A Multi-Strategy Approach for Parsing of Grammatical Relations in Transcripts of Parent-Child Dialogs

Kenji Sagae

June 2, 2006
CMU-LTI-06-003

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Thesis Committee:
Alon Lavie, Chair
Brian MacWhinney
Lori Levin
Jaime Carbonell
John Carroll, University of Sussex

Abstract

Automatic analysis of syntax is one of the core problems in natural language processing. Despite significant advances in syntactic parsing of written text, the application of these techniques to spontaneous spoken language has received more limited attention. The recent explosive growth of online, accessible corpora of spoken language interactions opens up new opportunities for the development of high accuracy parsing approaches to the analysis of spoken language. The availability of high accuracy parsers will in turn provide a platform for development of a wide range of new applications, as well as for advanced research on the nature of conversational interactions. One concrete field of investigation that is ripe for the application of such parsing tools is the study of child language acquisition.

In this thesis, we describe an approach for analyzing the syntactic structure of spontaneous conversational language in parent-child interactions. Specific emphasis is placed on the challenge of accurately annotating the English corpora in the CHILDES database with grammatical relations (such as subject, objects and adjuncts) that are of particular interest and utility to researchers in child language acquisition. This work involves rule-based and corpus-based natural language processing techniques, as well as a methodology for combining results from different parsing approaches. We present novel strategies for integrating the results of different parsers into a system with improved accuracy.

One practical application of this research is the automation of language competence measures used by clinicians and researchers of child language development. We present an implementation of an automatic version of one such measurement scheme. This provides not only a useful tool for the child language research community, but also a task-based evaluation framework for grammatical relation identification.

Through experiments using data from the Penn Treebank, we show that several of the techniques and ideas presented in this thesis are applicable not just to analysis of parent-child dialogs, but to parsing in general.

Contents

Contents i

List of Figures v

List of Tables vii

Acknowledgements ix

1 Introduction 1

1.1 Motivation . . . 2

1.2 Research Goals . . . 3

1.3 Thesis Summary . . . 4

1.3.1 Background, Overview and Related Work . . . 4

1.3.2 The CHILDES GR Annotation Scheme . . . 5

1.3.3 A Rule-Based GR Parsing Approach . . . 6

1.3.4 Data-Driven GR Parsing . . . 9

1.3.5 Combining Different Approaches for GR Parsing . . . 12

1.3.6 Automated Measurement of Syntactic Development . . . 15

1.4 Thesis Contributions . . . 17

2 Overview, Data, and Previous Work 19

2.1 Background . . . 20

2.1.1 Dependency parser evaluation . . . 21

2.1.2 Constituent parser evaluation . . . 23

2.1.3 The role of annotated training data in parsing . . . 24

2.2 The multi-strategy parsing approach . . . 25

2.3 Data . . . 26
2.3.1 The CHILDES Database . . . 27
2.3.2 The Penn Treebank . . . 28
2.4 Related Work . . . 29
2.4.1 Related work in rule-based and data-driven parsing . . . 29
2.4.2 Related work in parser combination . . . 33

3 The CHILDES GR Annotation Scheme 35
3.1 Representing GRs with Labeled Dependencies . . . 35
3.2 Specific GR Labels . . . 36
3.3 Related GR Annotation Schemes . . . 48
3.4 Specific Representation Choices . . . 49
3.5 Word Insertion in Child Language Annotation . . . 51
3.6 Inter-Annotator Agreement . . . 52

4 A Rule-Based GR Parsing Approach 53
4.1 Grammar-driven Syntactic Parsing . . . 53
4.2 A Rule-Based Syntactic Analysis System . . . 57
4.2.1 MOR and Part-of-Speech Tagging . . . 60
4.2.2 LCFlex . . . 61
4.2.3 Statistical Disambiguation . . . 62
4.2.4 Tailoring an Analysis System for Adult Utterances . . . 63
4.2.5 Grammar-Driven Parsing and Child Utterances . . . 68
4.3 Conclusion . . . 69

5 Data-Driven GR Parsing Approaches 71
5.1 GR Parsing with a Generative Statistical Parser . . . 74
5.1.1 Text Preprocessing . . . 74
5.1.2 Unlabeled Dependency Identification . . . 75
5.1.3 Dependency Labeling . . . 76
5.2 GR Analysis with Classifier-based Parsing . . . 78
5.2.1 Classifier-Based Constituent Parsing in Linear Time . . . 79
5.2.2 Classifier-Based Dependency Parsing in Linear Time . . . 87
5.3 Conclusion . . . 93

6 GR Parser Combination **95**

6.1 Dependency voting schemes . . . 96

6.1.1 Unweighted voting . . . 96

6.1.2 Weighted voting . . . 97

6.1.3 POS-weighted voting and label-weighted voting . . . 98

6.2 Voting with the WSJ corpus . . . 98

6.2.1 Parsers . . . 99

6.2.2 Parser selection . . . 100

6.2.3 Voting results . . . 101

6.3 Voting with CHILDES data . . . 102

6.3.1 Parsers . . . 102

6.3.2 Voting results . . . 103

6.4 Parser combination as reparsing . . . 105

6.4.1 GR combination with maximum spanning trees . . . 106

6.4.2 GR combination with the CYK algorithm . . . 107

6.5 Identifying GRs in utterances by young children . . . 108

6.6 Conclusion . . . 112

7 Automated Measurement of Syntactic Development **115**

7.1 The Index of Productive Syntax (IPSyn) . . . 116

7.2 Automating IPSyn . . . 117

7.3 Evaluation . . . 119

7.3.1 Test Data . . . 120

7.3.2 Results . . . 120

7.3.3 Error Analysis . . . 122

7.4 Relating parser accuracy to IPSyn accuracy . . . 123

8 Conclusions and Future Directions **127**

8.1 Possible Directions for Future Research . . . 129

8.1.1 Classifier-based shift-reduce parsing . . . 129

8.1.2 Parser combination . . . 130

8.1.3 Parsing different languages and different genres . . . 131

8.1.4 Measurement of syntactic development in child language . . . 131

Bibliography **133**

A Head Percolation Table for Lexicalization of Constituents **141**

B Complete GR Results on the Eve GR Test Set **143**

C IPSyn Items and Scoring Sheet **155**

List of Figures

2.1 Three syntactic representations for a sentence. 21

2.2 A general parser combination scheme. 27

3.1 Sample sentences annotated with CHILDES GRs. 37

4.1 Simple context-free grammar and sentences. 55

4.2 Parse trees for "I saw the boy with the dog" and "I saw the dog with the telescope" (trees rooted with S* represent dispreferred analyses). The trees in (b) and (c) can be generated with the given grammar, but the desired trees are (a) and (c). Adding a rule to allow for the analysis in (a) has the undesired side effect of also allowing the (incorrect) tree in (d). 56

4.3 A simple grammar composed of a context-free backbone and unification equations. 57

4.4 Analysis of "He sees the ball" according to the grammar in figure 4.3 59

4.5 Three syntactic analyses for the sentence "Salespeople sold the dog biscuits." 65

5.1 An example of the binarization/de-binarization transform. 81

5.2 Features used for classification. The features described in items 1 – 5 are more directly related to the lexicalized constituent trees that are built during parsing, while the features described in items 6 – 11 are more directly related to the dependency structures that are built simultaneously to the constituent structures. 83

5.3 Constituent and dependency trees for the sentence "I eat cake with Joe". . 89

5.4 Accuracy of SVMdep trained on Eve GRs with different amounts of data. . 92

5.5 Accuracy of SVMdep trained on the Penn Treebank with different amounts of data. 92

7.1 Histogram of point differences between GR and HUMAN scores (black), and CP and HUMAN (white). 121

C.1 Part 1 of the IPSyn scoring sheet, from (Scarborough, 1990). 158
C.2 Part 2 of the IPSyn scoring sheet, from (Scarborough, 1990). 159

List of Tables

4.1 Settings, coverage, precision and recall for each pass in the multi-pass parsing strategy. 67

5.1 Precision, recall and f-score (harmonic mean) of selected GRs obtained with the statistical parsing approach. 78

5.2 Summary of results obtained with the classifier-based constituent parser as labeled precision and recall of constituents, dependency accuracy, and time required to parser the test set (unk denotes unknown values). *The parsers of Yamada and Matsumoto (Y&M) and Nivre and Scholz (N&S) do not produce constituent analyses. 87

5.3 Precision, recall and f-score (harmonic mean) of GRs obtained with classifier-based dependency parsing. 91

6.1 Evaluation of single parsers on the WSJ test corpus. 100

6.2 Results for voting experiments using the WSJ corpus, evaluated as unlabeled attachment score (UAS) and root accuracy (RA). In all conditions, UAS improvements from unweighted to weighted combination are statistically significant at the 5% level according to the computationally-intensive randomization test described by Yeh (2000a). Improvements from weighted to POS-weighted combinations are not statistically significant. 102

6.3 Accuracy of single parsers on the Eve test set. For the rule-based parsers, in addition to the labeled dependency accuracy (LDA) over all words, we show in parenthesis the LDA over only those words for which the parser found an analysis. 103

6.4 Results of POS-weighted voting and label-weighted voting on the Eve test set. Labeled dependency accuracy over all words and precision/recall values for four GRs are shown. 104

7.1 Precision, recall and F-score (harmonic mean) of the MST combination system on selected GRs. 119

7.2 Summary of IPSyn score evaluation (point difference and point-to-point reliability). GR is our implementation of IPSyn based on grammatical relations, CP is Long et al.'s (2004) implementation of IPSyn, and HUMAN is manual scoring. 120

7.3 IPSyn structures where errors occur most frequently, and their percentages of the total number of errors over 41 transcripts. 123

A.1 Head percolation table. 142

B.1 Cdep: GR results. 144

B.2 SVMdep: GR results. 145

B.3 MBLdep: GR results. 146

B.4 RL-SVMdep: GR results. 147

B.5 RL-MBLdep: GR results. 148

B.6 RB: GR results. 149

B.7 POS-weighted voting: GR results. 150

B.8 Label-weighted voting: GR results. 151

B.9 CKY-combination: GR results. 152

B.10 MST-combination: GR results. 153

Acknowledgements

I would like to thank my advisors Alon Lavie and Brian MacWhinney for their guidance in every aspect of my graduate education at Carnegie Mellon. One of the central themes in this thesis is the value of combining different approaches to perform a single task. In retrospect, this highlights how fortunate I have been as a graduate student with respect to my advisors. The general direction of the work described here stresses the impact of my advisors' individual backgrounds and strengths, while reflecting the freedom I have had in shaping my own research. I thank Alon and Brian for being excellent advisors, both individually and in combination. I have been able to count on their support during my entire time at CMU.

I would also like thank the other members of my thesis committee. Lori Levin contributed with her expertise in Linguistics and NLP, in addition to general advice on writing a dissertation. Jaime Carbonell provided insights and advice that could only come from someone with immense experience and breadth of knowledge in language technologies and artificial intelligence. Finally, John Carroll's expertise in syntactic parsing and identification of grammatical relations helped me strengthen this thesis in several aspects, from high-level discussions to technical details. John's comments were also valuable in situating this work within the context of the constantly growing body of parsing research.

Many others contributed in less direct but still important ways to this work. When I was still an undergraduate at UCLA, Ed Stabler first introduced me to computational linguistics, and then gave me a great job and much needed advice on pursuing graduate studies in this field. Also at UCLA, I had my first opportunity to do research and present it in a conference thanks to Jim Kroger (now faculty at New Mexico State University). Since arriving at CMU, I have benefited from discussions with many students and faculty members in the Language Technologies Institute. I have also received considerable help from the LTI administrative staff.

Lastly, but more importantly, I thank my family for their encouragement and support in everything I do. Most of all, I thank my wife Alicia, who took care of everything for both of us for several months while I worked on this thesis, and simply always makes me happy.

Chapter 1

Introduction

Automatic analysis of syntax is one of the core problems in natural language processing. The process of determining one or more syntactic structures corresponding to an input sentence is commonly referred to as *syntactic parsing*, or simply *parsing*, and it is generally regarded as a necessary step in natural language understanding, as well as in several natural language applications such as machine translation or question answering. After decades of research in linguistics, computer science and related fields, reasonably accurate natural language parsers are now available for a number of languages. However, most of this research has focused on written text, and the application of these analysis methods to spontaneous spoken language corpora has received more limited attention. At the same time, the recent explosive growth of online, accessible corpora of spoken language interactions opens up new opportunities for the development of high accuracy parsing approaches to the analysis of spoken language. The availability of high accuracy parsers will in turn provide a platform for developing a wide range of new applications, as well as for advanced research on the nature of conversational interactions. One concrete field of investigation that is ripe for the application of such parsing tools is the study of child language acquisition. The CHILDES database (MacWhinney, 2000), containing several megabytes of transcripts of spoken interactions between children at various stages of language development with their parents, provides vast amounts of useful data for linguistic, psychological, and sociological studies of child language development. In this thesis, we will present an approach for syntactic parsing of the child and adult language in CHILDES transcripts, focusing on a syntactic representation that suits the needs of the child language research community.

Despite the enormous impact CHILDES has had on the study of language acquisition, the lack of tools for automatic analysis above the word-level stands as a barrier to the realization of the full potential of this system for research on normal language develop-

ment and the evaluation of language disorders. Currently, the corpus includes no syntactic information, because parsing technology specifically targeted to the production of such annotations has not been available. Lacking these tools, several major research groups[1] have been forced to engage in resource-intensive syntactic annotation of small segments of the database. Because this work is not based on shared community standards, the resulting analyses are discrepant and non-replicable, and current corpus-based research on syntactic development has been only weakly cumulative.

The work presented in this thesis aims to correct this situation through the development of a new system specifically tailored to the needs of the child language community. Developing technology for reliable analysis of the syntax of spontaneous spoken language is a challenging task, and shaping this technology for the productions of young children is a still greater challenge. We present an annotation scheme designed to represent syntactic information in CHILDES corpora in a way that addresses specific needs of the child language community. We then investigate the use of different parsing technologies to analyze transcripts according to our annotation scheme. Finally, we will show that novel ways of combining different parsing approaches allow for accurate analysis of syntax in both child and adult language in CHILDES transcripts.

1.1 Motivation

Explaining the enigma of child language acquisition is one of the main challenges facing cognitive science. Although all normal children succeed in learning their native tongue, neither psychology nor linguistics has yet succeeded in accounting for the many complexities of language learning. Within this general area, there has been particular attention to the acquisition of grammar, as it is expressed through morphosyntax, stimulated in large measure by Chomsky's theory of Universal Grammar and its attendant claims regarding innate principles and parameters. Beyond its theoretical importance, the measurement of morphosyntactic competence is crucial to applied work in fields such as developmental language disorders, schooling and literacy, and second language acquisition.

To examine the development of morphosyntax, researchers have come to rely increasingly on large corpora of transcript data of verbal interactions between children and parents. The standard database in this area is the CHILDES database (MacWhinney, 2000; http://childes.psy.cmu.edu), which provides 300MB of transcript data for over 25 human languages, as well as a large amount of digitized audio and video linked to the transcripts.

[1] Such as those in the University of Winsconsin (Madison), the University of California (San Diego and Berkeley), Rutgers, Purdue, Cornell, the Massachusetts Institute of Technology, Nagoya University, the University of Potsdam, among others.

There are now several hundred studies that have used the CHILDES database to study the development of morphosyntax. However, these studies have typically been forced to use the database in its raw lexical form, without tags for part-of-speech, syntactic parses, or predicate-argument information. Lacking this information, researchers have devoted long hours of hand analysis to locating and coding the sentences relevant to their hypotheses. If syntactic parses were available, these analyses could be automated, allowing investigators to conduct a wider variety of tests in a more reliable fashion. Automatic syntactic analysis systems would also be of great value in clinical settings, allowing clinicians and clinical researchers to construct profiles of language delays by comparing small speech samples collected in structured interviews with a larger database of normed data.

Producing automated analyses of child language corpora is one instance of the more general challenge of developing a comprehensive NLP approach to the analysis of spontaneous speech. Although many of the ideas presented in this thesis are applicable to other genres of spoken language, we focus on the child language problem for three reasons. First, a large, publicly accessible corpus of parent-child spoken language interactions is available, and its importance in the study of child language makes the task of syntactic analysis immediately meaningful in a practical sense. Second, research in this area works towards bridging the gap between cognitive science and natural language processing, thereby improving interaction between these fields. Third, the child language field has already developed a clear set of criteria for the standardized measurement of morphosyntactic development (Scarborough, 1990). The automation of these pre-existing measures provides clear benchmarks for new NLP systems.

1.2 Research Goals

The research described in this dissertation encompasses five main objectives:

- Defining a suitable syntactic representation for annotating syntactic information in transcripts in the CHILDES database. This representation should address practical needs of the child language research community.
- Developing effective parsing approaches for high accuracy identification of grammatical relations in CHILDES data (including utterances spoken by adults and children at various stages of first language acquisition).
- Developing effective methods for combining the results of multiple sources of syntactic information, including rule-based and statistical parsers, in order to improve the

accuracy of the combined system by accounting for specific strengths of the different approaches;

- Evaluating the performance of our resulting grammatical relation parsing approaches using the standard performance metrics of precision and recall;
- Validating the utility of the resulting systems to child language acquisition research by automating a standard application for measuring the grammatical complexity of child language corpora.

1.3 Thesis Summary

1.3.1 Background, Overview and Related Work

Natural language syntactic parsers are systems that output one or more syntactic structures corresponding to an input sentence. Parsing is a complex problem, and several different approaches have demonstrated some amount of success in different aspects of the task. Parsers differ with respect to what underlying algorithm they use to determine syntactic structures, what kind of syntactic structures they provide, whether they output one or more syntactic structures for a single sentence, among other things. One characteristic that varies significantly across several parsing approaches is how much the system relies on linguistic knowledge encoded as rules, such as with a manually-written grammar, and how much the system learns based on manually annotated examples (sentences with their correct syntactic structures). In chapter 2 we start by looking at this source of variation, as one of the issues addressed in this thesis is the combination of rule-based and data-driven parsing strategies: by leveraging on the strengths of approaches that tackle the parsing challenge in different ways, we attempt to achieve overall results that surpass the performance of the state-of-the-art of the different approaches taken in isolation. We then turn to the data used in this work, and discuss what data we target for syntactic analysis, what data is used for evaluation of the techniques we develop, and what data is used for development and training of our systems.

The CHILDES Database

The standard source for corpus-based research in child language is the CHILDES Database (MacWhinney, 2000), which contains hundreds of megabytes of transcripts of dialogs between children and parents in several languages. CHILDES transcripts have been used in over 1,500 studies in child language acquisition and developmental language disorders.

We focus on the English portion of the database, and pay particular attention to the Eve corpus (Brown, 1973), which we partition into training, development and testing sections that we use with the different parsing strategies we pursue.

The Penn Treebank

The Penn Treebank (Marcus et al., 1993) contains a large amount of text annotated with constituent (phrase structure) trees. The Wall Street Journal (WSJ) corpus of the Penn Treebank contains about one million words of text from news stories taken from the Wall Street Journal. Although the type of language in the WSJ domain (written, formal, edited text) differs significantly from our target domain (CHILDES transcripts with utterances from children and parents), there are two important reasons for our use of the WSJ corpus of the Penn Treebank in this work: (1) it contains a much larger amount of text annotated with syntactic structures than would be feasible us to annotate manually using CHILDES data, and certain data-driven systems require very large training corpora; (2) it allows us to compare our methods directly to a large and accessible body of research in parsing that has used the WSJ corpus for training, development and testing.

1.3.2 The CHILDES GR Annotation Scheme

One crucial aspect of producing useful automatic syntactic analysis for child-parent dialog transcripts is the definition of a suitable annotation scheme that targets the specific type of syntactic information needed by the child language community. To address this need, we have developed the CHILDES Grammatical Relation (GR) annotation scheme, described chapter 3.

Syntactic information in CHILDES transcripts is represented according to our scheme as labeled dependency structures. Such structures are trees where each node is a word in the sentence, and there are labels associated with each edge that reflect the type of syntactic relationship (or grammatical relation) that exists between the parent word (the head), and its children (the dependents). The GRs specified by the dependency labels include relations such as subject, object, adjunct, etc. The specific set of GR types (or labels) included in our annotation scheme was developed by using the GRs in the annotation scheme of Carroll et al. (2003) as a starting point, and adapting the set to suit specific needs of child language research. Sources of refinement included a survey of the child language literature (Fletcher and MacWhinney, 1995), a review of existing measures of syntactic development (MacWhinney, 2000), and input from child language researchers.

In general, content words (such as nouns and verbs) are considered to be the heads of

function words (such as determiners and auxiliaries), except in the cases of prepositional phrases, where the preposition is the head of its object (usually a noun), and coordination, where a conjunction is the head of its coordinated phrases. Only surface grammatical relations are considered, and the annotation scheme does not deal with issues such as control and ellipsis (although extensions for handling such phenomena are possible). We have measured inter-annotator agreement for the CHILDES GR annotation scheme at 96.5%.

1.3.3 A Rule-Based GR Parsing Approach

Chapter 4 describes the use of rule-based parsing for automatically annotating CHILDES corpora with the grammatical relation scheme described in chapter 3. The form of rule-based parser we consider here is more specifically described as grammar-driven parsing, which uses a set of production rules (a grammar) that specify how each syntactic constituent may be expanded into other constituents or words as a model of natural language.

The use of a grammar effectively shapes the search space of possible syntactic analyses for a sentence, since the rules specify exactly how words may combine to form larger structures. It is then important to use a carefully designed grammar that allows the system to analyze as many syntactic configurations in the text as possible, while constraining the space of analyses so that ambiguities present in the grammar can be resolved effectively, and a correct analysis for each sentence can be found. In practice, this results in a trade-off between coverage (the portion of input sentences that are covered by the rules in the grammar) and ambiguity (how many possible analyses for each sentence are licensed by the grammar). Grammars that allow for combinations of words and phrases in too many ways can be highly ambiguous, making the process of choosing a correct analysis for a given sentence (in other words, disambiguation) a difficult task. On the other hand, more restrictive grammars may not cover all syntactic structures in the input text. In spontaneous spoken language this problem is exacerbated, since sentence structures do not always conform to the more rigid standards of written text.

We developed a syntactic analysis system based mainly on grammar-driven robust parsing using the LCFlex parser (Rosé and Lavie, 2001). The system analyzes utterances from CHILDES transcripts in three steps: (1) word-level analysis (morphology and part-of-speech tagging); (2) grammar-driven syntactic analysis; and (3) statistical disambiguation.

Word-level Analysis: MOR and Part-of-Speech Tagging

We use the morphological analyzer MOR (MacWhinney, 2000), which was designed for use with CHILDES data, and returns a set of possible morphological analyses for each word in

a input sentence. For part-of-speech tagging and disambiguation of the output of MOR, we employ data-driven methods that must be trained on annotated data. Because only a small amount of such data was available for CHILDES data, while much larger amounts of out-of-domain data exist in the Penn Treebank, we designed a scheme that involves MXPOST (Ratnaparkhi, 1996), a part-of-speech tagger trained on the Penn Treebank WSJ corpus, and a classifier that converts the output of MXPOST to the CHILDES POS tag set. With this approach, we obtain over 96% accuracy of part-of-speech tagging using the CHILDES tag set.

Grammar-driven Syntactic Analysis: LCFlex

LCFlex is a robust grammar-driven parser with features that are designed specifically for analysis of spontaneous spoken language. Through the tuning of a few parameters, the parser allows the insertion of pre-specified words or constituents into the input sentence, or the skipping of certain portions of the input. The ability to insert and skip allows the parser to analyze sentences that deviate from the language specified by the grammar in use. This results in a way to manage aspects of the coverage/ambiguity trade-off while keeping the grammar constant.

Grammars used by LCFlex consist of a set of rules, each containing a context-free backbone and a set of unification equations. A bottom-up chart-parsing algorithm with top-down left-corner predictions is used to process sentences according to the context-free backbone of the grammar, and unification according to the equations in each rule is performed in interleaved fashion. When a properly written grammar (one that identifies grammatical relations between words) is used, LCFlex outputs feature structures that can be easily converted to labeled dependency structures simply by reading the grammatical relations and their participating words from the feature structure. To parse CHILDES data with LCFlex, we designed a custom grammar for our domain, containing unification features that correspond to the GRs in the CHILDES annotation scheme. The grammar is relatively small, and it covers about 65% of the Eve test set. While 35% of the sentences in our test set cannot be parsed with this grammar, ambiguity is kept low. This means that while the recall of specific GRs is low, precision is high. By allowing the parser to insert and skip according to its robustness parameters, we can parse more sentences, while introducing more ambiguity. In other words, we increase recall at the expense of precision.

Statistical Disambiguiation: Two-level PCFG

When the grammar allows for multiple analyses of a single sentence, we choose one of the possible analyses using statistical disambiguation. In our system, disambiguation is done with a two-level probabilistic context-free grammar (two-level PCFG) module available in LCFlex.

PCFGs are generative models that simply have a probability associated with each context-free rule in the grammar (and the probabilities for all rules with the same left-hand side sum to 1.0). According to a PCFG, the probability of an analysis is the product of the probabilities of each rule used in the derivation of the analysis. A two-level PCFG considers rule bigrams (while a regular PCFG considers rule unigrams). In other words, instead of each rule having a probability (as in the case of PCFGs), each pair of rules has a probability. Similar to the parent annotation described by Johnson (1998), this model creates bounded sensitivity to context. The rule-bigram probabilities in our two-level PCFG are determined by maximum likelihood estimation using the (manually annotated) Eve training corpus.

Tailoring an Analysis System for Adult Utterances

While the adult language in the CHILDES corpora generally conforms to standard spoken language, the child language in the corpora varies from the language of a child in the very early stages of language learning to fairly complex syntactic constructions. Child and adult utterances differ significantly enough that we can analyze them more accurately by doing so separately, often with different strategies. We first explore the "easier" (in the sense that it is better defined) problem of analyzing adult utterances, which are theoretically of equal importance as child utterances, since theories of learning depend heavily on consideration of the range of constructions provided to children in the input (MacWhinney, 1999; Moerk, 1983).

Although the grammar covers about 65% of the sentences in the Eve test set, this does not mean that the system returns the correct analysis for each of these sentences. What it does mean, however, is that the correct analysis is included in the set of analyses the parser finds for a sentence, and it is up to the disambiguation procedure to find it. Allowing the parser to use its robustness features (skip portions of the input and insert lexical and non-lexical items in the parsing process, as needed) increases coverage, while also increasing ambiguity (and making the search for the correct analysis more difficult). In other words, by controlling the amount of flexibility that the parser is allowed to use, we can manipulate the trade-off between precision and recall of GRs. For example, consider the GR SUBJ

(subject). If we look at the output of system when it is not allowed to insert of skip at all, we obtain high precision (94.9%) and low recall (51.6%). This is because ambiguity and coverage were limited. If we increase coverage and ambiguity by allowing the system to insert a noun phrase, and to skip one word, precision drops due to the increased ambiguity (84.3%), but recall improves due to the increase in coverage (76.2%).

Although the performance of our rule-based system is poor in certain aspects, it is not meant to be used by itself for analysis of CHILDES data. Instead, it is used in combination with other approaches described in this thesis. As we show in chapter 6, when combined with data-driven approaches, the rule-based system is quite valuable.

Grammar-driven Parsing of Child Utterances

While every adult utterance in CHILDES transcripts is expected to have valid grammatical relations, utterances spoken by children younger than five years old often deviate enough from standard adult speech that the recognition of GRs is not possible. Grammar-driven parsing is particularly appropriate in this situation, since it can be used to detect sentences where the system should not attempt to find GRs. Because the well-formed sentences spoken by young children are syntactically simpler than adult speech, our grammar covers those sentences quite well. Sentences that are rejected by the rule-based system are considered to be in the category of sentences where GR analysis is not possible. The rule-based system finds such sentences with accuracy above 85%. In chapter 6, we will see how this discrimination capability of our rule-based system can be combined with other parsing approaches for accurate GR analysis of child utterances.

1.3.4 Data-Driven GR Parsing

In chapter 5 we consider two data-driven approaches to syntactic analysis of CHILDES data. The first is based on existing statistical parsing using a generative model trained on the WSJ corpus of the Penn Treebank. The second is based on deterministic classifier-based parsing, where a classifier decides the actions of a shift-reduce parser. To compare this last approach to other work on parsing English, we first train and test it using the standard division of the WSJ corpus. We then apply it to our task of parsing CHILDES data, training and evaluating it on the appropriate sections of the Eve corpus.

GR Parsing with a Generative Statistical Parser

This first data-driven GR parsing approach we describe illustrates how existing natural language processing tools and resources can be utilized to produce a high-performance system for GR analysis in CHILDES data with relatively little cost. This approach is particularly useful in languages for which accurate parsers already exist. The main idea is to obtain syntactic parses for the target data using an existing parser, the well-known Charniak (2000) parser, and convert those analyses into GRs in the CHILDES annotation scheme. The GR system we developed according to this general idea analyzes CHILDES transcripts in three steps: text preprocessing, unlabeled dependency identification, and dependency labeling.

In the text processing step we simply use existing tools for processing transcripts in the CHILDES database to remove disfluencies such as false-starts, retracings and repetitions. This step also tokenizes the input text, providing text that is as clean as possible to the Charniak parser.

The second step of unlabeled dependency identification is where we actually find a bare-bones syntactic analysis for the input text, using the Charniak parser. Since the parser is trained on Penn Treebank constituent trees, and outputs the same style of syntactic representation, it is then necessary to convert the output of the parser into the dependency format used for representing syntactic structures in the CHILDES annotation scheme. We use the standard procedure of lexicalizing the output constituent trees using a head percolation table (Collins, 1996; Magerman, 1995) to extract an unlabeled dependency structure from the output of the Charniak parser. The rules in the head table we use are similar to the ones used in the Collins (1996) parser, but modified to reflect our dependency annotation scheme.

Once we have obtained unlabeled dependencies, we proceed to the third step: dependency labeling. This is accomplished by using a classifier that determines a GR label for each of the dependencies in a dependency structure. The input we pass to the classifier is a set of features extracted from the unlabeled dependency structure, including the head and dependent words in the dependency in question, their part-of-speech tags, the distance between the two words, and the label (phrase type) of the lowest constituent that includes both words in the original constituent tree output of the Charniak parser. The output of the classifier is one of the possible GR labels in the CHILDES GR annotation scheme. The classifier is trained by extracting pairs of correct labels and feature sets for every dependency in the Eve training corpus. The classifier used in our system is the k-nearest neighbors implementation in the TiMBL package (Daelemans et al., 2004).

This approach is quite effective in identifying GRs in CHILDES data, as we verify

by testing it on the Eve test corpus. When tested on WSJ data, the Charniak parser has an unlabeled dependency accuracy of over 92%. On the Eve test corpus, the unlabeled dependency accuracy of the parser is 90.1%. Despite the degradation in performance due to the differences in the training and testing domains, the accuracy of the parser on CHILDES data is still high. The accuracy of the dependency labeling step using the k-nn classifier (on gold standard unlabeled dependencies) is 91.4%. When we combine the unlabeled dependency parsing step and the dependency labeling step, the overall labeled dependency accuracy of the system is 86.9%. Although accuracy is high, the precision and recall of certain GRs is, in fact, quite low. The GRs corresponding to clausal complements COMP and XCOMP, for example, have precision and recall below 60%. Other frequent GRs, such as subject, objects and adjuncts, are recognized with high levels of precision and recall (above 80%).

GR Analysis with Classifier-Based Parsing

Although the statistical parsing approach to GR identification described in the previous section performs well for most GRs in our annotation scheme, there are important reasons for also using additional data-driven approaches to GR analysis. One of the main reasons, which is discussed in detail in chapter 6 and is directly related to one of the central themes in this thesis, is that we can improve the precision and recall of identification of individual GRs as well as overall system accuracy by utilizing several different GR identification approaches. Henderson and Brill (1999) have shown that, in the context of constituent parsing of the Penn Treebank, combining the outputs of different parsers can result in improved accuracy, even if each parser uses the same training data. As we show in chapter 6, this aspect of parser diversity is also applicable to dependency parsing, and combination schemes using different approaches to GR parsing result in improved precision and recall of GRs. An additional reason for pursing different analysis approaches is, of course, that our statistical parsing approach discussed in the previous section does not identify certain GRs (such as COMP and XCOMP) reliably. Finally, this second data-driven approach we consider allows us to develop a parser that works natively with the syntactic representation used in our CHILDES GR scheme.

We first present a general approach for classifier-based parsing with constituent structures, and evaluate two parsers developed in this framework using the standard split for training, development and testing of the Penn Treebank. The main idea of our classifier-based parsing approach is to have a classifier that decides on the parsing action of a shift-reduce parser. The parsing algorithm is very simple, and parsing is done in linear time. Each time the parser must decide on a shift or a reduce action, a set of features that reflect the current configuration of the parser is given as input to a classifier, which then outputs

a parser action. A parser using k-nearest neighbors for classification achieves slightly above 80% precision and recall of constituents, and 86.3% dependency accuracy on Penn Treebank WSJ data. Using support vector machines, the classifier-based parser achieves over 87% precision and recall of constituents, and 90.3% dependency accuracy. The SVM-based parser, although not as accurate as state-of-the-art statistical parsers that achieve about 89% precision and recall on the same data, is more accurate than several more complex parsers, while parsing considerably faster than popular statistical parsing approaches.

We then turn to the task of parsing CHILDES data using a classifier-based parser. First, we show that our general classifier-based parser for constituents can be easily adapted into a labeled dependency parser. The resulting parser is similar to the MALT parser (Nivre and Scholz, 2004), but it uses a slightly different set of features. Using support vector machines for classification, and training on the Eve training set, the performance of the parser on the Eve test set is surprisingly high. Overall labeled dependency accuracy is 87.3%, putting this approach on the same level as the one using the more complex Charniak parser (trained on the larger WSJ corpus). More interestingly, the precision and recall of GRs that were problematic for our previous approach is much improved. For example, the precision and recall of XCOMP are above 80%, and the precision and recall of COMP are above 70%.

1.3.5 Combining Different Approaches for GR Parsing

In chapter 6 we address the issues related to the combination of the several parsing approaches discussed in chapters 4 and 5 to achieve improved identification of GRs. We start with a simple voting scheme, examine progressively more sophisticated combination schemes, and arrive at a novel way of performing parser combination using a parsing algorithm. We then turn to the issue of analyzing utterances from young children using the rule-based parser to discriminate between utterances that contain GRs and those where GRs cannot be reliably identified.

Dependency Voting

First we consider the simplest case of combining unlabeled dependency structures created by different parsers. The simplest voting scheme assigns equal weight to the dependencies generated by every parser. Voting is done on a word-by-word basis. For each word, we check the head chosen by each of the parsers. Each of these heads receives one vote. The head with most votes is chosen as the head for the current word.

We tested this combination scheme using different parsers generated according to the classifier-based framework described in chapter 5. Accuracy of the combination on the

standard WSJ test set is 91.9%, a 14% error reduction over the most accurate parser in the combination. Similar but more complex schemes that assign weights to the votes of each parser result in further accuracy improvements, with the best scheme achieving 92.3% accuracy. For comparison purposes, the statistical parsers of Collins (1997) and Charniak (2000) have unlabeled dependency accuracy of 91.2% and 92.3%, respectively. When these two statistical parsers are included in the best-performing voting scheme in addition to the deterministic parsers, we achieve 93.9% accuracy, surpassing the highest published results in the WSJ test set. When applied to the Eve test set, a weighted voting combination gives us 94.0% unlabeled dependency accuracy (over 30% relative error reduction from the single best parser in the combination scheme).

Obtaining Well-Formed Dependency Trees

Although the voting schemes perform well in producing more accurate dependencies from multiple parsers, there is no guarantee that the resulting dependencies form a dependency tree. In fact, the resulting set of dependencies may form a structure with cycles, or even disconnected graphs. We present two novel ways to combine dependency structures that perform as well as the voting schemes in terms of accuracy, while building well-formed dependency structures.

In both cases, we start by creating a graph where each word in the sentence is a node. We then create directed edges between nodes corresponding to words for which dependencies are obtained from any of the parsers. In cases where more than one parser indicates that the same edge should be created, the weights are added, just as in the voting scheme. As long as any of the parsers creates a valid dependency tree for the sentence, the directed weighted graph created this way will be fully connected.

Once the graph is created, we can simply find its maximum spanning tree, using, for example, the Chu-Liu/Edmonds directed MST algorithm (Chu and Liu, 1965; Edmonds, 1967). The maximum spanning tree maximizes the votes for dependencies given the constraint that the resulting structure must be a tree. However, there is no guarantee against crossing branches. While this may seem undesirable, the resulting tree generated from the combination of parsers should rarely contain crossing branches (for English). In addition, this would be a suitable scheme for combining structures in free-word-order languages, where branches are expected to cross.

A second option is to use dynamic programming to "reparse" the sentence. We proceed just as if we were parsing the sentence using the CKY parsing algorithm for PCFGs, but we restrict the creation of new items in the CKY chart to pairs of words connected by the appropriate edges in the directed weighted graph, and assign these items the weight of

their respective edges. Instead of the usual multiplication of probabilities, we simply add the values associated with each item used in the creation of a new item, and the value of the graph edge that allowed that item to be created. The resulting syntactic structure is guaranteed to be a tree with no crossing branches, and accuracy is only slightly lower than with the best-performing voting method.

Handling Child Utterances by combining Rule-Based and Data-Driven Approaches

There are two challenges in parsing child language that are not addressed by the data-driven GR analysis methods presented in chapter 5:

1. Certain utterances by young children cannot be annotated according to our GR annotation scheme, simply because they do not contain the syntactic structure associated with GRs. Young children may produce word strings that do not conform to a grammar that we can interpret. Rather than to *guess* what the child is trying to say (but not saying), we should simply not annotate any GRs in such utterances. The data-driven systems we have developed, as most of the recent work in data-driven parsing, are inherently robust, assigning a syntactic structure to (almost) any string of words.

2. Young children may produce utterances that are mostly grammatical, but missing certain words (auxiliaries, possessive case markers, pronouns, prepositions, etc). Although the intent of the utterance is clear (and no guessing is required), words that are missing in the utterance make its analysis problematic for a data-driven parser trained on fully grammatical language.

We have seen that the data-driven parsing approaches outperform overall performance of our rule-based approach. However, the rule-based approach is better equipped to handle the two challenges mentioned above. First we address challenge (1). By using a small grammar with as little over-generation as possible, we can attempt to identify utterances where GRs should be found, and those where they should not. This is done simply by verifying if an utterance can be parsed using the grammar or not. While it may seem that the brittleness usually associated with rule-based systems may be a problem, there is a characteristic of the task that works to our advantage: the sentences in question tend to be very simple, so the grammatical coverage problem is greatly diminished. To address challenge (2), we use the rule-based parser's ability to perform limited insertions (one of the robustness features of LCFlex). By using the same grammar, we can also attempt to identify utterances where a word may be missing, and even determine what the word should be. If a missing word is

correctly identified, it can be inserted in the sentence, which can then be passed as input to a data-driven system or a combination of systems as described in the previous section.

In the task of determining whether or not a sentence should be analyzed for GRs, we achieve better than 85% accuracy. In the task of identifying missing words, the system performs well in the most frequent cases (missing copula, missing possessive case marker). Certain less frequent (and more challenging) word insertions cannot be performed reliably, such as with missing prepositions. These rule-based techniques allow the overall performance of GR identification in young children's utterances to go from 69% to 87% accuracy.

1.3.6 Automated Measurement of Syntactic Development

In chapter 7 we present a practical end-to-end application of the methods for syntactic annotation and automatic analysis described so far. The task we explore is the automated measurement of syntactic development in child language, which has both clinical and theoretical value in the field of child language acquisition. Specifically, we present a fully automated way of computing the Index of Productive Syntax, or IPSyn (Scarborough, 1990), a popular measure for syntactic development that has traditionally required significant manual effort by trained researchers or clinicians.

In addition to its inherent value to the child language community, automatic computation of IPSyn scores serves as a task-based evaluation for our GR analysis approach. Although researchers in natural language parsing have become accustomed to evaluating systems in terms of precision and recall of certain pieces of information (such as constituent bracketing, or grammatical relations, as we have done in previous chapters), a syntactic analysis system often operates as a piece in a larger system designed to perform a task that goes beyond determining parse trees or grammatical relations. Because task-based evaluations focusing on the effects of the performance of specific NLP components are relatively rare, the relationship between the standard precision/recall measures and the performance of larger systems that include these NLP components is still somewhat unclear. Through a task-based evaluation where we examine the results of an end-to-end system that computes IPSyn scores, we can determine the impact of the accuracy of our GR system in a practical setting. Accurate computation of IPSyn scores validates the usefulness of our annotation scheme and our approach to automatic GR analysis in its current levels of precision and recall of grammatical relations.

The Index of Productive Syntax (IPSyn)

The Index of Productive Syntax (Scarborough, 1990) is a measure of development of child language that provides a numerical score for grammatical complexity. IPSyn was designed for investigating individual and group differences in child language acquisition, and has been used in numerous studies. It addresses weaknesses in the widely popular Mean Length of Utterance measure, or MLU, with respect to the assessment of development of syntax in children. Because it addresses syntactic structures directly, it has gained popularity in the study of grammatical aspects of child language learning in both research and clinical settings.

Calculation of IPSyn scores requires a corpus of 100 transcribed child utterances, and the identification of 56 specific language structures in each utterance. These structures are counted and used to compute numeric scores for the corpus in four categories (noun phrases, verb phrases, questions and negations, and sentence structures), according to a fixed score sheet. IPSyn scores vary from zero to 112, with higher scores reflecting more syntactic complexity in the corpus. Language structures that must be identified for computation of IPSyn vary from simple patterns such as determiner-noun sequences to more complex structures such as embedded clauses, relative clauses and bitransitive predicates.

Automating IPSyn

Calculating IPSyn scores manually is a laborious process that involves identifying 56 syntactic structures (or their absence) in a transcript of 100 child utterances. Currently, researchers work with a partially automated process by using transcripts in electronic format and spreadsheets. However, the actual identification of syntactic structures, which accounts for most of the time spent on calculating IPSyn scores, still has to be done manually. Long et al. (2004) have attempted, with limited success, to automate the computation of IPSyn using patterns of part-of-speech tags to search for the syntactic structures that are used in IPSyn scoring. Their Computerized Profiling (CP) program can be used for identification of simpler structures within IPSyn, but reliability of overall scores is well below manual level. Syntactic analysis of transcripts as described in chapters 4, 5 and 6 allows us to go a step further, fully automating IPSyn computations and obtaining a level of reliability comparable to that of human scoring. The ability to search for syntactic patterns using both grammatical relations and parts-of-speech makes searching both easier and more reliable. Automating the process of computing IPSyn scores consists of two main steps: (1) parse each sentence in the input transcript to obtain GRs according to the CHILDES annotation scheme; and (2) use patterns of GRs to search each sentence in the transcript for each of the syntactic structures named in IPSyn.

Evaluation

We evaluate our implementation of IPSyn in two ways. The first is *Point Difference*, which is calculated by taking the (unsigned) difference between scores obtained manually and automatically. The point difference is of great practical value, since it shows exactly how close automatically produced scores are to manually produced scores. The second is *Point-to-Point Accuracy*, which reflects the overall reliability over each individual scoring decision in the computation of IPSyn scores. It is calculated by counting how many decisions (identification of presence/absence of language structures in the transcript being scored) were made correctly, and dividing that number by the total number of decisions. The point-to-point measure is commonly used for assessing the inter-rater reliability of metrics such as the IPSyn. In our case, it allows us to establish the reliability of automatically computed scores against human scoring.

Using two sets of transcripts (41 transcripts in total) with corresponding IPSyn scoring that were provided to us by child language research groups, we measured the average point difference of our GR-based IPSyn system to be 2.5, and the point-to-point reliability to be 93.3%. For comparison purposes, CP's average point difference on the same transcripts is significantly higher at 8.3, and its point-to-point reliability is significantly lower at 85.4%. Inter-rater reliability among human coders is about 94%.

Our experiments show that the results obtained from automatic IPSyn scoring using our GR analysis for CHILDES data are only slightly less reliable than human scoring, and much more reliable than scoring based on part-of-speech analysis alone. This validates not only our analysis approach, but also our CHILDES GR annotation scheme.

1.4 Thesis Contributions

The major contributions of this thesis are:

- A scheme for annotating syntactic information as grammatical relations in transcripts of child-parent dialogs focusing on information relevant to the study of child language;
- A linear-time classifier-based deterministic parsing approach for constituent structures and dependency structures;
- The development of rule-based and data-driven approaches to grammatical relation identification in transcribed spoken language;
- A novel methodology for combining grammatical relation analysis from multiple

parsers through a process of reparsing, taking into account the specific strengths of individual parsers and guaranteeing well-formed structures;

- A general approach to syntactic analysis of a language genre for which there is not much training data available;
- The demonstration of the effectiveness of the GR annotation scheme and GR identification approach through a task-based evaluation;
- An accurate automated tool for automatic measurement of syntactic development in child-language.

Chapter 2

Overview, Data, and Previous Work

Natural language syntactic parsers use computational models to analyze the syntactic structure of natural language. Parsing is a complex problem, and several different approaches have demonstrated some amount of success in different aspects of the task. Parsers differ with respect to what underlying algorithm they use to determine syntactic structures, what kind of syntactic structures they provide, whether they output one or more syntactic structures for a single sentence, among other things. One characteristic that varies significantly across several parsing approaches is how much the system relies on linguistic knowledge encoded as rules, such as with a manually-written grammar (often with a lexicon and a domain model), and how much the system learns from manually annotated examples (sentences with their correct syntactic structures). The multi-strategy approach to syntactic analysis described in this dissertation aims to take advantage of the existence of several accurate parsers, in an attempt to produce syntactic analyses that are more accurate than what each of the parsers in isolation is capable of producing. The idea of combining the output of several systems that perform the same task differently has been explored at length by the machine learning community. The multi-strategy parsing approach is closely related to ensemble methods (Dietterich, 2000) in machine learning, and can be viewed as a *parser ensemble* approach. This is not surprising, given that several recent syntactic analysis efforts have successfully framed the parsing task as essentially a machine learning problem (Henderson, 2004; McDonald et al., 2005; Nivre and Scholz, 2004; Sagae and Lavie, 2005; Yamada and Matsumoto, 2003). While this is how we view parsing with respect to the data-driven approaches described in chapter 5, our parser ensemble also takes advantage of

a rule-based parser (described in chapter 4) that has significantly different characteristics than those of data-driven parsers. As observed by Henderson and Brill (1999), the strength of parser combination schemes is in parser diversity. To be successful, our parser ensemble strategy should include several different parsers that operate on the same input data.

In this chapter we present an overview of our multi-strategy approach to identification of grammatical relations in child language transcripts, starting with background information, an informal argument for why combining different systems should be productive, and a description of the data used in the experiments described throughout this dissertation. Finally, we discuss previous research that is relevant to work presented here.

2.1 Background

Typically, the input unit that most parsers deal with is a sentence, and analysis is performed one sentence at a time. This is the case in all of the work described here, but our definition of sentence is somewhat loose, also including complete utterances that do not include a main verb (e.g. noun phrases, prepositional phrases, and other sentence fragments). At times we may refer to sentences as utterances. Unless explicitly indicated, these terms will be used interchangeably. In some cases, parsers expect the input sentence to be tagged with part-of-speech (POS) tags, such as NOUN, VERB, ADJECTIVE, etc. Once a parser is given a sentence as input (possibly with POS tags), it attempts to produce a syntactic structure of the sentence as output. The syntactic structure may take several forms, such as a constituent tree (also called c-structure, phrase structure, or simply parse tree), a syntactic feature structure (such as a functional structure, or f-structure), or a dependency structure. Figure 2.1 shows examples of each of these syntactic representations using the same sentence. These various representations of a sentence may differ in the level of information they contain (parts-of-speech, morphology, case, syntactic function labels, etc.), but each of them describes in some way the structure of how words combine to form a sentence. The choice of a particular representation format depends mainly on the purpose the syntactic analyses will serve. In this work, we use dependency structures to represent syntactic information, since they are particularly suited to our task. As in the example in figure 2.1, we use labeled dependencies to represent grammatical relations, such as subjects, objects and adjuncts. This is discussed in more detail in chapter 3. Constituent structures are also of interest, since much of the related past research on parsing has focused on constituent parsing. In particular, working with constituent structures allows for comparisons between existing parsing techniques and some of our novel parsing approaches (see chapters 5 and 6). In addition, constituent structures, as discussed later, can be easily converted into dependency structures.

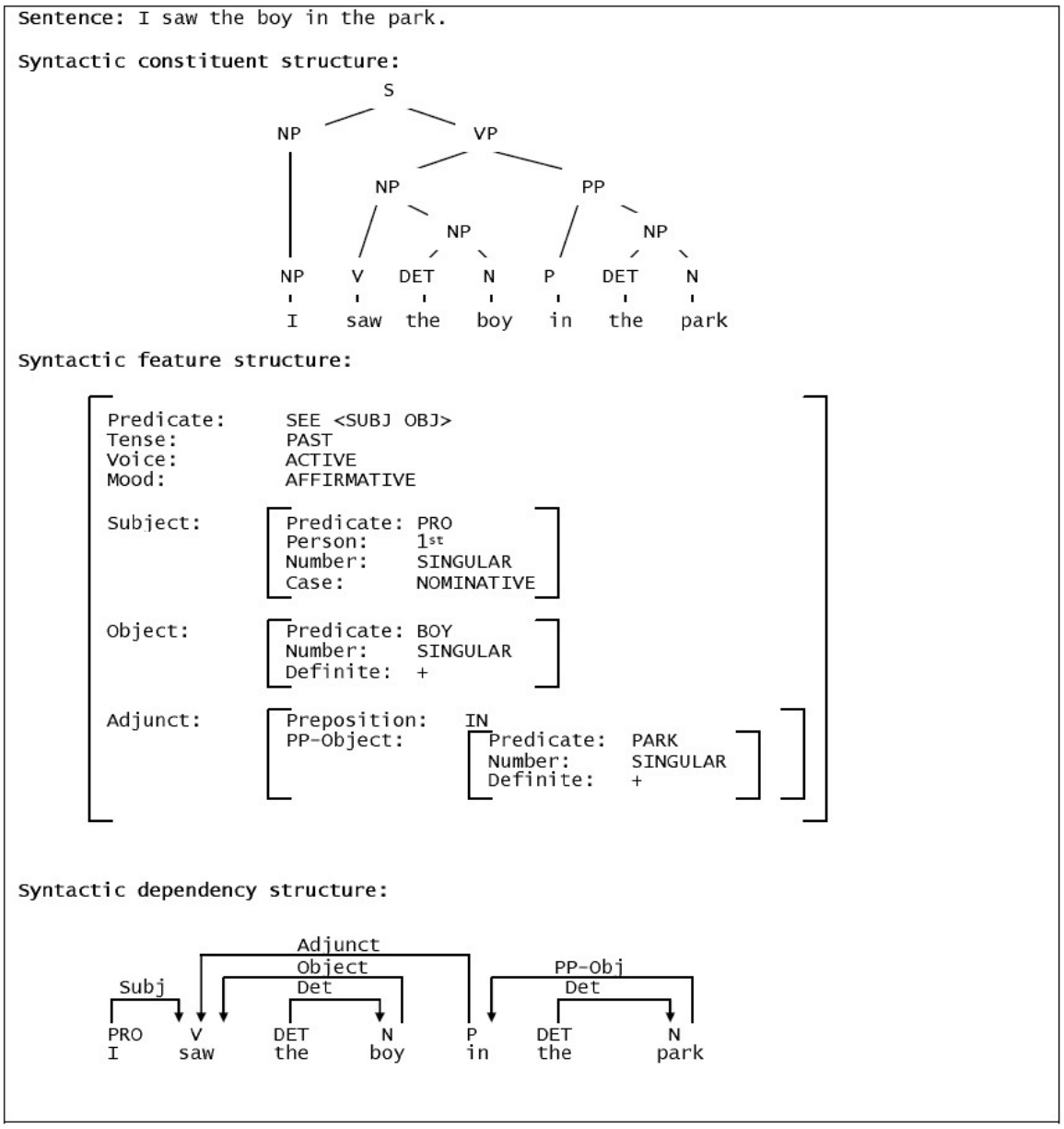


Figure 2.1. Three syntactic representations for a sentence.

2.1.1 Dependency parser evaluation

Dependency parsing has emerged as an attractive way of performing automatic syntactic analysis over the past decade (Eisner, 1996; McDonald et al., 2005; Nivre and Scholz, 2004; Yamada and Matsumoto, 2003). In dependency parsing, syntactic structure is represented by a dependency tree, where each word is a node, and the existence of an edge between two nodes indicates a syntactic relationship between the two words corresponding to the

two nodes. Of these two words, the one closer to the root of the tree (in terms of the dependency graph, not the linear order of the words in the sentence) is referred to as the *head* and other as the *dependent* in this particular syntactic dependency relationship. In addition to the head and dependent words, each of these relationships, or dependencies, also includes a *dependency label*, typically used to indicate the type of syntactic relationship that holds between the two words. If these labels are ignored, we are left with what Eisner (1996) calls barebones dependencies, and we will refer to as simply *unlabeled dependencies*.

Dependency structures are described further in chapter 3, but for now, with respect to how we evaluate dependency parsers in this dissertation, it suffices to say that a dependency structure (or tree, or graph) for a sentence with n words can be defined by simply listing n triples *[head, dependent, label]*. Each of these triples is an individual dependency. Typically, parser evaluation requires a test corpus of several sentences annotated with correct (gold-standard) syntactic structures. The most common way to evaluate a dependency parser is to determine the percentage of correct individual dependencies produced by the parser when it analyses the sentences in test corpus. A dependency produced by the parser is considered correct if the head, dependent and label of the dependency are identical to those of a dependency in the gold-standard dependency annotation for the same sentence in the test corpus. This evaluation metric is called *dependency accuracy*. Sometimes the unlabeled dependency accuracy is used. It is obtained in a similar way, but ignoring the dependency labels (only the head and dependent are used to determine if a dependency is correct).

Dependency accuracy is a convenient way to determine a single number that reflects the overall accuracy of a parser. A more detailed evaluation scheme is to consider *precision* and *recall* of individual dependency types (where a type is simply what dependency label the dependency has), in a similar way to the grammatical relation parser evaluation scheme proposed by Carroll et al. (2003). To calculate the precision and recall of a specific dependency for a parser, first we let c be the number of correct dependencies of that type in the output of a parser for the test set. Precision is calculated by dividing c be the total number of dependencies of that type produced by the parser for the test set. Recall is calculated by dividing c by the total number of dependencies of that type in the gold-standard annotations for the test set. Note that both precision and recall are computed over aggregate counts over the entire test set. It is sometimes useful to combine precision and recall into a single number. We do this by computing the *f-score*, or harmonic mean of precision and recall, specifically $(2 * precision * recall)/(precision + recall)$.

2.1.2 Constituent parser evaluation

Constituent structures continue to be a popular output representation for syntactic parsers. In fact, the term "parse tree" is often used interchangeably with constituent structure or constituent tree in natural language processing. Although in this dissertation we are concerned primarily with dependency structures, some of the more well-known related work involves constituent parsing. For that reason, we will in a few occasions apply some of the ideas developed in this dissertation to constituent parsing. This allows their performance to be compared to other work. To that end, we will use the constituent precision and recall measures as defined in the PARSEVAL (Black et al., 1991) scheme for constituent parser evaluation. To describe how the precision and recall measures are computed for constituents, first consider a constituent tree with n nodes. This tree can be represented as a list of n constituent triples *[label, begin, end]* (one triple corresponding to each node), where *label* is the node label of the corresponding tree node, *begin* is the index in the sentence of the leftmost word in the subtree rooted at that tree node, and *end* is the index of the rightmost word in that subtree. We consider a constituent triple to be correct if the *node*, *begin* and *end* in the triple are identical to those of a triple in the gold-standard annotation for the same sentence in the test corpus. Now, let c be the number of correct constituent triples the output of a parser for the test set. Precision is calculated by dividing c be the total number of constituent triples produced by the parser for the test set. Recall is calculated by dividing c by the total number of constituent triples in the gold-standard annotations for the test set. As with precision and recall for dependencies, constituent precision and recall are computed over aggregate counts over the entire test set, and a single number combining precision and recall can be obtained by taking their harmonic mean (f-score)[1].

While the evaluation of dependency parsers has been fairly uncontroversial, some have identified weaknesses in the PARSEVAL evaluation scheme for constituent parsers and proposed alternatives (Carroll et al., 1998, 2003; Lin, 1998; Sampson and Babarczy, 2003). It is clear that PARSEVAL is fairly coarse-grained compared to, for example, precision and recall of individual dependency types. It provides little information about specific strengths of parsers, and even as a measure of overall accuracy, PARSEVAL is still less well-motivated than overall dependency accuracy, since it really measures the accuracy of *bracketing* (essentially, identifying constituent boundaries), and not the correctness of the constituents themselves in relation to other constituents. On the positive side, PARSEVAL is easy to compute for systems that use the same constituent representation scheme, and it has been used extensively to evaluate parsers using the same training and test sets, allowing for a direct comparison of different parsing approaches. Despite its shortcomings, we use

[1]The PARSEVAL evaluation scheme also includes a measure for crossing brackets. Since that measure has not gained as much popularity in parsing research, we focus only on constituent precision and recall.

PARSEVAL exclusively for evaluation of constituent parsers, since it is still the de facto standard for constituent parser evaluation in the computational linguistics community, and our use of constituent parser evaluation is secondary to the development of techniques for grammatical relation identification.

2.1.3 The role of annotated training data in parsing

Natural language processing researchers have made use of a number of existing models and theories of language to develop systems that perform syntactic parsing with increasingly high levels of accuracy. The choice of how to represent syntactic information is just one of many distinguishing characteristics of different parsers. Another aspect that offers great variation in different approaches to parsing is the amount of (manually annotated) training material used by the system. By *training material* we mean a set of sentences annotated with correct syntactic structures. Often, parsers are built so that their outputs follow the same representation as their training data, but this is not always the case. For example, parsers trained on constituent structures may output dependency structures, and parsers trained on dependency structures may output functional structures, often relying on rules to go from one representation to another. To situate the parsing work described in the following chapters more clearly, we can imagine a spectrum of parsing approaches based on how much each different approach relies on annotated training data. At one end of this spectrum we have systems use no training material and rely solely on a set of linguistic principles, or on a carefully constructed grammar (sometimes combined with a rich lexicon, and possibly other knowledge sources or heuristics). At the other end, we have systems use no encoded a priori knowledge and learn their models of language based only on annotated training examples.

One intuitive way of thinking about how central the use of a training corpus is to a parsing approach is to consider the task of arriving at a syntactic parse for a sentence as a search problem. In a rule-based system, the search space is limited to syntactic structures that can be formed according to the rules provided to the system (such rules might include a grammar, a lexicon with detailed lexical features that interact with the grammar, a domain model that further restricts how grammar rules might be used, etc.). The more specific and restrictive these rules are, the smaller a search space it describes. With less restrictive rules, the search space increases accordingly, and learning syntactic disambiguation becomes a greater challenge. This typically causes ambiguity to increase, requiring the system to rely more on a disambiguation strategy to choose between competing analyses for the input. One such disambiguation strategy is to use a statistical model with parameters determined according to labeled training examples. In purely data-driven systems, having no grammar can be thought of as having a grammar that includes every possible rule over the set of

nonterminals (the least restrictive grammar possible in terms of search space). The search space of syntactic structures in that case is overwhelming in size even with a relatively small set of nonterminals, as any sequence of words could, without syntactic disambiguation, be assigned any possible structure. As Charniak (1997) points out, in this case the problems of parsing and disambiguation are in fact the same problem, as opposed to how the two tasks are often approached separately in the rule-based framework. One aspect of data-driven approaches is that they are inherently robust, as any sentence, no matter how unpredictable or unlikely, can be assigned some syntactic structure. This is appealing if there is an assumption that every sentence given to the parser is grammatical. This is a reasonable assumption for many practical settings where parsing would be applied. On the other hand, the concept of *ungrammaticality*, relatively prominent in rule-based parsing, becomes trivially nonexistent.

Our first approach to syntactic analysis of parent-child dialog transcripts, described in chapter 4, uses rule-based parsing. More specifically, it uses a manually written grammar that operates with part-of-speech (POS) tags as terminals, and a lexicon and a POS tagger that maps words to possible parts-of-speech. Although some systems that fit that description belong at the no-training-data end of the spectrum, in practice systems may use training examples for creating a model to perform disambiguation of the different competing possibilities allowed by the parser's rules. This is the case with the work we present in chapter 4. We then consider two data-driven approaches in chapter 5 that make no use of manually written rules, and instead automatically build a model from a large number of manually annotated examples. One of these data-driven systems uses an existing parsing approach that is an extension of lexicalized probabilistic context-free grammars with a generative statistical model trained on constituent structures. The other is based on a classifier-based approach and trained directly on dependency structures, using discriminative techniques.

2.2 The multi-strategy parsing approach

As its title suggests, the central theme in this dissertation is the identification of grammatical relations in transcripts of child-parent dialogs. To this end, we employ a multi-strategy approach, using several different parsers and combining their analyses in an attempt to achieve improved accuracy. To understand the motivation behind a parser combination approach, or ensemble parsing, let us consider the basic machine learning task of *supervised classification*. In a classification problem, a classifier (learner) is given a set of training examples, each taking the form of a vector of features associated with class. The classifier uses a learning algorithm to approximate a function that maps the input feature vectors to appropriate classes, ideally generalizing to feature vectors not seen as training examples.

In machine learning, the idea of combining the output of multiple classifiers to produce a classifier ensemble that is in general more accurate than the individual classifiers has been the subject of much research (Breiman, 1996; Dietterich, 2000; Krogh, 1995; Wolpert, 1992), which has determined that the general idea has both empirical and theoretical merits. A discussion of classifier ensembles is beyond the scope of this thesis, but just as a way to capture a general intuition for why combination may be productive, consider the following simplistic scenario: if we have several classifiers performing the same classification task with a reasonable level of accuracy, and the errors made by each classifier are independent of the errors made by the others, we can expect that a simple majority vote between the classifiers may be more accurate than the output of each of the individual classifiers. It is this intuition that motivates our parser ensemble approach. In the syntactic analysis task, a parser takes a sentence as input, and produces a syntactic analysis as output. If we have several different parsers that produce (possibly) different analyses for the same input sentence, we expect that there may be a way to combine these analyses to arrive at a more accurate final analysis, if the errors made by each of the parsers are independent enough from errors made by the other parsers. Figure 2.2 shows a high-level view of our multi-strategy parsing approach.

One complication in applying an ensemble strategy to the parsing task is that the output of a parser is typically a more complex structure than a simple class label. In our case, parsers output dependency trees. While simple class labels produced by classifiers can be combined in a number of straightforward ways (such as majority voting), combining complex structures is more challenging. Finding a way to combine such structures is one of the main issues in this thesis. In chapter 6, we present ways to combine dependency structures effectively, and also extend our approach to constituent structures.

2.3 Data

Now we turn our attention to the data used in this research. Our main source of data is the CHILDES database (MacWhinney, 2000), which, as mentioned in chapter 1, contains transcripts of verbal interactions between children and adults. We also use the Penn Treebank (Marcus et al., 1993), which contains a large amount of text annotated with corresponding syntactic structures in the form of constituent trees.

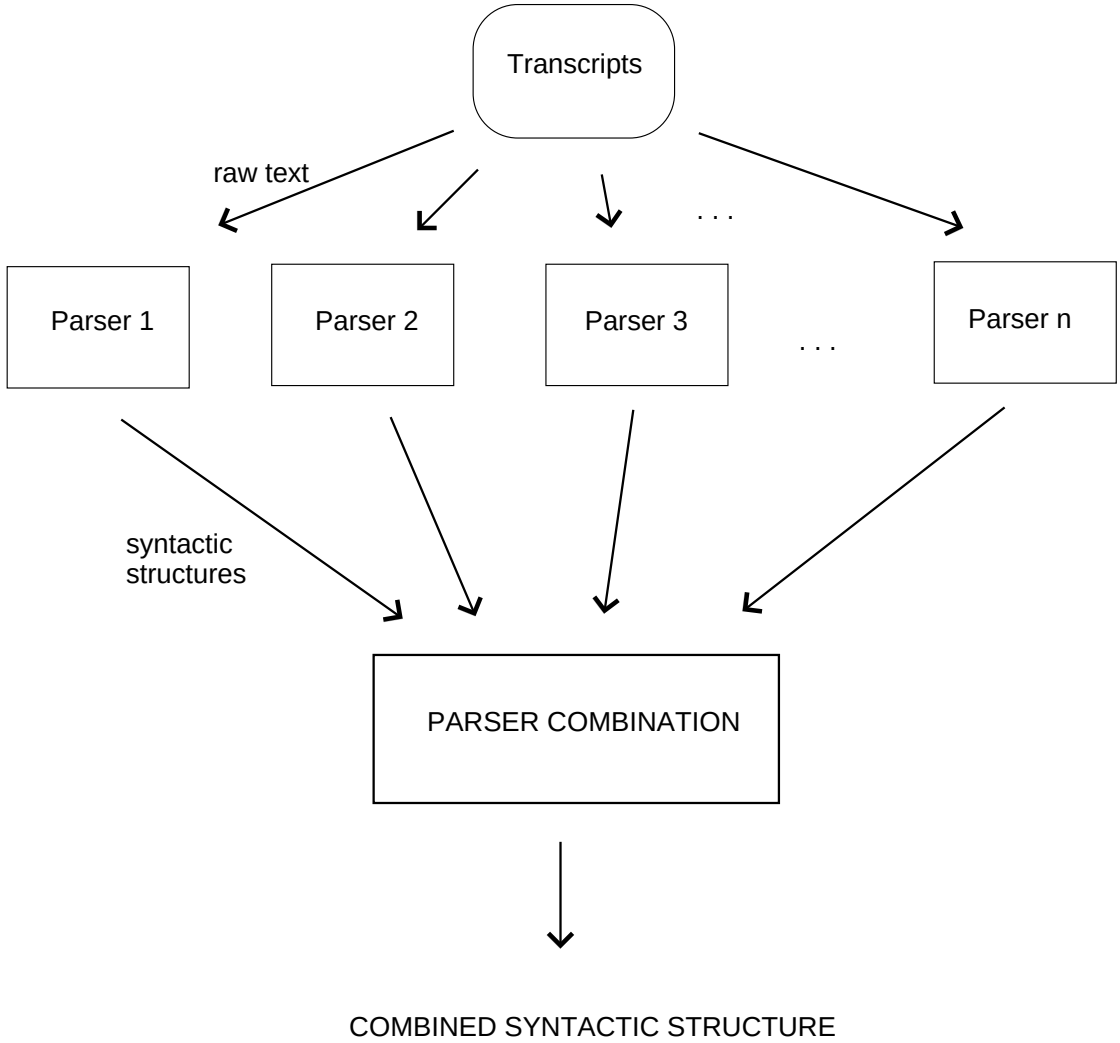


Figure 2.2. A general parser combination scheme.

2.3.1 The CHILDES Database

The Child Language Data Exchange System, or CHILDES (MacWhinney, 2000), was founded in 1984 and consists of data and tools for studying conversational interactions. The CHILDES database is the collection of data in the CHILDES project. The database contains audio, video and transcripts of verbal interactions between children and adults, and is the standard source of data for corpus-based child language research. There are now more than 1,500 published articles based on CHILDES data. CHILDES also includes a number of tools for processing the data, and these are organized in a software package called CLAN (Computerized Language Analysis). These tools include utilities for browsing transcripts (text), performing complex searches on the text data, tagging transcripts with

parts-of-speech and morphology, as well as utilities for processing audio and video data. For the purposes of this thesis, we use only the English text portion of the CHILDES database (although much of the work presented here could be extended to other languages in a straightforward manner). CHILDES transcript data follows the CHAT transcription format (MacWhinney, 2000), which includes explicit markers for spontaneous speech disfluencies such as pauses, retracings, repetitions and false starts.

To train, develop and test syntactic parsers for CHILDES transcripts, we have annotated approximately 8,000 words from Brown's Eve corpus (Brown, 1973) with grammatical relations (GRs) according to the GR annotation scheme described in chapter 3. Among the many corpora in CHILDES, we chose Brown's Eve corpus because it has been extensively studied (Brown, 1973; Moerk, 1983), and (perhaps for that reason) its transcriptions are cleaner than those of most of the other corpora in CHILDES. We call this corpus the CHILDES GR Eve corpus, and it is divided into five parts: (1) a 5,000-word set drawn exclusively from utterances spoken by adults to be used as a training set, (2) a 1,200-word set from utterances spoken by adults to be used as a test set, (3) an 800-word set from utterances spoken by Eve (the child) to be used as a test set for parsing of child utterances, (4) a 600-word set from adult utterances to be used as development data, and (5) a 400-word set from child utterances to be used as development data.

2.3.2 The Penn Treebank

The Penn Treebank (Marcus et al., 1993) is a large collection of text annotated with constituent structures. As of version 3 (released in 1999), the Penn Treebank includes text (with constituent trees) from Wall Street Journal (WSJ) stories from 1989, the Brown corpus (a balanced corpus), the ATIS corpus (air travel information), and the Switchboard corpus (transcripts of phone conversations). The WSJ section of the Penn Treebank contains syntactic structures and part-of-speech tags for about one million words from WSJ stories, and it has been heavily used in parsing research since its release (Charniak, 2000; Collins, 1996; Magerman, 1995; Ratnaparkhi, 1997). It has shaped much of the state-of-the-art in statistical parsing, and provided a common set of training and test corpora for the evaluation of data-driven parsers. Because of its widespread use, whenever it is productive to do so, we evaluate our work using what have become the standard sections for training (WSJ section 02 to 21), testing (WSJ section 23) and development (WSJ section 22). This allows for a direct comparison of our work to other well-known parsing approaches.

Although syntactic structure in the Penn Treebank is represented as constituent trees, a procedure for converting such structures into unlabeled dependency trees exists. It was first used on the WSJ Penn Treebank by Magerman (1995), and subsequently used by

nearly all work on lexicalized parsing using the Penn Treebank. The work of Collins (1996) helped popularize it, especially because its details were made public. The procedure was originally intended to lexicalize the constituent trees, adding a head-word to each tree node. The head-word associated with the tree node is one of the words dominated by that node. These head words are chosen according to a head percolation table, which is simply a set of rules that determine which word in a constituent should be its head. Heads are determined for each constituent bottom-up, so that when a head is chosen for a node, each of its children already has a head (and one of these heads is the one chosen as the current node's head). By lexicalizing the constituent tree, we automatically create an unlabeled dependency tree. The specific head rules used in our experiments are the same as the rules used by Yamada and Matsumoto (2003), Nivre and Scholz (2004), and McDonald et al. (2005). The complete head table is shown in Appendix A.

2.4 Related Work

A wide variety of approaches to syntactic analysis of natural language have been proposed and developed through the years. In this section, we review a few that are closely related to the work described in this dissertation. Only work that is related in a general sense to the main syntactic analysis task in this thesis is described here, while work that has more specific relevant aspects are mentioned throughout the description of our work where the discussion is more focused. Here, we cover rule-based and data-driven systems, generally moving from approaches that rely more on rules towards approaches that rely more on data. We then turn to previous parser combination schemes.

2.4.1 Related work in rule-based and data-driven parsing

The Generalized LR parser/compiler (Tomita, 1990) is a natural language parsing system that implements Tomita's GLR algorithm for context-free grammars augmented with unification equations (Tomita, 1987). The GLR algorithm itself is an extension of the LR algorithm that handles context-free grammars. Tomita's system uses a grammar formalism where each rule has a context-free backbone and a set of unification equations that operate on feature structures associated with each of the nonterminals in the context-free portion of the rule. The grammar formalism and its unification mechanism closely resemble Lexical Functional Grammar, or LFG (Kaplan and Bresnan, 1982), although the grammars themselves are not required to follow any linguistic theory in particular. The system produces as output a syntactic feature structure corresponding to the input sentence. Often, many of the features used in unification-augmented CFGs are the same types of grammatical

relations we investigate here. The GLR* parser (Lavie, 1996) extended Tomita's GLR algorithm to give it robustness features to handle spontaneous spoken language. Using the same grammar formalism, the GLR* algorithm allows the parser to skip portions of the input sentence and/or insert lexical or phrasal material into the input sentence according to a predefined set of parameters. These modifications to the original GLR parser allow GLR* to find analyses for input sentences where dropped words, filled pauses, retracings, and repetitions cause the sentence to deviate from the grammar used by the parser. In addition, a probabilistic model similar to that of Carroll (1993) and Briscoe and Carroll (1993) for GLR parsing was added to perform disambiguation. This probabilistic GLR model essentially associates probabilities with actions in the GLR table, and the probabilities associated with the actions in a derivation are multiplied to determine the probability of the resulting syntactic structure. Carroll points out that GLR tables can encode partial context information in states, so the probabilistic GLR model is capable of making distinctions that cannot be captured in a probabilistic context-free grammar (PCFG). Rosé and Lavie (2001) used the same framework for rule-based robust parsing with an LFG-like formalism as GLR* to create LCFlex, an efficient and robust active chart parser with left-corner predictions. Disambiguation in LCFlex is done with a statistical model of bigrams of context-free rules (see chapter 4). As is the case with the GLR* parser, the probabilistic model is used to disambiguate among possible analyses allowed by the grammar for a sentence. Although lexical information can be taken into account during parsing, in the form of features encoded in the lexicon and used in unification equations, the statistical model is unlexicalized. Exhaustive parsing is performed first, and each complete analysis is scored according to the disambiguation model in a separate pass. Our grammar-driven approach to GR identification is heavily based on LCFlex, and a more detailed description of that parser, including the grammar formalism also used in Tomita's GLR and Lavie's GLR* parsers, can be found in chapter 4. Like its predecessors, GLR and GLR*, LCFlex outputs feature structures corresponding to input sentences. Depending on how the grammar defines what feature structures can be produced, grammatical relations can be easily extracted from these feature structures simply by reading the relations themselves and their participating words in the feature structure.

Briscoe and Carroll (2002) present a mature parsing approach that has been thoroughly evaluated on identification of grammatical relations. Although their parser also uses a manually written unification-based grammar, it performs disambiguation using not only an unlexicalized probabilistic GLR model trained on a 100,000-word treebank, but also lexicalized subcategorization information extracted *automatically* from 10 million words (Briscoe and Carroll, 1997). This system, therefore, is significantly farther towards the data-driven side of the spectrum than the others mentioned here up to this point, although the text used to train the parser's subcategorization model was not manually annotated. Using

a broad-coverage phrase-structure grammar of English, the parser first finds constituent trees, and then rules are applied to convert those into GRs according to Carroll et al.'s GR scheme (Carroll et al., 1998). The parser identifies GRs with high accuracy for a largely domain-independent parser. In addition, Carroll and Briscoe (2002) showed that the balance between precision and recall of GRs can be manipulated so that very high precision can be achieved at the expense of recall.

A large number of data-driven parsing approaches that include no manually written grammar have been proposed. Several of these data-driven systems have in common the fact that parsing is framed as a learning task, and syntactic analysis is learned from manually annotated treebanks. While a variety of different approaches have shown success, such as the Data-Oriented Parsing model, or DOP (Bod et al., 2003), neural network parsing (Henderson, 2004), and decision-tree parsing (Magerman, 1995), to name just a few, we will focus our attention on approaches that are most closely related to our work.

Collins (1996, 1997) introduced statistical parsing models that have heavily influenced data-driven parsing. Collins used generative models with parameters estimated from the Penn Treebank using maximum-likelihood. Although these models were based on the general notion of probabilistic context-free grammars, a significant advancement was the introduction of generative parsing using markov-grammars[2], where instead of storing explicit PCFG rules, a parser uses a probability distribution of rules so that a probability can be assigned to any rule made of valid nonterminals. This way, PCFG rules read straight from constituent trees in the treebank are generalized into a data-driven grammar. In addition, that work also popularized the use of lexicalized PCFG-like models, where head words are associated with each node in a constituent tree. Although the Penn Treebank does not contain lexicalization information, Collins used a technique employed by Magerman (1995) to lexicalize constituent trees. This involves the use of a set of rules (a head table) that describes how to determine head words for specific constituent types based on the children of a tree node. An interesting part of the Collins model is that it stressed the importance of the word-to-word dependencies implicit in the lexicalized trees. These are very similar dependencies to the kind we use for representing GRs. Although Gildea (2001) later showed that the high accuracy of the Collins model was not due to the use of such bilexical dependencies as previously thought, the dependency models of Eisner (1996) indicate that such dependencies extracted from a treebank can in fact be used for high accuracy parsing on their own. Using markov-grammars introduces the complication that the search space of possible parses becomes extremely large. Collins deals with this by using a probabilistic version of the CYK parsing algorithm (Cocke and Schwartz, 1970; Kasami, 1965; Younger, 1967) with a beam-search, where only the most probable constituents are considered. In addition, estimation of parameters in lexicalized grammars is complicated by data-sparseness.

[2]The term *markov-grammar* was coined later by Charniak (1997).

Collins's approach to this problem was to use a linear combination of back-off estimates, where less specific statistics using part-of-speech tags that can be estimated more reliably are used as back-offs from the more fine-grained lexical statistics.

Charniak (2000) presents a statistical parser trained and tested on the same Penn Treebank data as Collins (1997), obtaining even more accurate results. Like Collins, Charniak uses the data-driven generative lexicalized PCFG-like framework of markov grammars. The main difference between the models of Collins and Charniak is that the latter uses additional information for conditioning probabilities and a different estimation technique that allows such information to result in higher parse accuracy. Charniak calls his model "maximum-entropy inspired" because the way different features are incorporated into the model resemble maximum-entropy models. Charniak does not actually perform maximum-entropy training or feature weight optimization, but he makes the observation that being able to represent conditioning events in terms of features makes it easier to test and incorporate different events into the model. One of our approaches to data-driven GR extraction uses the Charniak parser directly, and a head percolation table to lexicalize its output and extract a dependency structure.

A different approach to data-driven parsing is to use discriminative training instead of generative models. While some discriminative parsers treat the syntactic analysis task as structured prediction (Charniak and Johnson, 2005; Collins and Koo, 2003; McDonald et al., 2005; Taskar et al., 2004), the approaches that are more closely related to our work are those that deal with local parse decisions. Yamada and Matsumoto (2003) introduced the idea of performing *deterministic* unlabeled dependency parsing using a classifier to determine the actions of a bottom-up parser. Their parser achieved high dependency accuracy with an algorithm of quadratic run-time complexity and support vector machines (SVMs) for parser action classification. In a nutshell, the algorithm consists of moving a sliding window of two words over the input string from left to right, and at each position deciding whether there is a dependency with the word on the left as the head, with the word on the right as the head, or no dependency. If a dependency is found (and no further dependencies are expected to have the dependent word as a head), the dependent word is removed from the input string. Once the moving window reaches the end of the sentence, the process starts again at the beginning of the sentence. Parsing ends when a single word is left in the input string (and a complete dependency structure is built with the remaining word as the root), or no more dependencies can be formed (in which case the input string is rejected, although partial dependency structures can be recovered). The classifier used to make the parsing decisions is trained by running the algorithm on sentences for which the dependency structure is known in advance, and associating a set of features from the parser's configuration with the correct action. Yamada and Matsumoto used dependencies extracted from the Penn Treebank using a head percolation table. Features used for classification include the words

in the sliding window and their parts-of-speech (POS), words to the left and right of the window (with POS), and already determined dependents of the words in the window.

Nivre and Scholz (2004) introduced a related parser with lower accuracy than that of Yamada and Matsumoto, but with an algorithm that is linear on the size of the input string and handles *labeled* dependency parsing. Nivre and Scholz use a simple left-to-right stack-based shift/reduce algorithm. They used a memory-based learner (Daelemans et al., 2004), and features include the word on top of the stack and its POS, the next word in the input string and its POS, the next n words in the input string (with POS), and the previously determined dependents of the word on top of the stack. Unlike Yamada and Matsumoto's parser, where the only possible local decisions were to shift or create an unlabeled dependency, Nivre and Scholz's parser is actually designed for labeled dependencies, where the classifier also decides the dependency labels, and the labels can also be used as features. This results in having several different actions for the classifier to choose from. Training of the classifier is also done by running the algorithm on sentences with known dependency structures, and associating the correct parser actions with features that correspond to the parser's current state.

2.4.2 Related work in parser combination

Henderson and Brill (1999) introduced the idea of taking advantage of the existence of several accurate statistical constituent parsers trained on the same data from the Penn Treebank to combine their results in an attempt to produce more accurate analysis. In Henderson and Brill's parser combination work, three Penn Treebank parsers were used, and two general approaches were considered: parser hybridization and parser switching. In parser switching, a complete tree generated by one of the three parsers is chosen as the final tree. This decision can be made using a similarity metric, where the tree with most constituents in common with the other trees was chosen, or using a Naive Bayes classifier. Parser hybridization performed consistently better than parser switching, and it is the approach more closely related to our reparsing approach. Parser hybridization is the constituent equivalent of the simple voting scheme described for dependencies in the beginning of section 2. The idea is to first decompose the tree into constituents, and then select a constituent as part of the final combined tree if it appears at least $(m + 1)/2$ times, where m is the number of parsers used in combination. In other words, the combined parse tree contains a specific constituent if more than half of the m initial parse trees contain that constituent. Parametric and non-parametric versions were tested, with identical results. Although Henderson and Brill did get significantly more accurate parses using parser hybridization, the approach heavily favors precision. They obtained 92.1% precision and 89.2% recall (90.61 f-score) on the standard test section of the WSJ Penn

Treebank (section 23), which was well above the results obtained with the best parser they used (88.6% f-score).

Zeman and Žabokrtský (2005) applied the ideas of Henderson and Brill to dependency parsing of Czech. They confirmed that parser hybridization by voting is also an effective combination approach for dependencies. When m parsers each output a set of dependencies (forming m dependency structures) for a given sentence containing n words, it is easy to combine these dependency structures through a simple word-by-word voting scheme, where each parser votes for the head of each of the n words in the sentence, and the head with most votes is assigned to each word. However, the application of straight-forward voting to dependencies may result in invalid dependency structures, such as disconnected graphs or structures containing loops. Zeman and Žabokrtský deal with this problem using a variation of the parser switching idea, but no search for an optimal valid structure is performed, and a significant drop in accuracy results from enforcing constraints that ensure the creating of a valid dependency tree.

While the two combination approaches mentioned above are instances of parser voting, Yeh (2000b) explores a combination scheme that can roughly be described as parser stacking. First, the input sentence is analyzed by two existing GR parsers (Briscoe and Carroll, 2002; Buchholz et al., 1999). These two parsers use different representations for grammatical relations, and the representation desired as the output of the final system is different still than the ones for either initial parser. A classifier trained on a relatively small corpus is then used to determine the final analysis for the sentence, using as features the GR predictions made by the two initial parsers. This way, Yeh takes advantage of existing systems to create a new parser in a different domain (using a different GR set) using only a small training corpus.

Chapter 3

The CHILDES GR Annotation Scheme

One crucial aspect of producing useful automatic syntactic analysis for child-parent dialog transcripts is the definition of a suitable annotation scheme that targets the specific type of syntactic information needed by the child language community. To address this need, we have developed the CHILDES Grammatical Relation (GR) annotation scheme, described in this chapter.

3.1 Representing GRs with Labeled Dependencies

We represent syntactic information in CHILDES data in terms of *labeled dependencies* that correspond to *grammatical relations* (GRs), such as subjects, objects, and adjuncts. As in many flavors of dependency-based syntax, each GR in our scheme represents a relationship between two words in a sentence (or utterance, as mentioned in chapter 2): a *head* (sometimes called a parent or regent) and a *dependent* (sometimes called a child or modifier). In addition to the head and dependent words, a GR also includes a *label* (or GR type) that indicates what kind of syntactic relationship holds between the two words. Each word in a sentence must be a dependent of exactly one head word (but heads may have several dependents)[1]. The single exception to this rule is that every sentence has one "root" word that is

[1]This is a common constraint in dependency analysis, but one that disallows natural representation of certain syntactic structures. For example, in the sentence "he wants to leave," we may want to indicate that the word *he* is the subject of both *wants* and *leave*. However, it can only be a dependent of one word (in this case *wants*, and no subject is given to *leaves*). The advantage of enforcing this constraint is that

not a dependent of any other word in the sentence. To achieve consistency across the entire sentence (with each word having exactly one head), we make the root word a dependent of a special empty word, appended to the beginning of every sentence. We call this empty word the "LeftWall"[2]. Figure 3.1 shows the syntactic annotation of two sentences.

To describe the properties and constraints of syntactic structures defined in our annotation scheme, it is useful to think of these structures as graphs, where the words are nodes, and directed edges exist between each dependent-head pair, from the dependent to the head. A well-formed dependency structure must be connected. More specifically, from any word in the sentence (any node in the graph), there is a path to the root word (the single word in the sentence that is a dependent of the LeftWall). In addition, a well-formed dependency structure must be acyclic. Therefore, if we ignore the LeftWall and the directionality of the edges, a syntactic structure in this scheme is in fact a tree rooted at the root word.

One additional property that can be defined for dependency trees in our annotation scheme is projectivity. An in-order traversal of a projective dependency tree must list the words in the same left-to-right order as in the original sentence. In other words, a projective tree drawn above the sentence must have no crossing-branches. Syntactic structures in our scheme are *not* required to be projective. In fact, annotation of languages with free word order should produce dependency trees that frequently exhibit non-projectivity. English sentences rarely do[3], and most of the recent work on lexicalized syntactic parsing of English (using constituent trees or dependencies) assumes projectivity (e.g. Rathnaparkhi, 1996; Eisner, 1996; Collins, 1996; Charniak, 2000; Nivre, 2004). There are parsing approaches that deal with non-projectivity (McDonald et al., 2005; Nivre and Nilson, 2005), especially for parsing languages where non-projectivity is more common. Because in this thesis we are working exclusively with English, syntactic structures will be assumed to be projective, although this is not a constraint imposed by the annotation scheme.

3.2 Specific GR Labels

The specific set of GR types (or labels) included in our annotation scheme was developed by using the GRs in the annotation scheme of Carroll et al. (2003) as a starting point, and adapting the set to suit specific needs of child language research. Sources of refinement

the resulting annotation scheme is simpler to understand, annotate, and produce automatically. It should be noted that the annotation scheme could be extended with additional layers, with the existing annotation being a more superficial layer, and an additional layer representing control structures, co-reference, etc.

[2] We borrow the term LeftWall from Link Grammar (Sleator and Temperley, 1991), another dependency-based formalism. This is similar to the EOS word used in the definition of bare-bones dependency in Eisner (1996).

[3] Certain syntactic structures in English are most naturally represented with non-projective trees. These include cases of discontinuous constituents and wh-questions with dangling prepositions, for example.

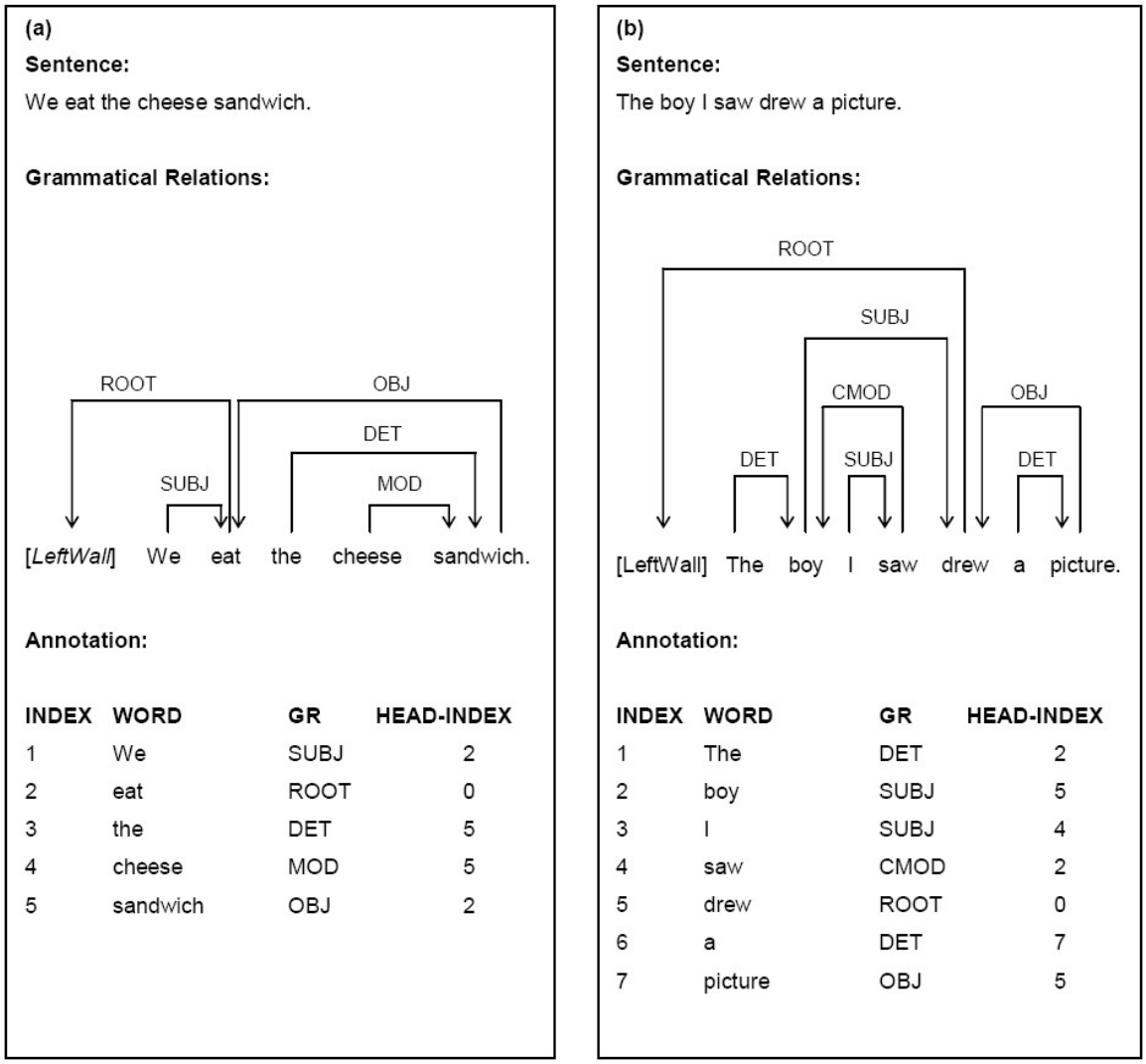


Figure 3.1. Sample sentences annotated with CHILDES GRs.

included a survey of the child language literature (Fletcher and MacWhinney, 1995), a review of existing measures of syntactic development (MacWhinney, 2000) and input from child language researchers.

Our final set of GRs is the product of two years of thoughtful development, paying close attention to the considerations mentioned above. Additionally, we note that a subset of these categories is sufficient to compute some of the most important measures of child language syntactic development – the Index of Productive Syntax, or IPSyn (Scarborough, 1990), the Developmental Sentence Score, or DSS (Lee, 1974), and the Language Assessment, Remediation, and Screening Procedure, or LARSP (Fletcher and Garman, 1988). An

automated procedure for computing IPSyn using this GR annotation scheme is presented in Chapter 7.

A comprehensive list of the grammatical relations in the CHILDS GR annotation scheme appears below. Example GRs appear in the form GR(dependent-di, head-hi), where GR is the GR label, dependent is the dependent word, di is the index of the dependent word (its position counting from left to right, starting at 1), head is the head word, and hi is the index of the head word.

SUBJ identifies the subject of clause, when the subject itself is not a clause. Typically, the head is a verb (the main verb of the subject's clause), and the dependent is a noun (or another nominal, such as a pronoun, or any head of a noun phrase). Clauses that act as subjects are denoted by CSUBJ or XSUBJ, depending on whether the clausal subject is finite or non-finite (see below). Note that throughout the labeling scheme, in general, C*GR* denotes a finite clausal version of the relation *GR*, and X*GR* denotes a non-finite clausal version of the relation *GR*, where the relation *GR* may be a subject, adjunct, predicate nominal, etc.

Example:

```
Mary saw a movie.

SUBJ(Mary-1, saw-2)
```

CSUBJ identifies the finite clausal subject of another clause. The head is typically the main verb of the main clause, and the dependent is the main verb of the clausal subject.

Example:

```
That Mary screamed scared John.

CSUBJ(screamed-3, scared-4)
```

XSUBJ identifies the non-finite clausal non-finite subject of another clause. The head is typically the main verb of the matrix clause, and the dependent is the main verb of the clausal subject.

Example:

```
Eating vegetables is important.

XSUBJ(eating-1, is-3)
```

OBJ identifies the first object of a verb. Typically, the head is a verb, and the dependent is a noun (or other nominal). The dependent must be the head of a required non-clausal and non-prepositional complement of the verb (head of OBJ). A clausal complement relation should be denoted by COMP or XCOMP (depending on whether the clausal complement is finite or non-finite, see below), not OBJ or OBJ2.

Example:

```
Mary saw a movie.

OBJ(movie-4, saw-2)
```

OBJ2 identifies the second object of a ditransitive verb, when not introduced by a preposition. Typically, the head is a ditransitive verb, and the dependent is a noun (or other nominal). The dependent must be the head of a required non-clausal and non-prepositional complement of a verb (head of OBJ2) that is also the head of an OBJ relation. A second complement that has a preposition as its head should be denoted by IOBJ, not OBJ2.

Example:

```
Mary gave John a book.

OBJ2(book-5, gave-2)
```

IOBJ identifies an object (required complement) introduced by a preposition. When a prepositional phrase appears as the required complement of a verb, it is the dependent in an IOBJ relation, not a JCT (adjunct) relation. The head is typically a verb, and the dependent is a preposition (not the complement of the preposition, see POBJ below).

Example:

```
Mary gave a book to John.

IOBJ(to-5, gave-2)
```

COMP identifies a finite clausal complement of a verb. The head is typically the main verb of the matrix clause, and the dependent is the main verb of the clausal complement.

Example:

```
I think that Mary saw a movie.

COMP(saw-5, think-2)
```

XCOMP identifies a non-finite clausal complement of a verb. The head is typically the main verb of the matrix clause, and the dependent is the main verb of the clausal complement. The XCOMP relation is only used for non-finite clausal complements, not predicate nominals or predicate adjectives (see PRED below).

Examples:

```
Mary likes watching movies.

XCOMP(watching-3, likes-2)


Mary wants me to watch a movie.

XCOMP(watch-5, wants-2)
```

PRED identifies a predicate nominal or predicate adjective of the subject of verbs such as *be* and *become*. The head of PRED is the verb, not its subject. The predicate may be nominal, in which case the dependent is a noun (or other nominal), or adjectival, in which case the dependent is an adjective. PRED should not be confused with XCOMP, which identifies a non-finite complement of a verb (some syntactic formalisms group PRED and XCOMP in a single category).

Examples:

```
Mary is a student.

PRED(student-4, is-2)

Mary got angry.

PRED(angry-3, got-2)
```

CPRED identifies a finite clausal predicate of the subject of verbs such as *be* and *become*. The head of CPRED is the main verb (of the matrix clause), not its subject.

Example:

```
The problem is that Mary sees too many movies.

CPRED(sees-6, is-3)
```

XPRED identifies a non-finite clausal predicate of the subject of verbs such as *be* and *become*. The head of CPRED is the main verb (of the matrix clause), not its subject.

Example:

```
My goal is to win the competition.

XPRED(win-5, is-3)
```

JCT identifies an adjunct (an optional modifier) of a verb, adjective, or adverb. The head of JCT is a verb, adjective or adverb. The dependent is typically an adverb, a preposition (in the case of phrasal adjuncts headed by a preposition, such as a prepositional phrase). Intransitive prepositions may be treated as adverbs, in which case the JCT relation applies, or particles, in which case the PTL relation (see below) applies. Adjuncts are optional, and carry meaning on their own (and do not change the basic meaning of their JCT heads).

Examples:

```
Mary spoke very clearly.
```

```
JCT(clearly-4, spoke-2)

JCT(very-3, clearly-4)

Mary spoke at the meeting.

JCT(at-3, spoke-2)

Mary is very tired.

JCT(very-3, tired-2)
```

CJCT identifies a finite clause that acts like an adjunct of a verb, adjective, or adverb. The head of CJCT is a verb, adjective, or adverb. The dependent is typically the main verb of a subordinate clause.

Example:

```
Mary left after she heard the news.

CJCT(heard-5, left-2)
```

XJCT identifies a non-finite clause that acts like an adjunct of a verb, adjective, or adverb. The head of CJCT is a verb, adjective, or adverb. The dependent is typically the main verb of a non-finite subordinate clause.

Example:

```
Mary left after hearing the news.

CJCT(hearing-4, left-2)
```

MOD identifies a non-clausal nominal modifier or complement. The head is a noun, and the dependent is typically an adjective, noun or preposition.

Examples:

```
Mary saw a red car.
```

```
MOD(red-4, car-5)

Mary saw the boy with the dog.

MOD(with-5, boy-4)

The Physics professor spoke clearly.

MOD(Physics-2, professor-3)
```

CMOD identifies a finite clause that is a nominal modifier (such as a relative clause) or complement. The head is a noun, and the dependent is typically a finite verb.

Example:

```
The student who visited me was smart.

CMOD(visited-4, student-2)
```

XMOD identifies a non-finite clause that is a nominal modifier (such as a relative clause) or complement. The head is a noun, and the dependent is typically a non-finite verb.

Example:

```
The student standing by the door is smart.

XMOD(standing-3, student-2)
```

AUX identifies an auxiliary of a verb, or a modal. The head is a verb, and the dependent is an auxiliary (such as *be* or *have*) or a modal (such as *can* or *should*).

Examples:

```
Mary has seen many movies.

AUX(has-2, seen-3)

Are you eating cake?
```

```
AUX(are-1, eating-3)

You can eat cake.

AUX(can-2, eat-3)
```

NEG identifies verbal negation. When the word *not* (contracted or not) follows an auxiliary or modal (or sometimes a verb), it is the dependent of a NEG relation (not JCT), where the auxiliary, modal or verb (in the absence of an auxiliary or modal) is the head.

Examples:

```
I am not eating cake.

NEG(not-3, am-2)

Speak not of that subject.

NEG(not-2, speak-1)
```

DET identifies a determiner of a noun. Determiners include *the*, *a*, as well as (adjectival) possessives pronouns (*my*, *your*, etc) and demonstratives (*this*, *those*, etc), but not quantifiers (*all*, *some*, *any*, etc; see QUANT below). Typically, the head is a noun and the dependent is a determiner. In cases where a word that is usually a determiner does not have a head, there is no DET relation.

Example:

```
The students ate that cake.

DET(the-1, students-2)

DET(that-4, cake-5)
```

QUANT identifies a nominal quantifier, such as *three*, *many*, and *some*. Typically, the head is a noun, and the dependent is a quantifier. In cases where a quantifier has no head, there is no QUANT relation.

Example:

```
Many students saw three movies yesterday.

QUANT(many-1, students-2)

QUANT(three-4, movies-5)
```

POBJ identifies the object of a preposition. The head is a preposition, and the dependent is typically a noun.

Example:

```
Mary saw the book on her desk.

POBJ(desk-7, on-5)
```

PTL identifies the verb particle (usually a preposition) of a phrasal verb. Intransitive prepositions that change the meaning of a verb should be in a PTL relation, not JCT (see above). The head is a verb, and the dependent is a preposition.

Example:

```
Mary decided to put off the meeting until Thursday.

PTL(off-5, put-4)
```

CPZR identifies a complementizer (usually a subordinate conjunction). The head is a verb, and the dependent is a complementizer. It is the verb (head of a CPZR relation) of an embedded clause that acts as the dependent in a relation involving the embedded clause and its matrix clause, not the complementizer (the verb is higher in the dependency tree than the complementizer).

Examples:

```
I think that Mary left.

CPZR(that-3, left-5)
```

```
She ate the cake because she was hungry.

CPZR(because-5, was-7)
```

COM identifies a communicator (such as *hey*, *okay*, etc). Because communicators are typically global in a given sentence, the head of COM is typically the root of the sentence's dependency tree (the dependent of the ROOT relation, see below). The dependent is a communicator.

Example:

```
Okay, you can read the book.

COM(okay-1, read-4)
```

INF identifies an infinitival particle (*to*). The head is a verb, and the dependent is always to.

Example:

```
Mary wants to read a book.

INF(to-3, read-4)
```

VOC identifies a vocative. As with COM, the head is the root of the sentence. The dependent is a vocative.

Example:

```
Mary, you may not eat the cake.

VOC(Mary-1, eat-5)
```

TAG identifies tag questions, where the tag is headed by a verb, auxiliary or modal. Tags of the type found in "this is red, *right*?" and "Let me do it, *okay*?" are identified as

dependents in a COM relation, not TAG. The head of a TAG relation is typically a verb, and the dependent is the verb, auxiliary or modal in the tag question.

Example (other relevant GRs also shown, for clarity):

```
This is good, isn't it?

TAG(is-4, is-2)

NEG(n't-5, is-4)

SUBJ(it-6, is-4)
```

COORD identifies coordination. The head is a conjunction (usually *and*), and several types of dependents are possible. The head coordinator may have two or more dependents, including the coordinated items. Once the COORD relations are formed between the head coordinator and each coordinated item (as dependents), the coordinated phrase can be thought of as a unit represented by the head coordinator. For example, consider two coordinated verb phrases with a single subject (as in "I walk and run"), where two verbs are dependents in COORD relations to a head coordinator. The head of COORD is then also the head of a SUBJ relation where the subject is the dependent. This indicates that both verbs have that same subject. In the case of a coordinated string with multiple coordinators, the COORD relation applies compositionally from left to right. In coordinated lists with more than two items, but only one coordinator, the head coordinator takes each of the coordinated items as dependents. In the absence of an overt coordinator, the right-most coordinated item acts as the coordinator (the head of the COORD relation).

Example (other relevant GRs also shown, for clarity):

```
Mary likes cats and dogs.

COORD(cats-3, and-4)

COORD(dogs-5, and-4)

OBJ(and-4, likes-2)

Mary likes birds and cats and dogs.

COORD(birds-3, and-4)

COORD(cats-5, and-4)
```

```
COORD(and-4, and-6)

COORD(dogs-7, and-6)

OBJ(and-6, likes-2)
```

ROOT this is the relation between the topmost word in a sentence (the root of the dependency tree) and the LeftWall. The topmost word in a sentence is the word that is the head of one or more relations, but is not the dependent in any relation with other words (except for the LeftWall). This word is the dependent in the ROOT relation, and the LeftWall is the head.

Example:

```
Mary saw many movies last week.

ROOT(saw-2, LeftWall-0)
```

3.3 Related GR Annotation Schemes

Our GR annotation approach shares characteristics with other recent annotation schemes. Like Carroll et al. (2003), we use a rich set of GRs that provide detailed information about syntactic relationship types. As previously mentioned, the GR set of Carroll et al. was used as a starting point in the development of the CHILDES GR set. Like the annotation used by Lin (1998) for parser evaluation, and the scheme of Rambow et al. (2002) for annotation of a small treebank of spoken English, we use dependency structures instead of the constituent structures that continue to be thought of as the primary form of syntactic representation by many. We share Rambow et al.'s findings that dependency structures, compared to constituent structures, provide increased ease and reliability of manual annotation. However, our choice to use dependency-based grammatical relations is motivated primarily by the fact that much of the syntax-based child language work that uses CHILDES data is predominantly GR-driven, and dependencies offer a natural way to encode such GRs.

In spite of the similarities mentioned above, our annotation scheme differs from those of Carroll et al. and Rambow et al. in important ways. The scheme of Carroll et al. does not annotate sentences with a full dependency structure, but rather lists a set of GRs

that occur in the sentence as a set of propositions. By doing away with the requirement of a complete and consistent dependency structure, their scheme allows for n-ary relations (while our GRs are strictly binary) and greater flexibility in working with GRs that may not fit together in a global graph structure. In our framework, n-ary relations (with n > 2) must be represented indirectly, using combinations of binary GRs. Their set of GRs, although detailed, is meant for general-purpose annotation of text, and does not explicitly denote specific pieces of information we have identified as important to the child language community, such as vocatives, communicators, tag questions[4]. In addition, they distinguish between initial (or deep) GRs, and actual (or surface) GRs, while we report surface GRs only. In summary, Carroll et al.'s GR scheme is more flexible than our CHILDES GR scheme, allowing for representation of syntactic structures our scheme does not support (such as control, raising, etc.). Our scheme, on the other hand, is tailored for consumption by child language researchers, and it is focused on representing the necessary information in a simple way, while also keeping in mind ease of automatic analysis. Our use of a simple and popular syntactic representation format allows much of the recent work on dependency parsing that support the same type of dependency trees (Eisner, 1996; McDonald et al., 2005; Nivre and Scholz, 2004; Yamada and Matsumoto, 2003) to be applied to our annotation scheme very easily.

The scheme of Rambow et al. is dependency-based, like our CHILDES GR annotation scheme. However, their dependency labels are limited to seven syntactic roles (SRole and DRole features, which can have the values of subj, obj, obj2, pobj, pobj2, adj and root). These seven roles suffice for some applications, but do not offer the granularity needed for CHILDES annotation. In contrast, we have 30 distinct labels for GRs. Like Carroll et al., Rambow et al. annotate surface and deep relations. Other schemes based on dependency trees for English (Lin, 1998; Nivre and Scholz, 2004) have been developed around what dependency and label information can be extracted from existing constituent treebanks. While this approach is convenient and results in fairly general representations, the information they can represent is for the most part limited to what is already encoded in the original constituent trees. The CHILDES GR scheme, on the other hand, was designed with a specific set of criteria in mind.

3.4 Specific Representation Choices

While the CHILDES GR annotation scheme is suitable for the identification of specific syntactic structures of interest to child language researchers, we should stress that the annotation scheme is in no way meant to be a theory of syntax, nor is it intended to

[4]It is likely that these structures may be identified indirectly.

represent all possible kinds of syntactic phenomena. In fact, the choice of grammatical relations and how they are represented was motivated primarily not by specific linguistic theories, but by two practical concerns: the usefulness of the information contained in the annotations to child language research, and (to a lesser extent) automatic annotation of transcripts using syntactic parsing. In addition, the scope of the annotation scheme is purely syntactic. Although semantic roles are undoubtedly valuable, this annotation scheme makes no attempt to represent such information. Therefore, while we address the identification of syntactic relations such as subjects and objects, we do not address the identification of semantic relations such as agents and themes.

Our GRs typically assign content words as heads and function words as dependents. For example, nouns are the heads of determiners (forming a GR of type DET with the noun as the head and the determiner as the dependent), and verbs are chosen as the heads of auxiliaries (forming a GR of type AUX with the verb as the head and the auxiliary as the dependent). An exception to this rule is coordination, where coordinated items are dependents of the coordinator (for example, in "boys and girls" each noun is a dependent of "and", forming two GRs of type COORD). This is further discussed below. When both words in a GR are members of lexical (content) categories, the direction of the relation follows common practice, where complements, adjuncts and modifiers are dependents of the words they complement or modify. For example, adjectives are typically dependents of nouns, and adverbs are typically dependent of verbs, adjectives or other adverbs. Nouns are dependents of verbs when the noun is a complement (such as subject or object), but verbs can also be dependents of nouns, as in the case of relative clauses (where a clause modifies a noun, and the verb is the root of the dependency subtree representing the clause). In the case of prepositional phrases, the preposition is chosen as the head of the prepositional object (in a POBJ relation). By convention, vocatives and communicators are dependents of the main verb of their sentences.

In cases where a clause has a relation to another clause, the verb of the lower (subordinate) clause is used as a dependent. Thus, in the relative clause of figure 3.1(b), the verb "saw" of the relative clause is treated as dependent in a CMOD (clausal modifier of a nominal) relation with the noun "boy". The other relations in this sentence are as expected: "The" and "a" are dependents in DET relations with "boy" and "picture", "boy" is the dependent in a SUBJ relation with "saw", and "picture" is in a OBJ relation with "drew". Finally, "drew" is the root of the dependency tree for this sentence, and by convention is the dependent in a ROOT relation with the special word LeftWall, which we append to the beginning of the sentence.

Coordination is known to be a difficult structure to represent intuitively using dependency trees. As stated above, in our representation the conjunction is the head of coordi-

nated items. This is also the way coordination is analyzed in FDG (Functional Generative Description) (Sgall et al., 1986), the theory used to guide annotation choices in the Prague Dependency Treebank (Hajic et al., 2001), a very large Czech dependency treebank. Other popular representations are to have each of the coordinated items be dependents of the word they actually modify (Tesnière, 1959), to form a linear chain where each item in the coordinated phrase is a dependent of the preceding item and the first item is a dependent of the word it actually modifies (Mel'cuk, 1988), or to treat coordination as a special case where dependencies do not apply and a constituent representation must be used (Hudson, 1984). Given that there is no agreement on what the correct or most intuitive representation should be, the motivation for using our representation of coordination is that it can be converted to any of the representation above in a straightforward manner, if necessary.

A point worth noting is that, in general, only words that appear in a sentence can be participants in a GR as either a head or a dependent (an exception to this statement, specific to child language annotation, is presented in section 3.5). This is in contrast to the scheme of Rambow et al., where an empty word e can be added to the sentence in control structures or ellipsis. For example, in the sentence "I wanted to run," Rambow et al. annotate the empty word e as the subject of run, while we do not annotate a subject for run.

3.5 Word Insertion in Child Language Annotation

Our annotation scheme, as described above, deals only with surface syntactic relationships between words that appear explicitly in a sentence. There is no analysis of ellipsis, traces, or any other phenomena that involves lexical items without a surface representation. There is, however, one situation where we found it necessary to include words that do not appear in the surface sentence as part of the syntactic analysis. As can be easily observed in CHILDES transcripts, some utterances in child language are fully intelligible, but differ from the adult norm of grammaticality only by the omission of certain lexical items (and these items can be identified reasonably unambiguously). For example, consider the following sentences (the items in square brackets are not part of the original sentences, but have been inserted for illustration purposes):

- This is Fraser ['s] pencil.
- This [is] my pencil.
- I want [to] go outside.

In such cases, we allow for the annotation of words that do not actually appear in the transcripts. When a word is inserted into an analysis, it is marked with an asterisk that indicates that it was not part of the original sentence or utterance. This mechanism is not intended for the annotation of ellipsis, and it is only applied in the annotation of child utterances. It is also not intended for unbounded reconstruction or repair of ungrammatical sentences. These limited insertions are only used when the utterance is already clear before the insertion (although not perfectly formed by adult standards), and when a clear choice can be made for the inserted word. This is a simple solution targeted specifically at the annotation of important information in child language research, and it represents only a small exception in the annotation scheme.

3.6 Inter-Annotator Agreement

Inter-annotator agreement was measured by having a corpus of 285 words (48 sentences from a CHILDES corpus) annotated manually by two annotators, independently. The annotators had at least a basic background in Linguistics, and one was familiar with the annotation scheme. The other was trained for about one hour before annotating a separate 10-sentence trial corpus under close guidance. Annotation of the 285-word corpus took about 90 minutes. Annotator agreement was 96.5%, which is about the same as inter-annotator agreement figures for related annotation schemes involving dependencies and grammatical relations. Carroll et al. report 95% agreement, while Rambow et al., report 94% agreement. Because of the size of the corpora used for rating annotator agreement, these differences are not significant. Out of 285 possible labeled dependencies, there were 10 disagreements between the annotators. Of particular interest, four of them were disagreements on the attachments of adjuncts, and three of them were incorrect labeling (with correct dependent-head links) involving COMP, PRED and OBJ. These three GRs are closely related, and the labeling mistakes were oversights on the part of one of the annotators and not genuine differences of opinion.

Chapter 4

A Rule-Based GR Parsing Approach

In the next three chapters we will address the central issue in this thesis of automatically annotating utterances in CHILDES corpora with the grammatical relations in the scheme described in chapter 3. To this end, we will explore two general approaches to syntactic parsing. This chapter deals with rule-based parsing, where a system depends a great deal on the linguistic knowledge encoded in a set of rules (a grammar) according to which syntactic structures are determined. In chapter 5 we will examine a data-driven or corpus-based approach to syntactic analysis, where the knowledge used by the system comes from a training corpus (a large set of sentences annotated with syntactic structures). Finally, in chapter 6 we present ways of combining the strengths of the rule-based and data-driven approaches, producing a highly accurate system for grammatical relation identification.

4.1 Grammar-driven Syntactic Parsing

In this chapter we present a GR analysis system that is based on grammar-driven parsing. In other words, the system uses a manually written grammar as its model of natural language. Although the parser we present is not purely rule-based, since it employs a statistical model that helps the system resolve ambiguities in the grammar, it is a predominantly grammar-driven approach. As native speakers of at least one language, equipped with linguistic intuition, a vast amount of world-knowledge and reasoning skills, we often fail to sense the magnitude of the ambiguity problem in natural language. However, when we come

to articulating a set of rules to guide a system in automatic parsing, dealing with the ambiguity of attachment, roles, interpretations, and relations in human language stands as a formidable challenge. The use of a grammar effectively shapes the search space of possible syntactic analyses for a sentence, since the rules specify exactly how words may combine to form larger structures. It is then important to use a carefully designed grammar that allows the system to analyze as many syntactic configurations in the text as possible, while constraining the space of analyses so that ambiguities present in the grammar can be resolved effectively, and a correct analysis for each sentence can be found. In practice, this results in a trade-off between coverage (the portion of input sentences that are covered by the rules in the grammar) and ambiguity (how many possible analyses for each sentence are licensed by the grammar). Very restrictive grammars that constrain the search space of possible syntactic trees too aggressively may not cover all syntactic structures in input text that was not seen during grammar development. On the other hand, grammars that allow for combinations of words and phrases in too many ways can be highly ambiguous, making the process of choosing a correct analysis for a given sentence (in other words, disambiguation) a difficult task. This can be addressed with the use of additional knowledge sources (such as a lexicon and a semantic model), and/or a statistical disambiguation model. The creation of knowledge sources or training material for a statistical model may be a very labor-intensive task, but one that may be eased significantly with the use of a carefully designed grammar.

In addition to the inherent ambiguity commonly expected in natural languages, parsers must consider a remarkably large number of alternative analyses for all but the simplest sentences. Unless additional knowledge sources are used, allowing for the analysis of a large number of syntactic constructions leads to the development of a model that suffers from more ambiguity than a more restrictive model. The balance of coverage and ambiguity is a crucial issue in parsing. As an example, consider the simple context-free grammar in figure 4.1 and sentences in figure 4.2.

In spite of the similar part-of-speech sequences in sentences S1 and S2, their syntactic structures differ in the place where the prepositional phrases (PP) “with the dog” and “with the telescope” should be attached. Let’s assume the preferred analysis of sentence S1 has the prepositional phrase attached to the noun phrase (figure 4.2a), yielding the paraphrase “I saw the boy who was with a dog.” In other words, the phrase “with the dog” modifies the phrase “the boy.” However, the grammar above lacks the rule used to make the prepositional phrase attachment in figure 4.2a, and the only analysis it allows for sentence S1 is the incorrect analysis seen in figure 4.2b (which might be paraphrased as “*I used the dog to see the boy”, where the prepositional phrase attaches to the verb, having the phrase “with the dog” modify the verb “saw”). Conversely, the preferred analysis of sentence S2 has the prepositional phrase “with the telescope” attached to the verb phrase (figure 4.2c),

```
(1)  S -> NP VP            (7)  DET -> the

(2)  NP -> DET N           (8)  N -> boy

(3)  NP -> PRO             (9)  N -> dog

(4)  VP -> V NP            (10) N -> telescope

(5)  VP -> VP PP           (11) P -> with

(6)  PP -> P NP            (12) PRO -> I

                           (13) V -> saw


(S1) I saw the boy with the dog.

(S2) I saw the dog with the telescope.
```

Figure 4.1. Simple context-free grammar and sentences.

thus modifying the verb "saw" and yielding the paraphrase "I used the telescope to see the dog." This is the only analysis of S2 allowed by our grammar. The incorrect attachment of "with the telescope" to the noun phrase "the dog" (which could be paraphrased as "*I saw the dog that had a telescope", figure 4.2d) is not allowed. In summary, the grammar above allows for exactly one syntactic analysis of each sentence, but in the case of sentence S1 the analysis is incorrect (since it leads to the dispreferred analysis).

For the grammar to cover the correct analysis of S1, we need to add an additional rule that allows prepositional phrases to modify noun phrases:

```
(2')  NP -> NP PP
```

However, the addition of a new rule to handle a previously uncovered syntactic structure may have adverse side effects in the overall performance of the grammar. Even though the addition of rule 2' to the grammar allows for the correct analysis of sentence S1 (figure 4.2a), the incorrect analysis (figure 4.2b) is still possible. What is even worse is that this modification allows for the incorrect analysis of sentence S2 (figure 4.2d), which could only be analyzed correctly and unambiguously before the addition of rule 2'. While there are ways to resolve the ambiguities in sentences S1 and S2 and produce the correct analysis in each case (for example, by using additional knowledge sources, more complex gram-

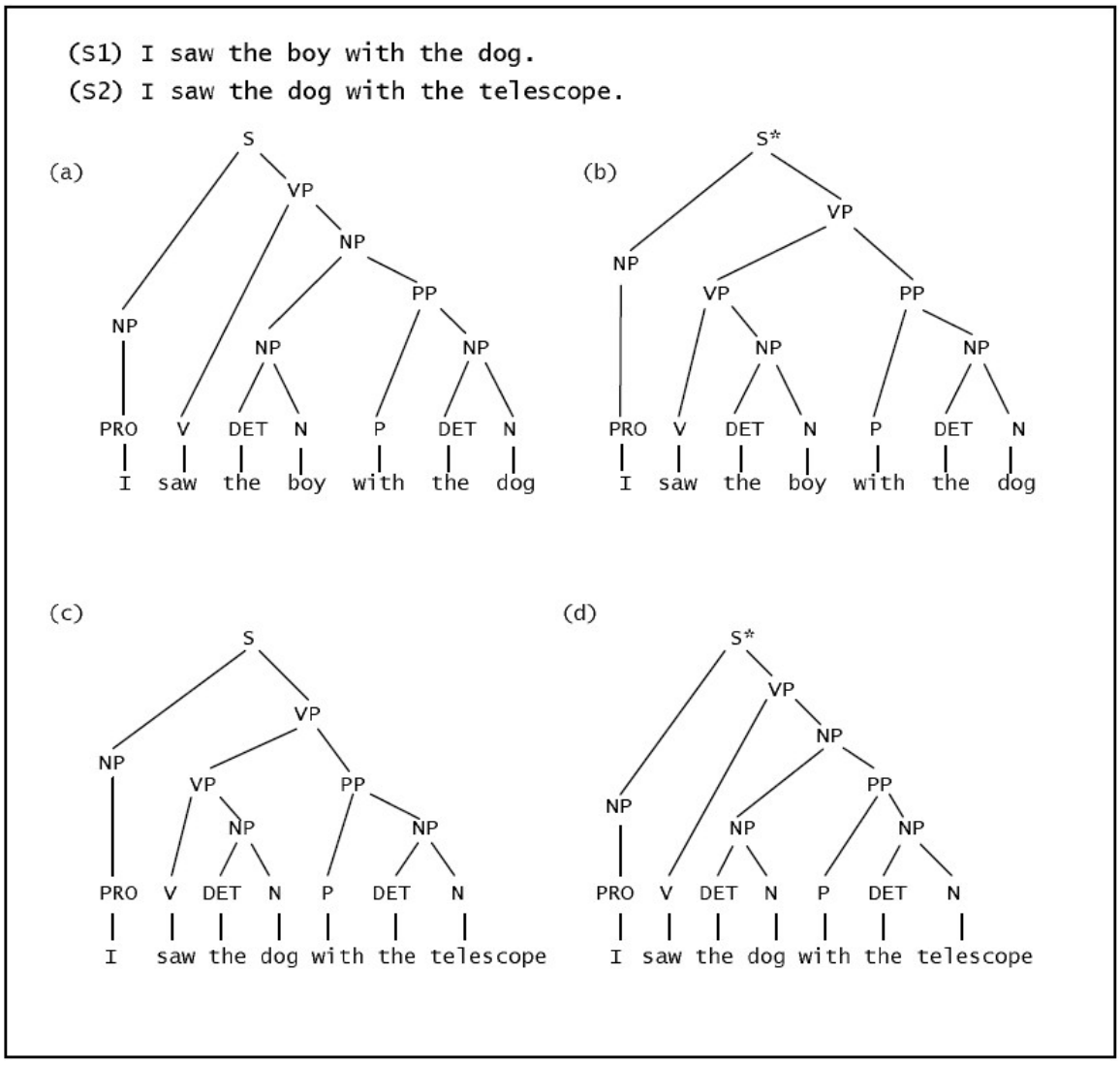


Figure 4.2. Parse trees for "I saw the boy with the dog" and "I saw the dog with the telescope" (trees rooted with S* represent dispreferred analyses). The trees in (b) and (c) can be generated with the given grammar, but the desired trees are (a) and (c). Adding a rule to allow for the analysis in (a) has the undesired side effect of also allowing the (incorrect) tree in (d).

mar constructions, feature unification, or statistical disambiguation models), this example illustrates how increasing the coverage of a grammar may result in unwanted ambiguity.

```
(1) S -> NP VP
    (x1 AGREEMENT) =c (x2 AGREEMENT)
    (x2 SUBJECT) = x1
    x0 = x2

(2) NP -> DET N
    (x2 DETERMINER) = x1
    x0 = x2

(3) NP -> PRO
    x0 = x1

(4) VP -> V NP
    (x1 OBJECT) = x2
    x0 = x1
```

Figure 4.3. A simple grammar composed of a context-free backbone and unification equations.

4.2 A Rule-Based Syntactic Analysis System

In the context of annotating CHILDES corpora with grammatical relations, we developed a syntactic analysis system based on grammar-driven robust parsing and statistical disambiguation. Grammar-driven parsers use a set of production rules that specify how each syntactic constituent may be expanded into other constituents or words as a model of natural language. The type of grammar used by our system follows a formalism based on a context-free backbone augmented with feature-unification constraints, originally developed for the Generalized LR parser/compiler (Tomita, 1987, 1990). As an example, figure 4.3 shows a simple grammar that can be used with our system. The feature structure for each nonterminal symbol in the context-free portion of the rules is referenced in the unification equations as `x`n, where n is an integer. `x0` corresponds to the feature structure associated with the symbol in the left-hand-side of an equation, `x1` to the first symbol in the right-hand-side, `x2` to the second symbol in the right-hand-side, and so on. The operator `=c` in equation 1 in the first equation of rule 1 is used for checking equality, while the operator `=` is used for unification of feature structures.

The grammar in figure 4.3 can be used to parse the sentence "He sees the ball" as follows:

a lexical analysis of the words in the input determines the part-of-speech and agreement features (in the cases of "he" and "sees") of each word. Rule 3 allows the formation of a noun phrase (NP) from the pronoun (PRO) "he". The single unification equation associated with that rule states that every feature in the first element of the right-hand side of the context-free portion of the rule (represented in the equation by x1 and corresponding to the pronoun) will be passed to the newly formed constituent, or the left-hand side of the context-free rule (the noun phrase, represented in the equation by x0). Rule 2 forms another noun phrase from the words "the" and "ball". The first equation in the rule specifies that the first element in the right-hand side of the context-free rule (DET, represented in the equation by x1) is the value of a feature named DETERMINER of the second element of the right-hand side. The second equation specifies that every feature of the second element of the right-hand side (including the newly specified DETERMINER feature) be passed to the newly created constituent (NP). Rule 4 can then be applied to form a verb phrase (VP) from the verb "sees" and the noun phrase "the ball". According to the first equation of rule 4, the noun phrase becomes the OBJECT of the verb. Finally, the noun phrase "he" and the verb phrase "sees the ball" can be combined by rule 1 to produce a sentence (S) constituent that spans the entire input string, completing the parse. The first equation of rule 1 requires that the value of the AGREEMENT features of the verb phrase and noun phrase match. These values are provided by the lexical analysis performed before the parsing process. The second equation makes the noun phrase the subject of the sentence. The final analysis can be seen in figure 4.4. Notice that a dependency structure that corresponds to the annotation scheme in chapter 3 can be easily extracted from the f-structure.

Our system is based on an instance of rule-based robust parsing technologies that seeks to augment the coverage of a parser by allowing it to analyze language phenomena that fall outside of the coverage of the parser's grammar (Lavie, 1996; Rosé and Lavie, 2001). Our use of robustness is targeted towards the analysis of unforeseen spoken language phenomena, or the lack of grammar support for specific language structures. This is achieved by allowing the parser to: (1) insert lexical or non-terminal items (constituents) that are not present in the input string, and (2) skip certain words in the input string. The specific uses of these techniques are discussed in a later section on parser flexibility. While the expansion of coverage provided by robust parsing increases the ambiguity problem faced by the analysis system, we employ statistical techniques to allow the parser to cope with such ambiguity. By providing a training corpus of manually disambiguated sentences (of much smaller size than what would be required to train a data-driven parser, see chapter 5), we can build a statistical model of grammar usage to make certain decisions in the parsing process that result in fairly accurate disambiguation.

The input to our system is a sequence of transcribed utterances, and the output is a

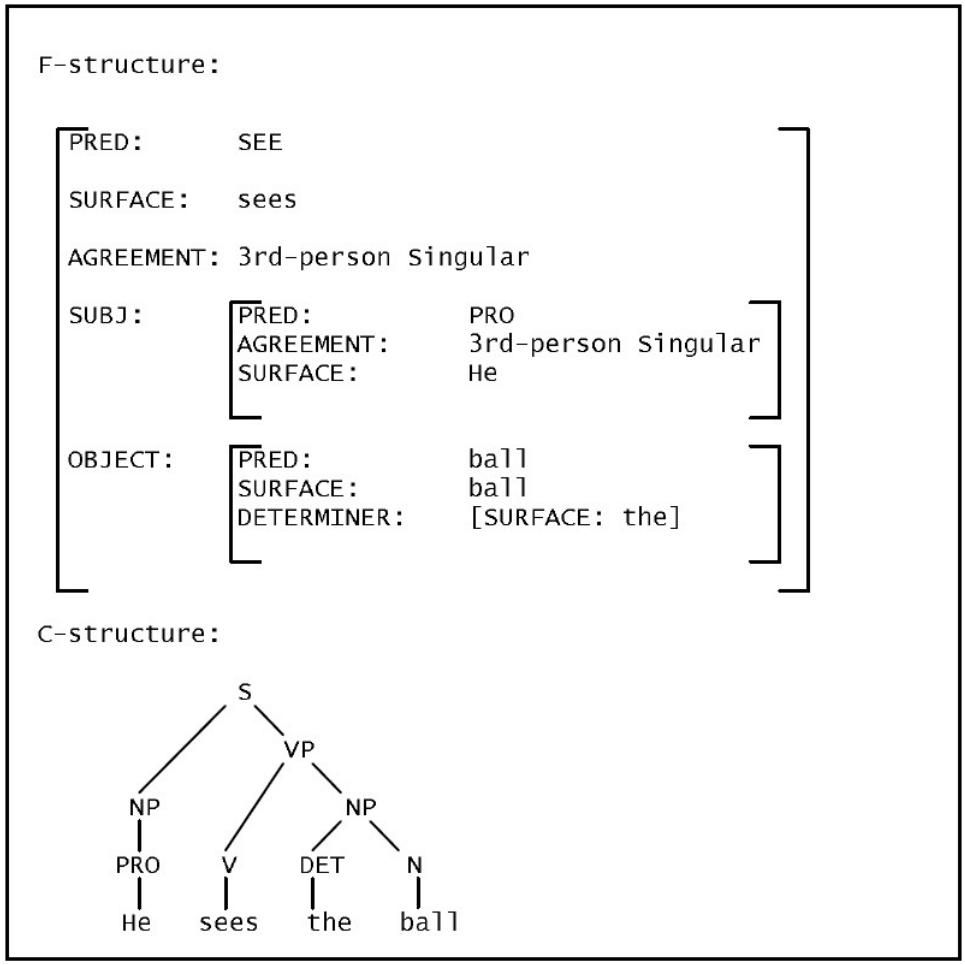


Figure 4.4. Analysis of "He sees the ball" according to the grammar in figure 4.3

syntactic analysis for each of those utterances. At a lower level, the system can be divided into three main components:

1. The MOR morphology analysis program (MacWhinney, 2000), and a part-of-speech tagger.
2. A robust parser for transcribed spoken language based on LCFlex (Rosé and Lavie, 2001).
3. A statistical disambiguation module to pick the correct analysis from the many produced by the parser.

4.2.1 MOR and Part-of-Speech Tagging

MOR is a lexical analyzer that uses a lexicon and morphological rules to provide all the possible parts-of-speech and morphological analyses for each word in transcribed utterances in CHILDES corpora. In some cases, such as with the determiner "the", the output of MOR is unambiguous. Sometimes, however, MOR outputs several possible analyses for a word. For example, the word "houses" may be a plural noun or a third person singular verb. In these cases, we need to perform lexical disambiguation to arrive at one single analysis for each word.

Although the system described in this chapter is largely rule-based, we use data-driven techniques for lexical disambiguation. The training data used here consists of approximately 2,000 words annotated with parts-of-speech according to the CHILDES tag set. This is a subset of the CHILDES Eve training set described in chapter 2. Additionally, we use a part-of-speech tagger, MXPOST (Ratnaparkhi, 1996), trained on the Penn Treebank (see chapter 2).

Our system expects words to be tagged according to the CHILDES tag set. Because we do not have enough text annotated with CHILDES part-of-speech (POS) tags to train a POS tagger with high accuracy, we attempt to take advantage of the availability of accurate POS taggers trained on then Penn Treebank. The reason for not using those one such tagger directly on CHILDES data is that Penn Treebank taggers are trained to produce output according to the Penn Treebank tag set, while the CHILDES tag set has been developed over several years specifically for child language data (MacWhinney, 2000), and they are trained on data in a very different language genre. Our approach to adapt a Penn Treebank POS tagger to the CHILDES domain and tag set is to first tag the input sentences using the MXPOST POS tagger (Ratnaparkhi, 1996), and then change the Penn Treebank tags assigned by MXPOST into CHILDES tags. When trained on the Penn Treebank, MXPOST achieves accuracy close to 97% using the Penn Treebank tag set on data in the domain of the text used to train the tagger. Our complete process for tagging CHILDES data is as follows. First, we parse the CHILDES input text using MXPOST. We then use the words, MXPOST-predicted (Penn Treebank) tags, and the manually annotated (CHILDES) tags in our CHILDES training set to train a classifier to predict CHILDES POS tags when given words with Penn Treebank POS tags. Tagging is then a classification task that is done one word at a time. For classification, we use TiMBL (Daelemans et al., 2004), a memory-based learner, with the following features: the word to be tagged and its context (previous two words and next two words), and for each of these words we list its POS tag assigned by MXPOST, whether or not the word is capitalized, and the last three letters of the word. Our POS tagging approach achieves 96.3% accuracy in assigning CHILDES POS tags to CHILDES data. This approach is very similar to the approach used for base-

phrase chunking (also known as shallow parsing) used by Kudo and Matsumoto (2001), but instead of predicting IOB chunk tags, we predict CHILDES POS tags. A similar idea has been used by Yeh (2000b) in the context of data-driven GR identification: first, two existing GR parsers (Briscoe and Carroll, 2002; Buchholz et al., 1999) process the data, and a post-processing system trained on a small domain-specific training corpus with different GR tags modifies the GR information assigned in the first parsing pass.

4.2.2 LCFlex

Because the corpora in the CHILDES database consist only of transcribed spontaneous speech, having a parser designed to handle spoken language is of great importance. Through a set of parameters, our version of the LCFlex parser (Rosé and Lavie, 2001) can be tuned to allow the insertion of specific missing syntactic constituents into a sentence, and to skip extra-grammatical material that would prevent an analysis from being found with the grammar in use. These features allow great flexibility in parsing spoken language, but their parameters must be tuned carefully to balance benefits and the increased ambiguity caused by allowing insertions and skipping.

LCFlex is an agenda-driven bottom-up chart-parser. A detailed description of how such parsers work is provided in (Allen, 1995). In a nutshell, bottom-up parsers start from words (or part-of-speech tags) to form the smallest constituents (such as noun phrases) first, placing them in a chart (in the case of a chart-parser). In an agenda-driven parser, newly formed constituents are inserted in an agenda. At each iteration of the parser, a constituent is taken from the agenda, and the parser consults its grammar rules to find out how that constituent may be combined with other constituents in the chart to form new, larger constituents, which will be then added to the agenda. Two constituents can be combined into a new constituent only if they are adjacent. Once the possibilities for a given constituent are exhausted, the parser inserts it in the chart. Parsing finishes when the agenda is empty. At that point we check the chart for constituents of certain types (usually "sentence" or S) that span the entire input sentence.

Grammars used by LCFlex are of the type discussed earlier in this section (context-free backbone with unification equations), and illustrated in figure 4.4. These grammars (including the one used in our system) can be edited manually in any text-editing program. LCFlex's capability for inserting missing syntactic constituents in the analysis of a given sentence is due to a modification in the general bottom-up chart-parsing algorithm. When the parser considers how a constituent (picked from the agenda) may combine with other existing constituents, it also considers the combination of the constituent with the specific constituents we allow the parser to insert. These inserted constituents do not correspond

to any actual words in the sentence. For example, if the sentence "went home" is given as input to the parser, and if the insertion of a noun phrase is allowed, the parser may find an analysis that includes a noun phrase with no lexical content as the subject of the input sentence. A different modification allows word skipping. When the parser considers how a constituent may combine with other constituents, it may also consider combinations of constituents that are one word apart (if we allow skipping of a single word), in which case the word between the two constituents is ignored. A detailed description of LCFlex and the modifications to the bottom-up chart-parsing algorithm that allows for limited insertions and word skipping can be found in (Rosé and Lavie, 2001).

The output of LCFlex is in the form of syntactic feature structures (that resemble functional structures), as shown in figure 4.4. Because each level of nesting in the structures produced with our grammar has a head word, extracting labeled dependency structures from the feature structures produced by LCFlex is simple: there is a dependency between the head word of each sub-structure and the head word of the outer structure containing the sub-structure in question. The word in the inner structure is the dependent, and the word in the outer structure is the head. The label of the dependency is the label of the function of the inner structure. In the case of the outermost structure, its head word is a dependent in a ROOT relation with the LeftWall (see chapter 3). Figure 2.1 in chapter 2 shows LCFlex-style feature structures and corresponding dependency structures.

4.2.3 Statistical Disambiguation

The idea behind statistical syntactic disambiguation in LCFlex is that each analysis of a particular utterance is obtained through an ordered succession of grammar rule applications, and the correct analysis should be the one resulting from the most probable succession of rules. The probability of each competing analysis is determined based on a statistical model of bigrams of rule applications (Rosé and Lavie, 2001) obtained from training examples.

The training corpus required by the statistical disambiguation model consists of sentences paired with their correct analyses. The parser uses the training corpus to count the usage of bigrams of grammar rules in the process of parsing a sentence into its correct analysis. This corresponds to a "two-level" probabilistic context-free grammar model. In a standard probabilistic context-free grammar, each rule in the grammar is associated with a probability. A training corpus containing correct parses can be used to count the usage of each rule and determine their frequencies. In the two-level case, or with bigrams of rule applications, instead of estimating the probability of each rule in isolation, we estimate the probability of each rule, given the rule used to generate the left-hand-side (LHS) of the current rule. For example, instead of having a rule $VP \rightarrow V\ NP$ with probability 0.8, we

might determine from training data that the probability of rule $VP \rightarrow V\ NP$ is 0.6 if its LHS (VP) was generate using the rule $S \rightarrow NP\ VP$, and 0.2 if its LHS was generated using $VP \rightarrow VP\ PP$. This allows for rule probabilities to be somewhat sensitive to structural context. Probabilistic grammars of this type are sometimes referred to as pseudo-context-sensitive. Details on the statistical disambiguation model used by LCFlex can be found in (Rosé and Lavie, 2001).

Our implementation differs slightly from the original implementation of disambiguation in LCFlex. The difference involves grammar rules with identical context-free portions, but different sets of unification constraints. In the original implementation, such rules must be written as a single rule, and the different unification constraints are represented as a disjunct of constraints. Our implementation allows different rules to have identical context-free portions. We found this change to be necessary because GRs are represented in the unification portion of the rules, and it is desirable to allow rules with different unification constraints to have different probabilities.

4.2.4 Tailoring an Analysis System for Adult Utterances

While the adult language in the CHILDES corpora generally conforms to standard spoken language, the child language in the corpora varies from the language of a child in the very early stages of language learning to fairly complex syntactic constructions. Child and adult utterances differ significantly enough that we can analyze them more accurately by doing so separately, often with different strategies. We first explore the "easier" (in the sense that it is better defined) problem of analyzing adult utterances, which are theoretically of equal importance as child utterances, since theories of learning depend heavily on consideration of the range of constructions provided to children in the input (MacWhinney, 1999; Moerk, 1983).

To obtain high quality syntactic analyses from a system composed of the pieces described in the previous section, each component must be tuned carefully, keeping in mind the behavior of the overall system. This section focuses on the specific issues that relate to the components of the parser, as well as their integration into a high performance analysis system.

Grammar

The grammar needed by LCFlex consists of context-free rules augmented with feature unification constraints. Although general-purpose English grammars are available, we found that they were not suitable for our analysis task. The main problems associated with off-

the-shelf grammars are related to the large amount of ambiguity allowed by such grammars (a practically unavoidable consequence of a grammar designed to analyze unconstrained English sentences), and the lack of support for certain phenomena we find in corpora in the CHILDES database (such as the extensive use of communicators and vocatives commonly used in casual spoken language, onomatopoeia, etc.). It has been reported that practical grammars used to analyze newspaper articles written in English produce an astronomical number of parses per sentence (Moore, 2000), with the vast majority of these parses being completely uninterpretable from a human point-of-view. As a simple example of this phenomenon, Charniak (1997) uses the sentence "Salespeople sold the dog biscuits" and a grammar not unlike the one discussed earlier in relation to figure 4.1, but including the rule $NP \rightarrow NP\ NP$.

Although this rule may seem unusual, it is used to analyze phrases such as "10 dollars a share," where both "10 dollars" and "a share" are noun phrases that combine to form a larger noun phrase. Charniak gives three analyses for his example sentence (figure 4.5). While the first two analyses can be easily interpreted as "dog biscuits were sold by the salespeople," and "biscuits were sold to the dog by the salespeople," respectively, the third analysis does not seem to have a meaningful interpretation. In fact, it was the result of the application of a rule designed to cover a syntactic construction not present in any plausible interpretation of this sentence. The interested reader is encouraged to see Charniak (1997) for a more detailed account of the ambiguity problem in practical natural language grammars.

Because of the nature and the domain of our target corpus, it does not contain many of the complex syntactic constructions found in newspaper-style text, or even adult conversations. We can take advantage of that fact and attempt to reduce ambiguity by designing a grammar that fits our target language more tightly. It is a fairly well accepted notion in natural language processing that parsing within a specific domain can be made more accurately with the use of domain-specific resources.

Starting with a general-purpose English grammar with about 600 rules, we pruned or simplified a large number of rules that would never (or rarely) be used in correct analyses for the target corpus. For example, the noun phrase rule mentioned above can be safely discarded, since constructions such as "10 dollars a share" are so rare in our target language (in fact, that construction was not observed in our development set) that their coverage does not justify the added ambiguity. The result was a completely rewritten compact grammar with 168 rules. This final grammar included rules to handle the specific language phenomena likely to appear in the CHILDES database, and represents a much cleaner and tighter model of the language in the domain we are attempting to analyze. As a result, the potential for ambiguity in parsing was significantly reduced.

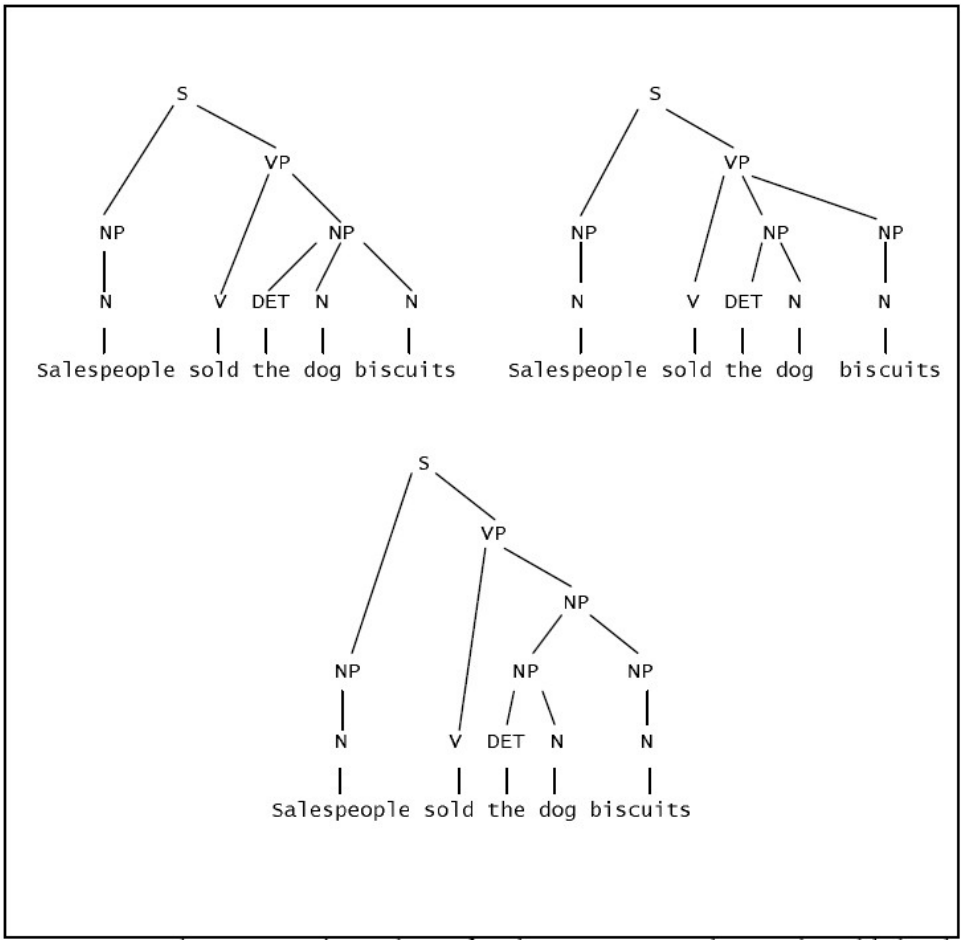


Figure 4.5. Three syntactic analyses for the sentence "Salespeople sold the dog biscuits."

Parser Flexibility

Even after grammar development, a large number of sentences in the Eve corpus still could not be parsed with our compact grammar due to specific characteristics of the casual conversational language in the corpus, errors and inconsistencies in the transcriptions, and infrequent syntactic structures not supported by the grammar. The majority of such sentences were not covered successfully because of omitted words and disfluencies that we failed to filter out due to transcriptions errors[1] in otherwise fully grammatical utterances, for example:

- Missing auxiliary verbs in questions ("[Do] You want to go outside?");

[1]The CHAT format used in CHILDES transcriptions specifies that all disfluencies be marked as such, making it possible to filter them out before parsing, but in practice not all disfluencies are identified by transcribers.

- Missing noun phrases, as elided subjects ("[I] Don't think she wants to play now."), and even as elided objects ("Give [it] to me.");
- Missing auxiliary verbs and noun phrases ("[Do] [you] Want to go outside?");
- Filled pause ("I'd like to tell you, *uh*, something" or "she went, *yeah*, to the store").

Adding explicit ad-hoc grammar rules to handle such sentences would cause the grammar to deviate from the clean model of language we were hoping to achieve, and add much harmful ambiguity to the analysis process. Instead, we turned to the robustness features of the parser to handle these sentences. LCFlex allows the addition of specific syntactic nodes to an analysis, or skipping of words in a sentence, making it conform to the grammar and leading to a successful analysis.

Tunning the tradeoff between precision and recall

The issues of parser coverage and ambiguity are closely related to the precision and recall of grammatical relations we can expect to obtain from a parser. Lack of coverage directly affects recall negatively, since no GRs are found in sentences that cannot be analyzed by the parser. However, we must be careful when increasing coverage, since this may lead to an increase in ambiguity, making the disambiguation task more difficult and hurting precision as a result. Now we look at how we deal with these issues in the grammar-driven parsing framework we described so far.

If we set LCFlex's robustness parameters not to allow any insertions or word skipping, and we use the grammar mentioned above with the two-level PCFG disambiguation model trained on 2,000 words (a subset of the Eve training set described in chapter 2), only about 31% of the words in the Eve adult development set can be parsed, so almost 70% of the words receive no analysis. If we consider only the accuracy of the GRs assigned to words in sentences that were covered by the grammar, we find that precision is very high, at about 0.94. However, because such a large portion of the words received no GRs at all, recall is very low, at 0.29. If we set LCFlex's robustness parameters to allow insertions of one noun phrase (NP) and/or one auxiliary, and to skip at most one word to increase the number of sentences covered by the parser, 63% of the words in the same development corpus are assigned a grammatical relation. Under these conditions[2], recall improves to 0.55, but at the cost of lowering precision to 0.82. This is largely because it is more difficult for the disambiguation model to choose the best parse in a significantly expanded search space of possible analyses for each sentence. Although increasing parser robustness improves overall accuracy (from 0.44 f-score to 0.66 f-score), the drop in precision is quite sharp. With more

[2]We arrived at these settings by manual inspection of the development set.

Settings	Coverage	Recall	Precision
no insertions, no skipping	31%	0.29	0.94
insertion of NP and/or aux	38%	0.35	0.92
no insertion, skipping at most one word	52%	0.47	0.90
insertion of NP and/or aux, skipping at most one word	63%	0.55	0.88

Table 4.1. Settings, coverage, precision and recall for each pass in the multi-pass parsing strategy.

parsing flexibility (for example, allowing the parser to skip two words in a sentence), more modest gains in recall are obtained with a greater loss in precision.

To decrease the loss in precision when attempting to increase recall by using insertions and word skipping, we use a multi-pass parsing strategy. The main idea is to use robustness (with an expanded search space) only when necessary, and attempt to parse each sentence with the most constrained parser that covers that sentence. In the first pass, we parse all sentences using no parser robustness. The analyses for sentences that were covered by the parser are recorded, and all sentences that were not covered are parsed again in the next pass. Each parsing pass uses more robustness than the previous pass, and the robustness settings are determined using development data. The new settings are the same as the settings for the previous pass, plus the change in robustness parameters (insertions and word skipping) that results in an increase in recall with the smallest drop in precision. A similar idea has been used by Clark and Curran (2004a) and Ninomiya et al. (2005) to improve parser efficiency. In CCG parsing using supertagging, Clark and Curran assign only a small number of categories to each word in a first parsing pass, and increase the number of categories in a second pass if the first pass fails. Ninomiya et al. use a tight beam in their HPSG parser in a first pass, and widen the beam if the first parsing pass fails. In both cases, like in our multi-pass strategy, parsing is attempted with a smaller search space, and upon parse failure the search space is expanded.

Table 4.1 shows the settings for each of the passes in out multi-pass strategy, along with what percentage of the words in the development set can be parsed up to that pass, and what the precision and recall of GRs are up to that pass. The multi-pass strategy allows us to obtain an overall f-score of 0.68, with 0.55 recall and 0.88 precision. More interestingly, table 4.1 shows how the precision/recall tradeoff can be manipulated through LCFlex's robustness settings.

The reason for not pursing more aggressive coverage-increasing techniques in further passes of parsing is that in the context of our overall parser combination strategy the strength of the rule-based approach is that it is capable of high precision. So we prefer having no analysis for an utterance to having an analysis that is very likely to be incorrect. As mentioned in chapter 2, the data-driven approaches we will consider in chapter 5 are

very robust, so it is more important that our rule-based approach be precise than having it achieve high coverage.

4.2.5 Grammar-Driven Parsing and Child Utterances

Syntactic analysis of child utterances poses unique challenges that distinguish it from the task of analyzing adult utterances. While a certain level of grammatical competence is expected from any adult native speaker of a language, verbal ability varies greatly not only among different children, but also in the same child at different ages. This variation makes the application of standard natural language processing techniques to child language in many ways more challenging than their application to adult utterances. While the same parsing technologies used to analyze adult utterances are effective in the analysis of language produced by normally-developing children over the age of five, these techniques are in large part inadequate for the analysis of the language produced by most children under the age of three.

One aspect of parsing that we have discussed in the work descried so far in this chapter is the issue of grammatical coverage. In language produced by adult native speakers, every utterance is expected to be grammatical, and so we attempt to find grammatical relations for every word in every utterance. As we have seen, this is a challenging problem for the rule-based system we have developed, and the system suffers from levels of recall of grammatical relations that are lower than what we would like as a final result. However, in the analysis of child language, the inability to parse certain utterances is, in fact, advantageous. Children at early stages of language acquisition often produce utterances that do not conform to the standard syntax of adult language. Many of the words in such utterances have no clear grammatical relations as they are defined in our syntactic scheme, because their syntactic structure is unclear even to a human linguist. Consider the following sentences from the Sarah section of the Brown corpus (Brown, 1973) in the CHILDES database, where the child is two years and three months old:

1. Daddy store.
2. Gone Mommy.
3. I carrots.
4. *I'm sleepy.
5. Sarah apple.
6. *My horsie.

While we can manually identify grammatical relations in utterances 4 and 6 without much controversy, the syntactic structures of utterances 1, 2, 3 and 5 cannot be determined reliably. If we were to annotate these utterances by hand according to the CHILDES GR annotation scheme, it would be counter-productive to guess if, for example, if "carrots" was meant to be an object in sentence 3. Trying to explain or speculating on the syntactic structures utterances 1, 2, 3 and 5 is an open research question in the child language community (MacWhinney, 1999) and falls outside of the scope of our work.

Our annotation scheme is designed to denote grammatical relations that clearly appear in an utterance. When a word does not have a GR according to the annotation scheme, it should be annotated as such (this is denoted by assigning the word a special GR lable of "NO-GR"). In other words, a system that identifies GRs automatically should also recognize when no GR is present. At the same time, it is important to be able to identify grammatical relations in utterances such as 4 and 6. One way to recognize when GRs are present or not is to determine if an utterance can be parsed by a given grammar or not. Fortunately, when child utterances where such recognition is necessary do contain GRs that should be identified, they are usually less syntactically complex than most adult utterances, and do not pose the same challenge in terms of parser coverage as adult utterances do. This means that we can use the rule-based approach to determine when to assign GRs to an utterance, and when to determine that GR identification should not be attempted. Naturally, in this case, the application of the parsing robustness (limited insertions and skipping) used to analyze adult utterances must be tuned differently, reflecting the goals of this task.

Because our approach to GR analysis in child utterances makes use of the data-driven techniques discussed in chapter 5 in addition to the rule-based techniques described in this chapter, we will postpone the presentation of how we analyze child utterances to chapter 6, which deals with combination strategies for rule-based and data-driven techniques.

4.3 Conclusion

We have presented a rule-based approach for grammatical relation identification in adult utterances in CHILDES data. As we will see in chapter 5, the accuracy of the rule-based system on adult utterances is far below the accuracy we achieve with data-driven systems described. In particular, even though the parser's precision is well above 0.85, recall is at 0.55 or below because of the system's lack of coverage of over 35% of the words in our test set. There is no doubt that the rule-based system described here can be improved. However, in the global scheme of producing a GR analysis system that takes advantage of rule-based and data-driven techniques, we will see that it is not crucial that the precision and recall of GRs of the rule-based system be improved. It should be noted that the rule-

based system is not meant to be used in isolation, but when combined with the approaches described in chapters 5 and 6, will play an important role in highly accurate GR analysis. In summary, the system described in this chapter plays three important roles in the automatic GR annotation of CHILDES data:

1. One of the data-driven systems we will describe in chapter 5 utilizes a large training corpus of close to one million words annotated with syntactic information (the Penn Treebank) to achieve its results. The rule-based system uses only 2,000 words of training data. Although a large corpus of syntactically annotated text exists in English, when analyzing a language for which training data is not available, the techniques presented here represent a sensible alternative to generating one million words of training data (the Penn Treebank is the result of several years of annotation work). Additionally, analyses produced by this system can serve as a starting point in the annotation of a lager training corpus that may be used with the techniques discussed in chapter 5, significantly reducing the amount of manual work required for annotation.

2. As we will see in chapter 6, because the techniques used in the rule-based system are quite different from those used in the data-driven system, the errors in the syntactic structures produced by the two types of systems are often different. This allows for the combination of the results from both systems to produce even higher levels of precision and recall of GRs.

3. The grammar-driven nature of the system is crucial in our overall strategy for parsing child utterances at critical stages of language development, as we began to explore in the previous section, and will fully address in chapter 6.

Chapter 5

Data-Driven GR Parsing Approaches

In chapter 2 we defined a spectrum of approaches to automatic syntactic analysis that reflects how much systems rely on annotated training examples to perform accurate analysis. At one extreme of this spectrum, we have systems that use only human insight encoded as rules, principles and heuristics to define the models of language used during parsing. Although the system presented in chapter 4 uses a relatively small number of annotated training examples (2,000 words) for syntactic disambiguation, that system fits well within the range of what is considered a rule-based parser, where a hand-crafted grammar is the primary source of knowledge. At the other end of the spectrum, we have systems that use no rules at all. Such approaches learn to perform syntactic analysis of text based solely on data, and we will refer to them as data-driven. As discussed in chapter 2, data-driven parsing is in general inherently robust, assigning a syntactic structure to any sequence of words. The coverage/ambiguity trade-off is shifted entirely towards ambiguity: the coverage problem is nearly nonexistent, and the parsing problem is then intrinsically connected to the ambiguity problem. It is generally expected that a larger set of annotated examples is needed for training data-driven systems, often hundreds of thousands of words. Because we are focusing only on the task of supervised syntactic analysis, the data used in these systems consist of sentences annotated with syntactic structure. In this chapter we will describe approaches to GR parsing that fall within this data-driven end of the spectrum.

The GR parsers described here use the two data sets described in chapter 2:

- The Wall Street Journal (WSJ) corpus of the Penn Treebank (Marcus et al., 1993),

which contains about one million words annotated with constituent structures (phrase structures, such as noun phrases, verb phrases, etc.). We divide this corpus into training, development and test sets as has become standard practice with Penn Treebank parsers (see chapter 2).

- The adult utterances in the CHILDES GR Eve corpus (see chapter 2). This includes approximately 5,000 words of training data, 1,200 words of test data, and a 600-word development set. The CHILDES GR Eve corpus is manually annotated according to the CHILDES GR annotation scheme (see chapter 3).

Much of the work on data-driven parsing of English in the past decade has utilized the WSJ corpus of the Penn Treebank. Although the text in the WSJ corpus belongs to a very different domain (newspaper text) than the one we focus on in this work (child-parent dialogs), and the annotation scheme used in the Penn Treebank (constituent trees) differs significantly from our targeted labeled dependencies, there are important reasons for our use of that corpus.

Despite the differences in how syntactic structure is represented in the Penn Treebank and in the CHILDES GR annotation scheme, a well-known procedure exists for converting Penn Treebank constituent structures into unlabeled dependencies (this procedure is described in chapter 2). Converting annotations in the CHILDES GR scheme into unlabeled structures is trivial: we simply remove the dependency labels. Thus, our first reason for using Penn Treebank data is that it provides a large amount of text annotated with unlabeled dependencies, which can be used for training, development and testing of parsing approaches that require large amounts of data. As we will see in this chapter, accurate parsing of text into unlabeled dependencies requires a much larger training set than accurate labeling of existing unlabeled dependencies. It is then possible to achieve high accuracy of GR identification by using a large amount of text annotated with unlabeled dependencies (obtained from the WSJ corpus), and a much smaller amount of text annotated with CHILDES GRs. This drastically reduces the manual annotation effort required for obtaining accurate data-driven GR parsers for CHILDES data. This reduction is a significant advantage, since manually creating a CHILDES GR corpus of size comparable to the WSJ corpus of the Penn Treebank would be a very expensive task.

An additional benefit of using Penn Treebank data is that it allows us to compare the performance of our methods to those of other parsing approaches. The Penn Treebank has been used extensively in parsing research, allowing for a direct comparison of performance of different parsers using the same training and test sets. In fact, the existence of such a platform for direct comparison of systems has fueled much of the progress in parsing in the last ten years.

Although the Penn Treebank is useful as a large training corpus of syntactic structures and as a way to evaluate our systems against the state-of-the-art in data-driven parsing, our ultimate goal is the analysis of CHILDES data. It is clear, then, that the annotated subset of the Eve corpus of the CHILDES database is of great importance. In chapter 4, the Eve training and test sets were used for statistical disambiguation and testing of the rule-based system, respectively. Here, the Eve training set serves as a small training corpus that, when combined with the much larger Penn Treebank training corpus, can be used for learning to produce syntactic structure that conforms to the CHILDES GR annotation scheme. As with the rule-based system, the performance of our data-driven parsers is measured using the Eve test set.

Before we begin our description of data-driven parsing approaches for CHILDES data, it is important to clarify our usage of the terms "data-driven" and "statistical" parsing. Data-driven parsers are frequently referred to as statistical parsers in current natural language processing literature. This makes sense, as most systems that are trained from data estimate statistical models used for parsing using that data. These statistical models often correspond to probabilistic versions of well-known parsing formalisms, such as context-free grammars, or CFGs (Charniak, 1997), combinatorial categorical grammars, or CCGs (Clark and Curran, 2004b), among others. In other cases, however, the term statistical parsing is used to refer to parsers that use statistics in a more indirect way, such as parsers based on neural networks (Henderson, 2004) or decision trees (Magerman, 1995). There are also parsers that rely heavily on statistics trained on corpora, but also use hand-written grammars, such as the parser of Briscoe and Carroll (2002) discussed in chapter 2. These are statistical parsers that fall somewhere in between the rule-based and the data-driven end of the spectrum, closer to the data-driven end than our rule-based parser (chapter 4), but closer to the rule-based end than the approaches described in this chapter. There are also cases where the parameters of the statistical models associated with hand-written grammars may even be set by hand, and not learned from data (Dzikovska et al., 2005). Therefore, our use of the term data-driven parsing refers to systems that learn to parse primarily from data, regardless of whether or not the use of statistical models is a central focus of the approach (although it often is). Conversely, we use the term statistical parsing to refer to systems that rely primarily on statistical models, regardless of the extent data is used to learn the parameters of those models.

In this chapter, we will consider two approaches to data-driven parsing for analysis of CHILDES GRs. The first approach is based on lexicalized probabilistic context-free grammars (lexicalized PCFGs), where a statistical generative model is used for syntactic analysis. The second uses classifier-based linear-time parsing, where a classifier is used to determine parser actions. The specific classifier-based parsers described here are also statistical parsers, since the classifiers used are based on statistical models. However, as

we will see, one aspect of the classifier-based parsing approach is that the classifier may be used as a black-box. The parsing approach is not concerned with the statistics inside the classifier, so we will refer to the approach simply as classifier-based. Both approaches presented here (one based primarily on a generative statistical model, and one based on discriminative classification) are data-driven.

As was the case in chapter 4, the evaluations using CHILDES data in this chapter use only adult utterances. As explained in the end of chapter 4, our approach to the analysis of child language uses a combination of the rule-based techniques described in that chapter, and the data-driven techniques described in this chapter. Child language analysis will be addressed in chapter 6, along with other issues related to the combination of diverse analysis strategies.

5.1 GR Parsing with a Generative Statistical Parser

This first data-driven GR parsing approach we describe illustrates how existing natural language processing tools and resources can be utilized to produce a high-performance system for GR analysis in CHILDES data with relatively little cost. This approach is particularly useful in languages for which accurate parsers already exist. The main idea is to obtain syntactic parses for the target data using an existing parser, and convert those analyses into GRs in the CHILDES annotation scheme. The GR system we developed according to this general idea analyzes CHILDES transcripts in three steps: text preprocessing, unlabeled dependency identification, and dependency labeling. In the following subsections, we examine each of these steps in detail.

5.1.1 Text Preprocessing

The CHAT transcription system (see chapter 2) is the format followed by all transcript data in the CHILDES database, and it is the input format we use for syntactic analysis. CHAT specifies ways of transcribing extra-grammatical material such as false-starts, retracings, and repetitions, common in spontaneous spoken language, especially in child language. Transcripts of spontaneous parent-child language may contain a large amount of extra-grammatical material that falls outside of the scope of the syntactic annotation system and our GR identifier, since it is already clearly marked in CHAT transcripts.

There are freely available text processing tools, distributed in the CLAN package (MacWhinney, 2000), that are designed specifically for processing transcripts in CHAT format. The CLAN tools allow us to easily remove the false-starts, retracings and repetitions

that have been marked in the transcripts. In the rule-based system described in chapter 4, this is accomplished by the use of the MOR program (which is also part of the CLAN tools). Removing this extra-grammatical material provides us with sentences that can then be more accurately analyzed by a parser trained on written text (such as the WSJ corpus). This type of preprocessing is comparable to the edit-detection techniques (Charniak and Johnson, 2001) for statistical parsing of the spoken language in the Switchboard corpus, which contains transcribed telephone conversations. In that work, extra-grammaticality is detected and marked automatically. In the case of CHILDES data, that detection and marking is done manually during transcription, so there is no need for automatic detection.

It is important to note, however, that the removal of material marked as extra-grammatical does not make every utterance in a transcript conform to the grammatical standard commonly expected from adult English usage. In fact, certain utterances still deviate significantly from typical written language. Not surprisingly, this is particularly common in utterances produced by children. In the rule-based parser, the use of LCFlex robustness settings is used both to remedy lack of grammatical coverage and to handle small grammatical deviations in adult utterances. Since data-driven parsing is inherently robust to grammatical deviations from the training material, we do not face the coverage issues we encountered with the rule-based system. The analysis of child utterances that do not conform to standard adult English grammar, however, is a greater challenge that we address with a combination of the rule-based work of chapter 4 and the data-driven work described in this chapter. For the moment, we will simply parse every sentence as if it were fully grammatical after the marked extra-grammatical material has been removed.

5.1.2 Unlabeled Dependency Identification

Once we have isolated the text that should be analyzed in each sentence, we parse it to obtain unlabeled dependencies. Although we ultimately need labeled dependencies, as this is how GRs are represented in the CHILDES annotation scheme, our choice to produce unlabeled structures first (and label them in a later step) is motivated by available resources. Unlabeled dependencies can be readily obtained by processing constituent trees, such as those in the Penn Treebank (Marcus et al., 1993), with a set of rules to determine the lexical heads of constituents. This lexicalization procedure is commonly used in statistical parsing (Magerman, 1995; Collins, 1996) and produces a dependency tree (see chapter 2). The dependency extraction procedure from constituent trees gives us a straightforward way to obtain unlabeled dependencies: use an existing statistical parser trained on the Penn Treebank to produce constituent trees, and extract unlabeled dependencies using the aforementioned head-finding rules. The parser we use is the Charniak (2000) parser, a popular statistical parser that uses a generative statistical model based on a lexicalized

probabilistic context-free grammar (PCFG). The Charniak parser is very accurate on WSJ data, and it is described in more detail in chapter 2.

CHILDES language is from a very different domain than the one of the data used to train the statistical parser (the Wall Street Journal section of the Penn Treebank), and any system trained on data is expected to suffer performance degradation when applied to data outside of the domain of the training data. Fortunately, the degradation in the parser's accuracy is surprisingly small. An evaluation using the 2,000-word Eve test set with manually annotated dependencies shows that the dependency accuracy of the parser is 90.1% on child language transcripts (compared to over 92% on the WSJ test set). In this evaluation, the dependency labels in the Eve test set were ignored, and unlabeled dependency accuracy was computed simply as the number of words with a correct head assignment (forming a correct unlabeled dependency) divided by the total number of words (the total number of dependencies) in the test set.

The high accuracy of the Charniak parser on CHILDES data may be explained by the fact that sentences in parent-child dialogs are much less syntactically complex than sentences in the Wall Street Journal. In addition, despite the many differences with respect to the domain of the training data, our domain features sentences that are much shorter (and therefore easier to parse) than those found in Wall Street Journal articles. The average sentence length varies from transcript to transcript, because of factors such as the age and verbal ability of the child, but it is usually less than 15 words, even for adult utterances.

Although we refer to the dependencies obtained at this point as unlabeled, we do in fact have labels associated with each dependency. These labels are simply the phrasal-type labels of the lowest constituents (NP, VP, etc.) containing the head and dependent words, and can be read directly from the constituent tree during the extraction of dependencies. Although these are constituent labels, and not truly meaningful as the labels of dependencies, they are helpful in determining the actual dependency labels in the final GR analysis step.

5.1.3 Dependency Labeling

After obtaining unlabeled dependencies using the Charniak parser and a set of rules to transform its constituent output into dependencies, we proceed to label those dependencies with the GR labels listed in chapter 3.

Determining the labels of dependencies is in general an easier task than finding unlabeled dependencies in text. Klein and Manning (2002) offer an informal argument that shows that constituent labels are much more easily separable in multidimensional space than constituents/distituents. Since dependencies are very closely related to constituents

(as we have seen, a small set of rules converts constituents to dependencies), the same argument applies to dependency labels and dependencies/non-dependencies. In practice, this means that a much smaller training corpus of labeled dependencies is necessary for training a system that performs accurate dependency labeling than what is required to train a dependency parser. In other words, although a corpus of almost one million words (the WSJ corpus) was used to train the parser that provides us with unlabeled dependencies, we can train a system to label those dependencies using the much smaller Eve GR training corpus (5,000 words).

Our approach to dependency labeling is to use a classifier, where the target classes are each of the 30 possible GR labels in the CHILDES GR scheme. For each dependency that we label, we give the classifier local features to the dependency and obtain the appropriate class. The classifer used for dependency labeling is TiMBL (Daelemans et al., 2003), a memory-based learner (set to use the k-nn algorithm with k=1, and gain ratio weighing). To train the classifier, we extract features from each of the 5,000 dependencies in the Eve GR training corpus, and pair each set of features with the correct GR label for that dependency. When we want to label dependencies extracted from the output of the Charniak parser, we simply extract features from each unlabeled dependency, and let the classifier choose a GR label.

The local features we extract for each dependency are:

- The head and dependent words;
- The head and dependent parts-of-speech;
- Whether the dependent comes before or after the head in the sentence;
- How many words apart the dependent is from the head;
- The label of the lowest node in the constituent tree that includes both the head and dependent words.

The accuracy of the classifier in labeling perfect unlabeled dependencies is 91.4%. This accuracy is determined by using TiMBL to label the dependencies of the Eve GR test set (to determine the accuracy of dependency labeling alone, we simply use the manually annotated dependencies, without their labels). Accuracy is calculated simply by counting the number of correct label assignments and dividing it by the total number of dependencies in the test corpus. The set of features listed above was tuned separately on the Eve development set (of 500 words).

When we combine the unlabeled dependencies obtained with the Charniak parser (and head-finding rules) and the labels obtained with the memory-based classifier, overall labeled

GR	Precision	Recall	F-score
SUBJ	0.94	0.93	0.93
OBJ	0.83	0.91	0.87
COORD	0.68	0.85	0.75
JCT	0.91	0.82	0.86
MOD	0.79	0.92	0.85
PRED	0.80	0.83	0.81
ROOT	0.91	0.92	0.91
COMP	0.60	0.50	0.54
XCOMP	0.58	0.64	0.61

Table 5.1. Precision, recall and f-score (harmonic mean) of selected GRs obtained with the statistical parsing approach.

dependency accuracy is 86.9%. Although not directly comparable, this result is in agreement with other state-of-the-art results in grammatical relation and labeled dependency parsing (see chapter 2). Certain frequent and easily identifiable GRs, such as DET, POBJ, INF, and NEG were identified with precision and recall above 98%. Among the most difficult GRs to identify were clausal complements COMP and XCOMP, which together amount to less than 4% of the GRs seen the training and test sets. Table 5.1.3 shows the precision and recall of GRs of particular interest. See appendix B for a complete list showing the precision and recall performance of this system on each of the 30 GRs in the CHILDES annotation scheme.

5.2 GR Analysis with Classifier-based Parsing

Although the statistical parsing approach to GR identification described in the previous section performs well for most GRs in our annotation scheme, there are important reasons for also using additional data-driven approaches to GR analysis. One of the main reasons, which is discussed in detail in chapter 6 and is directly related to one of the central themes in this thesis, is that we can improve the precision and recall of identification of individual GRs as well as overall system accuracy by utilizing several different GR identification approaches. Henderson and Brill (1999) have shown that, in the context of constituent parsing of the Penn Treebank, combining the outputs of different parsers can result in improved accuracy, even if each parser uses the same training data. As we show in chapter 6, this aspect of parser diversity is also applicable to dependency parsing, and combination schemes using different approaches to GR parsing result in improved precision and recall of GRs.

Another reason for considering additional GR analysis approaches is that the data-driven system described in the previous section is tied to a corpus of constituent structures.

The work described here deals with the syntactic analysis of English, and it would be unwise to ignore the existence of resources that facilitate accurate GR parsing and save us the cost of creating large amounts of training material. However, English is a special case where these resources (state-of-the-art parsers and large corpora annotated with syntactic structures) do exist. Applying many of the techniques we present here to other languages may require the creation of additional training material, or the use of corpora annotated with different types of syntactic structures. Since our annotation scheme is based on labeled dependencies, it would make sense that annotation efforts should follow that formalism. Additionally, if labeled dependency resources do exist, it would not make sense to depend solely on systems that require the existence of a corpus of constituent structures. Finally, as we will see, even a very small training corpus (5,000 words) of domain-specific data annotated in the CHILDES GR scheme allows us to achieve highly accurate results using a labeled dependency parser that works in a single step. To address these issues, in this section we present a GR analysis approach that makes it easy to quickly obtain a number of diverse parsers that use the same training material but can be used in combination to potentially obtain better results, and can be used in the context of constituent or (labeled or unlabeled) dependency parsing with only small modifications. This approach is classifier-based parsing.

Parsers that rely mainly on classifiers have recently been proposed by Yamada and Matsumoto (2003) and Nivre and Scholz (2004). These parsers work deterministically to build dependency structures for a sentence, using classifiers to determine parser actions (see chapter 2 for a description of these two parsers). Although Yamada and Matsumoto's parser is more accurate than Nivre and Scholz's, its algorithm is less efficient. Yamada and Matsumoto use an algorithm of quadratic run-time complexity, while Nivre and Scholz algorithm is linear. We present a linear-time parsing approach that extends this previous research in deterministic classifier-based parsing, resulting in a more general framework that allows for training and analysis using either constituent structures (such as those in the Penn Treebank) or dependency structures (such as those in the CHILDES GR annotation scheme). We first describe the more general case of a deterministic classifier-based constituent parser, and evaluate the performance of that parser on the WSJ corpus to provide a basis for comparison with other recent parsing research. We then show how it can be easily modified into an accurate dependency parser for CHILDES GRs, and evaluate the performance of the dependency parser on the Eve test corpus.

5.2.1 Classifier-Based Constituent Parsing in Linear Time

Our classifier-based parser is remarkably simple to understand and implement. An additional feature of our approach is its modularity with regard to the algorithm and the classifier that determines the parser's actions. This makes it easy for different classifiers

and different sets of features to be used with the same parser with very minimal work. Finally, its run-time complexity is linear on the size of the input sentence.

Like other deterministic parsers (and unlike many statistical parsers), our parser considers the problem of syntactic analysis separately from part-of-speech (POS) tagging. Because the parser greedily builds trees bottom-up in one pass, considering only one path at any point in the analysis, the task of assigning POS tags to words is done before other syntactic analysis. In the subsections that follow, we assume that the input to the parser is a sentence with corresponding POS tags for each word.

Our parser employs a basic bottom-up shift-reduce parsing algorithm, requiring only a single pass over the input string. The algorithm considers only trees with unary and binary branching. In order to use trees with arbitrary branching for training, or generating them with the parser, we employ an instance of the transformation/detransformation process described in (Johnson, 1998). In our case, the transformation step involves simply converting each production with n children (where $n > 2$) into $n - 1$ binary productions. Trees must be lexicalized, so that the newly created internal structure of constituents with previous branching of more than two contains only subtrees with the same lexical head as the original constituent. Additional non-terminal symbols introduced in this process are clearly marked. The transformed (or "binarized") trees may then be used for training. Detransformation is applied to trees produced by the parser. This involves the removal of non-terminals introduced in the transformation process, producing trees with arbitrary branching. An example of transformation/detransformation is shown in figure 5.1.

Algorithm Outline

The parsing algorithm involves two main data structures: a stack S, and a queue W. Items in S may be terminal nodes (POS-tagged words), or (lexicalized) subtrees of the final parse tree for the input string (since terminal nodes are also subtrees of the final parse tree, all items in S are in fact subtrees). Items in W are terminals (words tagged with parts-of-speech) corresponding to the input string. When parsing begins, S is empty and W is initialized by inserting every word from the input string in order, so that the first word is in front of the queue.

Only two general actions are allowed: shift and reduce. A shift action consists only of removing (shifting) the first item (POS-tagged word) from W (at which point the next word becomes the new first item), and placing it on top of S. Reduce actions are subdivided into unary and binary cases. In a unary reduction, the item on top of S is popped, and a new item is pushed onto S. The new item consists of a tree formed by a non-terminal node with the popped item as its single child. The lexical head of the new item is the same as

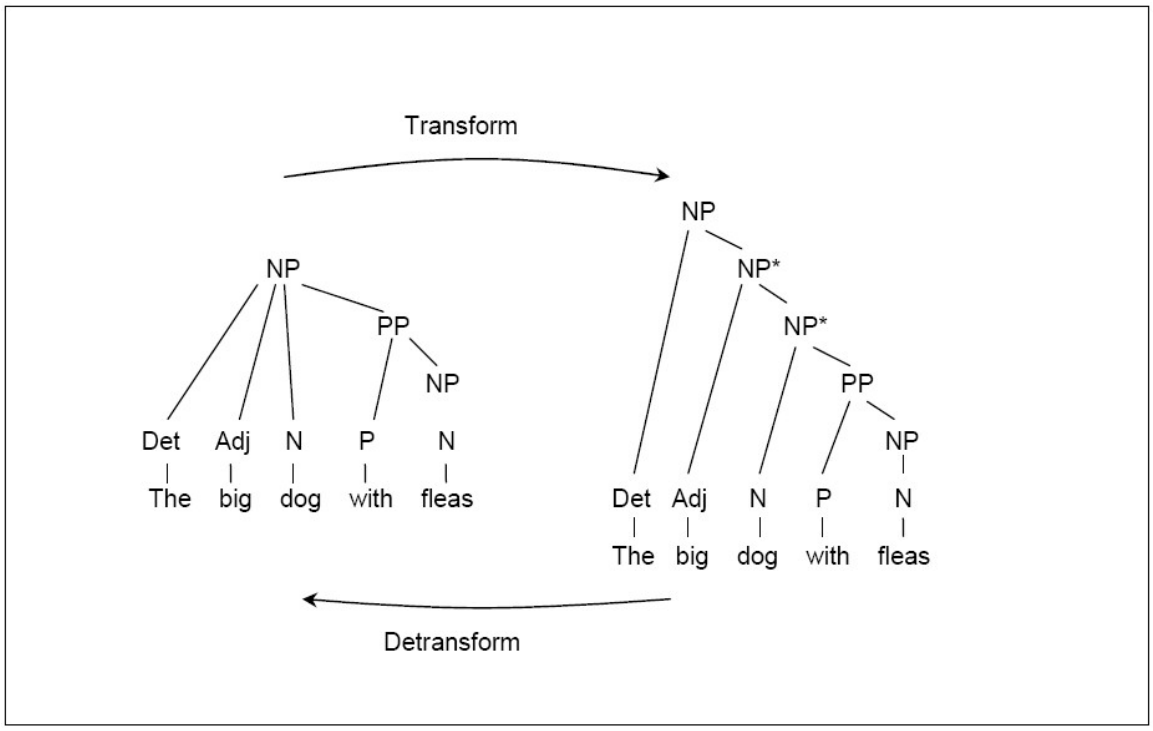


Figure 5.1. An example of the binarization/de-binarization transform.

the lexical head of the popped item. In a binary reduction, two items are popped from S in sequence, and a new item is pushed onto S. The new item consists of a tree formed by a non-terminal node with two children: the first item popped from S is the right child, and the second item is the left child. The lexical head of the new item is either the lexical head of its left child, or the lexical head of its right child.

If S is empty, only a shift action is allowed. If W is empty, only a reduce action is allowed. If both S and W are non-empty, either shift or reduce actions are possible. Parsing terminates when W is empty and S contains only one item, and the single item in S is the parse tree for the input string. Because the parse tree is lexicalized, we also have a dependency structure for the sentence. In fact, the binary reduce actions are very similar to the reduce actions in the labeled dependency parser of Nivre and Scholz (2004), but they are executed in a different order, so constituents can be built (we will address dependency parsing using a slightly modified version of this algorithm later in this chapter). If W is empty, and more than one item remain in S, and no further reduce actions take place, the input string is rejected.

Determining Actions with a Classifier

A parser based on the algorithm described in the previous section faces two types of decisions to be made throughout the parsing process. The first type concerns whether to shift or reduce when both actions are possible, or whether to reduce or reject the input when only reduce actions are possible. The second type concerns what syntactic structures are created. Specifically, what new non-terminal is introduced in unary or binary reduce actions, and which of the left or right children are chosen as the source of the lexical head of the new subtree produced by binary reduce actions. Traditionally, these decisions are made with the use of a grammar, and the grammar may allow more than one valid action at any single point in the parsing process. When multiple choices are available, a grammar-driven parser may make a decision based on heuristics or statistical models, or pursue every possible action following a search strategy. In our case, both types of decisions are made by a classifier that chooses a unique action at every point, based on the local context of the parsing action, with no explicit grammar. Because in this type of classifier-based parsing only one path is pursued with no backtracking, the parsing process can be viewed as greedy, or deterministic.

In order to determine what actions the parser should take given a particular parser configuration, a classifier is given a set of features derived from that configuration. This includes, crucially, the two topmost items (subtrees) in the stack S, and the item in front of the queue W (the next word in the input string). Additionally, a set of context features is derived from a (fixed) limited number of items below the two topmost items of S, and following the item in front of W. The specific features are shown in figure 5.2.1. These features are generally intended to represent the state of the parser through the relevant contents of the stack and queue, and also the pieces of syntactic structure that have been built up to the current point in the parsing process. The amount of context (the number of look-ahead items in the queue and the number of stack items) that is included in the representation of the state of the parser through these features was determined empirically by tuning on the development set.

The classifier's target classes are parser actions that specify both types of decisions mentioned above. These classes are:

SHIFT : a shift action is taken;

REDUCE-UNARY-XX : a unary reduce action is taken, and the root of the new subtree pushed onto S is of type XX (where XX is a non-terminal symbol, typically NP, VP, PP, for example);

REDUCE-LEFT-XX : a binary reduce action is taken, and the root of the new subtree

Let:

$S(n)$ denote the nth item from the top of the stack S, and
$W(n)$ denote the nth item from the front of the queue W.

Features:

1.	The head-word (and its POS tag) of: $S(0)$, $S(1)$, $S(2)$, $and S(3)$
2.	The head-word (and its POS tag) of: $W(0)$, $W(1)$, $W(2)$ and $W(3)$
3.	The non-terminal node of the root of: $S(0)$, and $S(1)$
4.	The non-terminal node of the left child of the root of: $S(0)$, and $S(1)$
5.	The non-terminal node of the right child of the root of: $S(0)$, and $S(1)$
6.	The POS tag of the head-word of the left child of the root of: $S(0)$, and $S(1)$
7.	The POS tag of the head-word of the right child of the root of: $S(0)$, and $S(1)$
8.	The linear distance (number of words apart) between the head-words of $S(0)$ and $S(1)$
9.	The number of lexical items (words) that have been found (so far) to be dependents of the head-words of: $S(0)$, and $S(1)$
10.	The most recently found lexical dependent of the head of the head-word of $S(0)$ that is to the left of the head-word of: $S(0)$ and $S(1)$
11.	The most recently found lexical dependent of the head of the head-word of $S(0)$ that is to the right of the head-word of: $S(0)$ and $S(1)$
12.	The previous parser action

Figure 5.2. Features used for classification. The features described in items 1 – 5 are more directly related to the lexicalized constituent trees that are built during parsing, while the features described in items 6 – 11 are more directly related to the dependency structures that are built simultaneously to the constituent structures.

pushed onto S is of non-terminal type XX. Additionally, the head of the new subtree is the same as the head of the left child of the root node;

REDUCE-RIGHT-XX : a binary reduce action is taken, and the root of the new subtree pushed onto S is of non-terminal type XX. Additionally, the head of the new subtree is the same as the head of the right child of the root node.

A Complete Classifier-Based Parser that Runs in Linear Time

When the algorithm described so far is combined with a trained classifier that determines its parsing actions as described above, we have a complete classifier-based parser. Training the parser is accomplished by training its classifier. To that end, we need training instances that consist of sets of features paired with their classes corresponding to the correct parsing actions. These instances can be obtained by running the algorithm on a corpus of sentences for which the correct parse trees are known. Instead of using the classifier to determine the parser's actions, we simply determine the correct action by consulting the correct parse trees. We then record the features and corresponding actions for parsing all sentences in the corpus into their correct trees. This set of features and corresponding actions is then used to train a classifier, resulting in a complete parser.

When parsing a sentence with n words, the parser takes n shift actions (exactly one for each word in the sentence). Because the maximum branching factor of trees built by the parser is two, the total number of binary reduce actions is $n-1$, if a complete parse is found. If the input string is rejected, the number of binary reduce actions is less than $n-1$. Therefore, the number of shift and binary reduce actions is linear with the number of words in the input string. However, the parser as described so far has no limit on the number of unary reduce actions it may take. Although in practice a parser properly trained on trees reflecting natural language syntax would rarely make more than $2n$ unary reductions, pathological cases exist where an infinite number of unary reductions would be taken, and the algorithm would not terminate. Such cases may include the observation in the training data of sequences of unary productions that cycle through (repeated) non-terminals, such as $A \rightarrow B \rightarrow A \rightarrow B$. During parsing, it is possible that such a cycle may be repeated infinitely.

This problem can be easily prevented by limiting the number of consecutive unary reductions that may be made to a finite number. This may be the number of non-terminal types seen in the training data, or the length of the longest chain of unary productions seen in the training data. In our experiments (described below), we limited the number of consecutive unary reductions to three, although the parser never took more than two unary reduction actions consecutively in any sentence. When we limit the number of consecutive

unary reductions to a finite number m, the parser makes at most $(2n-1)*m$ unary reductions when parsing a sentence of length n. Placing this limit not only guarantees that the algorithm terminates, but also guarantees that the number of actions taken by the parser is $O(n)$, where n is the length of the input string. Thus, the parser runs in linear time, assuming that classifying a parser action is done in constant time.

Experiments Using the WSJ Corpus

To compare the classifier-based parser approach to other well-known data-driven parsing approaches, we conducted experiments with our classifier-based parser for constituents using two different classifiers: TinySVM (a support vector machine implementation by Taku Kudo), and the memory-based learner TiMBL (Daelemans et al., 2004). We trained and tested the parser on the Wall Street Journal corpus of the Penn Treebank (Marcus et al., 1993) using the standard split used for evaluation of several English parsers (as described in chapter 2): sections 2-21 were used for training, section 22 was used for development and tuning of parameters and features, and section 23 was used for testing. The experiments reported here were performed on a Pentium 4 1.8GHz with 1GB of RAM.

Each tree in the training set had empty-node and function tag information removed, and the trees were lexicalized using similar head-table rules as those mentioned in (Collins, 1996). The trees were then converted into trees containing only unary and binary branching, using the binarization transform described earlier in this chapter. Classifier training instances of features paired with classes (parser actions) were extracted from each of the trees in the training set. The total number of training instances was about 1.5 million.

The classifier in the SVM-based parser (denoted by SVMpar) uses the polynomial kernel with degree 2, following the work of Yamada and Matsumoto (2003) on SVM-based deterministic dependency parsing, and a one-against-all scheme for multi-class classification. Because of the large number of training instances, we used Yamada and Matsumoto's idea of splitting the training instances into several parts according to POS tags, and training classifiers on each part. This greatly reduced the time required to train the SVMs, but even with the splitting of the training set, total training time was about 62 hours. Training set splitting comes with the cost of reduction in accuracy of the parser, but training a single SVM would likely take more than ten days. Yamada and Matsumoto experienced a reduction of slightly more than 1% in (unlabeled) dependency accuracy due to training set splitting, and we expect that a similar loss is incurred here.

When given perfectly tagged text (gold tags extracted from the Penn Treebank), SVMpar has labeled constituent precision and recall of 87.54% and 87.61%, respectively, and unlabeled dependency accuracy of 90.3% over all sentences in the test set (see section 2 for

a description of how constituent precision/recall and dependency accuracy are calculated). The total time required to parse the entire test set was 11 minutes. Out of more than 2,400 sentences, only 26 were rejected by the parser (about 1.1%). For these sentences, partial analyses were created by combining the items in the stack in flat structures, and these were included in the evaluation. Predictably, the labeled constituent precision and recall obtained with automatically POS-tagged sentences were lower, at 86.01% and 86.15%. Unlabeled dependency accuracy was only slightly lower, at 90.0%. The part-of-speech tagger used in our experiments was SVMTool (Giménez and Màrquez, 2004), and its accuracy on the test set is 97%.

The MBL-based parser (denoted by MBLpar) uses the IB1 algorithm, with five nearest neighbors, and the modified value difference metric (MVDM), following the work of Nivre and Scholz (2004) on MBL-based deterministic dependency parsing. MBLpar was trained with all training instances in under 15 minutes, but its accuracy on the test set was much lower than that of SVMpar, with constituent precision and recall of 80.0% and 80.2%, and dependency accuracy of 86.3% (24 sentences were rejected). It was also much slower than SVMpar in parsing the test set, taking 127 minutes. In addition, the total memory required for running MBLpar (including the classifier) was close to 1 gigabyte, while SVMpar required only about 200 megabytes (including all the classifiers).

Table 5.2 shows a summary of the results of our experiments with SVMpar and MBLpar, and also results obtained with the Charniak (2000) parser, the Bikel (2002) implementation of the Collins (1997) parser, and the Ratnaparkhi (1997) parser. We also include the dependency accuracy from Yamada and Matsumoto (2003)'s SVM-based dependency parser, and Nivre and Scholz (2004)'s MBL-based dependency parser. These results show that the choice of classifier is extremely important in this task. SVMpar and MBLpar use the same algorithm and features, and differ only on the classifiers used to make parsing decisions. While in many natural language processing tasks different classifiers perform at similar levels of accuracy, we have observed a dramatic difference between using support vector machines and a memory-based learner. We speculate that a relatively small difference in initial classifier accuracy results in larger differences in parser performance, due to the deterministic nature of the parser (certain errors may lead to further errors). Furthermore, SVMs may be more robust than MBL in classifying parser actions given parser configurations that deviate significantly from the ones seen in training data. Such configurations may result after the parser has already made a mistake, leading the process into the wrong path. We also believe classifier choice to be one major source of the difference in accuracy between Nivre and Scholz's parser and Yamada and Matsumoto's parser. It should be noted that the tuning of the feature set for our parser was done using MBL more heavily than SVMs, since MBL's training time is much faster. While the accuracy of SVMpar is below that of

Parser	Precision	Recall	Dependency	Time (min)
Charniak	89.5	89.6	92.1	28
Collins	88.3	88.1	91.5	45
Ratnaparkhi	87.5	86.3	Unk	Unk
Y&M*	–	–	90.3	Unk
N&S*	–	–	87.3	21
MBLpar	80.0	80.2	86.3	127
SVMpar	86.0	86.1	90.0	11

Table 5.2. Summary of results obtained with the classifier-based constituent parser as labeled precision and recall of constituents, dependency accuracy, and time required to parser the test set (unk denotes unknown values). *The parsers of Yamada and Matsumoto (Y&M) and Nivre and Scholz (N&S) do not produce constituent analyses.

lexicalized search-based statistical parsers, it is surprisingly good for a greedy parser that runs in linear time.

5.2.2 Classifier-Based Dependency Parsing in Linear Time

Obtaining unlabeled dependencies from the deterministic classifier-based constituent parser is a trivial task, since dependencies and constituents are determined simultaneously. A dependency is formed each time the parser takes a binary reduce action. In a REDUCE-LEFT action, the head word associated with the topmost node of the lexicalized subtree on top of the stack (or more simply, the head of that subtree) is a dependent of the head of the subtree of the second item (from the top) in the stack. Conversely, in a REDUCE-RIGHT action, the head of the second item in the stack is a dependent of the head of the subtree on top of the stack. Alternatively, dependencies could easily be extracted from the constituent output of the parser by lexicalization of the constituents using head-percolation rules, as was done with the output of the Charniak parser in section 5.1.2.

In this section, we address the situation where we have training data in the form of text annotated with dependencies, not constituents. The importance of such a scenario is clear, since the CHILDES GR annotation scheme is based on dependencies. If in-domain training data were to be annotated using the CHILDES GR scheme, it would be necessary to have a data-driven parser that can be trained from dependencies. As previously mentioned, the deterministic constituent parser can be easily transformed into a dependency parser.

In a well-formed dependency structure for a sentence, each dependency involves exactly two words, and each REDUCE by our deterministic parser corresponds the creation of a dependency, as mentioned above. Since no word-to-word dependency is created by unary reduce actions, there is no need for a parser to consider unary expansions. This means

that a dependency parser would not include actions of the type REDUCE-UNARY. Binary reduce actions in our classifier-based constituent parser had the form REDUCE-LEFT-*XX* or REDUCE-RIGHT-*XX*. In either case, *XX* corresponds to a constituent label. In the case of dependencies, *XX* corresponds to the dependency label (which is the GR type in the CHILDES annotation scheme). The last difference between constituent and dependency parsing in our classifier-based framework is the order in which syntactic structures are built. Consider the sentence "I eat cake with Joe," whose constituent and dependency structures are seen in figure 5.3. In constituent parsing, the dependency between "cake" and "eat" is formed before the dependency between "I" and "eat". This is allows the VP constituent to be formed before the S constituent. In dependency parsing there are no constituents to build, so the order in which dependencies are determined may differ from the order in constituent parsing. The single restriction on the order dependencies are built is that the dependent word in a newly constructed dependency must not be a head in any dependencies not yet built by the parser. This means that the dependency between "Joe" and "with" must be built before the dependency between "with" and "eat". This is accomplished trivially in constituent parsing because of the PP constituent, but in dependency parsing the restriction that only words that are not heads of remaining dependencies can be dependents must be placed explicitly. Therefore, when we process our example from left to right, the first dependency formed is between "I" (dependent) and "eat" (head), followed by "cake" (dependent) and "eat" (head). The dependency between "with" (dependent) and "eat" (head) cannot be formed at this point because "with" is the head of "Joe", so the next dependency formed is between "Joe" (dependent) and "with" (head). Finally, we can form the dependency between "with" (dependent) and "eat" (head).

In summary, the necessary changes to transform the constituent parser into a dependency parser are:

- No unary reduce actions exist;
- Binary reductions function in a similar manner as they do with constituents, but the dependency label must be decided instead of the constituent label;
- The substructures built as a result of reduce actions are dependency structures, not constituent subtrees;
- An explicit restriction on the dependency processing order must be placed so that only words that are not heads in any remaining dependencies can be dependents.

Making these changes to the constituent parser results in an algorithm similar to the one used in the deterministic dependency parser of Nivre and Scholz (2004). The main difference is that Nivre and Scholz's algorithm allows words to become dependents before all of their

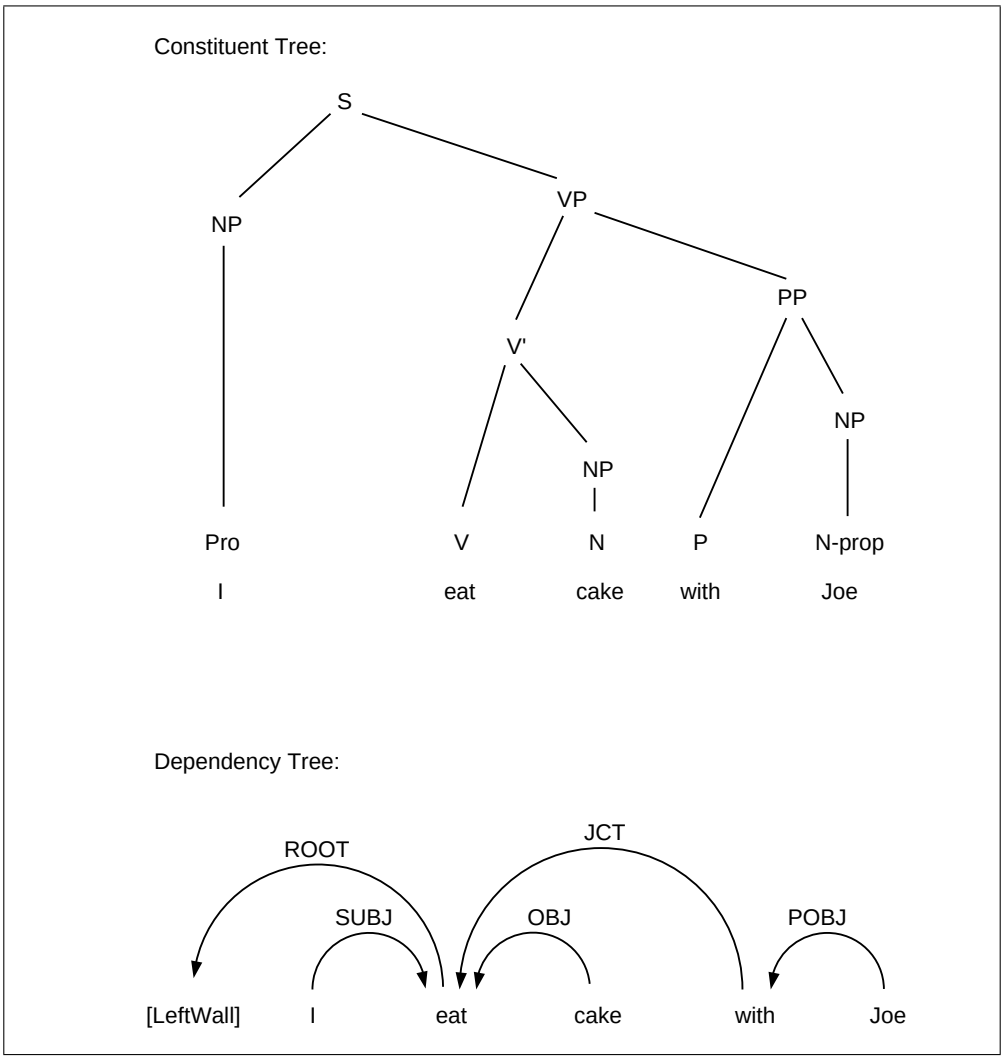


Figure 5.3. Constituent and dependency trees for the sentence "I eat cake with Joe".

own dependents have been found, while in our algorithm once a word is attached as a dependent, no other words can attach to it as dependents. This difference comes from a subtle difference in the two algorithm: Nivre and Scholz have separate parser actions for reducing (popping the stack) and for creating dependencies, while in our algorithm dependencies are always created in reduce actions. In addition, Nivre and Scholz use a slightly different set of features in their parser.

Experiments

We now have a data-driven dependency parser that can be trained directly on text annotated according to the CHILDES GR annotation scheme (and also produce output in

the CHILDES GR scheme directly). This allows us to address the important question of whether it is possible to obtain high precision and recall of CHILDES GRs using only our 5,000-word Eve training corpus. To address this issue, we trained the dependency parser with the Eve training corpus. As with our evaluation of the constituent parser on the WSJ corpus, we trained two versions of the parser: one using support vector machines, and one using k-nearest neighbors (using the TinySVM and TiMBL packages, as before). We will refer to these parsers as SVMdep and MBLdep. Because the training set used in these experiments is much smaller than the one used in the constituent parser experiments, there was no need to split the training data when training the SVM classifier. Training times for SVMdep and MBLdep were under two minutes.

The accuracy of SVMdep on the Eve test set is surprisingly high, given that the training set is so small. The overall labeled accuracy is 87.3%, putting the classifier-based deterministic dependency parser at the same level achieved with the system based on the Charniak parser, but using a much smaller training set (of domain-specific data). For comparison purposes, the unlabeled dependency accuracy is 90.0%. Unlike with SVMpar (the classifier-based constituent parser), SVMdep uses the linear kernel. Using the polynomial kernel (which gives us better results in SVMpar) results in lower accuracy, due to the much smaller size of the training set used by SVMdep. Tuning of all parameters, including kernel choice, was done using the 500-word Eve development set (the Eve test set was used only in final evaluations).

MBLdep's accuracy is lower than SVMdep's, but the gap between the two parsers is smaller than the difference between SVMpar and MBLpar. MBLdep's labeled dependency accuracy is 85.1% (with unlabeled dependency accuracy of 89.1%). Like MBLpar, MBLdep uses the k-nn algorithm with k=3 and the modifier value distance metric (MVDM). Table 5.3 shows the results obtained on several GRs by the two versions of the classifer-based dependency parser. Results on all GRs in the CHILDES annotation scheme can be found in appendix B. While SVMdep is in general more accurate than MBLdep, it is interesting that MBLdep is better at identifying coordination. We also note that the statistical approach based on the Charniak parser and a separate pass for dependency labeling is more accurate in identifying adjuncts than either one of the classifier-based parsers. Reasons for this include the use of a much larger training set in the Charniak parser, and the greedy nature of the classifier-based parsers.

To determine the effect of the amount of training data used with our classifier-based parsing approach, we trained SVMdep on subsets of the Eve GR training corpus containing different amounts of data. The results of this experiment are shown in figure 5.2.2. There is a large improvement in labeled accuracy (from 83.7% to 86.1%) when the size of the training set is increased from 3,000 words to 4,000 words. The absolute improvement from

	SVMdep			MBLdep		
GR Precision	Recall	F-score	Precision	Recall	F-score	
SUBJ	0.97	0.98	0.98	0.95	0.96	0.95
OBJ	0.89	0.94	0.92	0.87	0.85	0.86
COORD	0.71	0.76	0.74	0.83	0.76	0.79
JCT	0.78	0.88	0.83	0.77	0.82	0.80
MOD	0.94	0.87	0.91	0.95	0.85	0.90
PRED	0.80	0.83	0.82	0.75	0.81	0.78
ROOT	0.95	0.94	0.94	0.93	0.92	0.92
COMP	0.70	0.78	0.74	0.62	0.56	0.59
XCOMP	0.93	0.82	0.87	0.82	0.82	0.82

Table 5.3. Precision, recall and f-score (harmonic mean) of GRs obtained with classifier-based dependency parsing.

going from 4,000 words to 5,000 words is more modest, while still remarkable (from 86.1% to 87.3%). This trend suggests that further improvement in accuracy can be expected with a significantly larger training set.

For the purposes of comparing our classifier-based dependency parsing approach to other existing parsers, we also trained SVMdep on the Penn Treebank WSJ training set converted to unlabeled dependencies using a head percolation table (see chapter 2). The unlabeled accuracy of this version of SVMdep on the WSJ test set is close to 90% (using automatic POS tagging). The slight difference between this result and the unlabeled dependency accuracy of SVMpar is that SVMpar uses constituent labels as features, while the WSJ version of SVMdep works with unlabeled dependencies. While SVMdep's accuracy is below the unlabeled dependency accuracy that can be obtained by extracting dependencies from the output of search-based constituent parsers such as the Charniak parser (92.1%) or the Collins parser (91.5%), it is significantly higher than that of Nivre and Scholz's parser (86.4%), and similar to that of Yamada and Matsumoto's quadratic complexity parser (90.3%). Figure 5.2.2 shows the accuracy of SVMdep when it is trained on subsets of the Penn Treebank WSJ training set containing different amounts of data. It is interesting to note that the pace of improvements in accuracy is very steep up to 150,000 words, and decreases significantly when the size of the training data reaches 200,000 words (less than one quarter of the total size of the WSJ training set). From that point, improvement in accuracy is clearly visible, although at a much slower pace. Increasing the training set from 700,000 words to 920,000 words results in about 0.3% improvement in accuracy.

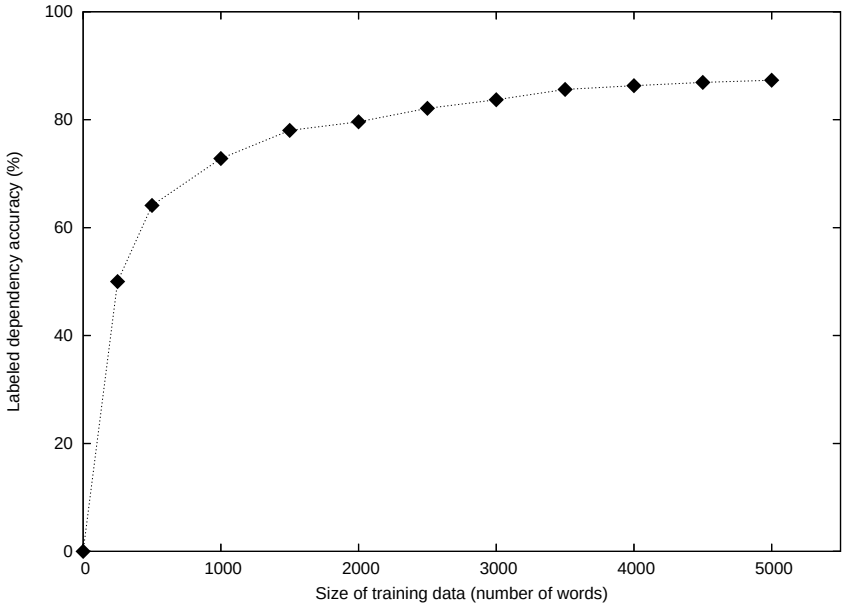


Figure 5.4. Accuracy of SVMdep trained on Eve GRs with different amounts of data.

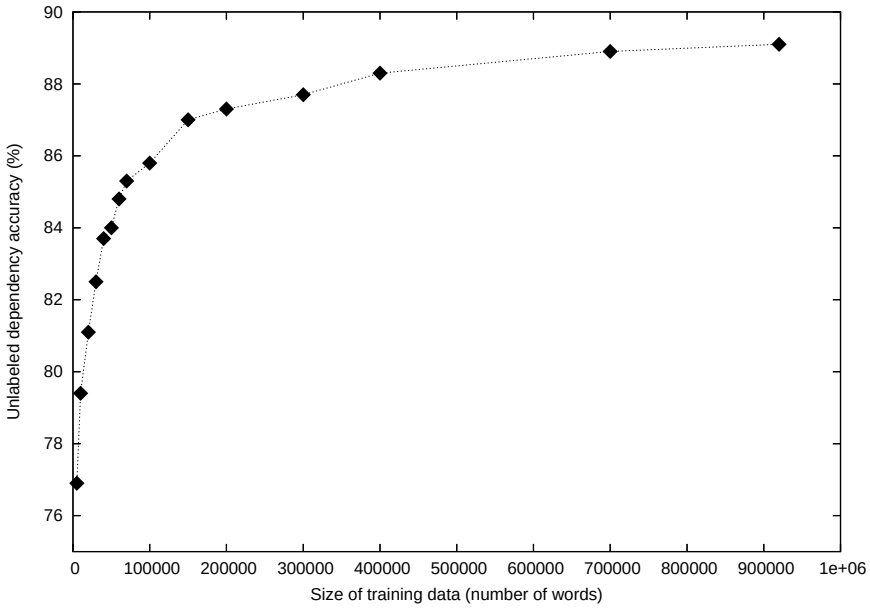


Figure 5.5. Accuracy of SVMdep trained on the Penn Treebank with different amounts of data.

5.3 Conclusion

We have presented two general approaches to data-driven analysis of grammatical relations in CHILDES data. The first approach illustrates how existing natural language processing resources can be used for accurate GR analysis with a small amount of additional domain-specific data. The system is relies on the Charniak (2000) parser (trained on the Penn Treebank WSJ corpus) to determine unlabeled dependencies, and uses a classifier (trained on CHILDES data) to label those dependencies in a multi-pass approach. As the second approach, we developed a classifier-based parsing framework. We showed how it can easily be applied to dependency parsing, allowing us to build simple but accurate dependency parsers that work natively with the CHILDES GR annotation scheme.

Both approaches achieve high levels of precision and recall in the identification of grammatical relations. The results in tables 5.1 and 5.2 point to the fact that the different parsers we have presented have different strengths in identifying grammatical relations in the CHILDES GR scheme, and that we may be able to obtain results that are superior to those of any of the individual parsers if a suitable way for combining their results can be found. This is one of the main issues we address in chapter 6.

Chapter 6

GR Parser Combination

After exploring rule-based and data-driven syntactic analysis approaches in chapters 4 and 5, we now turn our attention to strategies for combining different GR parsers in order to obtain improved accuracy. In the field of machine learning, it is a well-established notion that a combination of classifiers can achieve higher accuracy than that of each individual classifier (Dietterich, 2000). In natural language processing, the combination of the output of different systems has been applied successfully to a number of tasks, including part-of-speech tagging (Brill and Wu, 1998), word sense disambiguation (Florian and Yarowski, 2002; Pedersen, 2000), and constituent parsing (Henderson and Brill, 1999). The work of Henderson and Brill (discussed in more detail in chapter 2) is of particular interest, since it has shown that the output of three constituent parsers can be used to produce more accurate constituent structures. These structures are built by selecting constituents from each structure through classification or a simple voting scheme, with similar results. We start our investigation of GR parser combination by applying a simple voting scheme similar to that Henderson and Brill's, but adapted to the case of dependency structures with several parsers. We then examine progressively more sophisticated dependency combination schemes. Finally, we turn to the issue of combining rule-based and data-driven parsing for analysis of utterances produced by young children, where GR identification features unique challenges not present in the analysis of adult utterances.

To emphasize the intuitive appeal of combining the outputs of the different GR analysis approaches developed in chapters 4 and 5, consider the following situations:

- Suppose we have five GR parsers, and that experiments on a development set has shown that each of the parsers has overall accuracy greater than 85%. When each of the parsers is employed to analyze the sentence *I eat cake with a spoon*, four of the

parsers assign *eat* as the head of *with*, and one of the parsers assigns *cake* as the head of *with*. Although it is not guaranteed that a dependency is correct because it has been proposed by a majority of a group of fairly accurate parsers, it is possible that the accuracy of building structures based on a simple majority vote is higher than the accuracy of each individual parser, provided that the mistakes made by the parsers are different enough.

- Now suppose we have two parsers called A and B. Results on a development set indicates that parser A identifies direct objects (OBJ, in our GR scheme) with precision higher than 98%, but with recall lower than 55%, and it identifies predicate nominals (PRED) with 55% precision and 98% recall. Parser B identifies OBJs and PREDs with precision and recall of about 85%. If we were to choose a single system to identify OBJs and PREDs, parser B would be a better choice overall. However, if parser A identifies a specific GR as an OBJ, and parser B identifies the same GR as a PRED, we would be more likely to arrive at the correct GR by trusting the output of parser A in this case (since it hardly ever makes a mistake when it outputs an OBJ, and it hardly ever misses a PRED). By considering specific characteristics of each of the two parsers, it is likely that we can achieve results that are superior to those of either parser.

As in the description of classifier-based parsing in chapter 5, we first consider dependency parser combination strategies as a general parsing issue, and evaluate our methods on the standard split of the WSJ corpus of the Penn Treebank for parser development and evaluation. This serves as a way to situate the effectiveness of our approach in the context of a large body of recent research. We then apply these techniques to the task of parsing CHILDES data, and present an evaluation using the Eve corpus.

6.1 Dependency voting schemes

6.1.1 Unweighted voting

The first combination scheme we present is also the simplest one we consider, and we refer to it as *unweighted voting*. In this simple scheme, we first work with unlabeled dependencies. Unweighted voting combines n (with $n > 2$) unlabeled dependency structures, each generated by one of n different parsers given the same m-word sentence as input, and creates a single structure for that sentence. Each of the n parsers is given equal importance, regardless of the expected quality of the output they generate. Each of the dependency structures produced by each parser (and also by the combination procedure) is a set of

$(m-1)$ dependencies involving pairs of words in the sentence, plus one dependency between the root word and the artificial LeftWall symbol appended to the beginning of every sentence, for a total of m dependencies.

When n different parsers output a set of dependencies (forming a dependency structure) for a given sentence, combining these dependencies through unweighted voting is straightforward. Voting is done on a word-by-word basis (or dependency-by-dependency basis, since there is one exactly one dependency per word in the sentence, counting the root dependency to the LeftWall), where each of the parsers votes for what the head of each word should be. From the definition of the dependency structures in chapter 3, we know that each of the n parsers specifies exactly one head for each of the m words in the sentence. So, for each word, there are n votes (one by each parser) for what the head of that word should be, and the head is chosen by simple majority (with ties broken randomly). Because voting is done independently on a word-by-word basis, there is no guarantee that the final voted set of dependencies for a sentence will be a well-formed dependency tree. In other words, the final set of dependencies may describe a structure containing cycles, or even disconnected graphs.

An extension of unweighted voting to handle labeled dependencies is trivial: instead of having each parser vote for each head in a sentence, we have parsers vote for label-head pairs. However, in the sections that follow we will present more interesting schemes for labeled dependency combination.

6.1.2 Weighted voting

One simple improvement over unweighted voting is to allow different parsers to have different weights associated with their votes. In other words, if we expect parser A to be more accurate than parser B, we can allow parser A's vote to be more influential than parser B's vote simply by giving it a larger weight. This scheme is called *weighted voting*.

In weighted voting, we combine the outputs of n parsers in a way similar to unweighted voting. The difference is that there is a weight associated with each parser (one weight per parser), and instead of simply adding the number of votes for each head, we add the weight of each vote for each head. Unweighted voting is then one particular instance of weighted voting where every parser (and therefore every vote) has a weight of 1.

For weighted voting to be effective, we need to determine appropriate weights for each parser. We accomplish this by running each parser on a set of annotated sentences, which we will refer to as a weight training set. The weight training set must not contain any material used for training and/or development of any of the parsers used in the combination, since

the perforance of the parsers in the weight training set is intended to provide us with expectations of how each parser performs on unseen data. We then set the weight of each parser to be its unlabeled dependency accuracy on the weight training set.

Like unweighted voting, the extension of weighted voting to labeled dependencies is trivially obtained by having parsers vote for label-head pairs, instead of just heads.

6.1.3 POS-weighted voting and label-weighted voting

Instead of assigning a single weight to each parser (as in weighted voting), we can assign a set of weights per parser, with each weight being associated with the part-of-speech (POS) of the dependent word of the dependency being voted on. The idea behind *POS-weighted voting* is that certain parsers may be better at building dependencies for words with different parts-of-speech. By allowing weights to vary according to parts-of-speech, we attempt to capitalize on these differences. Like in weighted voting, the sets of weights for each parser are determined according to the parser's accuracy on a weight training set. In this case, however, the accuracy for each part-of-speech is measured separately, according to the part-of-speech of the dependent word.

While the trivial extension to labeled dependency voting is also applicable to POS-weighted voting, this combination scheme lends itself easily to a more interesting possibility for combining labeled dependency structures. Instead of setting weights by part-of-speech of dependent words, we can instead set weights per dependency label. In the case where dependency labels represent GRs, the intuitive idea is that certain parsers may be better at finding subjects, while other parsers may be better at finding adjuncts, and we wish to exploit each parser's strengths in the combination scheme. When we set weights according to dependency labels, we refer to this combination scheme as *label-weighted voting*, where each parser is assigned a set of weights, with each set containing exactly one weight associated with each of the possible dependency labels in a given dependency annotation scheme.

6.2 Voting with the WSJ corpus

In this section we evaluate each of the dependency voting schemes described in section 6.1 using the WSJ corpus (with constituent trees converted into dependency trees using a head percolation table as described in chapter 2). The parsers used in these experiments are data-driven parsers built using the standard WSJ training and development sections of the corpus. The voting schemes are tested on the WSJ test section. When necessary, section 00 of the WSJ corpus is used as the weight training set. Because we are using converted

WSJ data with no true dependency labels, the experiments described here deal only with unlabeled dependencies.

6.2.1 Parsers

We use the classifier-based dependency parsing framework described in section 5.2.2 to easily obtain several different dependency parsers. We use two different classification methods, support vector machines (SVM) and memory-based learning (MBL), to create parsers that differ with respect to how parsing decisions are made. In addition, we vary the directionality of parsing to achieve greater parser diversity. In other words, we use parsers that operate from left to right (LR), and parsers that operate from right to left (RL). In section 5.2.2, we described dependency parsing under the assumption of LR operation. RL parsers are obtained by simply initializing the queue in reverse order. In summary, we have four distinct classifier-based parsers, which we will denote by LR-SVM, RL-SVM, LR-MBL and RL-MBL.

In LR-SVM and RL-SVM, the third degree polynomial kernel is used, with the all-against-all strategy for multi-class classification. LR-MBL and RL-MBL use the k-nearest neighbors (k-NN) algorithm, with $k = 5$ and the modified value distance metric (MVDM).

To investigate how voting improves with the addition of different parsers, we use a fifth parser that also uses a classifier to make parser decisions, but that does not follow the framework described in section 5.2.2. Instead, it employs an quadratic-time algorithm similar to that of the SVM-based dependency parser developed by Yamada and Matsumoto (2003). A description of Yamada and Matsumoto's parser can be found in chapter 2. The difference between the parser used here and that of Yamada and Matsumoto's is that instead of moving a sliding window from left to right at each iteration over the input string, we alternate between moving the window from left to right and right to left. Because of its bidirectionality in processing the input string, we refer to this parser as BIDIR-SVM. The features used correspond to Yamada and Matsumoto's feature set with a context length of two words to the right and two words to the left. As with LR-SVM and RL-SVM, the third degree polynomial kernel is used, and multi-class classification is done in all-against-all fashion.

In addition to the five classifier-based parsers, we also use two statistical parsers: the Charniak (2000) parser and Bikel (2002)'s implementation of the Collins (1997) parser. These state-of-the-art statistical parsers produce constituent structures in the style of the Penn Treebank. We convert the trees they produce as output to dependency trees in the same way we converted the WSJ corpus to dependency trees (see chapter 2).

Parser	UAS	RA
LR-SVM	90.9	89.6
RL-SVM	90.1	86.3
LR-MBL	88.1	85.1
RL-MBL	86.3	83.9
BIDIR-SVM	89.6	89.1
Charniak	92.1	97.3
Collins	91.5	96.0

Table 6.1. Evaluation of single parsers on the WSJ test corpus.

Table 6.2.1 shows the test set accuracy of each of the seven parsers used in the WSJ parser combination experiments. The metrics used to evaluate each parser are:

- Unlabeled dependency accuracy, or unlabeled attachment score (UAS): the number of correct dependencies (including the root dependencies to the LeftWall) divided by the total number of dependencies;
- Root accuracy (RA): the number of correct dependency root assignments divided by the total number of root assignments.

In both metrics, punctuation symbols are excluded from calculations.

The right-to-left parsers are significantly less accurate than their left-to-right counterparts. BIDIR-SVM is less accurate than LR-SVM, despite making multiple passes over the input string using a quadratic run-time complexity algorithm. As expected from the classifier-based parser evaluation in chapter 5, each parser that used SVMs outperformed their MBL counterparts. The statistical constituent parsers have higher dependency accuracy than the classifier-based parsers, and much higher root accuracy. The large different in root accuracy is due to the classifier-based parser's greedy nature, compared to the search over a large number (often millions) of trees performed by the statistical parsers.

6.2.2 Parser selection

As in the case of Henderson and Brill's constituent voting, a potential problem with our dependency voting schemes is the assumption of independence among the errors generated by the different parsers. If multiple parsers systematically vote for the same incorrect dependencies, the accuracy of the voted dependency structures may be lower than that of one or more of the voting parsers. Additionally, adding parsers with low accuracy may also hurt the accuracy of the voted dependencies.

To avoid these problems, we do not use every parser available in the voting scheme. Instead, we use a set of parsers that we expect will give us the best results when used in combination. This set is determined by running the voting schemes with every set of three or more parsers on the weight training set, and selecting the set that achieves the highest UAS.

6.2.3 Voting results

We present results for each of the three voting schemes (unweighted voting, weighted voting, and POS-weighted voting) under three conditions:

1. Only the linear-time classifier-based parsers (LR-SVM, RL-SVM, LR-MBL and RL-MBL) are used;
2. All classifier-based parsers (LR-SVM, RL-SVM, LR-MBL, RL-MBL and BIDIR-SVM) are used;
3. The seven parsers described in section 6.2.1 are used (all classifier-based parsers and the two statistical parsers).

Perhaps not surprisingly, the parser selection procedure in each of the three conditions above determined that all parsers should be used, except for RL-MBL, which is the least accurate single parser.

Table 6.2.3 shows the voting results under each of the three conditions we considered. It is interesting to observe that the accuracy of the combination using only linear-time greedy parsers is at the same level as the accuracy of the Collins parser, and only slightly below that of the Charniak parser. Because the voting schemes also run in linear time, the total time required for running each of the parsers and the combination is linear[1]. Root accuracy, considered a weakness of deterministic parsers, is vastly improved with voting. However, although UAS increases from unweighted voting to weighted voting, and from weighted voting to POS-weighted voting, the same is not true of RA. The highest RA is observed with unweighted voting. This is due to the weights used in the weighted schemes are set according to UAS values on the weight training se, so the combination is then intended to optimize UAS, not RA.

When we combine all classifier-based and statistical parsers with POS-weighted voting, we obtain unlabeled dependency accuracy of 93.9% and root accuracy of 97.6%. These

[1]Although the time complexity of the parsers and the combination is linear, on average it is faster to run the Charniak parser or the Collins parser than it is to run three parsers and voting in series.

	Condition 1		Condition 2		Condition 3	
Voting Scheme	UAS	RA	UAS	RA	UAS	RA
Unweighted	91.7	95.6	91.9	96.0	93.5	97.8
Weighted	92.1	94.0	92.2	94.1	93.8	97.3
POS-weighted	92.1	94.7	92.3	94.9	93.9	97.6

Table 6.2. Results for voting experiments using the WSJ corpus, evaluated as unlabeled attachment score (UAS) and root accuracy (RA). In all conditions, UAS improvements from unweighted to weighted combination are statistically significant at the 5% level according to the computationally-intensive randomization test described by Yeh (2000a). Improvements from weighted to POS-weighted combinations are not statistically significant.

results surpass the best previously published results for dependency parsing of the WSJ corpus.

6.3 Voting with CHILDES data

We now evaluate voting schemes on data from the CHILDES database. In this evaluation we use the parsing approaches described in chapters 4 and 5. With the exception of the rule-based parser, which was trained with a subset of approximately 2,000 words of the Eve training corpus, the parsers used in these experiments were build using the Eve training set and Eve development set. An additional 1,102 words were annotated to serve as the weight training set. Voting schemes were evaluated on the Eve adult test set. In contrast with the evaluation with WSJ data, here we consider label dependencies.

6.3.1 Parsers

A total of six parsers are combined through voting to analyze CHILDES data:

- RB: the rule-based parser described in chapter 4. We try three different configurations of the rule-based parser: no insertions or skipping (RB), one word skipping (RB-skp), and one word skipping and insertion of a noun phrase and/or an auxiliary (RB-skp-ins). These configurations are not meant to be used at the same time, but to show how different trade-off points between precision and recall of GRs affect voting results.
- Cdep: the system based on the Charniak parser with classification for dependency labeling, described in section 5.1.
- SVMdep: the SVM-based dependency parser described in section 5.2.2.
- MBLdep: the MBL-based dependency parser described in section 5.2.2.

Parser	LDA	SUBJ		XCOMP		JCT		ROOT	
		Prec	Rec	Prec	Rec	Prec	Rec	Prec	Rec
RB	28.8 (93.6)	1.00	0.37	1.00	0.29	0.91	0.37	0.99	0.29
RB-skp	45.7 (89.9)	0.98	0.64	0.86	0.35	0.88	0.49	0.89	0.45
RB-skp-ins	51.3 (88.0)	0.93	0.67	0.86	0.35	0.80	0.55	0.90	0.55
Cdep	86.9	0.94	0.93	0.58	0.64	0.91	0.92	0.91	0.92
SVMdep	87.3	0.97	0.98	0.93	0.82	0.78	0.88	0.95	0.94
MBLdep	85.1	0.95	0.96	0.82	0.82	0.77	0.82	0.93	0.92
RL-SVMdep	85.1	0.92	0.96	0.93	0.82	0.72	0.88	0.92	0.91
RL-MBLdep	79.2	0.94	0.97	0.88	0.82	0.78	0.79	0.82	0.86

Table 6.3. Accuracy of single parsers on the Eve test set. For the rule-based parsers, in addition to the labeled dependency accuracy (LDA) over all words, we show in parenthesis the LDA over only those words for which the parser found an analysis.

- RL-SVMdep: a right-to-left version of SVMdep. As explained in section 6.2.1, a classifier-based parser based on the framework described in chapter 5 can be made to parse right-to-left simply by having its queue initialized in reverse order.
- RL-MBLdep: a right-to-left version of MBLdep.

The performance of each of the six parsers on the Eve test set can be seen in table 6.3.1. We show each parser's labeled dependency accuracy (LDA), as well as precision and recall of four GRs (SUBJ, XCOMP, JCT and ROOT). The precision and recall results for each of the parsers for every GR in the annotation scheme can be found in Appendix B.

6.3.2 Voting results

We use the parsers in the previous section to evaluate two voting schemes, both based on the more sophisticated of the voting schemes described in section 6.1. The first scheme simply uses a straight-forward application of POS-weighted voting for unlabeled dependencies (by ignoring the labels in the output of the parsers), and uses a classifier to label the resulting unlabeled dependencies (as done in section 5.1.3). The second scheme is label-weighted voting, just as described in section 6.1.3.

Parser selection was done in a similar way as in the experiments with WSJ data. The single difference in the selection procedure used here is that instead of running every possible set of at least three parsers, we restricted the sets to those that contained at most one of RB, RB-skp and RB-skp-ins. As in the WSJ experiments, the selection procedure eliminated only one parser in both voting schemes, and that parser was once again the MBL-based right-to-left parser. The set of parsers with higher accuracy in the parser selection phase included RB as the rule-based parser. Although the rule-based parsers have much lower

		SUBJ		XCOMP		JCT		ROOT	
Voting scheme	LDA	Prec	Rec	Prec	Rec	Prec	Rec	Prec	Rec
POS-weighted	89.5	0.96	0.94	0.65	0.76	0.83	0.81	0.95	0.94
Label-weighted	92.3	0.98	0.98	0.90	0.88	0.87	0.90	0.94	0.98
POS-noRB	88.0	0.96	0.98	0.91	0.59	0.61	0.86	0.94	0.93
Label-noRB	92.0	0.98	0.98	0.88	0.88	0.85	0.87	0.94	0.98
Label-RBskp	92.0	0.97	0.98	0.88	0.88	0.85	0.90	0.94	0.98
Label-RBskp-ins	92.0	0.98	0.98	0.88	0.88	0.85	0.90	0.94	0.98

Table 6.4. Results of POS-weighted voting and label-weighted voting on the Eve test set. Labeled dependency accuracy over all words and precision/recall values for four GRs are shown.

labeled dependency accuracy when we include all words in the test set in the calculations, they actually have a steady beneficial effect when used in the voting schemes, since the parsers only cast votes for dependencies involving words for which they find an analysis. When accuracy calculation of the rule-based parsers is restricted to the set of words for which they find an analysis, labeled dependency accuracy is very high.

Table 6.4 shows the voting results for POS-weighted voting and label-weighted voting with different sets of parsers. The precision and recall results on the entire set of GRs can be found in Appendix B. The first two rows of table 6.4 shows a summary of the results of applying POS-weighted voting and label-weighted voting with the set of parsers with the best performance on development data. This set includes the rule-based parser with no robustness, setting it to the highest precision. The set also includes every data-driven parser in our pool of parsers, except for RL-MBLdep. As expected, defining weights for each dependency type (GR) outperforms setting weights according to part-of-speech tags. Both strategies clearly outperform any of the single parsers in isolation, but the results obtained with label-weighted voting are visibly stronger in precision and recall of GRs (and, of course, overall dependency accuracy). Also of interest, the unlabeled dependency accuracy (not shown in the table) obtained with label-weighted voting is 94.7%. The only GR with either precision or recall below 0.80 is COMP, with 0.75 precision and 0.67 recall, for an f-score (harmonic mean of precision and recall) of 0.71. The upperbound on LDA for any of our combination schemes using these parsers (the percentage of words for which at least one of the parsers found the right labeled dependency) is 96.5%, while our best result using voting is 92.3%. This indicates that there may be other ways to set weights for different parsers in different contexts that could lead to further improvements in GR identification.

The remaining rows of table 6.4 show results for voting under different conditions with respect to rule-based parsing. Rows 3 and 4 show the results obtained with each of the voting schemes if rule-based parsing is not used (POS-noRB and label-noRB). Using the

high precision GRs produced by the rule-based parser has a strong positive impact on POS-weighted voting, resulting in a 12.5% relative reduction in error (1.5% absolute reduction). With labeled-weighted voting the effect of using a high-precision low-recall parser is less pronounced, but significant improvements are observed in a few GRs, such as adjuncts (JCT) and vocatives (VOC). Improvements obtained with versions of the rule-based parser that allowed insertions or skipping were negligible, but the addition of these parsers to the optimal combination did not hurt accuracy, as long as only one rule-based parser is used in the combination.

When our most accurate voting scheme is used (label weighted voting including the RB rule-based parser and several data-driven parsers), the most frequent cause of error is the attachment and/or labeling of adjuncts. Such errors account for nearly 24% of all errors in the parser combination, and are subdivided into three main groups: (1) error in attaching a prepositional phrase or adverb to the correct head; (2) confusion in labeling between an adverb and verb particles ("look *up*" as in towards the sky, vs. "*look up*" as in search); and (3) confusion between an adverb and a communicator (many communicators are words that are usually adverbs, such as *there* in "*there*, have more soup," which is clearly different than "have more soup *there*"). The next most frequent source of error is wh-words, which account for roughly 10% of the errors in the combined GR output. Because they can appear in the sentence far from their heads, often outside a verb phrase where they hold a semantic role, wh-words can be challenging. A common error was to attach a wh-word to the wrong verb (for example, in "what did you say you bought?" the head of *what* is *bought*, but *say* is found to be the head).

6.4 Parser combination as reparsing

Although label-weighted dependency voting using the GR parsers we presented in chapters 4 and 5 produces very high precision and recall levels of GR identification in CHILDES data, the voting schemes we presented have one significant shortcoming: because voting within a sentence is done on a word-by-word basis, there is no guarantee that the resulting set of dependencies is a well-formed dependency tree. In fact, the graphs generated by the voting procedures may contain cycles, and may not even be connected.

We address this problem with a general approach of *GR parser combination as reparsing*. In other words, once a sentence has been parsed by n parsers, and we have obtained n dependency structures corresponding to the input sentence, we use these n dependency structures to guide us in reparsing the sentence. With the knowledge of what structures were created by known parsers, we can attempt to parse the sentence once more and produce a more accurate dependency structure. We examine two instances of this approach. First

we treat the problem of dependency parsing as a search for a maximum spanning tree over a directed graph where the words are nodes, and dependencies are represented by weighted edges in the graph. Framing dependency parsing as a maximum spanning tree problem was recently proposed by McDonald et al. (2005), and here we apply that idea to dependency parser combination. The second instance of combination as parsing is done with a variant of the widely used Cocke-Younger-Kasami (or simply CYK) algorithm (Cocke and Schwartz, 1970; Kasami, 1965; Younger, 1967).

6.4.1 GR combination with maximum spanning trees

In the maximum spanning tree combination scheme, once the input sentence has been parsed by the n parsers to be combined, we start by creating a graph where each word in the original input sentence is a node. We then create directed edges between nodes corresponding to words for which dependencies are obtained from any of the parsers. Each edge in the graph goes from the node corresponding to a head word, to the node corresponding to a dependent of that head word. Each edge is weighted according to what parser is responsible for generating the edge, and possibly the label of the dependency that corresponds to the edge. These weights are the same weights used in weighted voting, POS-weighted voting, or label-weighted voting. In cases where more than one parser indicates that the same edge should be created, the weights are added, just as in the voting scheme. As long as any of the parsers creates a valid dependency tree for the sentence, the directed weighted graph created this way will be connected and contain no cycles. If none of the parsers creates a complete dependency tree, the combination may still be a well-formed dependency tree, although in that case there is no guarantee that it will be.

Once the graph is created, we can simply find its maximum spanning tree (MST), using, for example, the Chu-Liu/Edmonds directed MST algorithm (Chu and Liu, 1965; Edmonds, 1967), as described by McDonald et al. (2005). The maximum spanning tree maximizes the votes for dependencies given the constraint that the resulting structure must be a tree. In contrast with our classifier-based dependency parser (chapter 5) and other well known dependency parsing approaches (Eisner, 1996; Nivre and Scholz, 2004; Yamada and Matsumoto, 2003), there is no guarantee against crossing branches in a tree created with the MST algorithm. In other words, the resulting dependency structure may be non-projective. While projectivity is usually assumed in dependency parsing of English, this is usually the case only for efficiency reasons. When parsing languages such as Czech or Turkish, where dependency arcs are expected to cross more often than in English, being able to produce non-projective trees is important (Nivre and Nilson, 2005).

When the MST combination approach is applied to the WSJ data using the same sets of

weights as in the POS-weighted voting experiment, we obtain 93.8% unlabeled dependency accuracy, and 97.6% root accuracy. The dependencies created are essentially as accurate as those created with POS-weighted voting, but each dependency structure is guaranteed to be a tree. When applied to the Eve test set, the MST combination using the same weights as label-weighted voting has similar accuracy to voting, but the recall for COMP is significantly higher, putting the f-score for COMP at 0.80 (0.73 precision and 0.89 recall). The improvement in the identification of COMP highlights the role of the rule-based system. When RB is excluded from the combination, we obtain only 0.67 precision and 0.67 recall for COMP. The reason for the improvement in the precision and recall of COMP is that parsers often mislabel COMPs as ROOTs, and vice-versa. Because the ROOT relation is much more frequent than the COMP relation, these errors greatly affect the accuracy of COMPs, but their effect on the accuracy of ROOT is much less pronounced. This type of error is common because the contexts where COMPs and ROOTs appear are often similar. The dependent in either a COMP or a ROOT relation is typically a finite verb, and the difference is that in a ROOT relation that verb is the main verb of a sentence, and in COMP the verb belongs to a subordinate clause. The MST algorithm finds the tree with most votes, and because a tree has only one ROOT, the COMP and ROOT relations are less confusable. Among the results for all GRs in the MST combination scheme, only two GRs had f-scores below 0.85: COMP at 0.80 and PTL at 0.83. Complete results can be found in Appendix B.

6.4.2 GR combination with the CYK algorithm

The second instance of the reparsing combination framework we present is the combination of GR analysis using a variant of the CYK algorithm, which uses dynamic programming to find the parses of an input string, given a context-free grammar. The CYK algorithm for constituent trees requires a grammar that must be in Chomsky normal form. In a nutshell, the algorithm considers every substring, from shorter to longer, and determines which substrings can be combined according to the grammar to form a constituent. The probabilistic extension of the CYK algorithm is widely used in statistical parsing. In this GR combination scheme, we use a weighted variant of the CYK algorithm that is very similar to the probabilistic CYK algorithm. The main differences is that we use dependencies, and when structures are combined into larger structures, we add their weights instead of multiplying their probabilities.

The first step is the same as in the MST combination scheme: we build a graph that represents each word in the input string as a node, and each dependency as an edge. Edges are weighted according to dependency type and which parser produced the dependency represented by the edge. If multiple parsers are responsible for the creation of the same

edge, the weights for each parser are added. Once we have built this graph (described in more detail in section 6.4.1), we parse the input sentence according to the weighted CYK algorithm. However, instead of using a grammar, we restrict the creation of new items in the CYK chart to pairs of words whose corresponding nodes are connected directly. We assign a weight to each new item in the chart. The weight is determined by adding the weights of the items used in the creation of the new item, and the weight of the edge in the graph that allowed the new item to be created. The only difference between the weighted and probabilistic versions of the CYK algorithm is that in the weighted version we use arbitrary scores, not probabilities. The scores we use here are weighted votes, and we wish to find the parse for the sentence with the largest number of votes. In the probabilistic case, we look for the most probable parse, and the probability of a parse is found by multiplying the probabilities of each production. To find the parse with the largest number of votes, we add the votes for each parse, and that is why our search for the most voted parse involves the addition of scores.

The difference between results obtained with the MST combination and the CYK combination is that, although both methods are guaranteed to output a tree, the CYK combination also guarantees that the resulting tree will be projective (will have no crossing branches). Because the parsers used in our experiments do not produce structures with crossing branches, and the level of agreement among parsers is high (every parser is quite accurate on its own, although not as accurate as what we have achieved with combination), it is not surprising that the results obtained with CYK combination are essentially the same as what we obtain with MST combination. The sets of dependencies obtained using the two combination schemes were nearly identical in our experiments, and only negligible differences were observed in the overall accuracy or precision and recall of any GRs.

6.5 Identifying GRs in utterances by young children

We now turn to a different type of combination of rule-based and data-driven GR parsing, with the goal of addressing the challenge of analyzing utterances spoken by young children. The parsing approaches we have presented so far expect input utterances that roughly conform to standard spoken English. Children at a young age, however, may produce a mix of sentences that vary from perfectly well formed (according to what is expected from an adult speaker), to seemingly unintelligible word strings. Consider the following utterances from the Eve corpus:

1. I need tapioca in the bowl.
2. That's a hat.

3. In a minute.

4. ? I drinking milk.

5. ? I want Fraser hat.

6. * Warm puppy happiness a blanket.

7. * There briefcase

Here we see three general types of utterances[2]. Utterances 1, 2 and 3 conform perfectly to the adult standard of how grammatical relations appear in an utterance, and the same techniques applied to identification of grammatical relations in adult language may be used for these utterances. We will refer to this type of utterance as type I. Although utterances 4 and 5 deviate slightly from the adult standard, they contain clear grammatical relations that should be identified. Utterance 4 is missing the auxiliary *am*, and utterance 5 is missing the possessive marker *'s* after Fraser, but to a English speaker there is no doubt as to what these utterances mean and grammatical relations are present. However, when lexical material is missing in an utterance, we may not be able to identify grammatical relations reliably using the parsing approaches described so far. Utterances 4 and 5 are examples of type II utterances. Finally, utterances 6 and 7 contain no grammatical relations that can be reliably identified even by a human annotator. In such cases, we should annotate the words in these utterances as having no GRs. When we cannot determine the syntactic structure of certain utterances, it is better to indicate that no structure is found than to try to imagine what the intended structure should be. In other words, we are interested in GRs that actually appear in the transcripts, but not on those that might have been intended. We refer to this last kind of utterance as type III.

To determine grammatical relations in a transcript where utterances of types I, II and III may appear, out first step is to distinguish between the three types. This is an issue we started to address towards the end of chapter 4, when we discussed the rule-based GR system. The data-driven approaches discussed in chapter 5 are very robust. They are capable of assigning a set of GRs to almost any input utterance, no matter how little sense it seems to make. This type of smooth degradation is seen as an advantage when we parse corpora where every sentence is expected to contain meaningful syntactic structure, such as a corpus of adult spoken language, or written articles. On the other hand, in cases where the input might be a type III utterance, the output will may be a meaningless set of GRs. The rule-based system described in chapter 4 works with a grammar that defines a model of language that is much narrower than those of data-driven system. As a result, the recall

[2]We define these utterance types for the sole purpose of creating an effective system for GR identification. These type classifications are not intended for other purposes, or to reflect any assumption on child language acquisition.

of GRs obtained with the rule-based system was very low in the Eve GR adult test corpus. When parsing utterances from a child, this inability to parse certain sentences is actually an advantage, since it gives us a way to distinguish between utterances of types I, II and III. This is accomplished with a multi-pass process similar to the one described in chapter 4, but with a slightly different goal and different parameters. The general idea is to classify a sentence as type I if it is covered by the system under strict conditions, or type II if it is covered under slightly less strict conditions, or type III if the sentence is not covered.

To determine these conditions, we examine the Eve GR child development set. If a sentence in the set is annotated so that its words have the label NO-GR, indicating that no GR was annotated for these words, then the sentence is of type III. Otherwise, the sentence is of type I or II. Sentences of types I and II in the development set of child utterances are short, simple sentences. Manual inspection of these sentences reveals that the grammatical deviations found most often in sentences of type II are: the absence of an auxiliary *be* where it is required, the absence of a possessive *'s*, and the absence of certain prepositions. From the experiments using the rule-based GR system for adult utterances, we know that the insertion of auxiliaries may be a manageable task. The insertion of possessive *'s* is slightly more problematic, since a large number of type III utterances are two-noun strings ("Eve pencil," "Mommy store") and these would automatically become possessive constructions. This is undesirable because in an utterance such as "Eve pencil," it is not clear that what is meant really is *Eve's pencil*, and not *Eve wants a pencil*, or *give Eve a pencil*, for example. In a sentence such as "This is Eve pencil" it is much clearer that the intended message is *This is Eve's pencil*. Many of these two word utterances are of type III. The insertion of prepositions is a much more difficult problem. Prepositions can be inserted in different places in different types of sentences, potentially increasing ambiguity significantly in the process.

Based on the discussion above, we designed a procedure to classify utterances into one of the three types, and then proceed to identify GRs (or not, in the case of type III utterances) accordingly. To analyze a transcript that might contain the three different types of utterances, we first parse the transcript using the rule-based system, with robustness parameters set not to allow any insertions or skipping. The utterances covered by the system are classified as being of type I, regardless of the accuracy of the analyses they receive. Two-word utterances that were not covered by the system are classified as being of type III. If an utterance of only two words is not covered by the grammar, we make the reasonable assumption that there are no GRs that can be identified reliably. The remaining utterances are processed by the rule-based parser again, this time with robustness parameters set to allow the insertion of at most one auxiliary *be* or one possessive marker

's[3]. The complication mentioned above regarding the insertion of possessives is resolved by our earlier classification of two-word utterances as type III. The utterances covered by the system under these conditions are classified as being of type II. The remaining sentences are classified as being of type III. The practical result of this classification procedure is that all the words in type III utterances are marked with the NO-GR label. We ran this procedure on the Eve development of child utterances, and we found that the NO-GR label can be identified with surprisingly high accuracy of about 0.89 precision and 0.91 recall. Although this was an encouraging result, the procedure was created based on inspection of the development set, so the actual result on unseen test data could be much lower. Fortunatly, this was not the case. When tested on the Eve child test set, we obtain 0.88 precision and 0.89 recall of NO-GR identification.

Once type III utterances have been annotated with NO-GR labels, we still need to obtain GRs for utterances of types I and II. This is done using the same techniques previously applied to the Eve adult test set. The main difference is that if any insertions were made in the type classification procedure for an utterance, the lexical items corresponding to the insertions are actually inserted in the utterance before we parse it to obtain GRs. Running the MST combination scheme (using the same parsers as with the experiments on adult utterances) on utterances of type I and type II (with additional lexical items inserted automatically after the utterance is classified as being of type II, but before it is given to parsers with the goal of actually extracting GRs), and keeping the NO-GR labels on all words of utterances of type III, we obtain 88.0% labeled dependency accuracy on the Eve child test set. For comparison purposes, if we simply run the same combination scheme with the same parsers, but without first running the type classification procedure (in other words, if we treat the Eve child test set in the same way we do the Eve adult test set), we obtain 62.9% labeled accuracy. Most of the errors in this case come from assigning a grammatical relation to words that should be labeled as NO-GR. The complete set of results of the entire procedure for parsing the Eve child test set is listed in Appendix B.

To complete a strategy that can be applied fully automatic to child language transcripts, we still need a way to decide when to apply the rule-based first pass to classify utterance types. The experiments in chapter 4 indicate that always applying this approach is counter-productive, since one a child's language reaches a certain level the rule-based system would start to suffer from lack of coverage of type I utterances. At the same time, the type classification would not be needed, since at this point the child would be producing utterances that are much closer to the adult standard. Given a transcript of child utterances, we can decide on whether or not to use the rule-based type classification pass by computing the

[3]Although we also found that prepositions are often missing in type II utterances, we do not attempt to insert prepositions. Tests on the development set revealed that the system is much more likely to use inserted prepositions to arrive at incorrect analyses than to reconstruct legitimate type II sentences.

Mean Length of Utterance, or MLU (Brown, 1973), for the transcript. The MLU is a measure that can be easily computed automatically (MacWhinney, 2000) and roughly reflects the level of language development in children under the age of four. Manual inspection of CHILDES transcripts show that an MLU of 2.5 is a good cut-off point, meaning that if the MLU is higher than that, we do not assign any NO-GR labels. Interestingly, this agrees with the language stages proposed by Brown (1973). According to Brown, an MLU range of 2.0 to 2.5 corresponds to stage II in language development, and children in that stage typically produce utterances where an auxiliary *be* may be missing. An MLU range of 2.5 to 3.0 corresponds to Brown's stage III, where a child starts producing *'s* possessives, among other things. Although we arrived at the parameter settings for insertions and the 2.5 MLU cut-off empirically, it agrees to a remarkable extent with Brown's famous study on language stages.

A remaining concern is whether the efficacy of this procedure for parsing child language transcripts is specific to our Eve test set. The question of how well the same procedure would work on language from a different child clearly needs to be addressed. To that end, we annotated an additional 700 words from the Naomi section of the Sachs (1983) corpus. The GR results obtained on this Naomi test set were very similar to our results on the Eve test set. Using the same MST combination system as for adult GR identification, we obtain 67.4% accuracy, slightly above the accuracy on the Eve test set, due to fewer utterances in the Naomi set being of type III. Using the rule-based type classification pass, and then the entire MST combination, we obtain 86.9% labeled accuracy.

6.6 Conclusion

We have presented two general frameworks for combining different parsers with the goal of obtaining accurate GR analyses for CHILDES transcripts. The first framework is that of parser ensembles. We described a few different kinds of ensembles, from simple voting schemes to the reparsing approach, and showed that these types of parser combination work very well with the adult utterances in CHILDES transcripts, as well as with the WSJ Penn Treebank. We showed that setting different voting weights based on a held-out set result in improved accuracy by taking advantage of the specific strengths and weaknesses of each of the parsers in the ensemble. We also showed that going beyond word-by-word voting with combination by reparsing not only ensures that the result of the parser ensemble is a well formed dependency structure, but it can also improve the precision and recall of certain GRs.

The second combination framework is aimed specifically at the challenge of parsing utterances of young children where not all sentences have recognizable syntactic structure.

We argue for not attempting to find GRs if the syntactic structure of an utterance cannot be determined. This is accomplished by filtering out utterances that fall outside the coverage of a rule-based parser, and assigning the label of NO-GR to the words in these utterances. We show that this filtering step is quite accurate, and it allows for much improved accuracy in GR recognition in transcripts of utterances produced by young children. We also present a way to determine when this type of rule-based filtering should be applied, based on the transcript's Mean Length of Utterance. An interesting aspect of this work is that it agrees with the language stages proposed by Brown (1973).

Chapter 7

Automated Measurement of Syntactic Development

In this chapter we present a practical end-to-end application of the methods for syntactic annotation and automatic analysis described so far. The task we explore is the automated measurement of syntactic development in child language, which has both clinical and theoretical value in the field of child language acquisition. Specifically, we present a fully automated way of computing the *Index of Productive Syntax*, or IPSyn (Scarborough, 1990), a popular measure for syntactic development that has traditionally required significant manual effort by trained researchers or clinicians.

In addition to its inherent value to the child language community, automatic computation of IPSyn scores serves as a task-based evaluation for our GR analysis approach. Although researchers in natural language parsing have become accustomed to evaluating systems in terms of precision and recall of certain pieces of information (such as constituent bracketing, or grammatical relations, as we have done in previous chapters), a syntactic analysis system often operates as a piece in a larger system designed to perform a task that goes beyond determining parse trees or grammatical relations. Because task-based evaluations focusing on the effects of the performance of specific NLP components are relatively rare, the relationship between the standard precision/recall measures and the performance of larger systems that include these NLP components is still somewhat unclear. Through a task-based evaluation where we examine the results of an end-to-end system that computes IPSyn scores, we can determine the impact of the accuracy of our GR system in a practical setting. Accurate computation of IPSyn scores validates the usefulness of our annotation

scheme and our approach to automatic GR analysis in its current levels of precision and recall of grammatical relations.

7.1 The Index of Productive Syntax (IPSyn)

The Index of Productive Syntax (Scarborough, 1990) is a measure of development of child language that provides a numerical score for grammatical complexity. IPSyn was designed for investigating individual and group differences in child language acquisition, and has been used in numerous studies. It addresses weaknesses in the widely popular Mean Length of Utterance measure, or MLU, with respect to the assessment of development of syntax in children. Because it addresses syntactic structures directly, it has gained popularity in the study of grammatical aspects of child language learning in both research and clinical settings.

After about age three (Klee and Fitzgerald, 1985), differences in MLU become less significant and fail to properly distinguish between children at different levels of syntactic ability. For these purposes, and because of its higher content validity, IPSyn scores often tell us more than MLU scores. However, the MLU holds the advantage of being far easier to compute. Relatively accurate automated methods for computing the MLU for child language transcripts have been available for several years (MacWhinney, 2000).

Calculation of IPSyn scores requires a corpus of 100 transcribed child utterances, and the identification of 56 specific language structures in each utterance. These structures are counted and used to compute numeric scores for the corpus in four categories (noun phrases, verb phrases, questions and negations, and sentence structures), according to a fixed score sheet. Each structure in the four categories receives a score of zero (if the structure was not found in the corpus), one (if it was found once in the corpus), or two (if it was found two or more times). The scores in each category are added, and the four category scores are added into a final IPSyn score, ranging from zero to 112[1].

Some of the language structures required in the computation of IPSyn scores (such as the presence of auxiliaries or modals) can be recognized with the use of existing child language analysis tools, such as the morphological analyzer MOR (MacWhinney, 2000) and the part-of-speech tagger POST (Parisse and Le Normand, 2000). However, more complex structures in IPSyn require syntactic analysis that goes beyond what POS taggers can provide. Examples of such structures include the presence of an inverted copula or auxiliary

[1]See Appendix C for a complete listing of targeted structures and the IPSyn score sheet used for calculation of scores.

in a wh-question, conjoined clauses, bitransitive predicates, and fronted or center-embedded subordinate clauses.

7.2 Automating IPSyn

Calculating IPSyn scores manually is a laborious process that involves identifying 56 syntactic structures (or their absence) in a transcript of 100 child utterances. Currently, researchers work with a partially automated process by using transcripts in electronic format and spreadsheets. However, the actual identification of syntactic structures, which accounts for most of the time spent on calculating IPSyn scores, still has to be done manually.

By using part-of-speech and morphological analysis tools, it is possible to narrow down the number of sentences where certain structures may be found. The search for such sentences involves patterns of words and parts-of-speech (POS). Some structures, such as the presence of determiner-noun or determiner-adjective-noun sequences, can be easily identified through the use of simple patterns. Other structures, such as front or center-embedded clauses, pose a greater challenge. Not only are patterns for such structures difficult to craft, they are also usually inaccurate. Patterns that are too general result in too many sentences to be manually examined, but more restrictive patterns may miss sentences where the structures are present, making their identification highly unlikely. Without more syntactic analysis, automatic searching for structures in IPSyn is limited, and computation of IPSyn scores still requires a great deal of manual inspection.

Long et al. (2004) have developed a software package, Computerized Profiling (CP), for child language study, which includes a (mostly) automated[2] computation of IPSyn. CP is an extensively developed example of what can be achieved using only POS and morphological analysis. It does well on identifying items in IPSyn categories that do not require deeper syntactic analysis. However, the accuracy of overall scores is not high enough to be considered reliable in practical usage, in particular for older children, whose utterances are longer and more sophisticated syntactically. In practice, researchers usually employ CP as a first pass, and manually correct the automatic output. Section 7.3 presents an evaluation of the CP version of IPSyn.

Syntactic analysis of transcripts as described in chapters 4, 5 and 6 allows us to go a step further, fully automating IPSyn computations and obtaining a level of reliability comparable to that of human scoring. The ability to search for both grammatical relations and parts-of-

[2]Although CP requires that a few decisions be made manually, such as the disambiguation of the lexical item *'s* as copula vs. possessive marker, and the definition of sentence breaks for long utterances, the computation of IPSyn scores is automated to a large extent.

speech makes searching both easier and more reliable. As an example, consider the following sentences (keeping in mind that there are no explicit commas in spoken language):

(a) Then [,] he said he ate.

(b) Before [,] he said he ate.

(c) Before he ate [,] he ran.

Sentences (a) and (b) are similar, but (c) is different. If we were looking for a fronted subordinate clause, only (c) would be a match. However, each one of the sentences has an identical part-of-speech sequence. If this were an isolated situation, we might attempt to fix it by having tags that explicitly mark verbs that take clausal complements, or by adding lexical constraints to a search over part-of-speech patterns. However, even by modifying this simple example slightly, we find more problems:

(d) Before [,] he told the man he was cold.

(e) Before he told the story [,] he was cold.

Once again, sentences (d) and (e) have identical part-of-speech sequences, but only sentence (e) features a fronted subordinate clause. These limited toy examples only scratch the surface of the difficulties in identifying syntactic structures without syntactic analysis beyond part-of-speech and morphological tagging. In these sentences, searching with GRs is easy: we simply find a GR of clausal type (e.g. CJCT, COMP, CMOD, etc) where the dependent is to the left of its head.

For illustration purposes of how searching for structures in IPSyn is done with GRs, let us look at how to find other IPSyn structures[3]:

- Wh-embedded clauses: search for wh-words whose head, or transitive head (its head's head, or head's head's head...) is a dependent in GR of types [XC]SUBJ, [XC]PRED, [XC]JCT, [XC]MOD, COMP or XCOMP;
- Relative clauses: search for a CMOD where the dependent is to the right of the head;
- Bitransitive predicate: search for a word that is a head of both OBJ and OBJ2 relations.

Although there is still room for under- and overgeneralization with search patterns involving GRs, finding appropriate ways to search is often made trivial, or at least much more simple and reliable than searching without GRs. An evaluation of our automated version of IPSyn, which searches for IPSyn structures using POS, morphology and GR information, and a

[3]More detailed descriptions and examples of each structure are found in Appendix C, and are omitted here since the short descriptions are fairly self-explanatory.

GR	Precision	Recall	F-score
SUBJ	0.98	0.98	0.98
OBJ	0.94	0.94	0.94
COORD	0.94	0.91	0.92
JCT	0.87	0.90	0.88
MOD	0.97	0.91	0.94
PRED	0.86	0.89	0.87
ROOT	0.97	0.97	0.97
COMP	0.73	0.89	0.80
XCOMP	0.88	0.88	0.88

Table 7.1. Precision, recall and F-score (harmonic mean) of the MST combination system on selected GRs.

comparison to the CP implementation, which uses only POS and morphology information, is presented in the next section.

7.3 Evaluation

Because IPSyn uses only intelligible utterances, and the selection of such sentences is typically done at transcription time, in general it is not necessary to use rule-based filter for child utterances described in chapter 6 for the purposes of computing IPSyn automatically. In the experiments that follow, our GR analysis was done with the reparsing combination using maximum spanning trees (MST), as described in chapter 6 (the parsers used in the combination are the same as in the experiments in chapter 6: RB, Cdep, SVMdep, MBLdep, and RL-SVMdep). As a reminder, table 7.3 shows the precision and recall performance of that system on some important GRs.

We evaluate our implementation of IPSyn in two ways. The first is *Point Difference*, which is calculated by taking the (unsigned) difference between scores obtained manually and automatically. The point difference is of great practical value, since it shows exactly how close automatically produced scores are to manually produced scores. The second is *Point-to-Point Accuracy*, which reflects the overall reliability over each individual scoring decision in the computation of IPSyn scores. It is calculated by counting how many decisions (identification of presence/absence of language structures in the transcript being scored) were made correctly, and dividing that number by the total number of decisions. The point-to-point measure is commonly used for assessing the inter-rater reliability of metrics such as the IPSyn. In our case, it allows us to establish the reliability of automatically computed scores against human scoring.

System	Avg. Point Difference to HUMAN	Point-to-Point Reliability
GR (Total)	2.5	93.3%
CP (Total)	8.3	85.4%
GR (Set A)	2.9	92.9%
CP (Set A)	6.2	86.2%
GR (Set B)	2.1	93.6%
CP (Set B)	10.2	87.2%

Table 7.2. Summary of IPSyn score evaluation (point difference and point-to-point reliability). GR is our implementation of IPSyn based on grammatical relations, CP is Long et al.'s (2004) implementation of IPSyn, and HUMAN is manual scoring.

7.3.1 Test Data

We obtained two sets of transcripts with corresponding IPSyn scoring (total scores, and each individual decision) from two different child language research groups. The first set (A) contains 20 transcripts of children of ages ranging between two and three. The second set (B) contains 25 transcripts of children of ages ranging between eight and nine.

Each transcript in set A was scored fully manually. Researchers looked for each language structure in the IPSyn scoring guide, and recorded its presence in a spreadsheet. In set B, scoring was done in a two-stage process. In the first stage, each transcript was scored automatically by CP. In the second stage, researchers checked each automatic decision made by CP, and corrected any errors manually.

Two transcripts in each set were held out for development and debugging. The final test sets contained: (A) 18 transcripts with a total of 11,704 words and a mean length of utterance of 2.9, and (B) 23 transcripts with a total of 40,819 words and a mean length of utterance of 7.0.

7.3.2 Results

Scores computed automatically from transcripts parsed with the MST combination system were very close to the scores computed manually. Table 7.3.2 shows a summary of the results, according to our two evaluation metrics. Our system is labeled as GR, and manually computed scores are labeled as HUMAN. For comparison purposes, we also show the results of running Long et al.'s automated version of IPSyn, labeled as CP, on the same transcripts.

Figure 7.1. Histogram of point differences between GR and HUMAN scores (black), and CP and HUMAN (white).

Point Difference

The average (absolute) point difference between automatically computed scores (GR) and manually computed scores (HUMAN) was 2.5 (the range of HUMAN scores on the data was 21-91). There was no clear trend on whether the difference was positive or negative. In some cases, the automated scores were higher, in other cases lower. The minimum difference was zero, and the maximum difference was 12. Only two scores differed by 10 or more, and 17 scores differed by two or less. The average point difference between HUMAN and the scores obtained with Long et al.'s CP was 8.3. The minimum was zero and the maximum was 21. Sixteen scores differed by 10 or more, and six scores differed by 2 or less. Figure 7.1 shows the point differences between GR and HUMAN, and CP and HUMAN.

It is interesting to note that the average point differences between GR and HUMAN were similar on sets A and B (2.9 and 2.1, respectively). Despite the difference in age ranges, the two averages were less than one point apart. On the other hand, the average difference between CP and HUMAN was 6.2 on set A, and 10.2 on set B. The larger difference reflects CP's difficulty in scoring transcripts of older children, whose sentences are more syntactically complex, using only POS analysis.

Point-to-Point Accuracy

In the original IPSyn reliability study (Scarborough, 1990), point-to-point measurements using 75 transcripts showed the mean inter-rater agreement for IPSyn among human scorers at 94%, with a minimum agreement of 90% of all decisions within a transcript. The lowest agreement between HUMAN and GR scoring for decisions within a transcript was 89.1%, with a mean of 93.3% over the 41 transcripts used in our evaluation. Although comparisons of agreement figures obtained with different sets of transcripts are somewhat coarse-grained, given the variations within children, human scorers and transcript quality, our results are very satisfactory. For direct comparison purposes using the same data, the mean point-to-point accuracy of CP was 85.4% (a relative increase of about 100% in error).

In their separate evaluation of CP, using 30 samples of typically developing children, Long and Channell (2001) found a 90.7% point-to-point accuracy between fully automatic and manually corrected IPSyn scores[4]. However, Long and Channell compared only CP

[4]Long and Channell's evaluation also included samples from children with language disorders. Their 30 samples of typically developing children (with a mean age of 5) are more directly comparable to the data used in our evaluation.

output with manually corrected CP output, while our set A was manually scored from scratch. Furthermore, our set B contained only transcripts from significantly older children (as in our evaluation, Long and Channell observed decreased accuracy of CP's IPSyn with more complex language usage). These differences, and the expected variation from using different transcripts from different sources, account for the difference in our results and Long and Channell's.

In addition to the point-to-point measure used by Scarborough (1990) and Long and Channell (2001), we also report the kappa statistic (Cohen, 1960) for point-to-point decisions. The advantage of using kappa is that it takes into account agreement that could occur by chance. Kappa is calculated as (observed agreement – chance agreement)/(total number of observations – chance agreement). In our evaluation, we obtain a kappa value of 87.1, indicating very strong agreement.

7.3.3 Error Analysis

Although the overall accuracy of our automatically computed scores is in large part comparable to manual IPSyn scoring (and significantly better than the only option currently available for automatic scoring), our system suffers from visible deficiencies in the identification of certain structures within IPSyn.

Four of the 56 structures in IPSyn account for almost half of the number of errors made by our system (46.8% of the errors). Table 7.3.3 lists these IPSyn items, with their respective percentages of the total number of errors. Errors in items S11 (propositional complements), S13 (wh- clauses), and S14 (bitransitive predicates) are caused by erroneous syntactic analyses. For an example of how GR assignments affect IPSyn scoring, let us consider item S11. Searching for the relation COMP is a crucial part in finding propositional complements. However, COMP is one of the GRs that can be identified the least reliably in our set (precision of 0.73 and recall of 0.89) with the MST system as used in these experiments. As described in section 7.1, IPSyn requires that we credit zero points to item S11 for no occurrences of propositional complements, one point for a single occurrence, and two points for two or more occurrences. If there are several COMPs in the transcript, we should find about most of them (plus others, in error), and correctly arrive at a credit of two points. However, if there is only one or none, our count may be too high, since COMP precision is 0.73.

Most errors in item V15 (emphasis or ellipsis) were caused not by incorrect GR assignments, but by imperfect search patterns. The searching failed to account for a number of configurations of GRs, POS tags and words that indicate that emphasis or ellipsis exists. This reveals another general source of error in our IPSyn implementation: the search pat-

IPSyn item	Error
V15 (copula, modal or aux for emphasis or ellipsis)	16.1%
S11 (propositional complement)	12.9%
S13 (wh- clause)	9.7%
S14 (bitransitive predicate)	8.1%

Table 7.3. IPSyn structures where errors occur most frequently, and their percentages of the total number of errors over 41 transcripts.

terns that use GR analyzed text to make the actual IPSyn scoring decisions. Although our patterns are far more reliable than what we could expect from POS tags and words alone, these are still hand-crafted rules that need to be debugged and perfected over time. Given that only a handful of transcripts were used during development before evaluation of the complete system, most of the rules performed well. Although extensive debugging of IPSyn search patterns is beyond the scope of the work presented here, we speculate that the use of such a system by child language researchers in practical settings would generate feedback that would allow further improvement of the rules. As it stands, our automated version of IPSyn quite reliable when compared to human scoring and makes it clear that both our annotation scheme and our GR analysis are suitable for this task.

7.4 Relating parser accuracy to IPSyn accuracy

In our evaluation of an automated IPSyn system using the MST parser combination (our most accurate system for extracting GRs from child language transcripts), we determined that the IPSyn scores calculated automatically were almost as reliable as what can be expected of scores computed manually. In this section we examine the relationship between parser accuracy and the accuracy of automated IPSyn computations.

In chapter 6 we found the labeled dependency accuracy of the MST combination on the Eve adult test set to be over 92%, and 88% on the Eve child test. However, it is difficult to say what the accuracy of the system is on the transcripts used in our IPSyn experiments, since they include children of different ages, the transcription quality is at times inferior than that of Eve transcriptions (especially with regard to utterance breaks) and the children often talk about specific subjects not encountered in the Eve corpus (such as soccer, school and television). To determine the accuracy of the MST combination on the transcripts used in IPSyn calculations, and attempt to establish a relationship between IPSyn accuracy and labeled dependency parsing accuracy, we annotated six transcripts (three from each of the two data sets) manually according to the Eve GR annotation scheme. Using these six transcripts as gold standard data, we evaluated the labeled dependency accuracy of the

MST combination to be 85.9% on these transcripts. While this is lower than the accuracy on the Eve test sets, this confirms that our parsing techniques perform well on completely unseen data, even when transcription standards are slightly different than what is found in CHILDES transcripts. A second interesting point is that the IPSyn system is almost as reliable as manual computation, even though parsing accuracy is well below 90%. This is in part due to IPSyn being originally designed to have high reliability and be robust in the face of human error.

This brings us to two additional questions. How accurate would our automatic IPSyn computations be if we had a perfect GR parser? In other words, what is the upper bound on IPSyn accuracy, using our (imperfect) rules that convert GR annotations into IPSyn points? Given that parsing accuracy is only 85.9%, we might guess that there may still be some ground to be gained. On the other hand, considering the results of the experiments described above, and the fact that IPSyn was designed to be robust, we may already be close to the upper bound. The second question is about going the opposite direction. How much would IPSyn scores suffer if we used a less accurate parser? This may seem like a strange question, but it is relevant in practice, since we may not want to run five parsers and combine their results to obtain IPSyn scores if accurate scores can be computed using a single parser.

We answer the second question first, simply by repeating our IPSyn experiments using Cdep, the GR system based on the Charniak (2000) parser. That parser has labeled dependency accuracy of 86.9% on the Eve adult test set (compared to the MST combination's 92.3%), and 83.0% on the six transcripts in the IPSyn data sets that were manually annotated with GRs (compared to the MST combination's 85.9%). The Cdep-based IPSyn system does very well on the IPSyn task, considering its dependency accuracy is below 85%. The overall point-to-point reliability is 92.8% and the average point difference is 3.3 (compared to 93.3% and 2.5 using the MST combination system). One interesting observation is that when the MST combination is used, the IPSyn item where most errors occur is V15 (copula, modal or aux for emphasis or ellipsis) due to poor conversion from GRs to IPSyn points, while when the Cdep-based system is used, the IPSyn item where most errors occur is S11 (propositional complement) due to Cdep's poor precision and recall of COMPs (0.60 and 0.50, respectively).

The question of determining an upper bound for IPSyn accuracy could be more difficult to address, since it requires gold standard GRs. Fortunately, we already have six transcripts manually annotated with GRs, which can be used to determine parser accuracy. We use these transcripts to compare the accuracy of our IPSyn implementation using (close to) perfect vs. automatically generated GRs. On the six transcript set, the point-to-point reliability for the IPSyn system using gold standard GRs is 93.4%. The point-to-point

reliability of the system using MST combination on the same six transcripts is 93.0%. The average point difference obtained with gold GRs was 2.3, and with automatic GR analysis that goes up to 2.5. This shows that we are in fact very close to the upper bound of IPSyn accuracy using our rules to count IPSyn items from GR analyses. Many of the differences between the IPSyn scores computed manually and the IPSyn scores computed automatically with gold standard GRs are not necessarily errors made by the automatic system. Given that human inter-rater reliability for IPSyn is 94%, it is not too surprising that we do not see much improvement using gold standard GRs.

Chapter 8

Conclusions and Future Directions

This dissertation presents a multi-strategy approach for syntactic analysis of transcripts of parent-child dialogs. We have focused on the English portion of the CHILDES database, which has been the source of data for over 1,500 published articles on child language research. We started by defining a set of grammatical relations (GRs) to annotate in CHILDES transcripts, and proceeded to describe several approaches to performing the GR annotation task automatically through syntactic parsing. Rule-based and data-driven approaches were explored, as well as a parser combination framework that allows for high accuracy in syntactic analysis of both the adult and child utterances in CHILDES transcripts. Additional experiments using data from the Penn Treebank show that several of the ideas developed in this thesis are applicable not just to analysis of parent-child dialogs, but to parsing in general.

The general contributions of this thesis are: (1) the development of novel techniques for natural language syntactic parsing, (2) the application of these and other existing techniques to a language genre for which no manually annotated training data was previously available (in our case, transcripts of adult-child verbal interactions), and (3) an example of how syntactic analysis of child language can be used in the study of child language. The major specific contributions are:

- A scheme for annotating syntactic information as grammatical relations in transcripts of child-parent dialogs focusing on information relevant to the study of child language. The annotation scheme is based on labeled dependency structures that represent a broad set of grammatical relations (GRs), such as subjects, objects and adjuncts.
- The application of the rule-based robust parsing techniques of Lavie (1996) and Rosé and Lavie (2001) to a high-precision grammatical relation identification system for

parent-child dialogs. We use a multi-pass approach, allowing a gradual increase of coverage and ambiguity by setting robust parsing parameters. Although the recall of grammatical relations obtained with the system is low, precision is very high, allowing the system to contribute as one of the components in a high-accuracy combination system.

- A classifier-based deterministic parsing approach for constituent structures. By using classifiers to determine the actions of a shift-reduce parser, we obtain high levels of precision and recall of constituent structures with a parser that has linear run-time complexity.

- The development of data-driven parsers for grammatical relations, based on generative statistical parsing models, and classifier-based parsing approaches. First, using the Penn Treebank (Marcus et al., 1993) and the Charniak (2000) parser, we show how existing resources and parsing technologies can be adapted to a different domain using a different syntactic representation. We then develop a classifier-based approach for labeled dependencies, based on our classifier-based constituent parser. We show that the performance of a classifier-based parser trained on a small corpus is comparable to that of a more complex system trained on a much larger corpus of text in a different domain.

- The use of different weighted voting schemes for combining different dependency structures, and a novel methodology for using parsing algorithms to combine grammatical relation analysis from multiple systems. We extend the work of Henderson and Brill (1999) on parser combination to the case where several dependency parsers are combined. We also develop new ways of determining different voting weights, and show that more specific weights can produce improved accuracy. Finally, we present a novel way of using maximum spanning trees or the CKY algorithm to combine the results of different parsers producing well- formed dependency structures.

- The demonstration of the effectiveness of the GR annotation scheme and GR identification approach through a task-based evaluation. We implement an automated version of the Index of Productive Syntax, or IPSyn (Scarborough, 1990), a measure of syntactic development in child language used by clinicians and researchers. We show that by using our GR parsing approach, we can produce IPSyn scores fully automatically with accuracy comparable to that of manual scoring. This serves not only as a useful tool for the child language community, but as a way of showing the quality and value of our syntactic analysis approach in a practical setting.

8.1 Possible Directions for Future Research

There are several research directions in which the work presented in this thesis can be extended. We conclude by discussing some of these directions for future research.

8.1.1 Classifier-based shift-reduce parsing

Although the classifier-based parsers described in chapter 5 are deterministic, the classifier-based framework can be extended to perform probabilistic parsing. A best-first probabilistic shift-reduce parser can be created with the addition of a heap of parser states, where a parser state includes a stack and a queue that are used in the same way the stack and queue are used in a deterministic parser, in addition to a probability. The heap should be initialized to contain a single parser state with probability 1.0, where the stack and the queue would be initialized in the same way as in the deterministic case. Then, using a classifier that outputs not just one action to be applied to the stack and queue, but a distribution of actions with associated probabilities, new parser states are created by applying each action to the current state. The probabilities of the new parser states are computed by multiplying the probability of the current state by the probability of the parser action as determined by the classifier. The new parser states are inserted into the heap. A parser state is then popped from the heap, and made the current parser state. This procedure is iterated until the current state has a stack and a queue that meet the same termination conditions as in the deterministic case. The probabilistic model used in this parser would be similar to the probabilistic LR model described by Briscoe and Carroll (1993), where the probability of a parse is the product of each of the probabilities of each parser actions in the derivation of the parse.

Another aspect of the classifier-based parsing framework that can be explored further is the use of different learning techniques. The modularity of the parsing approach with respect to the algorithm and learning mechanism makes it simple to use different classification techniques. An attractive direction is the use of classifiers that support complex features. Although the features currently used in the parser are atomic, some of them are designed to represent aspects of trees, which are complex structures. Learners that are capable of using trees as features may be a more natural fit for classifier-based parsing. Possible learning approaches that have been applied successfully to NLP problems include support vector machines with tree kernels (Moschitti, 2004) and the tree boosting approach of Kudo and Matsumoto (2004).

8.1.2 Parser combination

In chapter 6 we described combination schemes for dependency parsers. An extension of the reparsing framework for parser combination is its application to constituent parsers. While the simple voting scheme has been applied to constituent parser by Henderson and Brill (1999) with impressive results, that approach produces output with much higher precision than recall. In Henderson and Brill's voting scheme, a constituent is added to the combined parse tree if it appears in the output of the majority of the parsers used in the combination. This is necessary in order to produce valid trees as output. In the worst case, where the structures produces by different parsers are completely different, this results in a completely flat tree with no useful syntactic information in it. The reparsing framework could remedy this weakness by removing the restriction that a constituent appear in the majority of the outputs of the parsers involved in the combination. In fact, the constituents from any of the parsers could be considered, and they would be weighted according to the performance of the different parsers on held-out data, as in the dependency case. The reparsing step would consist of running a parsing algorithm for weighted grammars on the input sentence. However, instead of using a grammar, the creation of new constituents would be restricted to those constituents observed in the outputs of the different parsers. The weight of a constituent would be the sum of weights of each instance of the constituent in the output of the parsers. This way, the construction of a well-formed tree as output would be guaranteed, as the parsing algorithm would search for the heaviest tree. Using all the constituents, this approach would favor recall over precision, but by setting a threshold on the weights of constituents that excludes constituents unlikely to be correct, precision and recall could be balanced. This is a generalization of Henderson and Brill's approach, since setting the threshold to consider only constituents that appear in the outputs of most of the parsers would result in exactly Henderson and Brill's voting scheme.

A different direction involving parser combination is to apply machine learning to the combination procedure. Our approach to parser combination takes advantage of the different strengths of the parsers used, but it assumes that the errors made by each parser are independent of the errors made by the other parsers. Knowing that this assumption is very often violated, we performed a parser selection step that measured the accuracy of a combination scheme using different sets of parsers on held-out data, and selected the set with best performance. Instead of using the simple sum of votes that relies on this independence assumption, we can instead use more sophisticated combination techniques based on machine learning. In addition, our best performing combination schemes relied only on the output of the different parsers to determine their weights in the combination. By also taking characteristics of the input into account, we may be able to better estimate how likely the output of each parser is to be correct. An existing framework for combining

different classifiers that could be applied to parser combination was developed by Bennett et al. (2005) in the context of text classification. Their framework takes into account the context-sensitive reliabilities of different classifiers, using rich probabilistic combination of inputs. The combination strategy is then a metaclassifier that learns from the output of the classifiers to be combined, as well as reliability indicators for each classifier based on features that represent the context in which the classification is being made. Decision trees or support vector machines are then used as the learning mechanism of the metaclassifier.

8.1.3 Parsing different languages and different genres

The issue of parsing different languages is one that is strongly tied to an important characteristic of the CHILDES database, which served as the source of much of the data used in our experiments. The CHILDES database contains transcripts in several languages, and the state of NLP resources for these languages varies greatly. While it cannot be expect that the vast resources available for NLP in English would be available in other languages, there have been several recent successful efforts into data-driven parsing other languages (McDonald et al., 2005; Nivre and Nilson, 2005; Wang et al., 2006; Zeman and Žabokrtský, 2005). An initial approach to parsing another language in CHILDES would involve repeating the procedure for parsing English described throughout this dissertation. The first step would be the design of an appropriate syntactic representation for the specific language. One of the areas in which our work on English may have to be extended significantly to handle other languages is the set of features used in the classifier-based parsing framework. For example, Wang et al. have suggested the use of Chinese-specific rhythmic features. Other languages may need features that reflect rich morphology, which is absent in English.

A related issue is that of parsing different language genres. The research presented here is a case of applying parsing techniques to a domain where no resources for syntactic parsing data were previously available (with the exception of a part-of-speech tagger). Many of the techniques developed for parsing CHILDES transcripts are applicable to other domains, as has been shown through experiments using Penn Treebank data. Like in the case of parsing other languages, the first step would be to define a set of grammatical relations that are suitable for the new domain and the task where parsing would be applied.

8.1.4 Measurement of syntactic development in child language

Finally, accurate parsing of child language transcripts allows for several future research directions in child language. One area that is directly related to an issue explored in this thesis is the measurement of grammatical complexity in child language. A measure such as

IPSyn (Scarborough, 1990) was designed with the knowledge that transcripts would have to be scored manually, and such a constraint shaped the design of the scoring task. Although the design of entirely new metrics that make better use of the currently available technology is a worthwhile area of research, it is also one that is likely to require a great deal of expertise in child language research. A more tractable goal for natural language processing research is to find better ways to utilize parsing technology to determine scores for existing metrics. In our current implementation of IPSyn, we simply use an automated system to mimic the manual scoring process. However, knowing the precision and recall levels of different GRs produced automatically allows us to use sets of transcripts with corresponding scores to increase the reliability of automatic scoring even further. For example, instead of using whole counts of IPSyn items, fractional counts could be used depending on the expected accuracy of identification of GRs involved in particular items. A more ambitious alternative is the use of machine learning techniques to go from a GR-annotated transcript to an IPSyn score without the use of manually encoded rules, allowing a system to learn the mapping from GRs to scores.

Bibliography

Bennett, P., S. Dumais and E. Horvitz, 2005: The combination of text classifiers using reliability indicatiors. *Information Retrieval*, **8**, 67–100.

Bikel, D., 2002: Design of a multi-lingual, parallel-processing statistical parsing engine. in *Proceedings of HLT2002*. San Diego, CA.

Black, E., S. Abney, D. Flickenger, C. Gdaniec, R. Grishman, P. Harrison, D. Hindle, R. Ingria, F. Jelinek, J. Klavans, M. Liberman, M. Marcus, S. Roukos, B. Santorini and T. Strzalkowski, 1991: A procedure for quantitatively comparing the syntactic coverage of english grammars. in *Proceedings of the February 1991 DARPA Speech and Natural Language Workshop*. Pacific Grove, CA.

Bod, R., R. Scha and K. Sima'an, 2003: *Data-Oriented Parsing*. CSLI Publications, University of Chicago Press, Chicago, IL.

Breiman, L., 1996: Bagging predictors. *Machine Learning*, **24**(2), 123–140.

Brill, E. and J. Wu, 1998: Classifier combination for improved lexical disambiguation. in *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, pp. 191–195. Montreal, Quebec, Canada.

Briscoe, E. and J. Carroll, 1993: Generalised probabilistic lr parsing of natural language (corpora) with unification-based grammars. *Computational Linguistics*, **19**(1), 25–59.

Briscoe, E. and J. Carroll, 1997: Automatic extraction of subcategorization from corpora. in *Proceedings of the Fifth ACL Conference on Applied Natural Language Processing*. Washington, DC.

Briscoe, E. and J. Carroll, 2002: Robust, accurate statistical annotation of general text. in *Proceedings of the Third International Conference on Language Resources and Evaluation*, pp. 1499–1504. LREC, Las Palmas.

Brown, R., 1973: *A first language: The early stages*. Harvard University Press, Cambridge, MA.

Buchholz, S., J. Veenstra and W. Daelemans, 1999: Cascaded grammatical relation assignment. in *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC)*. College Park, MD.

Carroll, J., 1993: *Practical unification-based parsing of natural language*. Ph.D. thesis, University of Cambridge.

Carroll, J. and E. Briscoe, 2002: High precision extraction of grammatical relations. in *Proceedings of the 19th International Conference on Computational Linguistics (COLING)*, pp. 134–140. Taipei.

Carroll, J., E. Briscoe and A. Sanfilippo, 1998: Parser evaluation: a survey and a new proposal. in *Proceedings of the First International Conference on Language Resources and Evaluation*, pp. 447–454. Granada, Spain.

Carroll, J., G. Minnen and E. Briscoe, 2003: Parser evaluation: Using a grammatical relation annotation scheme. in *Treebanks: Building and using syntactically annotated corpora*, pp. 299–316. Kluwer, Dordrect.

Charniak, E., 1997: Statistical techniques for natural language parsing. *AI Magazine*, **18**(4), 33–44.

Charniak, E., 2000: A maximum-entropy-inspired parser. in *Proceedings of the First Meeting of the North American Chapter of the Association for Computational Linguistics*, pp. 132–139. Seattle, WA.

Charniak, E. and M. Johnson, 2001: Edit detection and parsing for transcribed speech. in *Proceedings of the 2nd Meeting of the North American Chapter of the Association for Computational Linguistics*, pp. 118–126. Pittsuburgh, PA.

Charniak, E. and M. Johnson, 2005: Coarse-to-fine n-best parsing and maxent discriminative reranking. in *Proceedings of the 43rd meeting of the Association for Computational Linguistics*. Ann Arbor, MI.

Chu, Y. J. and T. H. Liu, 1965: On the shortest arborescence of a directed graph. *Science Sinica*, (14), 1396–1400.

Clark, S. and J. R. Curran, 2004a: The importance of supertagging for wide-coverage ccg parsing. in *Proceedings of the International Conference on Computational Linguistics (COLING)*. Geneva, Switzerland.

Clark, S. and J. R. Curran, 2004b: Parsing the wsj using ccg and log-linear models. in *Proceedings of the 42nd Meeting of the Association for Computational Linguistics*. Barcelona, Spain.

Cocke, J. and J. T. Schwartz, 1970: Programming languages and their compilers: Preliminary notes. Tech. rep., New York University.

Cohen, J., 1960: A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, **20**, 37–46.

Collins, M., 1996: A new statistical parser based on bigram lexical dependencies. in *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*. Santa Cruz, CA.

Collins, M., 1997: Three generative, lexicalized models for statistical parsing. in *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pp. 16–23.

Collins, M. and T. Koo, 2003: Discriminative reranking for natural language parsing. in *Proceedings of the Seventeenth International Conference on Machine Learning*. Washington, DC.

Daelemans, W., J. Zavrel, K. van der Sloot and A. van den Bosch, 2004: Timbl: Tilburg memory based learner, version 5.1, reference guide. *ILK Research Group Technical Report Series*, (04-02, 2004).

Dietterich, T. G., 2000: Ensemble methods in machine learning. in *Proceedings of the First International Workshop on Multiple Classifier Systems*, Lecture Notes in Computer Science, pp. 1–15. Springer, Cagliari, Italy.

Dzikovska, M., M. Swift, J. Allen and W. de Beaumont, 2005: Generic parsing for multi-domain semantic interpretation. in *The Ninth International Workshop on Parsing Technologies*, Vancouver, Canada.

Edmonds, J., 1967: Optimum branchings. *Journal of Research of the National Bureau of Standards*, (71B), 233–240.

Eisner, J., 1996: Three new probabilistic models for dependency parsing: An exploration. in *Proceedings of the International Conference on Computational Linguistics (COLING'96)*. Copenhagen, Denmark.

Fletcher, P. and M. Garman, 1988: Larsping by numbers. *British Journal of Disorders of Communication*, **23**, 309–21.

Fletcher, P. and B. MacWhinney, Eds., 1995: *The Handbook of Child Language*. Blackwell, Oxford.

Florian, R. and D. Yarowski, 2002: Modeling consensus: classifier combination for word sense disambiguation. in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 25–32. Philadelphia, PA.

Gildea, D., 2001: Corpus variation and parser performance. in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Pittsburgh, PA.

Giménez, D. and L. Màrquez, 2004: Svmtool: A general pos tagger generator based on support vector machines. in *Proceedings of the Fourth Conference on Language Resources and Evaluation*. Lisbon, Portugal.

Hajic, J., B. Hladká and P. Pajas, 2001: The prague dependency treebank: Annotation structure and support. in *Proceedings of the IRCS Workshop on Linguistic Databases*. Philadelphia, PA.

Henderson, J., 2004: Discriminative training of a neural network statistical parser. in *Proceedings of the 42nd Meeting of the Association for Computational Linguistics*. Barcelona, Spain.

Henderson, J. and E. Brill, 1999: Exploiting diversity in natural language processing: combining parsers. in *Proceedings of the Fourth Conference on Empirical Methods in Natural Language Processing*. College Park, MD.

Hudson, R., 1984: *Word grammar*. Blackwell, Oxford.

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

Kaplan, R. and J. Bresnan, 1982: Lexical-functional grammar: A formal system for grammatical representation. in *The mental representation of grammatical relations*, pp. 173–281. MIT Press.

Kasami, T., 1965: An efficient recognition and syntax-analysis algorithm for context-free languages. Tech. rep., Air Force Cambridge Research Lab.

Klee, T. and M. D. Fitzgerald, 1985: The relation between grammatical development and mean length of utterance in morphemes. *Journal of Child Language*, (12), 251–269.

Klein, D. and C. Manning, 2002: A generative constituent-context model for improved grammar induction. in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pp. 128–135. Philadelphia, PA.

Krogh, A. V., 1995: Neural network ensembles, cross validation, and active learning. in G. Tesauro, D. Touretzky and T. Leen, editors, *Advances in Neural Information Processing Systems*, Vol. 7, pp. 231–238. MIT Press, Cambridge, MA.

Kudo, T. and Y. Matsumoto, 2001: Chunking with support vector machines. in *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics*. Pittsburgh, PA.

Kudo, T. and Y. Matsumoto, 2004: A boosting algorithm for classification of semi-structured text. in *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*. Barcelona, Spain.

Lavie, A., 1996: *GLR*: A robust grammar-focused parser for spontaneously spoken language*. Ph.D. thesis, Carnegie Mellon University.

Lee, L., 1974: *Developmental Sentence Analysis*. Northwestern University Press, Evanston, IL.

Lin, D., 1998: A dependency-based method for evaluating broad-coverage parsers. *Natural Language Engineering*, **4**(2), 97–114.

Long, S. H. and R. W. Channell, 2001: Accuracy of four language analysis procedures performed automatically. *American Journal of Speech-Language Pathology*, **2**(10).

Long, S. H., M. E. Fey and R. W. Channell, 2004: Computerized profiling.

MacWhinney, B., 1999: *The emergence of language*. Lawrence Erlbaum Associates, Mahwah, NJ.

MacWhinney, B., 2000: *The CHILDES Project: Tools for Analyzing Talk*. Lawrence Erlbaum, Mahwah, NJ, 3rd edition.

Magerman, D., 1995: Statistical decision-tree models for parsing. in *Proceedings of the 33rd Meeting of the Association for Computational Linguistics*.

Marcus, M. P., B. Santorini and M. A. Marcinkiewics, 1993: Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, **19**.

McDonald, R., F. Pereira, K. Ribarov and J. Hajic, 2005: Non-projective dependency parsing using spanning tree algorithms. in *Proceedings of the Conference on Human Language Technologies/Empirical Methods in Natural Language Processing (HLT-EMNLP)*. Vancouver, Canada.

Mel'cuk, I., 1988: *Dependency Syntax: Theory and Practice*. State University of New York Press, New York, NY.

Moerk, E., 1983: *The mother of Eve as a first language teacher*. ABLEX, Norwood, N.J.

Moore, R. C., 2000: Improved left-corner chart parsing for large context-free grammars. in *Proceedings of the Sixth International Workshop on Parsing Technologies*. Seattle, WA.

Moschitti, A., 2004: A study on convolution kernels for shallow semantic parsing. in *Proceedings of the 42nd Meeting of the Association for Computational Linguistics*. Barcelona, Spain.

Ninomiya, T., Y. Tsuruoka, Y. Miyao and J. Tsujii, 2005: Efficacy of beam thresholding, unification filtering and hybrid parsing in probabilistic hpsg parsing. in *Proceedings of the Ninth International Workshop on Parsing Technologies*. Vancouver, BC.

Nivre, J. and J. Nilson, 2005: Pseudo-projective dependency parsing. in *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pp. 99–106. Ann Arbor, MI.

Nivre, J. and M. Scholz, 2004: Deterministic dependency parsing of english text. in *Proceedings of the 20th International Conference on Computational Linguistics*, pp. 64–70. Geneva, Switzerland.

Parisse, C. and M.-T. Le Normand, 2000: Automatic disambiguation of the morphosyntax in spoken language corpora. *Behavior Research Methods, Instruments, and Computers*, (32), 468–481.

Pedersen, T., 2000: A simple approach to building ensembles of naive bayesian classifiers for word sense disambiguation. in *Proceedings of the Sixth Applied Natural Language Processing Conference*, pp. 63–69. Seattle, WA.

Ratnaparkhi, A., 1996: A maximum-entropy part-of-speech tagger. in *Proceedings of the the First Conference on Empirical Methods in Natural Language Processing*. Philadelphia, PA.

Ratnaparkhi, A., 1997: A linear observed time statistical parser based on maximum entropy models. in *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*. Providence, RI.

Rosé, C. P. and A. Lavie, 2001: Balancing robustness and efficiency in unification-augmented context-free parsers for large practical applications. in A. van Noord and A. Junqua, editors, *Robustness in language and speech technology*. Kluwer, Amsterdam.

Sachs, J., 1983: Talking about the there and then: The emergence of displaced reference in parent-child discourse. in K. E. Nelson, editor, *Children's language*, Vol. 4. Lawrence Erlbaum Associates, Hillsdale, NJ.

Sagae, K. and A. Lavie, 2005: A classifier-based parser with linear run-time complexity. in *Proceedings of the Ninth International Workshop on Parsing Technologies*. Vancouver, BC.

Sampson, G. and A. Babarczy, 2003: A test of the leaf-ancestor metric for parse accuracy. *Journal of Natural Language Engineering*, **9**, 365–80.

Scarborough, H. S., 1990: Index of productive syntax. *Applied Psycholinguistics*, (11), 1–22.

Sgall, P., E. Hajicová and J. Panevová, 1986: *The meaning of the sentence in its pragmatic aspects*. Reidel.

Sleator, D. and D. Temperley, 1991: Parsing english with a link grammar. Tech. rep., CMU.

Taskar, B., D. Klein, M. Collins, D. Koller and C. Manning, 2004: Max-margin parsing. in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Barcelona, Spain.

Tesnière, L., 1959: *Elément de syntaxe structurale*. Klincksieck, Paris, France.

Tomita, M., 1987: An efficient augmented context-free parsing algorithm. *Computational Linguistics*, **13**, 31–46.

Tomita, M., 1990: The generalized lr parser/compiler - version 8.4. in *Proceedings of the International Conference on Computational Linguistics (COLING'90)*, pp. 59–63. Helsinki, Finland.

Wang, M., K. Sagae and T. Mitamura, 2006: A fast, accurate deterministic parser for chinese. in *Proceedings of the Joint Conference of the International Committee on Computational Linguistics and the Association for Computational Linguistics (COLING-ACL 2006)*. Sydney, Australia.

Wolpert, D., 1992: Stacked generalization. *Neural Networks*, **5**, 241–259.

Yamada, H. and Y. Matsumoto, 2003: Statistical dependency analysis using support vector machines. in *Proceedings of the Eighth International Workshop on Parsing Technologies*. Nancy, France.

Yeh, A., 2000a: More accurate tests for the statistical significance of result differences. in *Proceedings 17th International Conference on Computational Linguistics*, pp. 947–953. Saarbuken, Germany.

Yeh, A., 2000b: Using existing systems to supplement small amounts of annotated grammatical relations training data. in *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pp. 126–132. Hong Kong.

Younger, D. H., 1967: Recognition and parsing of context-free languages in time n3. *Information and Control*, **10**(2), 189–208.

Zeman, D. and Z. Žabokrtský, 2005: Improving parsing accuracy by combining diverse dependency parsers. in *Proceedings of the International Workshop on Parsing Technologies*. Vancouver, Canada.

Appendix A

Head Percolation Table for Lexicalization of Constituents

Conversion from constituent structures to dependency structures is done through a process of lexicalization of the constituent tree (Magerman, 1995; Collins, 1996). The rules used for converting the WSJ Penn Treebank constituent trees into unlabeled dependencies in our experiments are the same rules used originally by Yamada and Matsumoto (2003) and subsequently adopted by Nivre and Scholz (2004) and McDonald et al. (2005). Yamada and Matsumoto's rules are a slight variation of the rules published by Collins(1999).

Each row in the head percolation table corresponds to a rule that describes how to find the lexical head of constituent given its non-terminal symbol. Each rule contains three parts: (1) the non-terminal symbol of the constituent to be lexicalized, (2) a direction from where to look for the constituent's head, and (3) a priority list containing non-terminal symbols and preterminals (POS tags). Applying a rule from the table is a simple of scanning the children of the node being lexicalized in the direction specified, looking for a child of a type from the priority list. All children are scanned for an item in the priority list before other items are considered. When a match is found, the lexical head of the children is assigned as the lexical head of the current node. The specific rules are listed in table A.1.

NP	r	POS NN NNP NNPS NNS NX JJR CD JJ JJS RB
		QP NP
ADJP	r	NNS QP NN $ ADVP JJ VBN VBG ADJP
		JJR NP JJS DT FW RBR RBS SBAR RB
ADVP	l	RB RBR RBS FW ADVP TO CD JJR JJ IN NP
		JJS NN
CONJP	l	CC RB IN
FRAG	l	
INTJ	r	
LST	l	LS :
NAC	r	NN NNS NNP NNPS NP NAC EX $ CD QP
		PRP VBG
		JJ JJS JJR ADJP FW
PP	l	IN TO VBG VBN RP FW
PRN	r	
PRT	l	RP
QP	r	$ IN NNS NN JJ RB DT CD NCD QP JJR JJS
RRC	l	VP NP ADVP ADJP PP
S	r	TO IN VP S SBAR ADJP UCP NP
SBAR	r	WHNP WHPP WHADVP WHADJP IN DT S SQ SINV
		SBAR FRAG
SBARQ	r	SQ S SINV SBARQ FRAG
SINV	r	VBZ VBD VBP VB MD VP S SINV ADJP NP
SQ	r	VBZ VBD VBP VB MD VP SQ
UCP	l	
VP	l	VBD VBN MD VBZ VB VBG VBP VP ADJP NN
		NNS NP
WHADJP	r	CC WRB JJ ADJP
WHADVP	l	CC WRB
WHNP	r	WDT WP WP$ WHADJP WHPP WHNP
WHPP	l	IN TO FW
NX	r	POS NN NNP NNPS NNS NX JJR CD JJ JJS RB
		QP NP
X	r	

Table A.1. Head percolation table.

Appendix B

Complete GR Results on the Eve GR Test Set

This appendix provides the complete results for the various GR analysis systems presented in this thesis. The systems covered in this appendix are:

Cdep : the data-driven approach based on the Charniak (2000) parser and a separate step for dependency labeling using a classifier;

SVMdep : the deterministic parser using support vector machines to determine the parser's actions;

MBLdep : the deterministic parser using memory-based learning to determine the parser's actions;

RL-SVMdep : the deterministic parser using support vector machines to determine the parser's actions, and processing the input from right to left;

RL-MBLdep : the deterministic parser using memory-based learning to determine the parser's actions, and processing the input from right to left;

RB : the rule-based parser set for high precision;

POS-weighted voting : the word-by-word voting scheme, with votes weighted according to the parser generating the vote and the part-of-speech of the current word;

Label-weighted voting : the word-by-word voting scheme, with votes weighted according to the parser generating the vote and the label of the dependency of which the current word is a dependent;

CKY-combination : the reparsing approach using the CKY algorithm, with dependency weights set as in label-weighted voting;

MST-combination : the reparsing approach using maximum spanning trees, with dependency weights set as in label-weighted voting.

Unlabeled Dependency Accuracy: 91.3
Labeled Dependency Accuracy: 86.2

GR	Precision	Recall	F-score
CJCT	0.00	0.00	0.00
PRED	0.78	0.87	0.82
CPRED	0.00	0.00	0.00
TAG	0.00	0.00	0.00
PTL	0.83	0.83	0.83
INF	1.00	0.93	0.96
ESUBJ	1.00	1.00	1.00
JCT	0.73	0.82	0.77
AUX	0.97	0.95	0.96
VOC	0.00	0.00	0.00
XJCT	0.00	0.00	0.00
COORD	0.69	0.94	0.79
DET	0.98	1.00	0.99
OBJ	0.95	0.87	0.91
POBJ	0.96	0.96	0.96
MOD	0.89	0.87	0.88
COM	0.83	0.80	0.81
CPZR	0.89	0.80	0.84
XCOMP	0.71	0.59	0.65
ROOT	0.94	0.94	0.94
SUBJ	0.94	0.94	0.94
COMP	0.62	0.56	0.59
QUANT	0.67	1.00	0.80
NEG	0.87	0.50	0.63
CMOD	0.00	0.00	0.00

Table B.1. Cdep: GR results.

Unlabeled Dependency Accuracy: 91.1
Labeled Dependency Accuracy: 87.3

GR	Precision	Recall	F-score
CJCT	0.50	0.33	0.40
PRED	0.80	0.83	0.82
CPRED	0.00	0.00	0.00
TAG	1.00	1.00	1.00
PTL	0.67	0.33	0.44
INF	1.00	1.00	1.00
ESUBJ	1.00	1.00	1.00
JCT	0.78	0.88	0.83
AUX	0.96	0.97	0.97
PUNCT	1.00	1.00	1.00
VOC	0.33	0.50	0.40
XJCT	0.00	0.00	0.00
COORD	0.71	0.76	0.74
DET	1.00	1.00	1.00
OBJ	0.90	0.94	0.92
POBJ	0.92	0.94	0.93
MOD	0.94	0.87	0.91
COM	0.90	0.82	0.86
CPZR	1.00	0.70	0.82
XCOMP	0.93	0.82	0.87
ROOT	0.95	0.94	0.94
SUBJ	0.97	0.98	0.98
COMP	0.70	0.78	0.74
QUANT	0.80	1.00	0.89
NEG	1.00	0.96	0.98
CMOD	0.40	0.50	0.44

Table B.2. SVMdep: GR results.

Unlabeled Dependency Accuracy: 91.0
Labeled Dependency Accuracy: 86.0

GR	Precision	Recall	F-score
CJCT	0.50	0.17	0.25
PRED	0.75	0.81	0.78
CPRED	0.00	0.00	0.00
TAG	1.00	1.00	1.00
PTL	1.00	0.50	0.67
INF	1.00	1.00	1.00
ESUBJ	1.00	1.00	1.00
JCT	0.77	0.82	0.80
AUX	0.91	0.99	0.95
VOC	0.25	0.50	0.33
XJCT	0.00	0.00	0.00
COORD	0.83	0.76	0.79
DET	0.96	0.98	0.97
OBJ	0.87	0.84	0.86
POBJ	0.90	0.90	0.90
MOD	0.96	0.85	0.90
COM	0.83	0.89	0.86
CPZR	0.88	0.70	0.78
XCOMP	0.82	0.82	0.82
ROOT	0.93	0.92	0.92
SUBJ	0.95	0.96	0.95
COMP	0.62	0.56	0.59
QUANT	1.00	0.75	0.86
NEG	0.93	1.00	0.96
CMOD	0.00	0.00	0.00

Table B.3. MBLdep: GR results.

Unlabeled Dependency Accuracy: 89.0
Labeled Dependency Accuracy: 85.1

GR	Precision	Recall	F-score
CJCT	0.00	0.00	0.00
PRED	0.77	0.76	0.77
CPRED	0.00	0.00	0.00
TAG	0.00	0.00	0.00
PTL	0.83	0.83	0.83
INF	1.00	1.00	1.00
ESUBJ	1.00	0.50	0.67
JCT	0.72	0.88	0.79
AUX	0.91	0.97	0.94
PUNCT	1.00	1.00	1.00
VOC	0.25	0.50	0.33
XJCT	0.00	0.00	0.00
COORD	0.83	0.88	0.85
DET	0.96	1.00	0.98
OBJ	0.91	0.91	0.91
POBJ	0.94	0.94	0.94
MOD	0.98	0.85	0.91
COM	0.92	0.82	0.87
CPZR	1.00	0.70	0.82
XCOMP	0.93	0.82	0.87
ROOT	0.92	0.91	0.92
SUBJ	0.92	0.96	0.94
COMP	0.71	0.56	0.63
QUANT	0.44	1.00	0.62
NEG	1.00	0.88	0.94
CMOD	0.00	0.00	0.00

Table B.4. RL-SVMdep: GR results.

Unlabeled Dependency Accuracy: 82.7
Labeled Dependency Accuracy: 79.2

GR	Precision	Recall	F-score
CJCT	0.00	0.00	0.00
PRED	0.84	0.85	0.84
CPRED	0.00	0.00	0.00
TAG	0.50	1.00	0.67
PTL	0.83	0.83	0.83
INF	1.00	1.00	1.00
ESUBJ	1.00	1.00	1.00
JCT	0.78	0.79	0.78
AUX	0.83	0.97	0.89
VOC	0.17	0.50	0.25
XJCT	0.00	0.00	0.00
COORD	0.78	0.64	0.70
DET	1.00	0.98	0.99
OBJ	0.89	0.84	0.87
POBJ	0.92	0.92	0.92
MOD	0.97	0.78	0.87
COM	0.84	0.73	0.78
CPZR	0.88	0.70	0.78
XCOMP	0.88	0.82	0.85
ROOT	0.82	0.86	0.84
SUBJ	0.94	0.97	0.96
COMP	0.83	0.56	0.67
QUANT	0.80	1.00	0.89
NEG	0.83	0.92	0.87
CMOD	0.00	0.00	0.00

Table B.5. RL-MBLdep: GR results.

Unlabeled Dependency Accuracy: 29.2
Labeled Dependency Accuracy: 28.8

GR	Precision	Recall	F-score
CJCT	0.00	0.00	0.00
PRED	1.00	0.28	0.43
CPRED	0.00	0.00	0.00
TAG	0.00	0.00	0.00
PTL	0.00	0.00	0.00
INF	1.00	0.36	0.53
ESUBJ	0.00	0.00	0.00
JCT	0.91	0.37	0.53
AUX	0.97	0.46	0.62
PUNCT	1.00	0.25	0.40
VOC	0.00	0.00	0.00
XJCT	0.00	0.00	0.00
COORD	1.00	0.06	0.11
DET	1.00	0.44	0.62
OBJ	0.94	0.37	0.53
POBJ	0.93	0.29	0.44
MOD	0.91	0.22	0.36
COM	0.80	0.27	0.40
CPZR	0.00	0.00	0.00
XCOMP	1.00	0.29	0.45
ROOT	0.99	0.29	0.45
SUBJ	1.00	0.37	0.54
COMP	1.00	0.33	0.50
QUANT	1.00	0.25	0.40
NEG	1.00	0.23	0.38
CMOD	1.00	0.25	0.40

Table B.6. RB: GR results.

Unlabeled Dependency Accuracy: 94.0
Labeled Dependency Accuracy: 89.5

GR	Precision	Recall	F-score
CJCT	0.00	0.00	0.00
PRED	0.74	0.89	0.81
CPRED	0.00	0.00	0.00
TAG	0.00	0.00	0.00
PTL	0.83	0.83	0.83
INF	1.00	0.93	0.96
ESUBJ	1.00	1.00	1.00
JCT	0.83	0.81	0.82
AUX	0.92	0.96	0.94
PUNCT	0.99	1.00	1.00
VOC	0.00	0.00	0.00
XJCT	0.00	0.00	0.00
COORD	0.86	0.94	0.90
DET	0.98	1.00	0.99
OBJ	0.94	0.88	0.91
POBJ	0.96	0.98	0.97
MOD	0.91	0.90	0.91
COM	0.88	0.82	0.85
CPZR	1.00	0.90	0.95
XCOMP	0.65	0.76	0.70
ROOT	0.95	0.94	0.94
SUBJ	0.96	0.94	0.95
COMP	0.57	0.44	0.50
QUANT	1.00	1.00	1.00
NEG	0.81	0.96	0.88
CMOD	0.00	0.00	0.00

Table B.7. POS-weighted voting: GR results.

Unlabeled Dependency Accuracy: 94.8
Labeled Dependency Accuracy: 92.3

GR	Precision	Recall	F-score
CJCT	0.00	0.00	0.00
PRED	0.89	0.89	0.89
CPRED	0.00	0.00	0.00
TAG	1.00	1.00	1.00
PTL	0.83	0.83	0.83
INF	1.00	1.00	1.00
ESUBJ	1.00	1.00	1.00
JCT	0.87	0.90	0.88
AUX	0.96	0.97	0.97
PUNCT	1.00	1.00	1.00
VOC	0.00	0.00	0.00
XJCT	0.00	0.00	0.00
COORD	0.88	0.88	0.88
DET	0.96	1.00	0.98
OBJ	0.94	0.94	0.94
POBJ	0.90	0.98	0.94
MOD	0.97	0.90	0.93
COM	0.95	0.86	0.90
CPZR	1.00	0.90	0.95
XCOMP	0.90	0.88	0.89
ROOT	0.94	0.98	0.96
SUBJ	0.98	0.98	0.98
COMP	0.75	0.67	0.71
QUANT	1.00	1.00	1.00
NEG	1.00	0.96	0.98
CMOD	0.00	0.00	0.00

Table B.8. Label-weighted voting: GR results.

Unlabeled Dependency Accuracy: 95.0
Labeled Dependency Accuracy: 92.3

GR	Precision	Recall	F-score
CJCT	1.00	0.17	0.29
PRED	0.86	0.89	0.87
CPRED	0.00	0.00	0.00
TAG	1.00	1.00	1.00
PTL	0.83	0.83	0.83
INF	1.00	1.00	1.00
ESUBJ	1.00	1.00	1.00
JCT	0.85	0.91	0.88
AUX	0.94	0.97	0.95
PUNCT	1.00	1.00	1.00
VOC	1.00	0.50	0.67
XJCT	0.00	0.00	0.00
COORD	0.94	0.91	0.92
DET	0.96	1.00	0.98
OBJ	0.94	0.94	0.94
POBJ	0.92	0.98	0.95
MOD	0.97	0.91	0.94
COM	0.95	0.91	0.93
CPZR	1.00	0.90	0.95
XCOMP	0.88	0.88	0.88
ROOT	0.97	0.96	0.96
SUBJ	0.97	0.98	0.98
COMP	0.73	0.89	0.80
QUANT	1.00	1.00	1.00
NEG	1.00	0.96	0.98
CMOD	0.50	0.25	0.33

Table B.9. CKY-combination: GR results.

Unlabeled Dependency Accuracy: 95.2
Labeled Dependency Accuracy: 92.4

GR	Precision	Recall	F-score
CJCT	1.00	0.17	0.29
PRED	0.87	0.89	0.88
CPRED	0.00	0.00	0.00
TAG	1.00	1.00	1.00
PTL	0.83	0.83	0.83
INF	1.00	1.00	1.00
ESUBJ	1.00	1.00	1.00
JCT	0.85	0.91	0.88
AUX	0.94	0.99	0.96
PUNCT	1.00	1.00	1.00
VOC	0.00	0.00	0.00
XJCT	0.00	0.00	0.00
COORD	0.91	0.91	0.91
DET	0.96	1.00	0.98
OBJ	0.95	0.96	0.95
POBJ	0.92	0.98	0.95
MOD	0.97	0.91	0.94
COM	0.93	0.91	0.92
CPZR	1.00	0.90	0.95
XCOMP	0.88	0.88	0.88
ROOT	0.97	0.96	0.96
SUBJ	0.98	0.98	0.98
COMP	0.73	0.89	0.80
QUANT	1.00	1.00	1.00
NEG	1.00	0.96	0.98
CMOD	1.00	0.25	0.40

Table B.10. MST-combination: GR results.

Appendix C

IPSyn Items and Scoring Sheet

This appendix contains the original IPSyn scoring sheet that appeared in (Scarborough, 1990), in addition to the description of the grammatical structures that correspond to each IPSyn item.

IPSyn items:

N1 Proper, mass, or count noun

N2 Pronoun or prolocative, excluding modifiers

N3 Modifier, including adjectives, possessives, and quantifiers

N4 Two-word NP: nominal preceded by article or modifier

N5 Article, used before a noun

N6 Two-word NP after verb or preposition

N7 Plural suffix

N8 Two-word NP before verb

N9 Three-word NP (Det/Mod + Mod + N)

N10 Adverb modifying adjective or nominal

N11 Any other bound morpheme on N or adjective

N12 Other (not used)

V1 Verb

V2 Particle or prepositioin

V3 Prepositional phrase

V4 Copula linking two nominals

V5 Catenative (pseudo-auxiliary) preceding a verb

V6 Auxiliary be, do, have in VP

V7 Progressive suffix

V8 Adverb

V9 Modal preceding verb

V10 Third person singular present tense suffix

V11 Past tense modal

V12 Regular past tense suffix

V13 Past tense auxiliary

V14 ‘‘Medial’’ adverb

V15 Copula, modal, or auxiliary for emphasis or ellipsis

V16 Past tense copula

V17 Other -- e.g., bound morpheme on verb or on adjective (to make adverb)

Q1 Intonationally marked question

Q2 Routine do/go or existence/name question or wh- pronoun alone

Q3 Simple negation

Q4 Initial wh- pronoun followed by verb

Q5 Negative morpheme between subject and verb

Q6 Wh- question with inverted modal, copula or auxiliary

Q7 Negation of copula, modal, or auxiliary

Q8 Yes/no question with inverted modal, copula, or auxiliary

Q9 Why, when, which, whose

Q10 Tag question

Q11 Other: e.g., questions with negation and inverted copula/aux/modal

S1 Two-word combination

S2 Subject-verb sequence

S3 Verb-object sequence

S4 Subject-verb-object sequence

S5 Conjunction

S6 Sentence with two VPs

S7 Conjoined phrases

S8 Infinitive without catenative, marked with ‘‘to’’

S9 Let/Make/Help/Watch introducer

S10 Adverbial conjunction

S11 Propositional complement

S12 Conjoined sentences

S13 Wh- clause

S14 Bitransitive predicate

S15 Sentence with three or more VPs

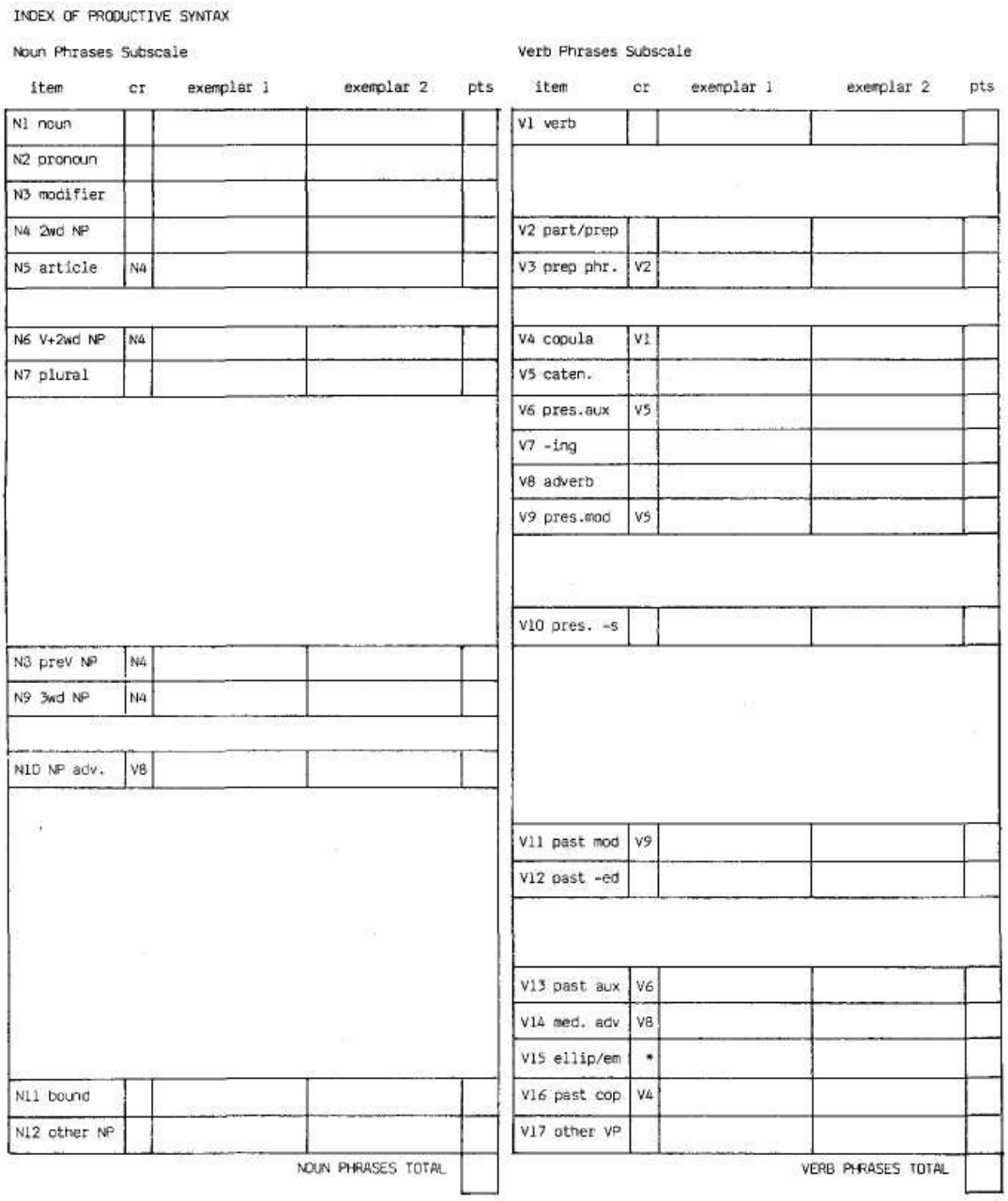
S16 Relative clause, marked or unmarked

S17 Infinitive clause: new subject

S18 Gerund

S19 Fronted or center embedded subordinate clause

S20 Other: e.g., passive constructions, tag comments/intrusions



INDEX OF PRODUCTIVE SYNTAX

Noun Phrases Subscale

item	cr	exemplar 1	exemplar 2	pts
N1 noun				
N2 pronoun				
N3 modifier				
N4 2wd NP				
N5 article	N4			
N6 V+2wd NP	N4			
N7 plural				
N8 preV NP	N4			
N9 3wd NP	N4			
N10 NP adv.	V8			
N11 bound				
N12 other NP				
			NOUN PHRASES TOTAL	

Verb Phrases Subscale

item	cr	exemplar 1	exemplar 2	pts
V1 verb				
V2 part/prep				
V3 prep phr.	V2			
V4 copula	V1			
V5 caten.				
V6 pres.aux	V5			
V7 -ing				
V8 adverb				
V9 pres.mod	V5			
V10 pres. -s				
V11 past mod	V9			
V12 past -ed				
V13 past aux	V6			
V14 med. adv	V8			
V15 ellip/em	*			
V16 past cop	V4			
V17 other VP				
			VERB PHRASES TOTAL	

Figure C.1. Part 1 of the IPSyn scoring sheet, from (Scarborough, 1990).

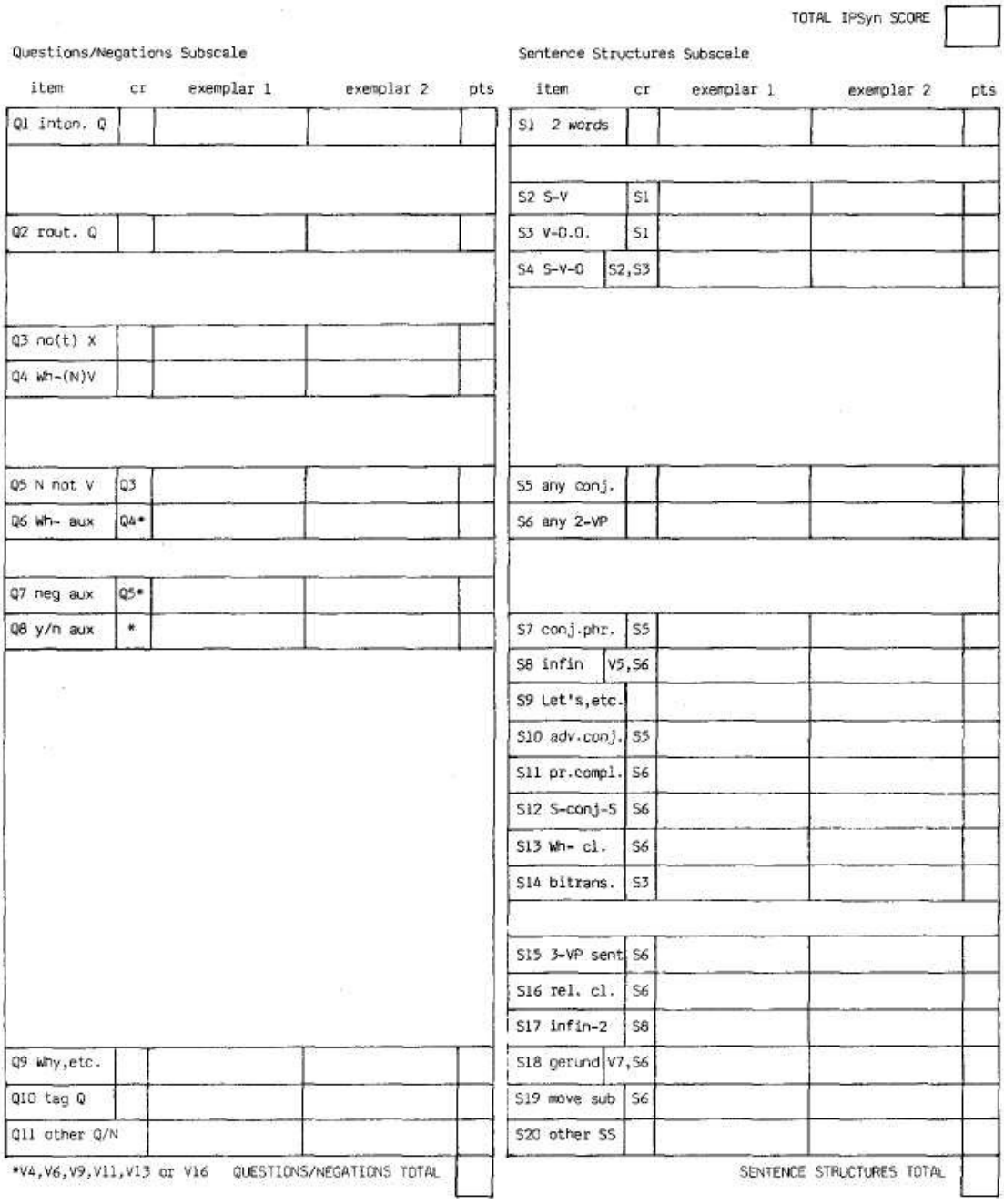


Figure C.2. Part 2 of the IPSyn scoring sheet, from (Scarborough, 1990).