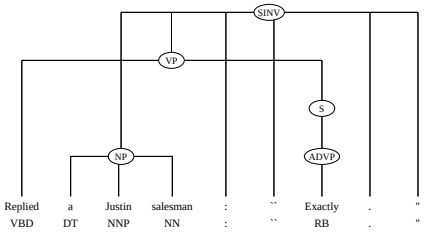


Figure 5.23.: Correct parse (though not conforming to gold standard, which appears to be inconsistent here) of a rare quotation construction where the sentence begins with the verb of saying

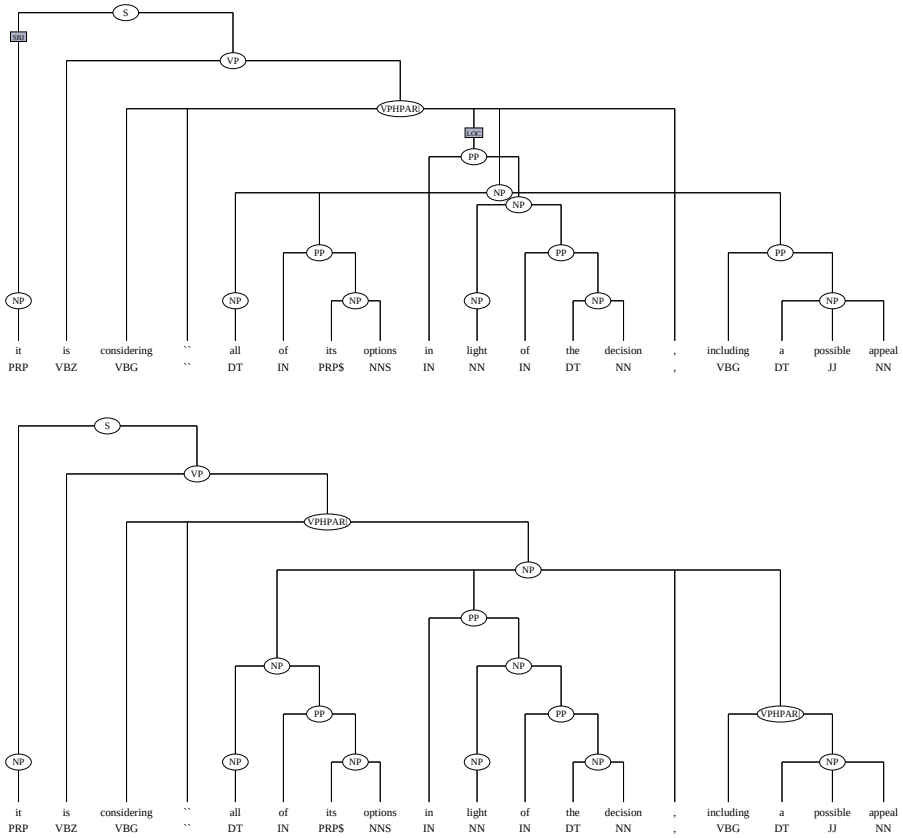


Figure 5.24.: Gold standard and parse: the NP *all of its options, including a possible appeal* is not recognized as discontinuous, an intervening PP is incorrectly attached within it.

5.1.5. Discontinuous dependency

"Discontinuous dependency" subsumes all those instances of discontinuity that arise through the transformation of coindexed *ICH* null elements as described in Section 3.3.3. A frequent case is that within a VP, a phrase a depending on the verb is followed by a phrase b also depending on the verb and then by a phrase c which depends on a or something within a. a and c then form a discontinuous constituent. Where this construction occurs in the WSJ corpus, b is often a temporal phrase while c often expresses a quantitative comparison. An example is shown in Figure 5.20. Other examples are right-dislocated modifiers to the subject (see e.g. Figure 5.21), cases of two dependents on the same head where the gold standard adjoins the closer dependent

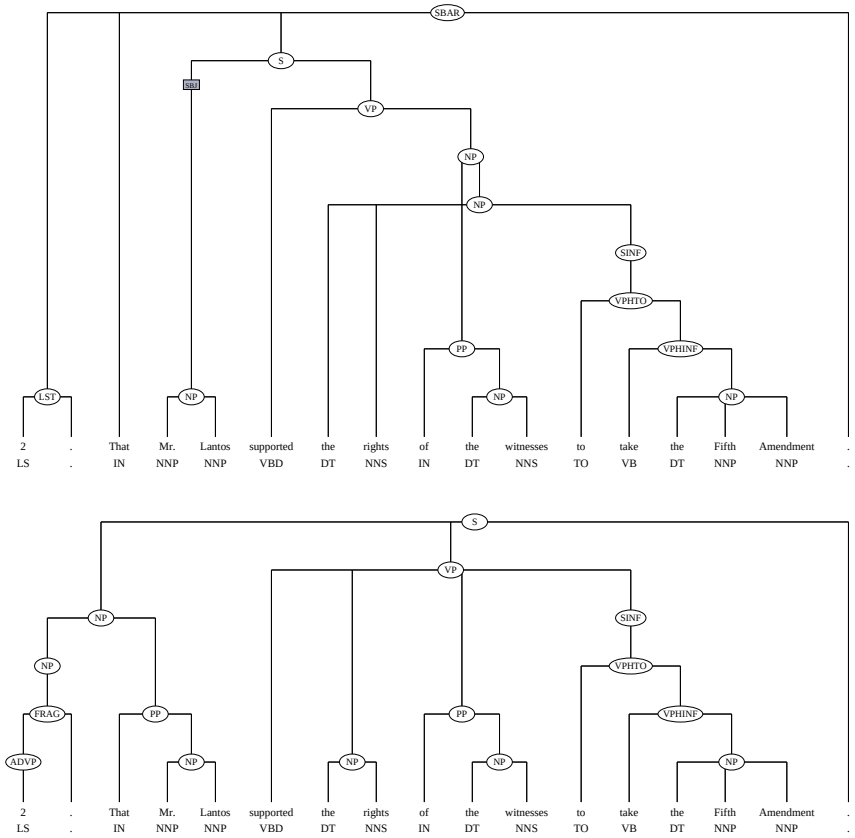


Figure 5.25.: Gold standard and parse: the NP *the rights* and the SINF *to take the Fifth Amendment* are not recognized as forming a discontinuous constituent together, but are at least attached at the same level.

higher than the further removed dependent (Figure 5.22), and right-dislocated quotations where the matrix sentence has an inverted word order (Figure 5.23).

Of the last two types, one instance of each was correctly recognized by the parser. In no other of the 30 cases of discontinuous dependency in the test set was a discontinuity recognized. There are varying degrees to which the parses still resemble the desired analysis. In three cases, the constituency of the discontinuous phrase is preserved at the price of incorrectly attaching the intervening material within the same constituent, as exemplified in Figure 5.24. In six cases, the left and right part are recognized as individual constituents and attached to the lowest node that properly dominates both the left part and the intervening material (Figure 5.25), roughly corresponding to a systematic transformation of the discontinuous tree to a context-free tree as described e.g.

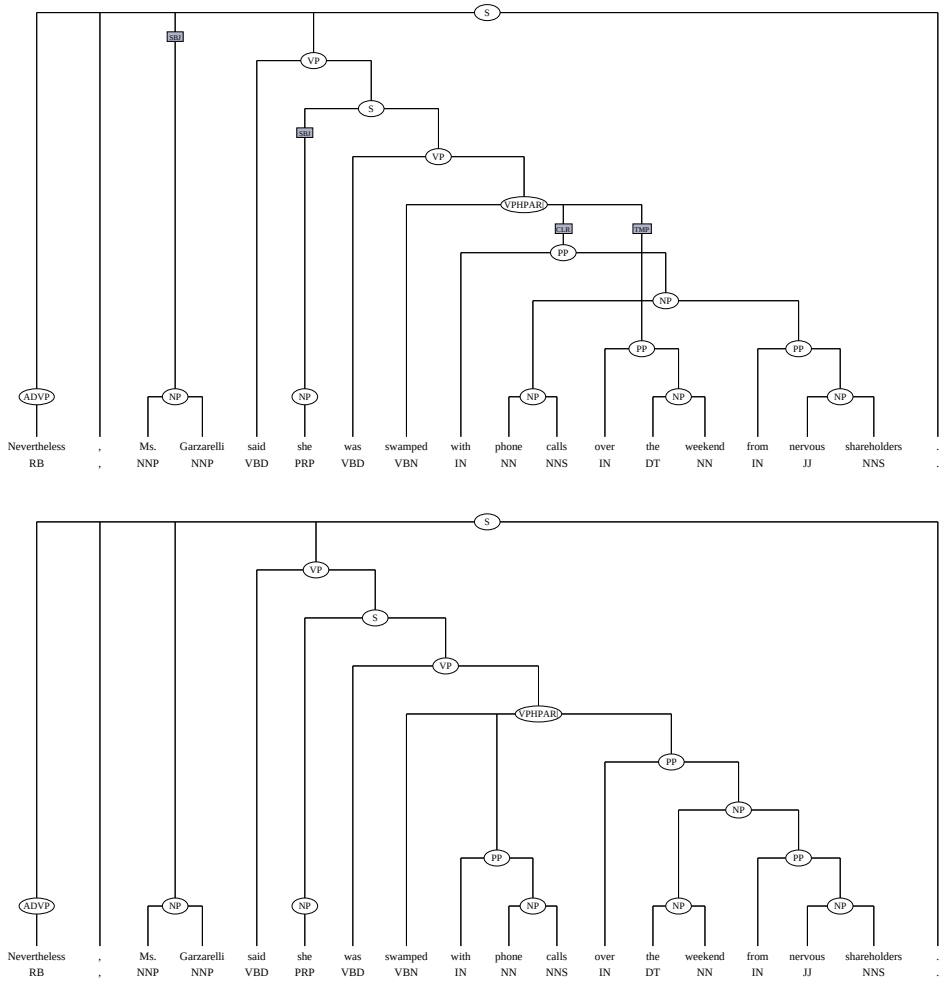


Figure 5.26.: Gold standard and parse: the NP *phone calls* and the PP *from nervous shareholders* are not recognized as forming a discontinuous constituent together. The latter is attached deeply within the intervening material.

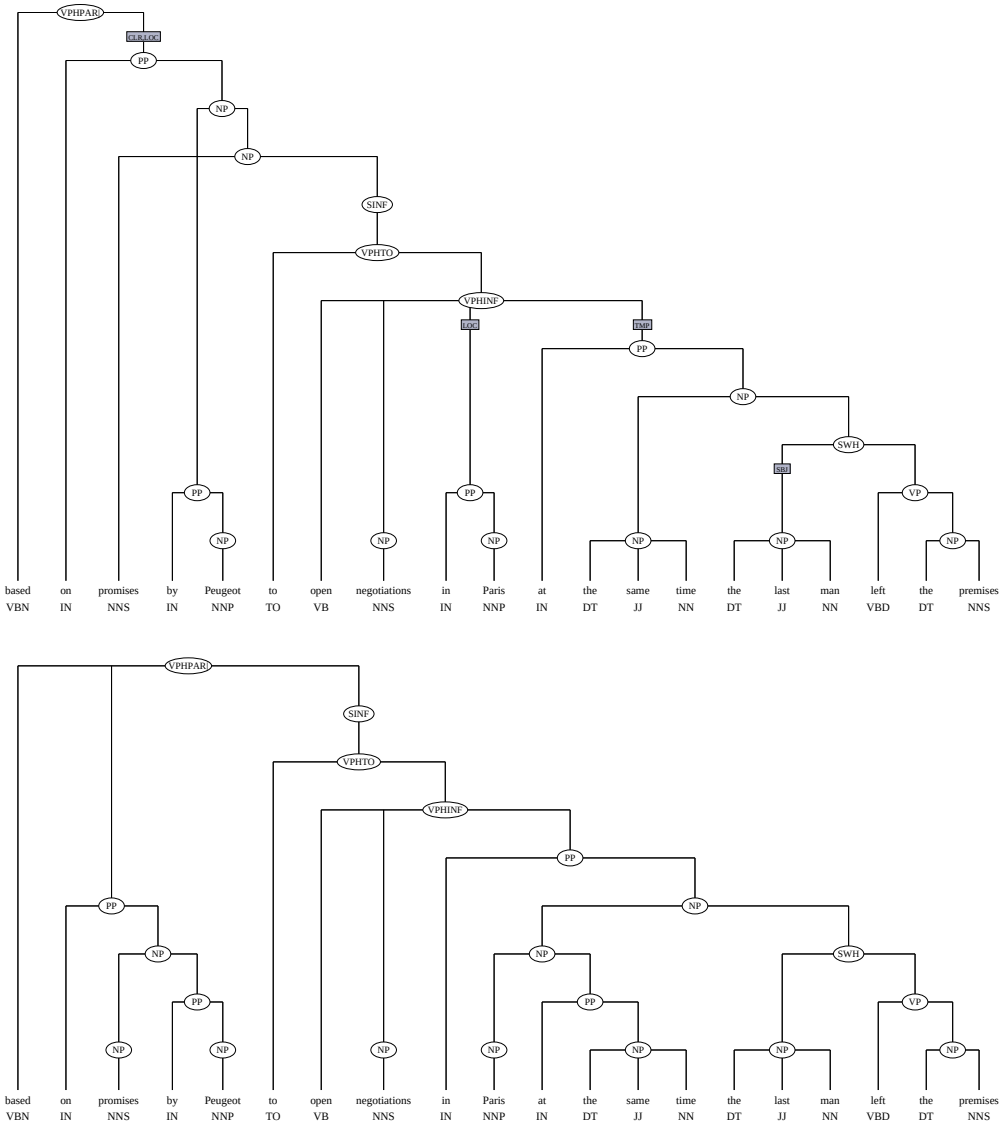


Figure 5.27.: Gold standard and parse: the NP *promises* and the SINF *to open negotiations in Paris at the same time the last man left the premises* are not recognized as forming a discontinuous constituent together. The latter is attached at a higher node in the tree than the former.

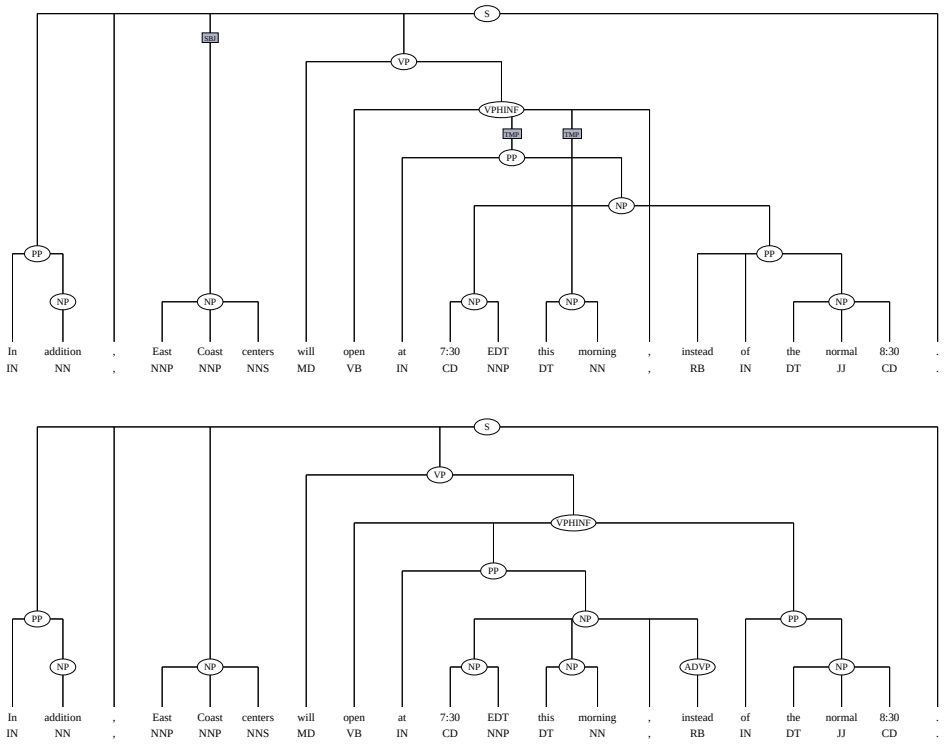


Figure 5.28.: Gold standard and parse: not only aren't the NP *7:30 EDT* and the PP *instead of the normal 8:30* recognized as forming a discontinuous constituent together, but the latter is not recognized as a constituent at all.

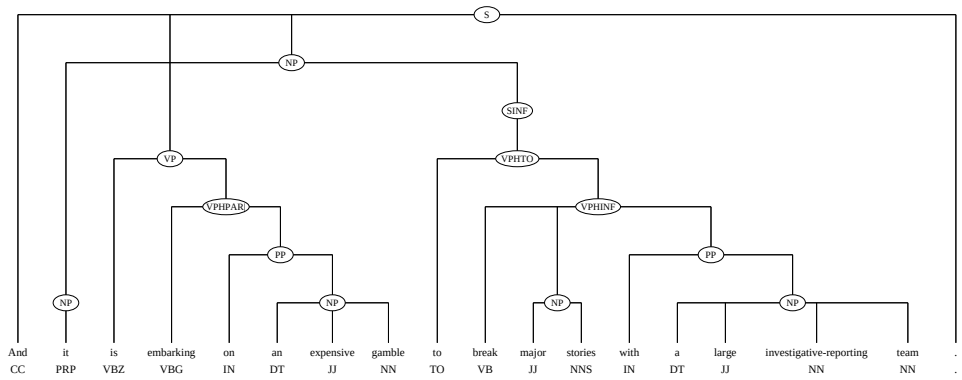


Figure 5.29.: Correct parse of a discontinuous constituent resulting from *it*-extraposition

in Kübler (2005). However, three of these parse trees are further flawed by intervening material attaching within the left or the right part of the discontinuous constituent. In seven cases, the right part instead attaches at a node within the intervening material (Figure 5.26) and in three cases at a node higher in the tree (Figure 5.27). The remaining nine instances can only count as failures to recognize both the left and the right part as constituents (Figure 5.28).

The near-zero recognition rate on this class of discontinuity can be explained by the fact that it consists of rather infrequent and isolated constructions that do not have much in common in terms of LCFRS rules, thus suffering massively from sparse data. What they do have in common that it is almost always easy for the two parts of a discontinuous constituent to be interpreted as individual constituents and to attach locally, the latter due to attachment ambiguity, a well known problem also in context-free parsing. The correct phrase for e.g. PPs as in Figure 5.26 to attach to often depends heavily on the particular preposition (here: *from*) and the possible head words it could modify (better *calls* than *weekend*). The present parsing model deduces the syntactic structure from a mere sequence of part-of-speech tags which does not contain such lexical information. Extending it to take some lexical clues into account might help to recognize this type of discontinuity.

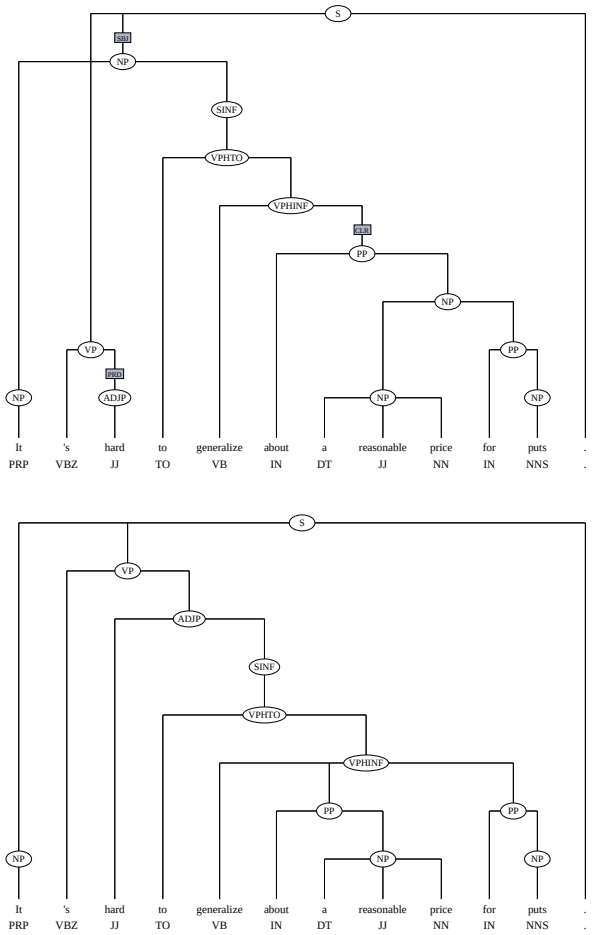


Figure 5.30.: Gold standard and parse: The NP *it* and the SINF *to generalize about a reasonable price for puts* are not recognized as forming a discontinuous constituent together. The latter is attached deeply in the intervening material.

5.1.6. it-extraposition

it-extraposition (cf. Section 3.3.4) is similarly infrequent and poorly recognized. Of the 14 such discontinuities in the test corpus, only two are correctly parsed. One correct parse is shown in Figure 5.29. The range of shapes the incorrect parses take is less wide than for discontinuous dependency in general: here, the right part or part of it attaches at a node within the intervening material in all cases (Figure 5.30).

5.1.7. False positives

This section discusses cases of discontinuities in the parse trees where the gold standard has no discontinuity. These instances cannot simply be classified as false positives of the six causes of discontinuities. What can be seen in the parse trees is which LCFRS rules were used to produce the discontinuities, and these can to some extent be traced back to certain types of discontinuities in the training corpus. Note however that not only are some rules involved in more than one type of discontinuity, it is also the case in all LCFRSs extracted by the algorithm described in Section 2.2 that each instance of discontinuity involves at least two rules specific to discontinuity, the two of which can stem from different types of discontinuity. This possibility of recombination is a potential source of error and nonsensical parses on one hand, on the other hand it can afford useful generalizations, just like the possibility for extracted rules to recombine in new ways in treebank grammars in general. Introducing some handy new terminology, the two rules minimally involved in any instance of discontinuity are:

1. A "mixing rule" where two variables that occur in different arguments of one RHS element occur as part of the same argument on the LHS. For example, the mixing rule of the discontinuity in Figure 5.1 is $\mathrm{SWH}_1(XYZ) \rightarrow \mathrm{NP}_1(Y)\mathrm{VP}_2(X, Z)$. A mixing rule is used to expand a node whose yield does not contain some gap that the yield of one of its children contains. Thus, looking at it top-down, a mixing rule "introduces" a gap into a parse tree while looking at it bottom-up, it "removes" it. The "mixing category" of a discontinuity is the LHS non-terminal of its mixing rule.

2. A "distributing rule" where two variables that occur in different arguments on the LHS occur in arguments of different elements of the RHS. For example, the distributing rule of the discontinuity in Figure 5.1 is $\mathrm{VPHPART}_2(X, Y) \rightarrow \mathrm{WHNP}_1(X)$ $\mathrm{VBN}_1(Y)$. A distributing rule is used to expand a node whose yield contains a gap

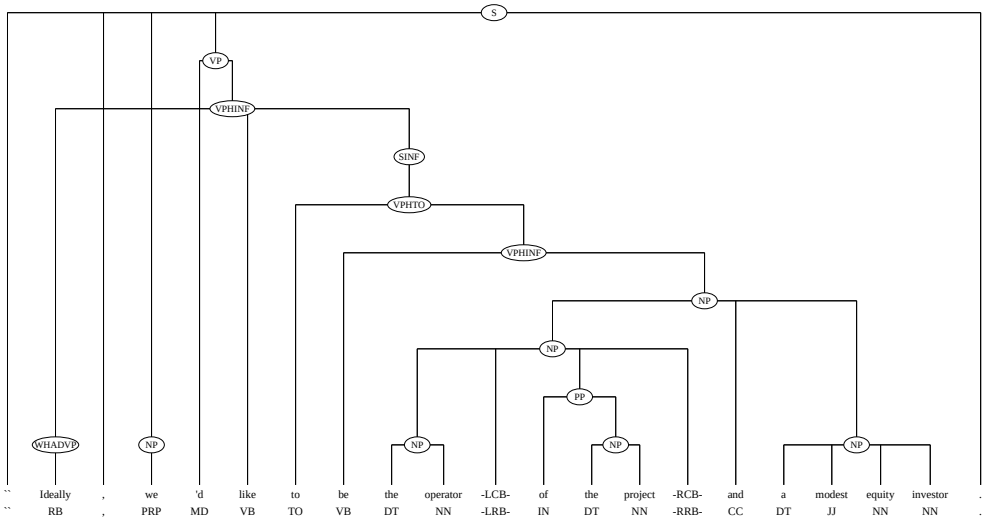


Figure 5.31.: Incorrectly recognized *wh*-movement. The rule $\text{WHADVP}_1(X) \rightarrow \text{RB}_1(X)$ exists due to tagging errors in the training corpus.

that none of the yields of its children contains. Thus, looking at it top-down, a distributing rule "removes" a gap from a parse tree while looking at it bottom-up, it "introduces" it. The "distributing category" of a discontinuity is the LHS non-terminal of its distributing rule.

Optionally, a number of "carrying rules" can be involved in an instance of discontinuity:

3. A "carrying rule" is a rule where two variables that occur in different arguments on the LHS occur in different arguments of the same RHS element. For example, the carrying rules of the discontinuity in Figure 5.1 are $\text{VP}_2(X, YZ) \rightarrow \text{MD}_1(Y)$ $\text{VPHINF}_2(X, Z)$, $\text{VPHINF}_2(X, YZ) \rightarrow \text{VB}_1(Y)$ $\text{VPHINF}_2(X, Z)$ and $\text{VPHINF}_2(X, YZ) \rightarrow \text{VB}_1(Y)$ $\text{VPHPART}_2(X, Z)$.

A useful way to classify false discontinuities is by looking at their distributing and mixing rules. Recall from Section 4.1.2 that in each rule one RHS element can be marked as the "head" of the rule, corresponding to the head child of a node in a tree that is expanded using this rule. For example, in the distributing rule of Figure 5.1 mentioned above, the head is $\text{VBN}_1(Y)$. It appears "right of the gap" since its only variable is on the right among the LHS arguments.

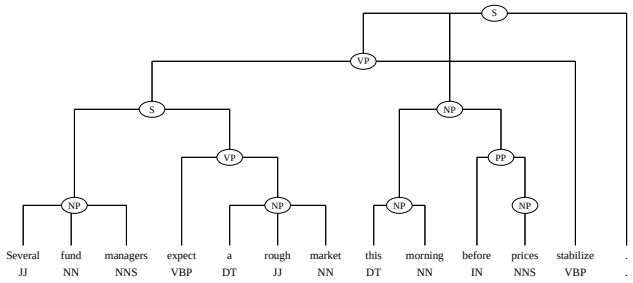


Figure 5.32.: Incorrectly recognized fronting

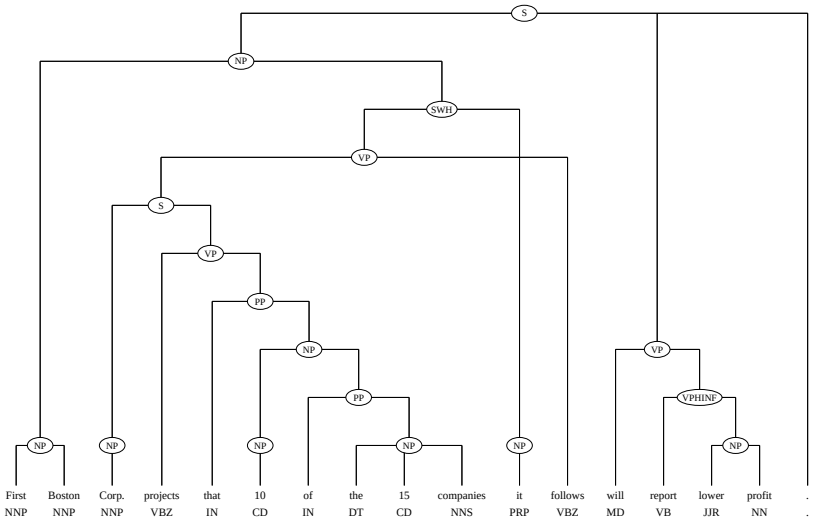


Figure 5.33.: Incorrectly recognized fronting with mixing category SWH

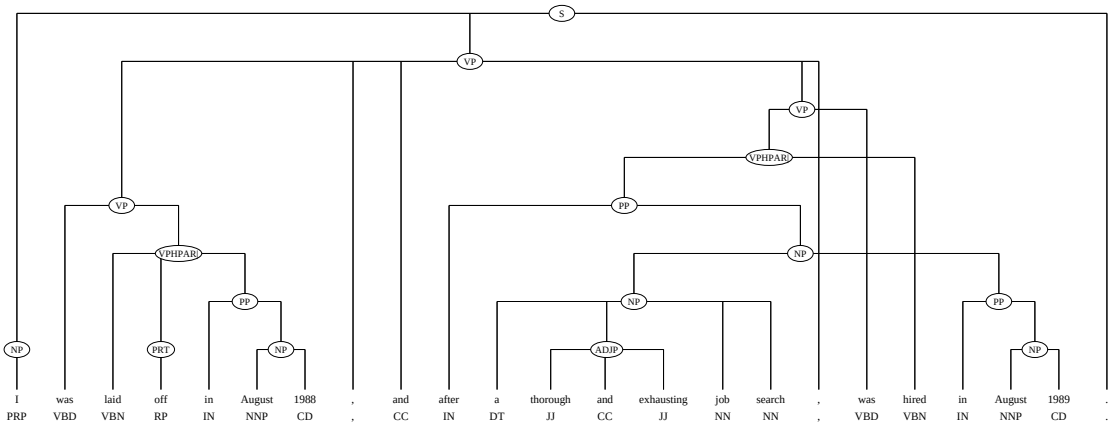


Figure 5.34.: Two entangled instances of incorrectly recognized fronting with VP mixing categories

Discontinuities arising through ***wh*-movement** are characterized by the fact that the head of the distributing rule is right of the gap and that a child with a WH category appears left of the gap. There are two false positives in the test corpus where both is the case, one of which is shown in Figure 5.31. Excluding cases of incorrectly recognized circumpositioned dependents (see Section 5.1.4 below), there are 25 further false positives where the head of the distributing rule is right of the gap. These can be classified as false positives of **fronting**. The vast majority (23) of them have a clausal category (i.e. starting with S) as mixing category (Figure 5.32), also a characteristic of actual fronting. In ten of these instances, the clausal category is SWH – here, mixing rules for *wh*-movement and distributing rules for fronting have been combined (Figure 5.33). Only in the remaining two instances is the mixing category VP, and it is actually the same mixing node shared by the two instances in the same sentence (and by one further instance of incorrectly recognized extraposition), shown in Figure 5.34.

The failure of the PP *in August 1989* in the last example to attach to the more plausible VPHPART node can be seen as another illustration for the artificial infrequency of rules due to the separation of non-terminals with the same syntactic category depending on fan-out, as discussed in Section 5.1.4. Here, a reason for the bad parse may be that the required rule $\mathrm{VPHPART}_2(X, YZ) \rightarrow \mathrm{PP}_1(X)\mathrm{VBN}_1(Y)\mathrm{PP}_1(Z)$ with its two prepositional phrases, one of which is fronted, is too infrequent to beat even an analysis involving the also rare construction involving a discontinuous PP containing a discontinuous NP as shown in the diagram. Had the parser been allowed to take the frequency of the continuous rule $\mathrm{VPHPART}_1(XYZ) \rightarrow \mathrm{VBN}_1(X)\mathrm{PP}_1(Y)\mathrm{PP}_1(Z)$ into account in expanding the discontinuous category $\mathrm{VPHPART}_2$, the probabilities might have been more representative of the training corpus and thus might have led to a better parse. Note, however, that since the first PP is fronted in the discontinuous and not fronted in the continuous rule, this example raises the additional point that the *order* of RHS non-terminals might have to be factored out of the expansion of a rule as well as the separation. This in turn touches upon the factorization afforded by horizontal Markovization (cf. Section 4.1.1), underscoring that the topic merits a great deal of thought – more than it can be given in the present work.

False positives of **discontinuous dependency** and **it-extraposition** cannot be distinguished from each other since rules that arise through the latter, such as $\mathrm{NP}_2(X, Y) \rightarrow \mathrm{NP}_1(X)\mathrm{SINF}_1(Y)$, appear also with other yields for the NP than *it*, in which case they represent instances of discontinuous dependency. Between them, the two types have 44 false positives, exactly as many as there are actual instances of them in the test corpus

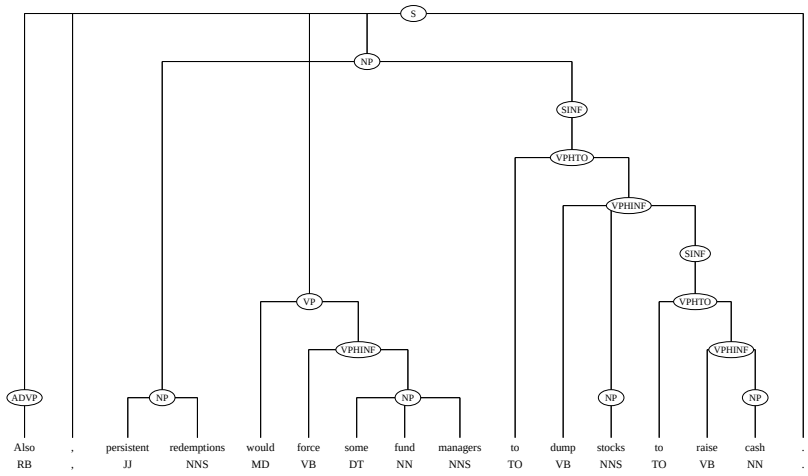


Figure 5.35.: Incorrectly recognized discontinuous dependency with mixing category S

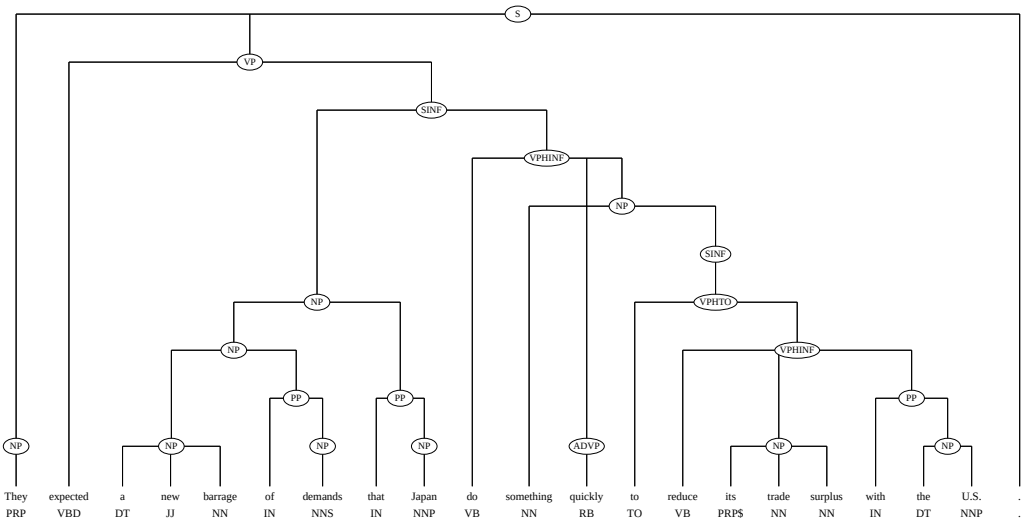


Figure 5.36.: Incorrectly recognized discontinuous dependency with mixing category VPHINF

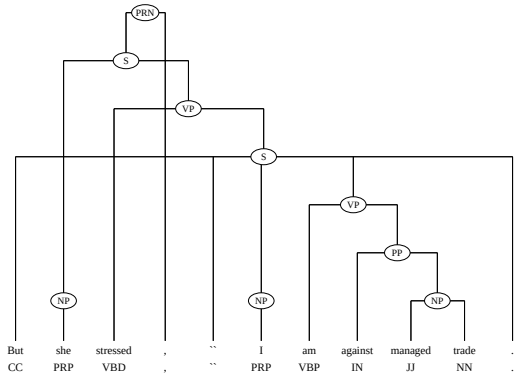


Figure 5.37.: An incorrectly recognized circumpositioned dependent, including a sentence-initial adjunct

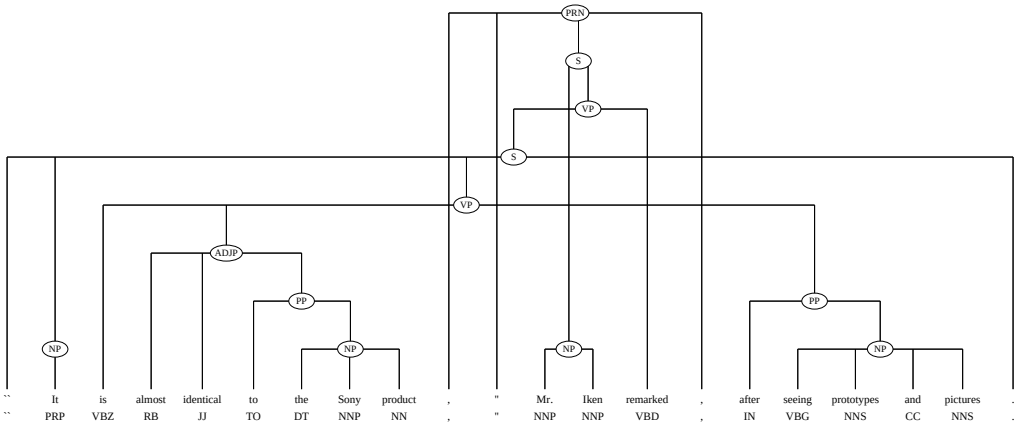


Figure 5.38.: An incorrectly recognized circumpositioned dependent, including a sentence-final adjunct

(almost none of which are correctly recognized, as seen in Sections 5.1.5 and 5.1.6). This further highlights the fact that at least with the current parsing model, instances and non-instances of these types of discontinuities are very hard to distinguish. As is characteristic for *it*-extraposition and also common for discontinuous dependency, the mixing category here is clausal in most (39) cases (Figure 5.35 shows an example), and as with fronting, some (5) have mixing category SWH. The remaining five instances have other mixing categories such as VPHINF (Figure 5.36).

Discontinuities arising through **circumpositioned dependents** are characterized by clausal distributing categories and the mixing category PRN. There are six false

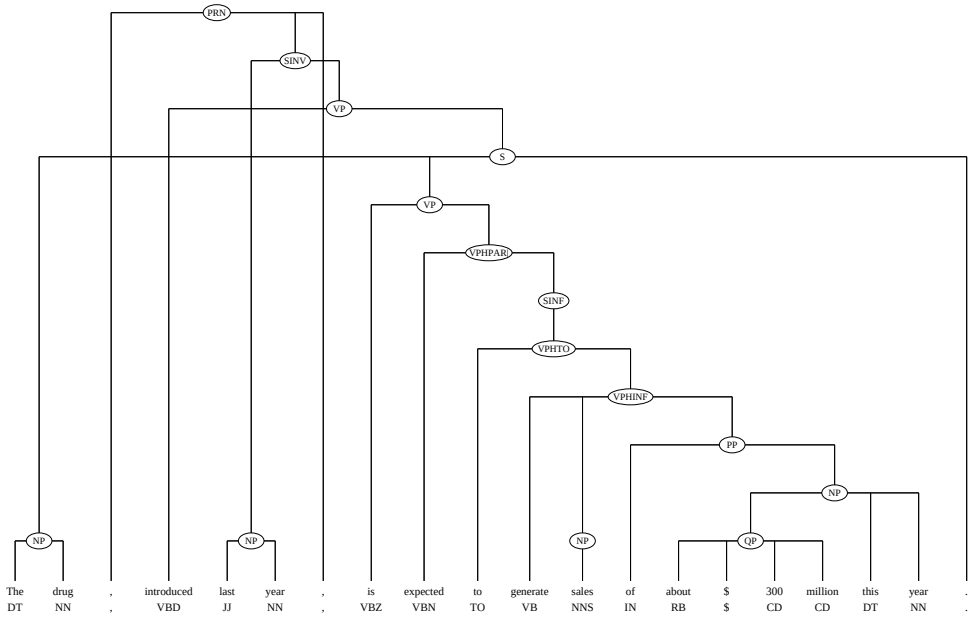


Figure 5.39.: An incorrectly recognized parenthetical matrix clause with circumpositioned dependent

positives, three where a sentence-initial adjunct is interpreted as part of a non-fronted quotation (Figure 5.37), two where a sentence-final adjunct is interpreted as part of a fronted quotation (Figure 5.38) and one where there is no quotation at all, but a VP adjoined to the subject and surrounded by commas is interpreted as a matrix sentence (Figure 5.39).

5.1.8. Summary and Conclusions

This section has analyzed parser performance in the discontinuous experiment specifically with respect to discontinuities, broken down by the types of grammatical constructions causing them. The analysis has shown that *wh*-movement, fronted quotations and circumpositioned dependents (mostly also quotations) are recognized reliably, with few false negatives and few false positives. Fronted dependents other than quotations, instances of discontinuous dependency and of *it*-extraposition, on the other hand, are recognized correctly only in a vanishingly low number of cases while the PLCFRS rules extracted from such constructions are involved in many incorrectly recognized instances of discontinuity.

The difference between the well recognized and the badly recognized constructions is to some degree one of frequency and thus of both available training data and a skewed test set, resulting in good odds for the parser to get it right by picking a frequent construction (as would be the case in comparable freely occurring text). A possibly even more important difference is in the presence of reliable clues in the parser input (a sequence of part-of-speech tags) for each type of construction. For *wh*-movement, such a clue would be a *wh*-word (recognizable by its tag) followed by a sentence. For fronted and circumpositioned quotations, combinations of the parts of two sentences (matrix and embedded clause) as well as certain punctuation patterns are characteristic. Discontinuous dependency and *it*-extraposition lack such reliable clues since they almost invariably give rise to attachment ambiguities that would probably require taking into account lexical clues, if not a deeper semantic analysis, to resolve reliably.

Apart from ignoring important lexical information, a problem with the parsing model used is that its probabilistic component treats as unrelated the different variants of one syntactic category with respect to the degree of discontinuity (as given by gap degree or fan-out), resulting in data sparseness especially for the higher-fan-out variants of categories, as discussed in Sections 5.1.4 and 5.1.7. Certain difficulties of the parser that can be attributed to this problem indicate a need for either vastly more training data or a factoring of rules to make the expansion of rules more independent from the different possible ways for them to be separated by discontinuity. Lastly, the way punctuation, especially quotation marks, is attached in the transformed treebank as a legacy of the original context-free annotation, proves to be problematic and could be improved (Section 5.1.1), possibly along with other properties of the annotation scheme that might have an impact on parser performance.

5.2. An experiment limited to the transformation of *T* and *RNR* dependencies

As discussed in Sections 5.1.5, 5.1.6 and 5.1.7, instances of discontinuous dependency and *it*-extraposition are almost never recognized, but LCFRS rules used in such constructions are involved in many cases of incorrectly recognized discontinuities, so presumably they doubly affect overall evaluation scores negatively. Resolving with reasonable reliability the attachment ambiguities underlying the recognition and nonrecognition of such

	discontinuous
million items produced	1056
labeled recall	77.40%
labeled precision	80.44%
labeled f-measure	78.89%
unlabeled recall	80.65%
unlabeled precision	83.83%
unlabeled f-measure	82.21%

Table 5.5.: EVALB-style evaluation of the discontinuous parsing experiment limited to transformation of *T* and *RNR* dependencies

	discontinuous		context-free	
T	$\frac{319}{439}$	73%	$\frac{137}{439}$	31%
T-PRN	$\frac{23}{32}$	72%	$\frac{8}{32}$	25%
RNR	$\frac{2}{4}$	50%	$\frac{2}{4}$	50%
other	$\frac{29973}{35964}$	83%	$\frac{29657}{35964}$	82%
total	$\frac{30317}{36439}$	83%	$\frac{29804}{36439}$	82%

Table 5.6.: Dependency evaluation of 1) the parser output in the limited discontinuous experiment and 2) the parser output in the unchanged context-free experiment against the limited discontinuous gold standard

discontinuities would require the inclusion of lexical clues into the parsing model. For the present model, it thus seems wise to exclude both types from the transformation (done simply by not considering *ICH* and *EXP* null elements) and content oneself with invariably local attachment as in the original context-free annotation. The discontinuous parsing experiment was repeated with the so modified training data, and the evaluation of both the new discontinuous and the unchanged context-free experiment was repeated with the so modified gold standard. The results are shown in Tables 5.5 and 5.6. While the overall EVALB scores for the discontinuous experiment and the dependency evaluation of the context-free experiment are affected only marginally, the discontinuous experiment without *ICH* and *EXP* transformation sees a strong rise in attachment score for dependencies of the *T* type, a slight decrease for those of the *T*-PRN type and, again, only marginal overall change.

Chapter 6.

Conclusion and Outlook

This thesis presents first results for large-scale data-driven discontinuous constituent parsing of English text. To this end, the PLCFRS parsing methodology presented by Maier and Kallmeyer (2010) has been transferred from parsing German to parsing English. For training and testing, the Wall Street Journal part of the Penn Treebank has been used after being converted to a format where certain types of non-local dependencies are represented using discontinuous constituents rather than null elements and coindexation, making them accessible to the data-driven constituent parser. The method for converting the treebank has been developed and discussed in detail. A detailed analysis of parsing performance on specific linguistic phenomena involving discontinuities has been provided.

The results for discontinuous constituent parsing of English are promising. While the accuracy figures lag behind the state of the art for English, this does not call the viability of discontinuous constituent parsing into question. When trained on a context-free treebank, the parser used is at its core a PCFG parser, just like state-of-the-art context-free parsers (Klein and Manning, 2003b; Petrov et al., 2006). The context-free parsing experiments conducted show about the same lack in accuracy as the discontinuous experiments, reflecting the ample room for improvement left by techniques not yet implemented, viz. refinements of the algorithms for parsing and for grammar annotation that are used in state-of-the-art parsing. Since PLCFRS is a generalization of PCFG, prospects are good that the applicability and effects of such optimizations will continue to carry over from the context-free to the discontinuous case.

Especially promising in the direction of grammar annotation are category splits. A small set of them was shown to already have a strong beneficial effect on parsing accuracy. This could be exploited further by either adding more linguistically motivated

splits manually in the fashion of Klein and Manning (2003b) or automatically in the fashion of Ule (2003) or Petrov et al. (2006).

No improvement in accuracy could be elicited using the limited form of Markovization tried out in this work. However, full Markovization and a range of further parameters remain to be explored, e.g. making the use of horizontal Markovization depend on the parent category as successfully done by Kübler (2005) on the German treebank TüBa-D/Z.

Concerning parsing performance specifically on discontinuous constituents, *wh*-movement, fronted and circumpositioned quotations are recognized well in the current parsing model while performance on discontinuous dependency and *it*-extraposition is poor and fronting outside of quotations is too rare allow for a substantiated assessment. Specific problems caused by the presence of discontinuity were not found. However, difficulties well known in PCFG parsing appear in their old guises in PLCFRS parsing, and additionally in some new guises that involve discontinuous constituents. As Petrov et al. (2006) put it, independence assumptions made by probabilistic grammars tend to be "too strong in some places (...) and too weak in others". As a possible example of too strong independence assumptions involving discontinuous constituents, consider the case of discontinuous rules from different constructions that sometimes combine to produce more or less nonsensical analyses because the current grammar model does not capture connections of certain "mixing rules" with specific "distributing rules" (Section 5.1.7). As a possible example of too weak independence assumptions (resulting in data sparseness), consider the complete separation of syntactic categories and the rules expanding them based on fan-out, which could be addressed by factoring rules so that expansions and their possible separations into two or more ranges are determined separately (Section 5.1.4). In the case of discontinuous dependency, the main problem appears to be massive attachment ambiguity that might be mitigated by making lexical information available to the parser.

The experiments have shown that parsing with the grammar extracted from the discontinuous treebank is significantly more complex in terms of items produced, and therefore more time-consuming, than context-free parsing. An important factor for parsing complexity is the fan-out of the grammar which is 1 in context-free parsing and has been 3 in all discontinuous parsing experiments presented. However, a fan-out this high has been found to be almost exclusively due to the way punctuation is attached in circumpositioned quotations, suggesting that 2-LCFRS is sufficient for English and requires only small modifications to the annotation scheme. Further structural simpli-

fications to the annotation scheme may also prove beneficial since certain linguistically motivated distinctions appear to complicate the parsing task far more than they are relevant in subsequent processing, e.g. attaching fronted adjuncts at sentence vs. VP level (Section 5.1.3) or nesting NP adjuncts in a specific order (Section 5.1.5). Parsing accuracy could also gain from a harmonization of the annotation scheme with respect to fine points such as the attachment of quotation marks, as discussed in Section 5.1.1.

A final topic for further work that should be mentioned here concerns the relation between discontinuous constituent parsing and other methods to the end of including non-local dependencies in syntactic representation, such as those mentioned in Section 1.3. Even discontinuous trees cannot represent all types of non-local dependencies, thus some additional processing will remain necessary in order to recover e.g. control relations or attachment to multiple interpretation locations. Further research might be able to show how to adapt algorithms for doing so from context-free to discontinuous input trees and whether the additional information encoded in the discontinuous trees also helps with the recovery of those other dependencies.

Appendix A.

Corrections to the Penn Treebank annotation

During the development of the algorithm described in Chapter 3, certain annotation errors and inconsistencies in the Penn Treebank (Release 2, Wallstreet Journal corpus) showed up by leading to errors in applying the algorithm or in sanity checks carried out on the transformed treebank. This appendix contains the full list of manual corrections made to the treebank before the final run of the algorithm which produced the transformed treebank used in the experiments.

Sentence	Problematic element	Solution
575	* coindexed with root	changed it to *T*
2512	*T* without referent	removed *T* and placeholder ADVP
5659	*T* coindexed with S	coindexed it with lower S
6323	*T* coindexed with SINV	coindexed it with lower S
6580	VP embedding quotation	reannotated complete VP
7757	* coindexed with root	changed it to *T*
11680	missing PRN	added it as parent of SINV
15577	*T* coindexed with S	coindexed it with lower S
15755	*T* coindexed with SINV	coindexed it with lower S
15770	*T* coindexed with SINV	coindexed it with lower S
15983	* with wrong index	coindexed it with surface subject NP
21763	*ICH* missing referent	coindexed it with last SBAR
23986	*T* missing referent	coindexed it with WHADVP
26789	missing PRN	added it, embedding parenthetical S

40235	missing PRN and SINV	added them, embedding parenthetical VP
40356	missing PRN	added it, embedding parenthetical S
41561	*T* coindexed with with SBAR	coindexed it with WHADVP
48536	*T* coindexed with top S	coindexed it with new topicalized S node
48618	ADVP with *T* attached too high	attached it in lowest VP

Bibliography

Bies, A., Ferguson, M., Katz, K., and MacIntyre, R. (1995). Bracketing Guidelines for Treebank II Style Penn Treebank Project. Technical report, University of Pennsylvania.

Boullier, P. (1998). A generalization of mildly context-sensitive formalisms. In *Proceedings of the Fourth International Workshop on Tree-Adjoining Grammars and Related Formalisms (TAG+4)*, pages 17–20, Philadelphia, USA.

Brants, S., Dipper, S., Hansen, S., Lezius, W., and Smith, G. (2002). The TIGER treebank. In *Proceedings of the First Workshop on Treebanks and Linguistic Theories (TLT2002)*, pages 24–41, Sozopol, Bulgaria.

Bresnan, J., Kaplan, R. M., Peters, S., and Zaenen, A. (1982). Cross-serial dependencies in Dutch. *Linguistic Inquiry*, 13(4):613–635.

Cahill, A., Burke, M., O'Donovan, R., Van Genabith, J., and Way, A. (2004). Long-distance dependency resolution in automatically acquired wide-coverage PCFG-based LFG approximations. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 319–326, Barcelona, Spain.

Campbell, R. (2004). Using linguistic principles to recover empty categories. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 645–652, Barcelona, Spain.

Charniak, E. (2000). A Maximum-Entropy-inspired parser. In *1st Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 132–139.

Collins, M. (1999). *Head-Driven Statistical Models for Natural Language Parsing*. PhD thesis, University of Pennsylvania.

de Marneffe, M.-C., MacCartney, B., and Manning, C. D. (2006). Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC 2006*.

Dienes, P. and Dubey, A. (2003). Deep syntactic processing by combining shallow methods. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 431–438, Sapporo, Japan. Association for Computational Linguistics.

Gabbard, R., Kulick, S., and Marcus, M. (2006). Fully parsing the Penn Treebank. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 184–191, New York City, USA. Association for Computational Linguistics.

Gómez-Rodríguez, C., Kuhlmann, M., Satta, G., and Weir, D. (2009). Optimal reduction of rule length in Linear Context-Free Rewriting Systems. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 539–547, Boulder, Colorado, USA. Association for Computational Linguistics.

Hockenmaier, J. (2003). *Data and models for statistical parsing with Combinatory Categorial Grammar*. PhD thesis, School of Informatics, University of Edinburgh.

Jijkoun, V. and de Rijke, M. (2004). Enriching the output of a parser using memory-based learning. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 311–318, Barcelona, Spain.

Johnson, M. (2002). A simple pattern-matching algorithm for recovering empty nodes and their antecedents. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 136–143, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Joshi, A. K. (1985). Tree adjoining grammars: How much context-sensitivity is required to provide reasonable structural descriptions? In Dowty, D., Karttunen, L., and Zwicky, A., editors, *Natural Language Parsing*, pages 206–250. Cambridge University Press.

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

Kallmeyer, L. and Maier, W. (2010). Data-driven parsing with probabilistic linear context-free rewriting systems. In *Proceedings of the 23rd International Conference*

on Computational Linguistics (Coling 2010), pages 537–545, Beijing, China. Coling 2010 Organizing Committee.

Klein, D. and Manning, C. D. (2003a). A* parsing: Fast exact Viterbi parse selection. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, Edmonton, Alberta, Canada.

Klein, D. and Manning, C. D. (2003b). Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 423–430, Sapporo, Japan. Association for Computational Linguistics.

Kübler, S. (2005). How do treebank annotation schemes influence parsing results? or how not to compare apples and oranges. In *Proceedings of RANLP 2005*, Borovets, Bulgaria.

Levy, R. (2005). *Probabilistic models of word order and syntactic discontinuity*. PhD thesis, Stanford University.

Levy, R. and Manning, C. (2004). Deep dependencies from context-free statistical parsers: Correcting the surface dependency approximation. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 327–334, Barcelona, Spain.

Lezius, W. (2002). *Ein Suchwerkzeug für syntaktisch annotierte Textkorpora*. PhD thesis, IMS, University of Stuttgart.

Lin, D. (1995). A dependency-based method for evaluating broad-coverage parsers. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI 1995)*, Montreal, Quebec, Canada.

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, Los Angeles, California, USA. Association for Computational Linguistics.

Maier, W. and Kallmeyer, L. (2010). Discontinuity and non-projectivity: Using mildly context-sensitive formalisms for data-driven parsing. In *Proceedings of the Tenth International Conference on Tree Adjoining Grammars and Related Formalisms (TAG+10)*, pages 119–126, New Haven, Connecticut, USA.

Maier, W. and Lichte, T. (2009). Characterizing discontinuity in constituent treebanks. In *Proceedings of the 14th Conference on Formal Grammar (FG-2009)*, Bordeaux, France.

Maier, W. and Søgaard, A. (2008). Treebanks and mild context-sensitivity. In de Groote, P., editor, *Proceedings of the 13th Conference on Formal Grammar (FG-2008)*, pages 61–76, Hamburg, Germany. CSLI Publications.

Marcus, M. P., Santorini, B., and Marcinkiewicz, M. A. (1993). Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19:313–330.

McCawley, J. D. (1982). Parentheticals and discontinuous constituent structure. *Linguistic Inquiry*, 13(1):91–106.

McDonald, R., Pereira, F., Ribarov, K., and Hajic, J. (2005). Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 523–530, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Nederhof, M.-J. (2003). Weighted deductive parsing and Knuth's algorihtm. *Computational Linguistics*, 29(1):135–143.

Nivre, J. (2003). An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*, pages 149–160, Nancy, France.

Nivre, J. (2006). Constraints on non-projective dependency parsing. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 73–80, Trento, Italy. Association for Computational Linguistics.

Nivre, J., Hall, J., Nilsson, J., Chanev, A., Eryiğit, G., Kübler, S., Marinov, S., and Marsi, E. (2007). MaltParser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2):95–135.

Petrov, S., Barrett, L., Thibaux, R., and Klein, D. (2006). Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 433–440, Sydney, Australia. Association for Computational Linguistics.

Plaehn, O. (2004). Computing the most probable parse for a Discontinuous Phrase-Structure Grammar. In Bunt, H., Carroll, J., and Satta, G., editors, *New Developments in Parsing Technology*, pages 91–106. Kluwer.

Rehbein, I. and van Genabith, J. (2007). Evaluating evaluation measures. In *Proceedings of NODALIDA 2007*.

Santorini, B. (1990). Part-of-speech tagging guidelines for the Penn Treebank Project (3rd revision, 2nd printing). Technical report, University of Pennsylvania.

Seki, H., Nakanishi, R., Kaji, Y., Ando, S., and Kasami, T. (1991). On multiple context-free grammars. *Theoretical Computer Science*, 88(2):191–229.

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

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, Washington, DC, USA. Association for Computational Linguistics.

Skut, W., Krenn, B., Brants, T., and Uszkoreit, H. (1998). A linguistically interpreted corpus of German newspaper text. In *Proceedings of LREC-98*, Granada, Spain.

Ule, T. (2003). Directed treebank refinement for PCFG parsing. In *Proceedings of the Second Workshop on Treebanks and Linguistic Theories (TLT 2003)*, Växjö, Sweden.

Vijay-Shanker, K., Weir, D. J., and Joshi, A. K. (1987). Characterizing structural descriptions produced by various grammatical formalisms. In *Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics*, pages 104–111, Stanford, California, USA. Association for Computational Linguistics.

Villemonte de la Clergerie, E. (2002). Parsing mildly context-sensitive languages with thread automata. In *COLING 2002: The 19th International Conference on Computational Linguistics*.

Weir, D. J. (1988). *Characterizing Mildly Context-Sensitive Grammar Formalisms*. PhD thesis, University of Pennsylvania.

List of Tables