Learning Parse and Translation Decisions From Examples With Rich Context

by

Ulf Hermjakob, Dipl.-Inform.

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

May 1997

Learning Parse and Translation Decisions

From Examples With Rich Context

Approved by
Dissertation Committee:

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

To my parents Udo and Juliane
for their loving support and continuous encouragement for a good education

Acknowledgments

I am very thankful for the contributions of my first advisor, the late Robert F. Simmons, who helped me develop the basic ideas underlying my doctoral research. I also deeply appreciate the dedicated support of my current advisor, Raymond J. Mooney, who guided the completion of my dissertation with many helpful insights. Many thanks for valuable comments also to the other members of my committee, Risto Miikkulainen, Benjamin J. Kuipers, Manfred Krifka and Elaine A. Rich.

To Elaine Rich and Jim Barnett, who led the Knowledge Based Natural Language group at the Microelectronics and Computer Technology Corporation, I owe many stimulating ideas and much support during the four years I worked at MCC.

Many thanks also to everybody in the Natural Language Acquisition discussion group at our computer science department; besides Bob Simmons, Ray Mooney and Risto Miikkulainen in particular to John Zelle, for the many interesting discussions we had since the group was started in 1992.

Finally I would like to give thanks to Brent Adamson, Ulrich Ehrenberger, Ralf Gerlich, Thomas Göhlert, Mike Hoefelein, Martin Kracklauer, Michael Menth, Florian Mertens, Christoph Püntmann and Joe Sullivan for volunteering as translation evaluators.

Ulf Hermjakob

The University of Texas at Austin
May 1997

Learning Parse and Translation Decisions From Examples With Rich Context

Publication No. ____________________

Ulf Hermjakob, Ph.D.
The University of Texas at Austin, 1997

Supervisor: Raymond J. Mooney

The parsing of unrestricted text, with its enormous lexical and structural ambiguity, still poses a great challenge in natural language processing. The difficulties with traditional approaches, which try to master the complexity of parse grammars with hand-crafted rules, have led to a trend towards more empirical techniques.

We therefore propose a system for parsing and translating natural language that learns from examples and uses some background knowledge. As our parsing model we choose a deterministic shift-reduce type parser that integrates part-of-speech tagging and syntactic and semantic processing, which not only makes parsing very efficient, but also assures transparency during the supervised example acquisition. Applying machine learning techniques, the system uses parse action examples to generate a parser in the form of a decision structure, a generalization of decision trees. To learn good parsing and translation decisions, our system relies heavily on context, as encoded in currently 205 features describing the morphological, syntactical and semantical aspects of a given parse state. Compared with recent probabilistic systems that were trained on 40,000 sentences, our system relies on more background knowledge and a deeper analysis, but radically fewer examples, currently 256 sentences.

We test our parser on lexically limited sentences from the Wall Street Journal and achieve accuracy rates of 89.8% for labeled precision, 98.4% for part of speech tagging and 56.3% of test sentences without any crossing brackets. Machine translations of 32 Wall Street Journal sentences to German have been evaluated by 10 bilingual volunteers and been graded as 2.4 on a 1.0 (best) to 6.0 (worst) scale for both grammatical correctness and meaning preservation. The translation quality was only minimally better (2.2) when starting each translation with the correct parse tree, which indicates that the parser is quite robust and that its errors have only a moderate impact on final translation quality. These parsing and translation results already compare well with other systems and, given the relatively small training set and amount of overall knowledge used so far, the results suggest that our system CONTEX can break previous accuracy ceilings when scaled up further.

Contents

Acknowledgments **v**

Abstract **vi**

Chapter 1 Introduction **1**

1.1 Previous Approaches 2

1.1.1 Traditional Approaches 2

1.1.2 Empirical Approaches 2

1.2 A Context-Oriented Machine Learning Approach 3

1.3 Summary of Experimental Results 4

1.4 Organization of Dissertation 4

Chapter 2 System Architecture **6**

2.1 Task Definition 6

2.1.1 Corpus 6

2.2 Transfer-Based vs. Interlingua Approach 6

2.3 Major System Components 7

2.3.1 Features 9

Chapter 3 Background Knowledge **11**

3.1 Knowledge Base 12

3.2 Monolingual Lexicons 13

3.3 Acquisition of Lexicon and KB 16

3.4 Bilingual Dictionary 17

3.5 Subcategorization Tables 19

3.6 Morphology 21

3.6.1 Specific Pipeline Elements 24

Chapter 4 Segmentation and Morphological Analysis **27**

4.1 Segmentation 27

4.2 Morphological Analysis and Parse Entries 28

4.2.1 List of Morphological Forms 30

Chapter 5 Parsing **31**
5.1 Basic Parsing Paradigm . . . 31
5.2 The Core Parsing Mechanism . . . 33
5.3 The Parse Actions in Detail . . . 34
5.3.1 Parse Entry Paths . . . 34
5.3.2 Shift . . . 36
5.3.3 Reduce . . . 37
5.3.4 Add Into . . . 38
5.3.5 Empty Category Instantiation . . . 39
5.3.6 Co-Index . . . 40
5.3.7 Mark . . . 40
5.3.8 Expand . . . 40
5.3.9 Done . . . 41
5.3.10 Multiple Parse Actions . . . 41
5.3.11 Triggered Computations . . . 42
5.4 Anaphora Resolution . . . 42
5.5 Parsing Safeguards . . . 43
5.6 Features . . . 44
5.6.1 General Feature Structure . . . 45
5.6.2 Syntactic and Semantic Category Features . . . 46
5.6.3 Syntactic and Semantic Role Features . . . 46
5.6.4 Form Features . . . 47
5.6.5 Other Unary Boolean Features . . . 48
5.6.6 Binary Boolean Features . . . 48
5.6.7 Other Features . . . 48
5.7 Training the Parser . . . 49
5.8 Learning Decision Structures . . . 51
5.8.1 Review of Decision Trees and Lists . . . 51
5.8.2 Decision Hierarchies . . . 54
5.8.3 Syntax of Decision Structure Definitions . . . 56
5.8.4 Decision Structure Lists . . . 57
5.8.5 Feature Selection Bias in Decision Trees . . . 61
5.8.6 Feature Selection Preferences . . . 62
5.9 Garden Paths . . . 62

Chapter 6 Parsing Experiments **64**
6.1 Corpus . . . 64
6.2 Test Methodology and Evaluation Criteria . . . 64
6.3 Accuracy on Seen Sentences . . . 65
6.4 Testing on Unseen Sentences . . . 65
6.4.1 Contribution of Individual Feature Selection Preferences . . . 68
6.4.2 Contribution of the Subcategorization Table . . . 68
6.4.3 Contribution of Rich Context . . . 69
6.4.4 Contribution of Decision Structure Type . . . 72

Chapter 7 Transfer **74**
7.1 Simple Transfer . . . 74
7.2 Ambiguous Transfer . . . 75
7.3 Complex Transfer . . . 77
7.4 Added Concepts . . . 79

Chapter 8 Generation **81**
8.1 Ordering . . . 82
8.2 Morphological Propagation and Generation . . . 83

Chapter 9 Translation Experiments **87**
9.1 Test Methodology and Evaluation Criteria . . . 87
9.1.1 Comparative Translations . . . 88
9.1.2 Evaluators and Evaluation Criteria . . . 89
9.2 Test Results . . . 90
9.2.1 Overview . . . 90
9.2.2 Variation Analysis . . . 92
9.2.3 Correlation Analysis . . . 93

Chapter 10 Related Work **97**
10.1 Simmons and Yu . . . 97
10.2 Parsers Learning From Treebank Examples . . . 98
10.2.1 Comparison between Penn Treebank and CONTEX Parse Trees . . . 98
10.2.2 Declarative vs. Operational Parse Information . . . 101
10.2.3 SPATTER . . . 102
10.2.4 IBM Language Modeling Group . . . 103
10.2.5 Bigram Lexical Dependencies . . . 103
10.2.6 Comparison of Results . . . 103

Chapter 11 Future Work **105**
11.1 Scaling Up Further . . . 105
11.1.1 New Domains . . . 106
11.1.2 Lexical Expansion . . . 106
11.1.3 Considering Context Beyond the Sentence Boundary . . . 108
11.2 Incomplete Knowledge . . . 108
11.2.1 Unknown Words . . . 108
11.2.2 Incomplete Subcategorization Table . . . 109
11.2.3 Incomplete Knowledge Base . . . 109
11.3 Moving Beyond ‘English to German’ . . . 110
11.3.1 German to English . . . 110
11.3.2 Adding More Languages . . . 110
11.3.3 Grammar Checking . . . 111
11.4 Speed-Ups . . . 112

Chapter 12 Conclusions **113**

Appendix A WSJ Corpus **115**

Appendix B Features **125**

B.1 Features Used for Parsing English . . . 125

B.2 Features Used in English to German Translation Disambiguation Decision Structures 130

Appendix C Parse Example **131**

C.1 After Segmentation and Morphological Processing . . . 131

C.2 Parse State and Actions Step by Step . . . 132

C.3 Resulting Parse Tree . . . 133

Appendix D Translation Evaluation Questionnaire and Key **137**

D.1 Translation Evaluation Questionnaire . . . 137

D.2 Translation Evaluation Key . . . 152

Appendix E Abbreviations **158**

Bibliography **160**

Vita **165**

Chapter 1

Introduction

Natural language processing in general and machine translation in particular face many difficulties. Consider for example the following sentences:

(1) He had the right to go. ('right' is a noun)
(2) He was right in deciding not to go. (adjective)
(3) The exit door was right in the line of fire. (adverb)
(4) Should he wait until the problem might right itself? (verb)

(5) I ate the pasta with a fork. (PP attaches to verb)
(6) I ate the pasta with a delicious tomato sauce. (PP attaches to preced. NP)

(7) I sent a package to New York. (PP attaches to verb)
(8) I booked a flight to New York. (PP attaches to preceding NP)

(9) I saw the Grand Canyon flying to New York. (semantics vs. syntactic bias)
(10) I will pick up the car at the airport. (attachment unclear and irrelevant)

(11) I know your friend in New York. ("to know" $\rightarrow$ "conocer")
(12) I know your friend is in New York. ("to know" $\rightarrow$ "saber")

A parser[1] has to cope with ambiguous words like "right" (sentences 1-4), proper structural attachment (sentences 5-8), even if the preposition phrases are identical (sentences 7&8). Sometimes the existence of ambiguity is relatively obvious to the human reader, particularly when syntactic and semantic clues point in opposite directions (sentence 9), while in other cases the ambiguity is more subtle and possibly irrelevant (sentence 10). Even when there seems to be little ambiguity for words like "to know", other languages often make distinctions, depending on how exactly a word is used. In the case of sentences 11&12, "to know" has different translations for most languages, e.g. "conocer"/"saber" in Spanish.

[1]A *parser* is a program that analyses text to find structural descriptions of the text that encode useful information of some kind about it. The term *parser* is used for the analysis of both source code of computer programs and natural languages, such as English and Swedish. In natural language understanding, the structural descriptions built by the parser typically include a tree reflecting the syntactic structure of the text, but can also contain semantic and other information. A good introductory textbook on natural language understanding is (Allen, 1995).

(13) The statue of liberty in New York, which was sent to the United States as a gift from France, was dedicated in 1886.

(14) The airline said that it would report a loss for the first quarter, but that it would be less than $100 million.

Sentences 13&14 demonstrate the problem of anaphora resolution[2]. The German translation of the second occurrence of the personal pronoun 'it' in sentence 14 for example depends on whether that pronoun refers to "the airline", "a loss", or "the first quarter", because the grammatical gender of the translated German pronoun (*"sie", "er"* or *"es"*) has to agree with the grammatical gender of its antecedent ("the airline" $\rightarrow$ *"die Fluggesellschaft"* (feminine), "a loss" $\rightarrow$ *"ein Verlust"* (masculine), or "the first quarter" $\rightarrow$ *"das erste Quartal"* (neuter)). The same holds for the relative pronoun "which" in sentence 13.

These few examples already show that an automatic parsing and machine translation system has to make many difficult decisions, including part of speech assignment, proper structural analysis, word sense disambiguation and anaphora resolution. Natural language processing is a truly formidable task.

1.1 Previous Approaches

1.1.1 Traditional Approaches

The parsing of unrestricted text, with its enormous lexical and structural ambiguity, still poses a great challenge in natural language processing (NLP). Traditional approaches try to master the complexity of parse grammars with hand-crafted rules, as in augmented transition networks (Bates, 1978), definite clause grammars (Pereira & Warren, 1980), lexical functional grammars (Kaplan & Bresnan, 1982), functional unification grammars (Kay, 1982), tree-adjoining grammars[3] (Joshi, 1985), or generalized phrase structure grammars (Gazdar, Klein, Pullum, & Sag, 1982). The manual construction of broad-coverage grammars turned out to be much more difficult than expected, if not impossible. In his *Machine Translation 'forum'* lead paper, Somers (1993) reflects on the relatively modest advancement in natural language processing in the 70's and 80's by asking "What's wrong with classical second generation[4] architecture?".

1.1.2 Empirical Approaches

In recent years, there has been a trend towards more empirical approaches that augment or replace the hand-coded rule paradigm with statistical and machine learning techniques. These techniques automatically derive parse grammars and other classification structures from examples.

[2]An *anaphor* is an expression which cannot have independent reference, but refers to another expression, the so-called *antecedent* (Radford, 1988). Examples of anaphora in sentences 13&14 are the pronouns "which" and "it" (both occurrences), which refer to the antecents "the statue of liberty", "the airline", and "a loss". The process of finding the proper antecedent of an anaphor is called *anaphora resolution*.

[3]Tree-adjoining grammars are based on hand-coded trees instead of rules.

[4]By *second generation architecture* Somers refers to systems which incorporate most of the typical design features such as "linguistic rule-writing formalisms with software implemented independently of the linguistic procedures, stratificational analysis [in particular morphology - surface syntax - deep syntax] and generation, and an intermediate linguistically motivated representation which may or may not involve the direct application of contrastive linguistic knowledge."

A number of researchers have already applied machine learning techniques to various NLP tasks like accent restoration by Yarowsky (1994), who achieves 99% accuracy, relative pronoun disambiguation (Cardie, 1992, 1993, 1996), (Japanese) anaphora resolution (Aone & Bennett, 1995), part of speech tagging (Weischedel & al., 1993; Brill, 1995; Daelemans, Zavrel, Berck, & Gillis, 1996), where tagging accuracies between 96% and 97% are achieved, cue phrase classification (Siegel & McKeown, 1994; Litman, 1996), and word sense disambiguation (Yarowsky, 1992, 1995; Ng & Lee, 1996; Mooney, 1996).

Many of the statistical approaches have been made possible by the increased availability of large corpora of natural language data like the Brown corpus (Francis & Kucera, 1982), which consists of about a million words, all labeled with their parts of speech, or the 4.5 million word Penn Treebank (Marcus, Santorini, & Marcinkiewicz, 1993), in which more than half of the sentences have been annotated for their skeletal syntactic structure.

Newer treebank-based probabilistic approaches (Magerman, 1995; Collins, 1996) have produced encouraging results. It is however not clear how these systems can still be improved significantly, because (1) using about 40,000 training sentences, they have already exhausted the large Penn Treebank corpus and (2) they still use a fairly limited context and sharply restrict the amount of background knowledge.

To cope with the complexity of unrestricted text, parse rules in any kind of formalism will have to consider a complex context with many different morphological, syntactic, semantic and possibly other features to make good parsing decisions. This can present a significant problem, because even linguistically trained natural language developers have great difficulties writing and even more so extending explicit parse grammars covering a wide range of natural language. On the other hand it is much easier for humans to decide how *specific* sentences should be analyzed and what feature might discriminate a pair of *specific* parse states.

1.2 A Context-Oriented Machine Learning Approach

We therefore propose an approach to parsing based on learning from examples with a very strong emphasis on context, integrating morphological, syntactic, semantic and other aspects relevant to making good parse decisions, thereby also allowing the parsing to be deterministic. In order to reduce the complexity of the learning task, we break the parsing process into a sequence of smaller steps, so-called *parse actions*. This reduces parsing to a decision problem, where the system has to learn which parse action to perform next. As we have shown at the beginning of this introduction, these decisions can be quite difficult. We therefore provide (1) a rich context, encoded by currently 205 morphological, syntactic and semantic features and (2) some background knowledge, in particular dictionaries, a concept hierarchy, and a subcategorization table.

As our parsing model we choose a deterministic shift-reduce type parser, which not only makes parsing very efficient, but also assures transparency during the example acquisition, when parse action examples are collected interactively: the partially trained parser proposes the next parse action and a human supervisor confirms or overwrites the proposal. Compared to the original shift-reduce parser (Marcus, 1980), our parser includes additional types of operations and allows additional operation parameters, so that the parser can produce a deeper analysis of the sentence. The parse tree integrates semantic information, phrase-structure and case-frames. Not only does this lead to a final parse tree that is powerful enough to be fed into a transfer and a generation module to complete the full process of machine translation, it also provides much richer intermediate results, which are used to improve the quality of parsing decisions.

Applying machine learning techniques, the system uses the acquired parse action examples to generate a parse grammar in the form of an advanced decision structure, an extension of decision trees (Quinlan, 1987, 1993). Such a parse grammar can be used to parse previously unseen sentences into parse trees. Based on such parse trees, our system translates the sentences, re-using the empirical methods developed for parsing with only minor adaptations.

Following this corpus based approach, we relieve the NL-developer from the hard if not impossible task of writing explicit grammar rules and make grammar coverage increases very manageable. Compared with recently published probabilistic methods, our system relies on some background knowledge, a rich context, a deeper analysis and more supervision per sentence, but radically fewer examples, currently 256[5], as opposed to 40,000 sentences for the treebank-based approaches cited above.

1.3 Summary of Experimental Results

We tested our parser on lexically limited sentences from the Wall Street Journal and achieved accuracy rates of 89.8% for labeled precision[6], 98.4% for part of speech tagging and 56.3% of test sentences without any crossing brackets. Our accuracy results are about as good as for competing probabilistic systems trained on 40,000 sentences. Based on our learning curve, we expect that, when increasing the number of our training sentences from 256, but still staying far below the 40,000 training sentences used by probabilistic systems, our results will be significantly better than for those treebank-based probabilistic system.

Machine translations of 32 randomly selected Wall Street Journal sentences from English to German have been evaluated by 10 bilingual volunteers and been graded as 2.4 on a 1.0 (best) to 6.0 (worst) scale for both grammatical correctness and meaning preservation. Our system has been trained and tested on the same lexically limited corpus, but despite this advantage we believe that it is quite an achievement that after only very limited training, our system already produces better translations than all three competing commercial systems. The translation quality was only minimally better (2.2) when starting each translation with the correct parse tree, indicating that the parser is quite robust and that its errors have only a moderate impact on final translation quality.

1.4 Organization of Dissertation

Chapter 2 presents the task definition and gives an overview of the architecture of the system and its major components. Chapter 3 describes the background knowledge we use, namely the knowledge base, the monolingual lexicons, the bilingual dictionary, the subcategorization table, and the so-called morphological pipelines. Chapter 4 briefly explains the two pre-parsing modules, segmentation, and morphological analysis. The key chapter 5 on parsing explains our parsing paradigm

[5]Note however that these 256 sentences contain some 11,000 individual parse action examples.

[6]See chapter 6.2 for exact definitions of parsing metrics.

and presents the details of parse actions, features, the training of the parser and the decision structures we use for learning. Chapter 6 evaluates the parser, including a series of ‘ablation’ tests to investigate the practical contributions of various system components. Chapter 7 describes how the techniques used for parsing can be applied to the parse tree transfer to the target language. Chapter 8 explains how that target language tree is reordered and morphologically propagated in order to compute the surface forms of words and phrases in the target language. Chapter 9 reports the results of our translation experiments, including comparisons to three commercial systems. Chapter 10 describes related work and chapter 11 discusses several potential future extensions of our system, before chapter 12 concludes.

Chapter 2

System Architecture

2.1 Task Definition

Our goal is to parse and translate natural language. In this dissertation, we investigate how machine learning techniques can be used to first parse input text into a parse tree and subsequently, based on this parse tree, translate the text into another language.

To develop a parser for English and a translator from English to German, we use training sentences from the Wall Street Journal. We evaluate the parser by testing it on sentences from the Wall Street Journal that were not used for training. We use standard evaluation criteria for parsing, including precision, recall and violations of constituent boundaries ("crossing brackets"). The German translations are graded by outside bilingual evaluators.

2.1.1 Corpus

The WSJ corpus used in our work is a subset of sentences from Wall Street Journal articles from 1987, as provided on the ACL data-disc. In order to limit the size of the required lexicon, we work on a reduced corpus that includes all those sentences that are fully covered by the 3000 most frequently occurring words (ignoring numbers etc.) in the entire corpus. The lexically reduced corpus contains 105,356 sentences, a tenth of the full corpus.

For our training and testing we use the first 272 sentences from this lexically restricted corpus. They vary in length from 4 to 45 words, averaging at 17.1 words. We believe that it is important to use sentences from the "real world" instead of artificially generated sentences in order to capture the full complexity of natural language. A complete listing of the 272 training and testing sentences can be found in appendix A and a more detailed discussion of the testing and training corpus is presented in section 6.1 of the chapter on "Parsing Experiments".

2.2 Transfer-Based vs. Interlingua Approach

The overall system architecture follows the classical subdivision of parsing, transfer and generation, thereby rejecting a pure interlingua approach (see figure 2.1). As the authors and interlingua proponents of (Nirenburg, Carbonell, Tomita, & Goodman, 1992) point out, the major distinction between the interlingua- and transfer-based systems is the "attitude toward comprehensive analysis of meaning". While the pure interlingua approach might be quite appealing academically, we believe that the transfer approach is easier and more practical to handle and can demonstrate the main paradigm and ideas introduced by our system equally well.

An interlingua is a formal language which has to represent the meaning of source text fully and unambiguously. The task to extract all information contained in natural language text and to resolve all shades of ambiguity is so complex and would require such an amount of world knowledge that it would overburden any system development.

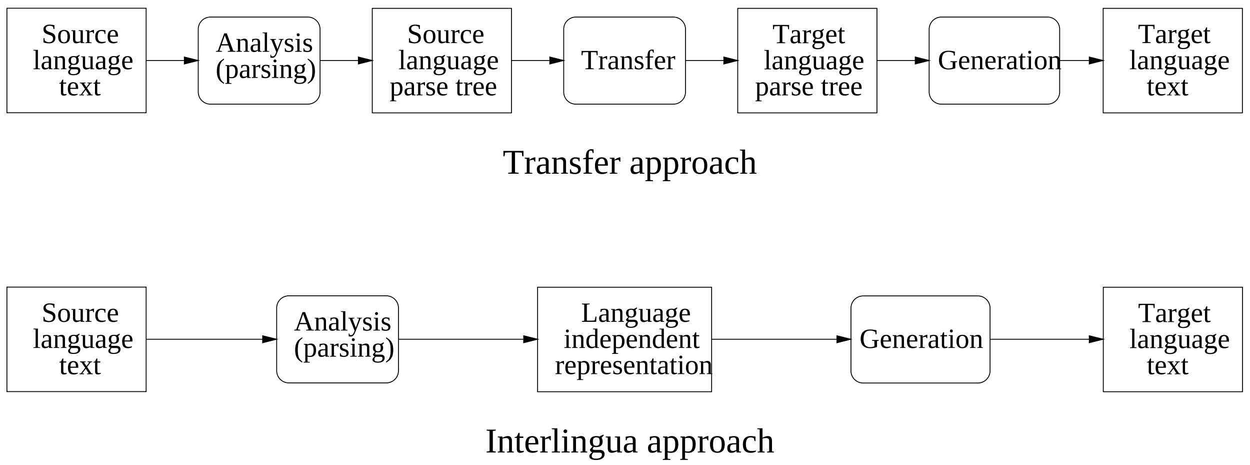


Figure 2.1: The transfer and interlingua approaches

For specific tasks such as machine translation between particular languages, a lot of this extracted and disambiguated information will be irrelevant, thus wasting both computing resources at runtime and human resources at development time.

On the other hand, classical transfer systems, which during parse time analyze the source text only up to some syntactic level and avoid a semantic analysis, will often not only face insurmountable difficulties in selecting the correct target concepts during the transfer phase, but also experience significant control problems during the parse phase itself, e.g. when making decisions concerning preposition phrase attachment. We therefore include a "shallow" semantic level during parsing, thereby substantially incorporating a major characteristic of the interlingua approach.

Another argument in favor of interlingua is that it requires only one analysis and one generation module per language involved, thus only $2 * n$ modules for n languages, as opposed to an additional $n * (n - 1)$ transfer modules. This argument could be refuted by finding a way to largely automatically construct a transfer module from language L_A to L_C from existing modules for transfer from L_A to L_B and from L_B to L_C. It appears that this would be quite feasible to do given the way our system is set up. This last argument is nevertheless somewhat moot as long as machine translation is not mature enough to perform well with a single language pair.

2.3 Major System Components

Figure 2.2 gives an overview of the system architecture. Boxes represent data, ovals represent programs and arrows indicate the flow of information. In the case of solid arrows, the entire data

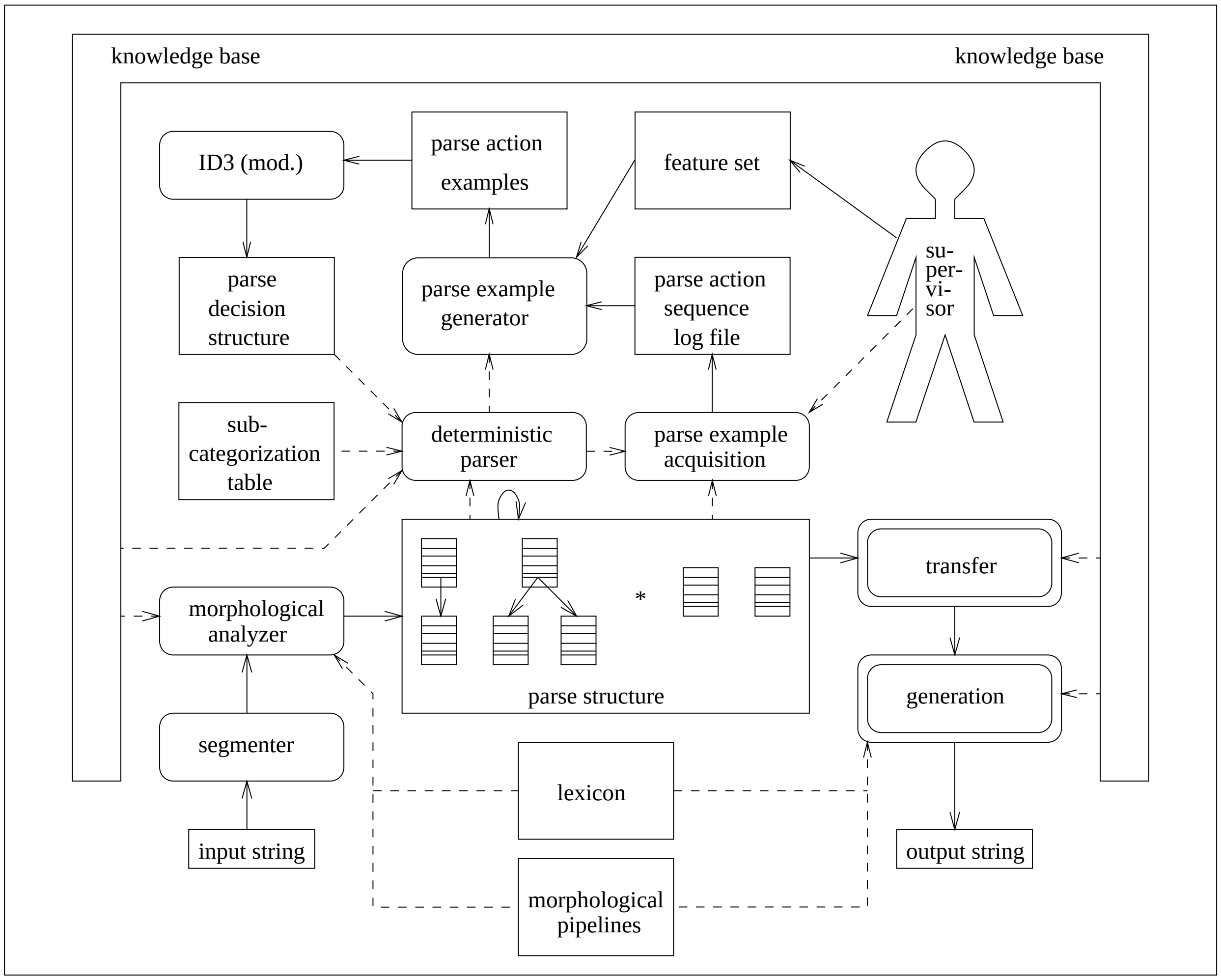


Figure 2.2: System Architecture Overview with Focus on Parsing

structure is passed on, whereas in the case of the dashed arrows, only the information requested by the recipient is passed on.

Before the parsing itself starts, the input string is segmented into a list of words including punctuation marks, which then are sent through a morphological analyzer. The parsing itself is broken into an ordered sequence of small and manageable parse actions. Using the output of the morphological analyzer as an initial parse state, the *deterministic parser* applies a sequence of *parse actions* to the parse structure until it decides that it is done. The parse structure is composed of a *parse stack* and an *input list*. The asterisk in the parse structure in figure 2.2 marks the boundary between the parse stack (left) and the input list (right) and can be interpreted as the 'current position' in a partially parsed sentence. The two most typical parse actions, *shift*, which moves an element from the input list to the parse stack, and *reduce*, which combines one or more elements on the parse stack into a more complex element, give the shift-reduce parser its name. The input list is initialized with the output of the morphological analyzer and the result of the parsing is the typically single but complex element that is left on the parse stack. The resulting parse tree is then sent through a transfer module that maps it to an equivalent tree in the target language and a generation module that reorders the tree and generates the proper surface words and phrases in the target language.

The key data structure that the deterministic parser uses to decide what the next parse action should be is the *parse decision structure*. This structure, a hybrid extension of decision trees and lists, is learnt from parse action examples. Parse action examples are collected by interactively running training sentences through the parser. Initially, when there are no parse action examples and hence no parse decision structure yet, a human supervisor provides the full and correct sequence of parse actions for a sentence. These are collected in a *log file*, which, together with the feature set, is used to construct a set of full parse action examples. The feature set is provided by the supervisor and describes the context of a parse state. The same feature set is used for all parse action examples. For each parse step, the parse action example generator makes the parse engine of the deterministic parser automatically determine the values for all features in the feature set and then complements the feature vector with the correct parse action from the log file as the classification of the parse action example. The resulting parse action examples are then used by a machine learning program, an extension of the classic decision tree constructor ID3 (Quinlan, 1987), to build the parse decision structure. Starting with the second sentence, this parse decision structure can be used by the deterministic parser to propose the next parse action to the supervisor. As the number of parse action examples grows and the parse decision structure becomes more and more refined, the supervisor has to correct fewer and fewer of the proposed parse actions.

Note that we directly use the proper parse action sequence to train our parser, whereas in most empirical methods, the example input consists of the final parse trees only. By providing examples with actual parse actions to the machine learning unit, our system provides more information and a more direct and therefore better guidance. The various types of background knowledge that support the parsing process (lexicons, morphological pipelines, knowledge base and subcategorization table) are described in the chapter 3.

2.3.1 Features

To make good parse decisions, a rich context in the form of a wide range of features at various degrees of abstraction have to be considered. To express such a wide range of features, we defined a *feature language*. The following examples, for easier understanding rendered in English and not in feature language syntax, further illustrate the expressiveness of the feature language:

- the general syntactic class of $frame_{-1}$ (the top frame of the parse stack[1]): e.g. verb, adj, np,
- whether or not $frame_{-1}$ could be a nominal degree adverb,
- the semantic role of $frame_{-1}$ with respect to $frame_{-2}$: e.g. agent, time; this involves pattern matching with corresponding entries in the verb subcategorization table,
- whether or not $frame_{-2}$ and $frame_{-1}$ agree as np and vp.

The feature collection is basically independent from the supervised parse action acquisition. Before learning a decision structure for the first time, the supervisor has to provide an initial set of features that can be considered obviously relevant. Later, the feature set can be extended

[1]The parse stack and the input list form the parse structure. For more details, see section 5.2.

as needed. All concepts and methods introduced in this overview are described in detail in the following chapters, notably features in section 5.6.

Chapter 3

Background Knowledge

In our approach to natural language processing, learning is complemented by some prespecified knowledge. This knowledge can be accessed both indirectly through the use of features when learning and using decision structures for parsing and transfer, as well as directly, as happens for example with the morphological knowledge during generation. The background knowledge is organized in a general *knowledge base (KB)*, *monolingual lexicons*, *a bilingual dictionary*, *subcategorization tables* and what we call *morphological pipelines*.

Much of the background knowledge is used at various stages of the translation process, e.g. the general KB at virtually all stages and the morphological pipelines for both pre-parsing analysis and generation, while other knowledge is used exclusively in a single phase, as e.g. the bilingual dictionary in transfer.

Before describing in further detail *what* background knowledge we use, we first want to give reasons *why* we use it. In chapter 1 we have already explained why we use machine learning in making parse and transfer decisions: essentially because the lexical and structural ambiguity of unrestricted natural language is so complex that hand-crafted approaches as tried in the past have failed. Why then not learn everything, and do without any background knowledge? Statistical approaches such as SPATTER (Magerman, 1995) or the Bigram Lexical Dependency based parser (Collins, 1996) produce parsers on very limited linguistic background information and Inductive Logic Programming systems such as CHILL (Zelle & Mooney, 1994; Zelle, 1995) have even generated linguistically relevant categories such as *animate*.

In short, the background knowledge we use is at least qualitatively easy to provide and mostly already conceptually available in the form of traditional (paper) dictionaries and grammar books; using available background knowledge, we can let the learning focus better on core decision problems in parsing and transfer, on decisions that are known to be extremely hard if not impossible to make using hand-written rules. With a better focus, training sizes can be smaller, because less has to be learnt; no wheels need to be reinvented.

The nature of the background knowledge we use is much simpler than what we try to learn, because it is what we call *micromodular*. We use the term *micromodular* for knowledge that is composed of very small pieces that are very independent of each other, e.g. single entries in a monolingual lexicon, a bilingual dictionary, or a subcategorization table. These entries are without any claim of completeness: the nominal lexical entry for *high* for example only asserts that there is a concept which can manifest itself as a noun with the stem form *high*, but does *not* preclude any other interpretations for *high* or indicate which interpretation might be most appropriate in a given context. Similarly, subcategorization table entries and other elementary assertions make no claim about the existence or preferences of potentially competing entries. This micromodular nature of the background knowledge makes it easy for humans to express it. The acquisition of background knowledge is therefore mostly a quantitative problem, calling for tools and methods that allow rapid information acquisition. In contrast to this micromodular background knowledge, the knowledge encompassing parse grammars is qualitatively very complex; it is this knowledge that has to disambiguate natural language text with all its regularities, sub-regularities, pockets of exceptions, and idiosyncratic exceptions.

The following sections describe the different types of background knowledge in detail. Section 3.3 shows how additions to the KB and lexicon are supported.

3.1 Knowledge Base

The *knowledge base (KB)* consists of concepts linked by relations. The vast majority of its currently 4356 concepts are semantic or syntactic.

Most semantic concepts represent a specific word in a specific language. A separate notion of a concept is necessary, because words are too ambiguous. Not only can words have different meanings in different languages, e.g. *gift* can mean *poison* in Swedish; some words, called homonyms, have unrelated meanings, e.g. *go*, a common verb and a Japanese board game, or have related meanings, but differ in their syntactic function, in which case the word is sometimes also referred to as a homomorph, as e.g. for *increase* which can be both a verb and a noun. Finally, concepts allow to distinguish between different shades of meanings of a word.

Even if two words from the same or different language basically share the same meaning, they are still represented by different concepts, because their meanings typically never match perfectly due to possibly ever so slight different connotations.

Semantic concepts are typically represented by identifiers of the form I-<language-tag><part-of-speech-tag>-<specific-tag>, e.g. I-EN-PILOT, I-EN-TANGIBLE-OBJECT or I-GADJ-DEUTSCH, where the 'I' stands for *internal concept*, the 'E' for *English*, the 'G' for *German*, the 'N' for *noun*, the 'ADJ' for *adjective*. Other semantic concepts include semantic roles, e.g. R-AGENT or R-TO-LOCATION.

Syntactic concepts include parts of speech, prefixed by "S-", e.g. S-VERB or S-TR-VERB, for verbs and, more specifically, transitive verbs respectively. Other syntactic concepts include forms, e.g. F-NUMBER, F-PLURAL, F-TENSE, F-FINITE-TENSE, F-PAST-TENSE and syntactic roles such as R-SUBJ.

In its current form, there is only one relation in the KB, the binary *is-a*. 4518 such *is-a* links connect the 4356 concepts to form an acyclic graph. More than 95% of the network is semantics, the rest syntax (90 links), forms (to express concepts for case, tense etc.; 52 links), roles (46 links) and miscellaneous (10 links).

While specific words are represented by one or more concepts specifically tied to the language the word is from, many generalizations such as I-EN-TANGIBLE-OBJECT are shared as superconcepts for concepts representing words from different languages.

More than 90% of the KB concepts are leaves that lexical entries refer to. The granularity

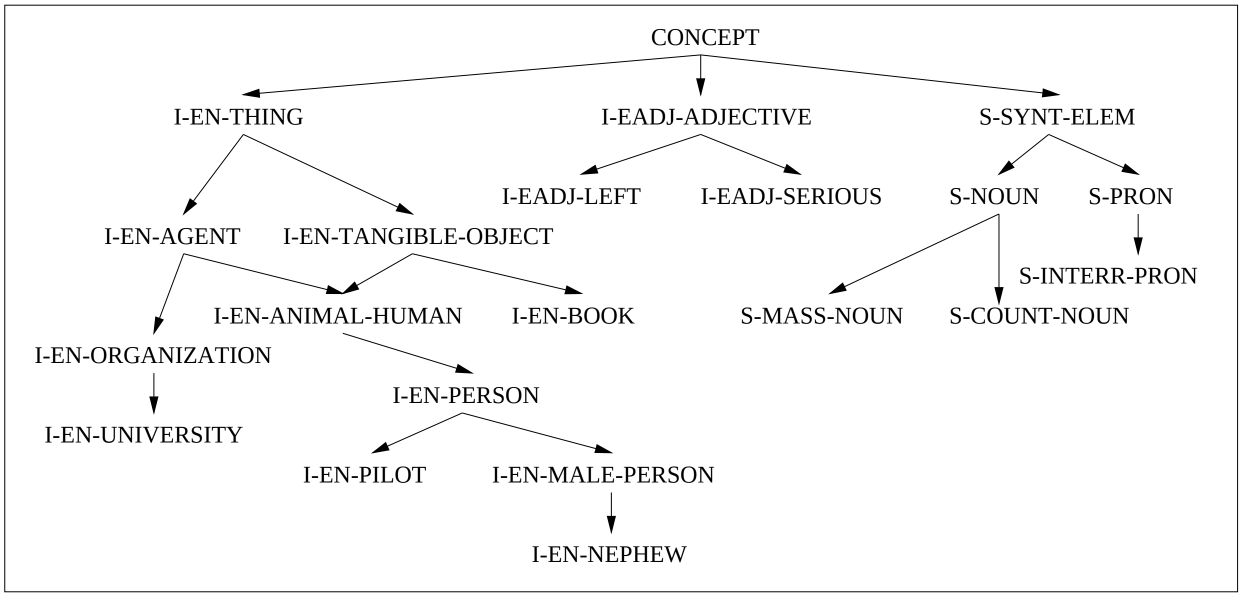


Figure 3.1: A subsection of the knowledge base

of the KB is basically application driven. Concepts can for example be grammatically relevant because they can help determine

- the need for an article (*S-COUNT-NOUN*)
- the proper antecedent based on natural gender (*I-EN-MALE-PERSON*)
- the semantic roles a constituent can fill (*I-EN-AGENT*).

As the direct superconcepts of I-EN-BOOK and I-EADJ-LEFT in figure 3.1 already suggest, a very coarse classification is often sufficient. A distinction between mammals and reptiles for example would hardly help in parsing and machine translation. We don't even have a concept for *noninherent* adjectives, a class of adjectives that typically can not be used in predicative position (as in *"The president is *former*."). While this could be a useful concept for a grammar checker, it probably doesn't serve a purpose in machine translation, where the source sentences are expected to be (more or less) correct and where it is probably best to conserve any semantic oddity in the translation.

3.2 Monolingual Lexicons

A lexicon is a set of lexical entries, which link surface words to semantic concepts as used in the KB. Besides the surface word and a corresponding concept, the lexicon entry will always include the part of speech for the word. Optional attributes include:

- irregular forms
- *value*, the numerical value for numerals
- the predicates *is-short-form* and *is-abbreviation*

- *grade* for superlative and comparative forms of adjectives
- *prob-rank* a relative rank of a priori likelihood for ambiguous words (see explanation below)
- *dummy-index, class, restriction* and *forms* for dummy words (see explanation below)
- *components* for contracted words (e.g. German "im" = "in dem")
- *connective-form* for German noun compounds *(as explained later)*
- *gender, genetive-type, plural-type* for German nouns
- *separable-prefix, has-inseparable-prefix, is-strong-verb, perfect-auxiliary* for German verbs

The attribute *prob-rank* affects the order of morphological analyses of words with ambiguous part of speech at the beginning of parsing, after morphological processing. Based on the following excerpt from the English lexicon, the English word "man" will be recognized as both a noun and a verb. With a default *prob-rank* of 1, the nominal analysis will be placed at the first position of the list of alternatives, and the verbal analysis at place two, since it has a *prob-rank* value of 2, indicating that it is a priori less likely.

The English lexicon currently contains 3015 entries, fully covering the restricted WSJ vocabulary. The German stands at 1039 entries, sufficient to cover all words that could occur in our translation training and test sentences.

From the English lexicon:

```
((LEX "man") (CONCEPT I-EN-MAN) (SYNT S-COUNT-NOUN)
  (IRR-FORMS ("men" (NUMBER F-PLURAL))))
((LEX "man") (CONCEPT I-EV-MAN) (SYNT S-TR-VERB)
  (PROPS (PROB-RANK 2)))
((LEX "Sen") (CONCEPT I-EN-SENATOR-TITLE) (SYNT S-COUNT-NOUN)
  (PROPS (IS-ABBREVIATION TRUE)))
((LEX "sixty") (CONCEPT I-ENUM-SIXTY) (SYNT S-CARDINAL)
  (PROPS (VALUE 60)))
((LEX "be") (CONCEPT I-EV-BE) (SYNT S-AUX)
  (IRR-FORMS
    ("am" (TENSE F-PRES-TENSE) (PERSON F-FIRST-P) (NUMBER F-SING))
    ("are" (TENSE F-PRES-TENSE) (PERSON F-SECOND-P) (NUMBER F-SING))
    ("is" (TENSE F-PRES-TENSE) (PERSON F-THIRD-P) (NUMBER F-SING))
    ("are" (TENSE F-PRES-TENSE) (NUMBER F-PLURAL))
    ("was" (TENSE F-PAST-TENSE) (PERSON F-FIRST-P) (NUMBER F-SING))
    ("were" (TENSE F-PAST-TENSE) (PERSON F-SECOND-P) (NUMBER F-SING))
    ("was" (TENSE F-PAST-TENSE) (PERSON F-THIRD-P) (NUMBER F-SING))
    ("were" (TENSE F-PAST-TENSE) (NUMBER F-PLURAL))
    ("been" (TENSE F-PAST-PART))
    ("being" (TENSE F-PRES-PART))))
```

```
((LEX "'be") (CONCEPT I-EV-BE) (SYNT S-AUX)  ;;; contracted form
  (IRR-FORMS
    ("'m" (TENSE F-PRES-TENSE) (PERSON F-FIRST-P) (NUMBER F-SING))
    ("'re" (TENSE F-PRES-TENSE) (PERSON F-SECOND-P) (NUMBER F-SING))
    ("'s" (TENSE F-PRES-TENSE) (PERSON F-THIRD-P) (NUMBER F-SING))
    ("'re" (TENSE F-PRES-TENSE) (NUMBER F-PLURAL))
    (NO-FORM (TENSE F-PRES-INF))
    (NO-FORM (TENSE F-PAST-TENSE))
    (NO-FORM (TENSE F-PAST-PART))
    (NO-FORM (TENSE F-PRES-PART)))
  (PROPS (IS-SHORT-FORM TRUE)))
((LEX "better") (CONCEPT I-EADJ-GOOD) (SYNT S-ADJ)
  (PROPS (GRADE COMPARATIVE)))
((LEX "SOMEBODY_1") (SYNT S-NP)
  (PROPS (DUMMY-INDEX 1) (CLASS I-EN-PERSON)))
```

The lexicon does not only contain normal word entries, but also *dummies*. These words will not occur in any normal text to be processed, but only in the bilingual dictionaries, where they mark the place for words or phrases. A typical English component of such a dictionary entry is the verb phrase **"to get SOMEBODY_1 TO_DO_SOMETHING_2"**. The English lexicon describes SOMEBODY_1 as a dummy for a noun phrase semantically restricted to be of class I-EN-PERSON. The possible restriction attributes *forms* and *restriction* offer alternative ways to indicate restrictions by respectively providing limitations on forms, e.g. tense or number, or through the reference to the name of a hard-coded predicate.

From the German lexicon

```
((LEX "Mann") (CONCEPT I-GN-MANN) (SYNT S-COUNT-NOUN)
  (PROPS (PLURAL-TYPE F-INFL+ER)
         (GENETIVE-TYPE F-INFL-ES)
         (GENDER F-MASC)))
((LEX "abbestellen") (CONCEPT I-GV-ABBESTELLEN) (SYNT S-TR-VERB)
  (PROPS (IS-STRONG-VERB :FALSE)
         (SEPARABLE-PREFIX "ab")
         (HAS-INSEPARABLE-PREFIX :TRUE)
         (HAS-SEPARABLE-PREFIX :TRUE)))
((LEX "im") (CONCEPT I-GP-IM) (SYNT S-PREP-PLUS-ART)
  (PROPS (COMPONENTS ("in" "dem"))
         (HAS-COMPONENTS TRUE)))
((LEX "Westen") (CONCEPT I-GN-WESTEN) (SYNT S-COUNT-NOUN)
  (PROPS (GENETIVE-TYPE F-INFL-S)
         (PLURAL-TYPE F-INFL-)
         (GENDER F-MASC)
         (CONNECTIVE-FORM "West")))
```

The identifiers for the genetive and plural types ("F-INFL...") already indicate the way in which the respective forms are constructed. The sign following F-INFL, + or -, indicates whether or not the form requires an umlaut and the rest of the identifier represents the respective ending. In the case of *Mann* (which means *man*), the (singular) genetive form is *Mannes* and the plural (nominative) form is *Männer*. Other forms can be constructed in a regular way from these forms. All German nouns possess a grammatical gender (masculine, feminine or neuter).
German nouns can take on a special connective forms in noun compounds. The head noun (at the last position) remains in the usual form with the appropriate number and case, whereas modifying nouns appear in their connective forms, typically not separated by any spaces, e.g. *Westafrika*.

3.3 Acquisition of Lexicon and KB

The entries in the KB and the lexicons are basically entered manually, facilitated by some useful tools though. So, to get the above entries for "Mann" and "abbestellen" (*to cancel*), along with some others, one can just enter
Aquisition example 1:

```
e "der Mann -es +er" nc spec I-EN-MALE-PERSON "der Junge -n -n"
  "der Bruder -s +"
```

Junge: "boy"; *Bruder*: "brother"; *nc* stands for *count noun*, *-es* etc. for genetive and plural endings
This entry command produces the following system feedback:

```
To lex:
   ((LEX "Mann") (CONCEPT I-GN-MANN) (SYNT S-COUNT-NOUN)
                 (GENDER F-MASC) (PLURAL-TYPE F-INFL+ER)
                 (GENETIVE-TYPE F-INFL-ES))
   ((LEX "Junge") (CONCEPT I-GN-JUNGE) (SYNT S-COUNT-NOUN)
                  (GENDER F-MASC) (PLURAL-TYPE F-INFL-N)
                  (GENETIVE-TYPE F-INFL-N))
   ((LEX "Bruder") (CONCEPT I-GN-BRUDER) (SYNT S-COUNT-NOUN)
                   (GENDER F-MASC) (PLURAL-TYPE F-INFL+)
                   (GENETIVE-TYPE F-INFL-S))
To KB:
   (I-GN-MANN IS-A I-EN-MALE-PERSON)
   (I-GN-JUNGE IS-A I-EN-MALE-PERSON)
   (I-GN-BRUDER IS-A I-EN-MALE-PERSON)
OK [y n l k]?
```

For German verbs, the double dash as used in the following example is not part of the word, but indicates that the preceding prefix is separable; similarly, a single dash indicates an inseparable prefix.

Aquisition example 2:

```
e "ab--be-stellen" vt spec I-EV-PROCESS "ab--stellen"
  "be-stellen" "stellen"
```

This entry command produces the following system feedback:

```
To lex:
   ((LEX "abbestellen")
       (CONCEPT I-GV-ABBESTELLEN) (SYNT S-TR-VERB)
       (SEPARABLE-PREFIX "ab") (HAS-SEPARABLE-PREFIX TRUE)
       (HAS-INSEPARABLE-PREFIX TRUE) (IS-STRONG-VERB FALSE))
   ((LEX "abstellen")
       (CONCEPT I-GV-ABSTELLEN) (SYNT S-TR-VERB)
       (SEPARABLE-PREFIX "ab") (HAS-SEPARABLE-PREFIX TRUE)
       (HAS-INSEPARABLE-PREFIX FALSE) (IS-STRONG-VERB FALSE))
   ((LEX "bestellen")
       (CONCEPT I-GV-BESTELLEN) (SYNT S-TR-VERB)
       (HAS-SEPARABLE-PREFIX FALSE) (HAS-INSEPARABLE-PREFIX TRUE)
       (IS-STRONG-VERB FALSE))
   ((LEX "stellen")
       (CONCEPT I-GV-STELLEN) (SYNT S-TR-VERB)
       (HAS-SEPARABLE-PREFIX FALSE) (HAS-INSEPARABLE-PREFIX FALSE)
       (IS-STRONG-VERB FALSE))
To KB:
   (I-GV-ABSTELLEN IS-A I-EV-PROCESS)
   (I-GV-ABBESTELLEN IS-A I-EV-PROCESS)
   (I-GV-BESTELLEN IS-A I-EV-PROCESS)
   (I-GV-STELLEN IS-A I-EV-PROCESS)
OK [y n l k]?
```

Entries for English verbs and nouns are simpler because English nouns don't have a grammatical gender or case and the English equivalent of a prefix is always separate (e.g. "take off"). There are special tables for irregular nouns and verbs. The irregular forms contained in these tables are automatically added to the lexical entries.

The part of speech of a word, as well as applicable information about grammatical gender, inflectional class, prefixes etc., are commonly listed in traditional (paper) dictionaries, but will typically be obvious to the educated speaker of a language.

Section 11.1.2 in the chapter on future work sketches how the lexical acquisition process could be further automated.

3.4 Bilingual Dictionary

A *bilingual dictionary* links words and expressions between different languages and is used for translation. The *surface* dictionary is a listing of words and phrases very similar to as one would find them in a traditional (paper) dictionary. Its format was designed to be very intuitive and user-friendly so that additions (or changes) can be made very easily. The following samples from the English-German transfer lexicon is presented in a table for better legibility and closely follows the actual format of the surface dictionary file, which, for the first table entry would be *("be" S-VERB "sein")*.

"be" "sein"	S-VERB
"to be dissatisfied with SOMETHING_1" "mit ETWAS_1 unzufrieden sein"	S-VP
"to be lucky" "Glück haben"	S-VP
"because" "weil"	S-CONJ
"book" "Buch"	S-NOUN
"downtown PLACENAME_1" "im Stadtzentrum von ORTSNAME_1"	S-ADV S-PP
"know" "kennen"	S-VERB
"know" "wissen"	S-VERB
"to make SOMETHING_1 equal to SOMETHING_2" "ETWAS_ACC_1 auf ETWAS_ACC_2 bringen"	S-VP
"primarily" "in erster Linie"	S-ADV S-PP
"some" "einige"	S-ADJ
"some" nil	S-ADJ
"up QUANTITY_1" "ein Plus von QUANTITAET_1 "	S-PARTICLE-PHRASE S-NP
"it takes SOMEBODY_3 SOMETHING_1 TO_DO_SOMETHING_2" "JEMAND_3 braucht ETWAS_ACC_1, um ETWAS_ZU_MACHEN_2"	S-CLAUSE
I-EART-INDEF-ART I-GART-INDEF-ART	
I-EN-PERSONAL-PRONOUN I-GN-PERSONAL-PRONOUN	
I-EPRT-'S I-GP-GEN-CASE-MARKER	

A typical dictionary entry consists of a pair of words or phrases along with part of speech restrictions. Note that the parts of speech don't necessarily have to be the same (e.g. "primarily"/"in erster Linie").

Phrases can contain variables, e.g. *SOMETHING_1*, *PLACENAME_1*, or *ETWAS_ZU_MACHEN_2*. Variables in the source and target phrase are linked by a common index. They have special entries in their respective monolingual lexicons, which list any syntactic, semantic or form restrictions. The German lexicon for example restricts *JEMAND_3* to be a noun phrase (SYNT S-NP), a person (CLASS I-EN-PERSON), and nominative case (CASE F-NOM).

The dictionary can also contain a pair of concepts, e.g. I-EART-INDEF-ART/I-GART-INDEF-ART. There are only relatively few such direct concept pair entries, but these entries allows the pairing of concepts whose level of abstraction can not easily be captured by specific words.

Note that a word or phrase can have multiple translations (e.g. "know" $\leftrightarrow$ "kennen"/"wissen") or an empty one (e.g. "some" $\leftrightarrow$ "eingige"/*nil*). (Chapter 7 will describe how the system learns to make the proper choice.)

When a *surface* dictionary is loaded, it is compiled into an *internal* dictionary. All strings are parsed (in their respective languages) and mapped to a KB concept or a complex parse tree. So the internal dictionary does not map from surface word or phrase to surface word or phrase, but from parse tree to parse tree.

The transformation of transfer entries from surface strings to their parse trees can be accomplished by using the **same** parsers that are used for the parsing of normal input surface sentences. Of course parsers for both the source and the target language are necessary, but since the phrases in the lexicon are not anywhere as complex and ambiguous as normal input surface sentences, the 'parsing power' requirements for the bilingual dictionary are relatively small, so that fairly few parse examples are necessary for a language that is only used as a machine translation target language.

The intuitive surface representation, which closely follows the format of good traditional dictionaries, enormously facilitates the building of the transfer lexicon; it is very transparent, can be built, extended and checked easily, and is not very susceptible to errors.

3.5 Subcategorization Tables

Consider the following two sentences:

1. I booked a flight to New York.

2. I sent a letter to New York.

Superficially, both sentences look very similar, but in sentence 1, the prepositional phrase *to New York* has to attach to the preceding noun phrase *a flight*, whereas in sentence 2, it belongs directly to the predicate *sent*. When deciding where to attach the prepositional phrase, an analysis of the parts of speech will not suffice, because it will not discriminate between the two sentences.

The fundamental difference lies in the different argument structure of the verbs and nouns. While both take a direct object, *to send* also allows an indirect object or a *to-location* argument. Additionally, the noun *flight*, a nominal form of *to fly*, allows a *to-location* argument.

This argument information can be represented in *subcategorization tables*. The following is a subset of the currently 242 entries in the **English Verb Patterns**. The asterisk (*) indicates that the following argument is optional.

```
("to book SOMETHING")

("to decline *I-EN-QUANTITY *C-TO-QUANT"
    (SUBJ = THEME) (OBJ = DIFF-QUANT))
("to decline by I-EN-QUANTITY *C-TO-QUANT"
    (SUBJ = THEME) (APP = DIFF-QUANT))
("to decline TO-DO-SOMETHING"
    (INF-COMPL = THEME))

("to get SOMETHING"
    (SUBJ = BEN))
("to get SOMEBODY TO-DO-SOMETHING"
    (OBJ = BEN) (INF-COMPL = THEME))
("to get out of SOMETHING"
    (APP = THEME))
("to get through SOMETHING"
    (SUBJ = THEME) (APP = PRED-COMPL))
("to get under way"
    (SUBJ = THEME) (APP = PRED-COMPL))

("to send *SOMEBODY SOMETHING")
("to send SOMETHING C-TO-LOCATION")
```

Each subcategorization table entry consists of a phrase, e.g. a verb phrase in a verb subcategorization table, plus a mapping of syntactic to semantic roles for the phrase pattern. The default mappings are

- subject (SUBJ) $\rightarrow$ agent
- object (OBJ) $\rightarrow$ theme
- indirect object (IOBJ) $\rightarrow$ beneficiary (BEN)
- infinitival complement (INF-COMPL) $\rightarrow$ purpose
- adjective complement (ADJ) $\rightarrow$ predicate complement (PRED-COMPL)
- C-TO-LOCATION $\rightarrow$ TO-LOCATION
- C-TO-QUANT $\rightarrow$ TO-QUANT

With this information, the subcategorization tables can not only be used to decide where to attach phrases, but also help in finding the proper semantic role of the new sub-component in the larger phrase.

Assuming that the two sentences at the beginning of this section have been partially parsed into a noun phrase, verb, noun phrase and prepositional phrase, i.e. (I) (sent/booked) (a flight/letter) (to New York), the parser could now exploit the verb subcategorization table by evaluating the feature "(SEMROLE OF -1 OF -3)". As explained in more detail in section 5.6, this feature is interpreted as "what is the semantic role of the last item (at position -1; here *to New York*) with respect to the item at position -3 (in our examples the verb)?". After matching the partially parsed phrase to the best verb pattern, the system would return feature values 'UNAVAIL' for *(I) (booked) (a flight) (to New York)* and 'R-TO-LOCATION' for *(I) (sent) (a letter) (to New York).* Using this discriminating feature, a parser can make the decision that will properly attach the prepositional phrase *to New York.*

The currently 242 entries of the verb subcategorization table have been entered manually. Some dictionaries and grammar books such as (Hornby, 1974; Engel, 1988) contain verb patterns that could serve as a basis for subcategorization tables, but the entries were actually made without any such support, because the micromodular nature of the knowledge and its intuitive format already made the task easy. In future extensions it might be useful to have a similar subcategorization table for nouns.

3.6 Morphology

At least from a pragmatic computational standpoint, morphological processing has basically already been solved for English as well as other Germanic and Romance languages, Japanese, and, we suspect, all other languages. Morphology is computationally much simpler than syntax or semantics, because it operates only very locally.

Some approaches might be more elegant or linguistically motivated than others, but inflection tables, regardless of how many classes, cases, tenses, numbers, genders, voices, and levels of definiteness or politeness etc. have to be considered, pose no problem to computational processing, nor do irregular forms, and any approach can be made fast by caching results.

In our system, the symmetric morpholgy module is used for both analysis and generation. For analysis, given the surface form of a word, say *(("increases"))*, it finds its annotated stem-forms, e.g.

```
(((lex "increase")
  (surf "increases")
  (synt s-count-noun)
  (forms (((number f-plural))))
  (concept i-en-increase))

 ((lex "increase")
  (surf "increases")
  (synt s-tr-verb)
  (forms (((tense f-pres-tense) (person f-third-p) (number f-sing))))
  (concept i-ev-increase)))
```

In the other direction, given the annotated stem-form, e.g.

```
(("increase"
     (synt s-noun)
     (number f-plural)
     (concept i-en-increase)))
```

it finds the corresponding surface form, here *(("increases"))*.

Such generation is done by sending the annotated forms through a *morphological pipeline*, a data structure with elements that are interpreted to manipulate these annotated forms. A morphological pipeline can automatically be inverted. A generation pipeline thus implicitely also defines an analysis pipeline, which can then be used to gradually manipulate initially unannotated forms to become fully annotated stem forms.

A morphological pipeline is a sequence of *pipeline elements*. Each of these elements manipulates or filters an annoted form, yielding a *set* of annotated output forms. Manipulations include the replacement of a word-ending string by another string, change of stems, the addition or deletion of umlauts (for German) or the addition of an annotation; filters include tests whether or not the word has a certain ending or whether or not it has a certain prefix. Finally the pipeline element can be a set of parallel sub-pipelines, a so-called *fork* element.

The following figures illustrate this morphological generation and analysis. Morphological pipelines are represented by polygons, the annotated forms at various stages of processing by circles.

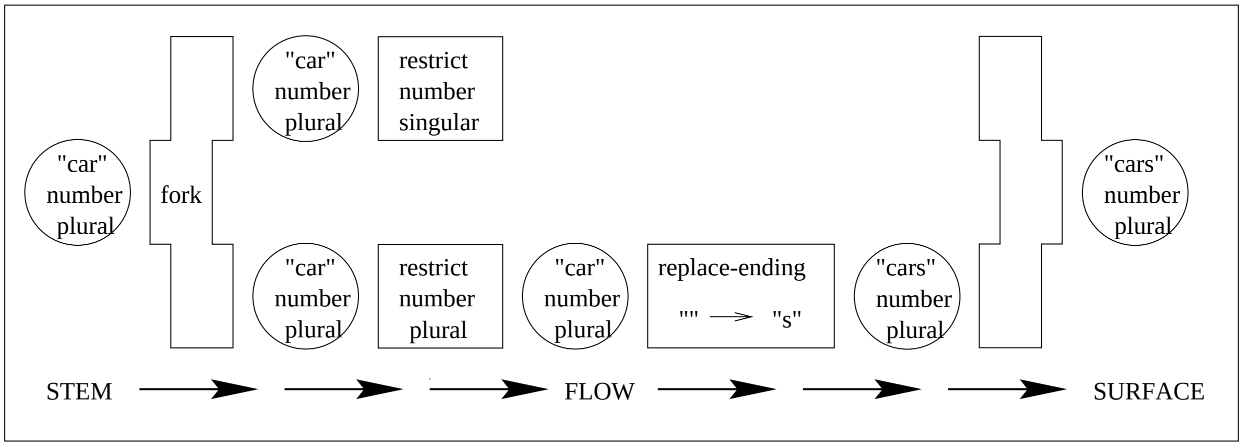


Figure 3.2: Generating the plural form of "car"

Figure 3.2 depicts a simplified noun pipeline. With "car" annotated by the restriction *number plural* as input, the pipeline forks, passing along the data to both sub-pipelines. While the *restrict number singular* element filters out the data because of the conflicting number, the plural filter element passes the data on to the next element, which adds an "s" to the word. The closing fork bracket combines the result of both sub-pipelines, in this case only the "cars" from the plural sub-pipeline.

The pipeline depicted in figure 3.2 is textually represented as

```
'((fork ((restrict number f-sing))
        ((restrict number f-plural)
         (repl "" "s"))))
```

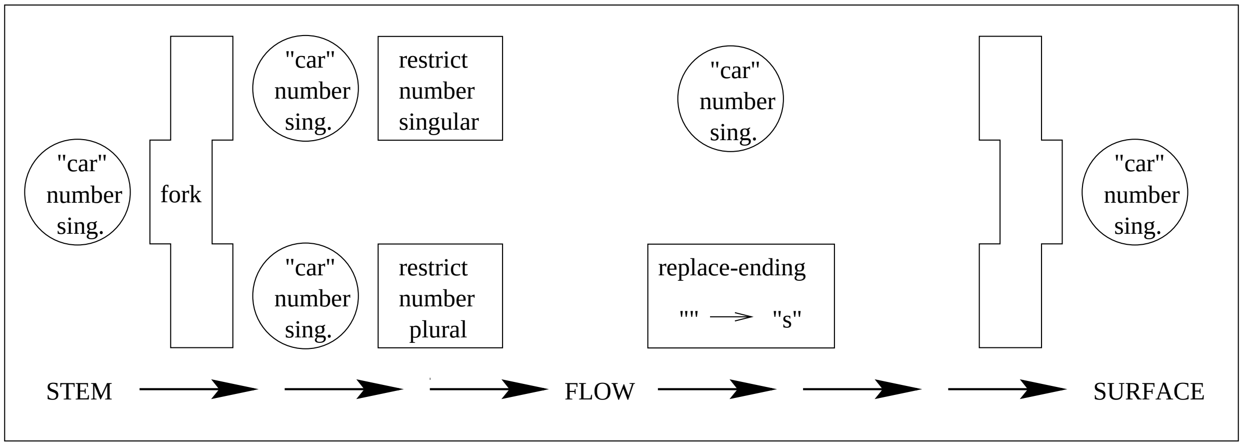


Figure 3.3: Generating the singular form of "car"

Figure 3.3 depicts the same pipeline as figure 3.2, but different data. In this case, the plural sub-pipeline filters out the incoming data.

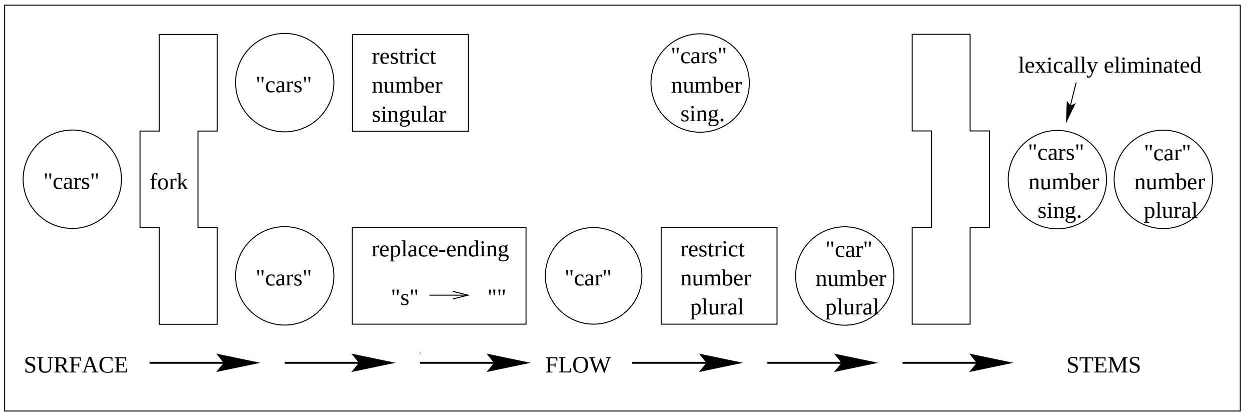


Figure 3.4: Analyzing the word "cars"

Figure 3.4 depicts the automatically inverted pipeline, which is used for analysis. Since the input data does not have any prespecified restrictions, the data passes through both sub-pipelines and finally yields a result of two units. In the full system, other pipeline elements add the filter *restriction synt noun* and perform a lexical check, which eliminates the stem form "cars", because no such noun stem will be found in the lexicon. Had the input been "bus" instead, the result *"bus" number singular* would have been kept and *"bu" number plural* been lexically eliminated.

Once morphological pipelines are defined, they can be used as sub-pipelines thereafter. This can be particularly useful since there are related morpological phenonoma even across parts of speech. The standard plural formation, obviously more complex than the simplified version depicted above *replace-ending* *“” → “s”*, since it also has to cover like *bush/bushes*, *baby/babies*, e.g. closely resembles the third person singular present tense formation with *push/pushes* and *cry/cries*.

When a pipeline element such as *replace-ending “s” → “”* encounters a data unit that it can not operate on, e.g. “car”, where no final “s” can be deleted, that pipeline just filters out that non-applicable data unit.

(Very) irregular forms are handled by forking off a special sub-pipeline that basically performs an irregular form table lookup; and the “regular” sub-pipeline filters out any forms that conflict with the irregular form table. Special pipeline elements for German handle umlauts and stem changes for strong verbs, in the latter case with access to irregular verb tables.

3.6.1 Specific Pipeline Elements

The **replacement** of a word-ending string by another string, e.g. the pipeline element *(repl “e” “ed”)* [1] would modify a form with “increase” to one with “increased”; if the incoming word does not match the first ending, the output is empty.

There are pipeline elements to add or delete umlauts (in German), e.g. *(add-umlaut)* changes “Apfel” to “Äpfel” and *(delete-umlaut)* does the opposite.

(double-last-letter) and *(undouble-last-letter)* do what their names suggest, e.g. “run” is changed to “runn” and vice versa.

Restrictions add annotations, and automatically filter out annotated forms with conflicting annotations. E.g. *(restrict number f-plural)* changes an incoming (“increase”) to (“increase” (number f-plural)), leaves an incoming (“increase” (number f-plural)) as is, and returns an empty output for an incoming (“increase” (number f-sing)).

Other specialized pipeline elements can perform **stem changes** or take care of **irregular forms** using lookup-tables.

Filters, which return a set containing the incoming annotated form or an emtpy set, check word-endings, e.g. (ends-in “ed”), (not-ends-in “ing”) or (memb (nth-last-letter x *n*) <*set of characters*>), which checks whether or not the *n*th last letter is a member of the given character set.

The last important pipeline element of a pipeline is a *fork*, which channels incoming input forms into parallel sub-pipelines.

[1] Since the system is implemented in Lisp and morphological changes tend to occur at the end of words, words are actually represented as their reversed character lists, e.g. “increases” is represented as (#\s #\e #\s #\a #\e #\r #\c #\n #\i). For easier reading, strings are left as such in the following descriptions of the morphological module.

Using a previously defined sub-pipeline *needs-doubling* the following sub-pipeline describes the formation of the regular past tense and past participle for English: [2]

```
`((fork ((repl "e" "ed"))                         ;;; like, liked
        ((not-memb (nth-last-letter x 2)
                   '(#\a #\e #\i #\o #\u))   ;;; pray, prayed
         (repl "y" "ied"))                        ;;; cry,  cried
        (,needs-doubling               ;; prev. def. sub-pipeline
         (double-last-letter)                     ;;; stop, stopped
         (repl "" "ed"))
        ((not-ends-in x "e")                      ;;; push, pushed
         (or (not-ends-in x "y")
             (memb (nth-last-letter x 2)
                   '(#\a #\e #\i #\o #\u)))
         (not ,needs-doubling)         ;; prev. def. sub-pipeline
         (repl "" "ed"))))
```

A major advantage of these pipelines is that they can easily be inverted automatically, so that the preceding sub-pipeline could also be used to find stem forms for regular past tense or particple forms. The inverse for *(repl "e" "ed")* for example is *(repl "ed" "e")* and inverting other pipeline elements and pipelines is equally straightforward:

element	inv(element)
replace-ending s1 s2	replace-ending s2 s1
any filter	*stays the same*
double-last-letter	undouble-last-letter
undouble-last-letter	double-last-letter
sequence (s1 ... sn)	*sequence* (inv(sn) ... inv(s1))
fork (s1 ... sn)	fork (inv(s1) ... inv(sn))

Referring to the just defined pipeline as *pv-ed*, and given similarly defined sub-pipelines for present tense (*pv-prt*) and present participle (*pv-prp*) forms, a pipeline for regular English verbs would then be[3]

```
`((fork ((restrict tense f-pres-inf))
        (append '((restrict tense f-past-tense))
                pv-ed)      ;;; previously defined sub-pipeline
        (append '((restrict tense f-past-part))
                pv-ed)      ;;; previously defined sub-pipeline
        (append '((restrict tense f-pres-tense))
                pv-prt)     ;;; previously defined sub-pipeline
        (append '((restrict tense f-pres-part))
                pv-prp)))   ;;; previously defined sub-pipeline
```

[2]The x is defined to refer to the word without its annotations.

[3]*append* concatenates two morphological pipelines

If we call this sub-pipeline *pv*, and with a similarly defined sub-pipelines for nouns, *pn*, the top-level pipeline for regular words looks like:

```
`((fork ,(append '((restrict synt s-verb))
                 pv)
        ,(append '((restrict synt s-noun))
                 '((restrict  person f-third-p))
                 pn)
        ((restrict synt s-prep))
        ((restrict synt s-adv))
        ...))
```

The monolingual lexicon is used to link stem-forms to actual lexicon entries which then provide more detailed information like (semantic) concept, a more detailed syntactic category and other annotations.

A morphological pipeline for English was defined in a file with 178 lines, whereas the German equivalent needed 640 lines.

Chapter 4

Segmentation and Morphological Analysis

Before the actual parsing starts, the input text is preprocessed. The text, already segmented into sentences in the original corpus, is first segmented into words and punctuation marks; the resulting words are then morphologically analyzed. The list of morphologically analyzed words and punctuation marks serves as input to the parser.

Segmentation and morphological analysis are not particularly challenging, because both steps need to consider only very local properties of substrings and words (at least for Germanic and Romance languages, at least from a pragmatic perspective, and at least qualitatively). Segmentation is fairly simple and morphology is well described in the literature (Lederer, 1969; Quirk, Greenbaum, Leech, & Svartvik, 1985).

4.1 Segmentation

Given a string of text, e.g. a sentence, the first task is to segment the string into a list of words, numbers and punctuation marks. E.g. the string “In the 1970s, the St. Louis-based 'company' traded at $2.45.” gets segmented into

```
((“In” 0 0 2 2) (“the” 2 3 6 6) (1970 6 7 11 11) (“s” 11 11 12 13 ,)
(D-COMMA 12 12 13 13) (“the” 13 14 17 17) (“St” 17 18 20 21 .)
(D-PERIOD 20 20 21 21) (“Louis” 21 22 27 27) (D-DASH 27 27 28 28)
(“based” 28 28 33 33) (D-APOSTROPHE 33 34 35 35) (“company” 35 35 42 43 ')
(D-APOSTROPHE 42 42 43 43) (“traded” 43 44 50 50) (“at” 50 51 53 53)
(D-DOLLAR-SIGN 53 54 55 55) (2.45 55 55 59 60 .) (D-PERIOD 59 59 60 60))
```

The four integers following the word, number or punctuation tokens indicate the *soft* and *hard* boundary positions of a token. The *hard* boundaries include only the token itself, whereas the *soft* boundaries include preceding whitespaces and some following punctuation.

This spacing information is maintained and used in later stages of parsing to reproduce precisely the surface strings associated with partially parsed sentence segments. Consider for example the two sentences "The new Japans - Taiwan and South Korea - were thriving economically." and "In tests it has worked for many heart-attack victims." The character '-' serves as a dash in the first example and as a hyphen in the second. Since spaces are not part of the segmented words per se, we attach the spacing information to the segmented words so that the original surface string can easily be reconstructed, as in the display of partially parsed sentences.

The spacing information also allows the surface string associated with partially parsed trees to be kept in the proper order, which is not trivial, because of discontinuous elements. E.g. when reducing the sentence elements "has been" + "always", we want to represent the resulting tree, even if it is structured [[has been] always], by its portion of the original surface string "has always been" for easier legibility. Finally, the spacing information is also used to properly represent so-called empty categories (see 5.3.5).

In many languages, including English, segmentation is relatively simple, since words are fairly easily identifiable as such. For languages with significant compounding, such as German or Swedish, where noun compounds are typically written as one word, an advanced segmenter needs to break noun compounds down into its components. Yet more difficult, in languages such as Japanese, where words are generally not separated by spaces, sophisticated segmentation tools, such as JUMAN (Matsumoto, Kurohashi, Utsuro, Myoki, & Nagao, 1994), have to be used. Since we currently only translate *from* English, and therefore only need to parse general text in English, a relatively simple segmenter is sufficient and so that's all we have implemented at this point.

4.2 Morphological Analysis and Parse Entries

The words put out by the segmenter are then morphologically analyzed. As already described in more detail in section 3.6, the morphological analyzer, given the surface form of a word, computes a list of *parse entries*, i.e. frames containing information about the lexical and surface form of a word, its syntactic and semantic category, form restrictions such as number, person, tense and other attributes that apply to certain types of words only, such as a value attribute for numerals. *Complex* parse entries have *subentries*, i.e. parse entries annotated by one or more roles, that can be syntactic or semantic, and describe the function of the subentry with respect to the superentry. *Simple* parse entries, which is what the morphological analyzer returns, do not have such subentries. Given for example the word *"increases"*, the morphological analyzer returns two simple parse entries, one for the nominal and one for the verbal interpretation of the word:

```
(((lex "increase")
  (surf "increases")
  (synt s-count-noun)
  (forms (((number f-plural))))
  (concept i-en-increase))

 ((lex "increase")
  (surf "increases")
  (synt s-tr-verb)
  (forms (((tense f-pres-tense) (person f-third-p) (number f-sing))))
  (concept i-ev-increase)))
```

Note:

- "**lex**" is the lexical root form of the head of the entry.
- "**surf**" is the surface string associated with an entry.
- "**ssurf**", not displayed here, contains spacing information.
- "**synt**" is the part of speech.
- "**forms**" displays any restrictions with respect to person, number, tense, etc. The following subsection contains a full list of these forms and their values.
- "**concept**" is the semantic class.
- "**subs**" denotes the list of sub-entries.
- "**props**" includes any other properties.

Morphological ambiguity is captured within a parse entry. E.g. the verb "put" would be analyzed as

```
(((lex "put")
  (surf "put")
  (synt s-tr-verb)
  (forms (((person f-first-p) (number f-sing) (tense f-pres-tense))
          ((person f-second-p) (number f-sing) (tense f-pres-tense))
          ((number f-plural) (tense f-pres-tense))
          ((tense f-pres-inf))
          ((tense f-past-part))
          ((tense f-past-tense))))
  (concept i-ev-put)))
```

The forms present different morphological alternatives. Each form consists of a set of form restrictions. Note that the attribute value pair "**(forms nil)**" indicates that there are no morphological alternatives, which, in pathological cases, can sometimes result when incompatible forms are unified, as for example when determining the number of the noun phrase "a cats", where the article imposes *singular* and the noun *plural*. On the other hand "**(forms (nil))**" indicates that there is one alternative without any restrictions. Words from syntactic categories without any morphological variation, e.g. adverbs, typically have this form.

4.2.1 List of Morphological Forms

In our system, we use the following morphological forms. Possible values, as they might arise in English or German, are given in parentheses.

- person (first, second, third)
- number (singular, plural)
- case (nominative, genitive, dative, accusative, object)
- voice (active, passive)
- tense (present, past, present perfect, past perfect, future, future perfect, past participle, present participle, present infinitive, present 'to'-infinitive, past infinitive, past 'to'-infinitive)
- gender (masculine, feminine, neuter)
- mood (indicative, subjunctive)
- mode (declaration, wh-question, yn-question, imperative, exclamation)
- aspect (continuous, non-continuous)
- det-adj-ending (primary, secondary, uninflected) [for German adjectives]
- you-caps (true, false) [for politeness capitalization of German pronouns]
- ref-number (singular, plural) [number of the noun phrase that a pronoun refers to]
- ref-person (first, second, third)
- ref-gender (masculine, feminine, neuter)

Chapter 5

Parsing

The parsing of unrestricted text, with its enormous lexical and structural ambiguity, still poses a great challenge in natural language processing. The traditional approach of trying to master the complexity of parse grammars with hand-crafted rules turned out to be much more difficult than expected, if not impossible. Many parsers leave ambiguity largely unresolved and just focus on an efficient administration of ambiguity. The Tomita parser (Tomita, 1986) for example, when running the sentence "*Labels can be assigned to a particular instruction step in a source program to identify that step as an entry point for use in subsequent instructions.*" on a relatively moderate sized grammar with 220 hand-coded rules returns a parse forest representing 309 parsing alternatives. For a more detailed grammar of 400 rules, the number of alternatives increases to 127,338.

Newer probabilistic approaches, often with only relatively restricted context sensitivity, have achieved only limited success even when trained on very large corpora. Magerman (1995) and Collins (1996) for example train on 40,000 sentences from the Penn Treebank Wall Street Journal corpus and it is doubtful whether the results can be further significantly increased by further enlarging the training corpus, because (1) a parsing accuracy ceiling might have been reached due to model limitations[1] and (2) a further significant enlargement of the manually annotated corpus is very expensive.

To cope with the complexity of unrestricted text, parse rules in any kind of formalism will have to consider a complex context with many different morphological, syntactic and semantic features. This can present a significant problem, because even linguistically trained natural language developers have great difficulties writing explicit parse grammars covering a wide range of natural language. On the other hand it is much easier for humans to decide how *specific* sentences should be analyzed.

5.1 Basic Parsing Paradigm

We therefore propose an approach to parsing based on example-based learning with a very strong emphasis on context. After the segmentation of the sentence into words and punctuation marks, and a morphological analysis of the words, the parser transforms the resulting word sequence into an integrated phrase-structure and case-frame tree, powerful enough to be fed into a transfer and

[1]For a more detailed discussion, see chapter 10, "Related Work".

a generation module to complete the full process of machine translation.

The parser is trained on parse examples acquired under supervision. Since during the training phase, the system is to be guided by a human supervisor, it is extremely important that the parsing process has a very transparent control structure that is intuitive to a human. As recent work in cognitive science, e.g. (Tanenhaus & al., 1996), has confirmed again, the human mind performs a continuous and deep interpretation of natural language input. A sentence is processed 'left-to-right' in a single pass, integrating part-of-speech selection and syntactic and semantic analysis.

As the basic mechanism for parsing text we therefore choose a shift-reduce type parser (Marcus, 1980). It breaks parsing into an ordered sequence of small and manageable parse actions such as *shift* and *reduce*. With the parse sequence basically mimicking the order people process sentences, we don't only make the parsing process intuitive to the supervisor during the training phase, but, with this paradigm, we also can be confident that at any point during the parse, the current morphological, syntactic and semantic context information of a partially parsed sentence will be sufficient for the computer program to make good decisions as to what parse action to perform next.

Like for humans, this approach also has the advantage of a single path, i.e. deterministic parsing[2], eliminating computation on 'dead end' alternatives and resulting in a temporal processing complexity that is **linear** in the length of a sentence, making parsing very fast.

Applying machine learning techniques, the system uses parse examples acquired under supervision to generate a deterministic shift-reduce type parser in the form of a decision structure. This decision structure's classification of a given parse state is the parse action to be performed next. The parse state used by the decision structure is described by its context *features*.

Relieving the NL-developer from the hard if not impossible task of writing an explicit grammar, the focus on relevant features at the same time requires a relatively modest number of training examples when compared to more statistically oriented approaches like for example (Magerman, 1995; Collins, 1996), where the parser is trained on 40,000 WSJ sentences. Our system is currently trained on only 256 WSJ sentences. The following chapter discusses parsing experiments and the influence of the number of features in detail.

The approach we take provides a high degree of manageability of the 'grammar' when increasing its coverage. All that is necessary, is to add more examples. The advantage of examples is that they are much more modular than rules. All we assert for an individual example is the next parse action that has to be taken for the specific parse state of a specific sentence, regardless of how different that parse action might be for a similar parse state of another sentence. We therefore don't have to change old examples when we try to increase coverage to include new sentences, however much a decision structure based on the old examples might have misparsed the new sentences by misclassifying new parse states. After new examples have been added, we automatically compute a new, more refined decision structure that will also cover the new sentences.

On the other hand, when increasing coverage in a rule-based system, it is typically not sufficient to add more rules. Old rules will often have to be modified, e.g. by specializing their antecedents. Given the enormous complexity of natural language, it can easily happen that sentences that were parsed properly before a rule modification no longer work afterwards. To make sure that the modified rules work, testing becomes necessary, and with that, examples. So we see

[2]For a brief discussion of the problem of garden path sentences, see section 5.9 later in this chapter.

that extending rule sets is not only qualitatively more difficult than extending example sets, but, at least in practise, examples are indispensable in some form or other anyway.

5.2 The Core Parsing Mechanism

As just described, we choose a shift-reduce type parser as our basic mechanism for parsing text into a shallow semantic representation. Parsing is broken down into an ordered sequence of small and manageable parse actions such as *shift* and *reduce*.

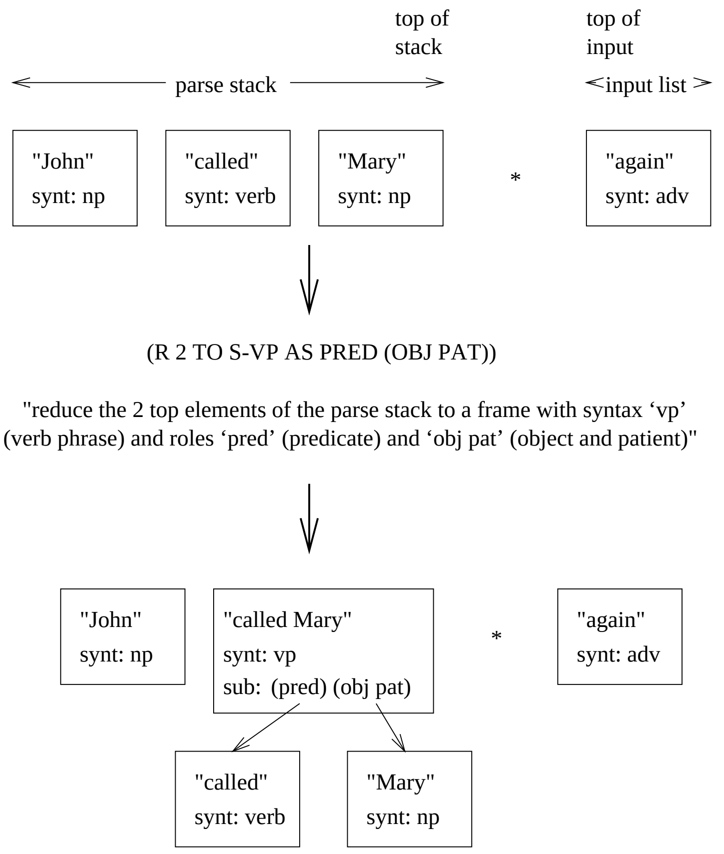

top of stack
top of input
parse stack
<input list>
"John" synt: np
"called" synt: verb
"Mary" synt: np
*
"again" synt: adv
(R 2 TO S-VP AS PRED (OBJ PAT))
"reduce the 2 top elements of the parse stack to a frame with syntax 'vp' (verb phrase) and roles 'pred' (predicate) and 'obj pat' (object and patient)"
"John" synt: np
"called Mary" synt: vp sub: (pred) (obj pat)
*
"again" synt: adv
"called" synt: verb
"Mary" synt: np

Figure 5.1: A typical parse action (simplified); boxes represent *parse entries*

The central data structure for the parser contains a parse stack and an input list. The parse stack and the input list contain trees of parse entries of words or phrases. Core slots of parse entries are surface and lexical form, syntactic and semantic category, subentries (of type parse entry) with syntactic and semantic roles, and form restrictions such as number, person, and tense. Other slots can include information like the numerical value of number words, flags whether or not a (German) verb has a separable or inseparable prefix etc.

Initially, the parse stack is empty and the input list contains the simple parse entries produced by the morphological analyzer. After initialization, the *deterministic parser* applies a sequence of *parse actions* to the parse structure. The most frequent parse actions are *shift*, which shifts a parse entry from the input list onto the parse stack or vice versa, and *reduce*, which combines one or several parse entries on the parse stack into one new parse entry. The parse entries to be combined are typically, but not necessarily, next to each other at the top of the stack. As shown in figure 5.1, the action

```
(R 2 TO VP AS PRED (OBJ PAT))
```

for example reduces the two top parse entries of the stack into a new parse entry that is marked as a verb phrase and contains the next-to-the-top parse entry as its predicate (or head) and the top parse entry of the stack as its object and patient. Other parse actions include 'add-into' which adds parse entries arbitrarily deep into an existing parse entry tree, 'mark' that marks some slot of some parse entry with some value and operations to introduce empty categories (traces and 'PRO', as in "She$_i$ wanted PRO$_i$ to win."). Parse actions can have numerous arguments, making the *parse action language* very powerful. In particular, when shifting in a new word, the mandatory argument of *shift* specifies the part-of-speech of the possibly ambiguous word that is being shifted in, thereby effectively performing the so-called tagging as part of the parsing.

5.3 The Parse Actions in Detail

This section describes the various parse action in more detail. We present the various basic types of parse action (shift, reduce, add-into, empty category instantiation, co-index, mark, expand as well as the pseudo-action 'done'). Many of these parse actions allow a sophisticated parameterization. Often these parameters designate parse entries that participate in an operation.

In the English parsing training examples, we use seven basic types of parse actions (all except 'expand' which is only used for German) with a current total of 265 different individual parse actions. These are quite unevenly distributed with the 10 most frequent parse actions accounting for 53% of all examples and 78 parse actions occurring only once, accounting for 0.66% of all examples.

The various parse actions are not preselected in any way. Instead, during training, as further explained in section 5.7, each parse action example is assigned the parse action that the supervisor deems to be appropriate in the context of a specific, partially parsed text. The range of parse actions therefore depends directly and exclusively on the specific set of training examples.

The first of the following subsections describes how parse entries that are referred to in features can be identified. The description of the individual parse action types is then followed by subsections on how these can be combined into multiple parse actions and how they implicitly trigger a number of internal computations.

5.3.1 Parse Entry Paths

Many parse actions have parameters that refer to parse entries on the parse stack or the input list. The participating parse entries are identified by *parse entry paths* with various degrees of complexity. Parse entries that are elements of the input list are identified by positive integers. n designates the nth element on the input list. Parse entries that are elements of the parse stack are identified by negative integers such that n designates the $|n|th$ element of the parse stack. The top row of numbers in figure 5.2 illustrates this scheme. The numbers in parentheses indicate positions between parse entries. Note that the active ends of the parse stack and the input list[3] face each other. The active position ("0") of a partially parsed sentence is marked by an asterisk (*).

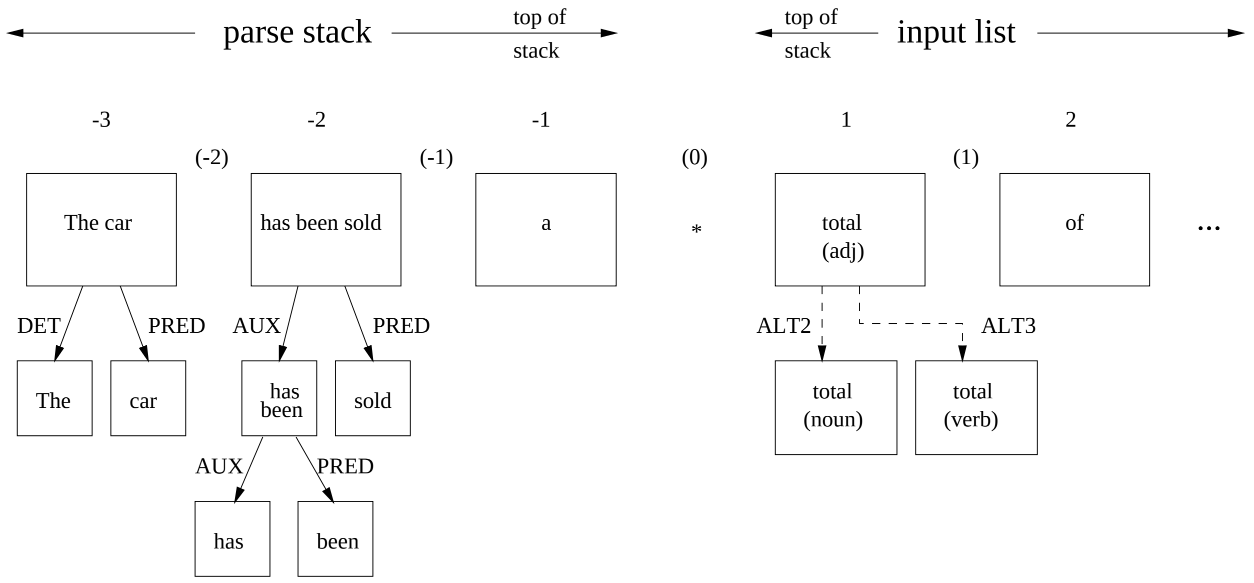


Figure 5.2: Parse stack and input list

Sub entries are described by paths such as (DET OF -3), in the current example the parse entry for '*The*'; (AUX OF -2), the parse entry for '*has been*'; and (PRED OF AUX OF -2), the parse entry for '*been*'. *PRED** designates the 'ultimate' predicate of a parse entry, i.e. the parse entry that is reached by repeatedly descending to sub-entry PRED until this is no longer possible. In figure 5.2, (AUX* OF -2) would designate the parse entry for '*has*', but the asterisk-extension is only used for predicates ("PRED").

Besides these concrete path descriptions, the system allows abstract paths that contain predefined elements such as *NP-1, ACTIVE-FILLER-1* and *(SURF-NP -2 BEFORE -1)*. Descriptors of form <restr>-<n> designate the nth last parse entry on the parse stack for which <restr> holds. NP-1 is the top (= rightmost) noun phrase parse entry on the parse stack, in the current figure '*The car*'. ACTIVE-FILLER-1 is the top parse entry on the parse stack that has a slot *gap* with the value '*active filler*'. (SURF-NP -2 BEFORE -1) is the penultimate surface noun phrase before position -1; for the SURF-NP construction, all right-branching embedded phrases of the last element before the position marked by the BEFORE argument count.

Path descriptions with such abstract elements can often better describe the essence of the path that leads to the involved path entry. Consider for example these two sentence fragments, separated by a '+': *I know the artist of this masterpiece, + which was painted 200 years ago.* Relative phrases like in this example often refer to the last surface noun phrase, here *this masterpiece.*

To express this relationship, no matter how deep this noun phrase is already embedded in a parse stack parse entry, we use (SURF-NP -1 BEFORE -1), indicating the last surface noun phrase before position -1. We could have used (PRED-COMPL OF MOD OF OBJ OF -2), the predicate complement of the modifier *of this masterpiece* of the object *the artist of this masterpiece* of the

[3]The input 'list' is actually a stack too, since the shift operation, as described in section 5.9, can also shift parse entries back onto the input 'list'.

second highest parse entry on the parse stack, but descriptions like that last concrete path can vary enormously from example to example and would not lend themselves very well to generalization. In order to *learn* parse actions well, it is important that essentially similar contexts should call for the same action to be taken. Abstract paths such as (SURF-NP -1 BEFORE -1) allow this parse action stability and therefore makes it easier for the machine learning component to learn appropriate parse actions.

5.3.2 Shift

Parse entries can be shifted from the input-list to the parse-stack and vice versa. The by far more common type of shift is from the input-list to the parse-stack (examples (1) and (2) below). Output products from morphological processing, i.e. parse entries in the input-list, can still be ambiguous, e.g. '*left*' can be interpreted as both a verb or an adjective. Since further processing on the parse stack requires syntactically disambiguated parse entries, this shift requires an argument identifying the proper alternative. When shifting in an ambiguous parse entry, the shift operation eliminates all alternatives that are not covered by the shift operation argument. Assigning a specific part-of-speech to a word is called *tagging*. Note that in our system, tagging is naturally integrated into the parsing process, whereas for some other parsers, tagging is required as a separate preliminary step.

In the current system, only syntactic restrictor arguments are used, as in example (1), but semantic restrictors as in example (2) could be used to make a yet more specific choice, such as for a word like *buck*.

(1)	**(S S-VERB)**	shifts parse-entry of syntactic type *verb* from input-list to parse-stack
(2)	**(S I-EN-ANIMATE)**	shifts parse-entry of semantic type *animate* from input-list to parse-stack
(3)	**(S -2)**	shifts two parse entries from the parse-stack back onto the input-list

The other type of shift (example (3)) shifts parse entries back onto the input list. The absolute value of the first argument indicates how many parse entries are to be shifted back. To better understand the purpose of this reverse shifting, consider the case of a noun phrase followed by a preposition. The prepositional phrase that this preposition probably starts might or might not belong to the preceding noun phrase. Making an attachment decision already at this point is typically still premature, because the decision would have to be based on the yet unprocessed components of the prepositional phrase. So instead of making an early PP attachment decision, the system shifts in the components of the prepositional phrase, processes it, and upon completion, decides whether or not an attachment to the preceding noun phrase is appropriate. If so, it performs the attachment through a *reduce* operation. Otherwise, it shifts the prepositional phrase, that turned out not to attach to the noun phrase after all, back onto the input-list so that the system can process the noun phrase otherwise, e.g. by attaching it to the preceding verb. Most shift-back actions involve a single parse entry, but some of them move two or even three parse entries at a time.

5.3.3 Reduce

The *reduce* operation typically combines one or more parse entries into a new parse entry. A typical parse action was already presented in figure 5.1. In the following examples, the plus sign (+) denotes the boundary between two adjacent parse stack elements.

(1) **(R 2 TO S-PP AS PRED PRED-COMPL CLASS C-BY-AGENT)**
reduces the top two parse-stack parse entries to a prepositional phrase of class *C-BY-AGENT* with roles *pred* and *pred-compl* respectively. Example: *by* + *a shareholder*

(2) **(R 3 AS MOD DUMMY PRED)**
reduces the top three parse-stack parse entries to a parse-entry with roles *mod*, *dummy* and *pred* respectively. Example: *heart* + '*-*' + *attack*
For the parser, 'dummy' is treated like any other role. We use it to mark components without any semantic meaning, mostly punctuation, that can later be discarded during transfer.

(3) **(R (-3 -1) AS AUX PRED)**
reduces the first and third parse entry of the parse-stack. Example: *have* + *clearly* + *put*

(4) **(R (-3 -1) AS AUX PRED AT -2)**
does the same, but places the result at position 2 of the parse stack.

(5) **(R 2 AS SAME (OBJ QUANT))**
merges the top parse entry into the second parse entry with roles *obj* and *quant*. Example: *will cost* + *$15*

(6) **(R (-2) AS SAME)**
moves the second parse entry on the parse stack to the top.

(7) **(R 2 TO LEXICAL)**
reduces the two top parse-stack parse entries to a parse entry that is listed in the lexicon. Example: *because* + *of*.

(8) **(R 1 TO S-NP AS PRED)**
upgrades the first parse entry on the parse stack to a noun phrase.

Often, the parse entries to be reduced are next to each other at the top of the parse tree (examples (1) (2) (5) (7) (8)). Disjoint constituents, e.g. *has* and *been* in "*has always been*", typically require a disjoint reduction such as in examples (3) and (4). Constituents also don't necessarily have to be at the top of the parse tree (example 6). Unless otherwise specified, as in example (4), the resulting parse entry is placed at the top of the parse stack. Also, unless otherwise specified, as in example (1), the new parse entry inherits properties like part-of-speech and semantic class from the component marked as predicate (*pred* or *same*). Sub parse entries can have one or more roles in the super parse entry. The second component in example (5) for example has the roles *obj* and *quant*. Reductions like the one in example (7), e.g. *New York* or *because of* reduce the components to a parse entry with an entry in the lexicon. The entry in the lexicon then determines the part-of-speech and semantic class of the new parse entry.

When reducing elements, a new level of hierarchy can be created (indicated by the keyword 'PRED'), or non-head elements are just added as extra sub-parse-entries to an existing complex parse entry, keeping the hierarchy flat (indicated by the keyword 'SAME' instead of 'PRED'), as illustrated in figure 5.3.

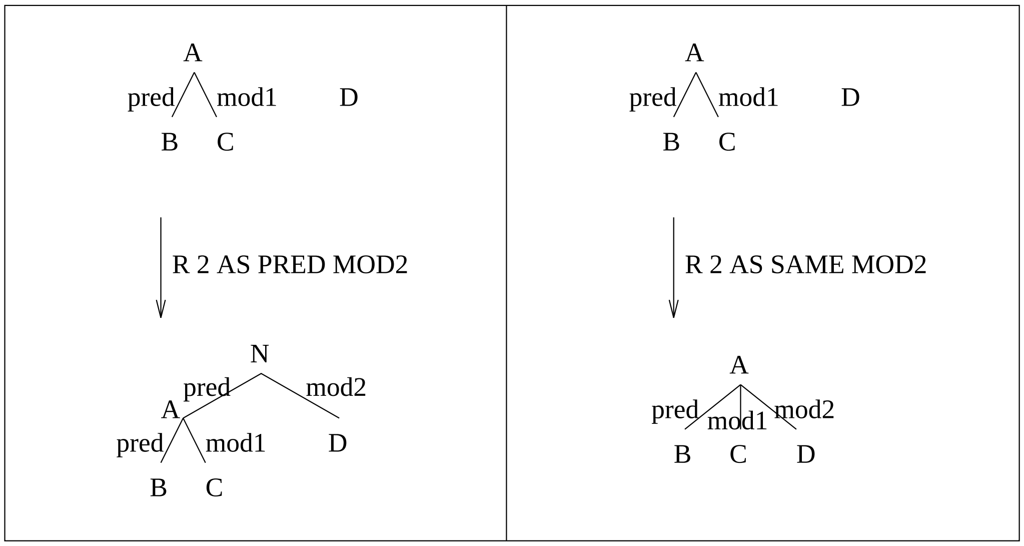


Figure 5.3: Difference between keywords 'pred' and 'same'

5.3.4 Add Into

Just like the *"reduce"* action, the *"add into"* action is used to combine parse entries. It is used to add more sub-entries to parse entries that are no longer at the top level of the parse stack, but rather (already) embedded underneath.

When reaching a comma while parsing sentences, it is often a good heuristic to use it as a signal to reduce all of what has been encountered so far before proceeding. In the following example (1), the main clause up to the comma is parsed into a sentence, before anything to the right of the comma is processed. When the sentence fragments *or* and *89 cents a share* are ready, they need to be added to the preceding disjunction element *a record $9.9 million*, the object of the third parse entry on the parse stack. The add-into action of example (1) accomplishes just that. Like for *reduce* actions, the *as*-clause provides the roles of the newly added sub-entries in the super entry.

(1) **(A (-2 -1) TO (SURF-NP -1 BEFORE -2) AS CONJ COORD))**
adds the top two parse-stack parse entries into the last surface noun phrase before position -2 with roles *conj* and *coord* respectively. Example: *In fiscal 1986, it earned a record $9.9 million, + or + 89 cents a share*

(2) **(A -1 TO (PRED* OF -2) AS PARTICLE)**
adds the top parse-stack parse entry into the ultimate predicate of the second parse-stack entry as a particle. Example: *had already been taken + out*

(3) **(A (-2 -1) TO (THAT-CLAUSE -1 BEFORE -2) AS CONJ COORD)**
adds the top two parse-stack parse entries into the last 'that-clause' before position -2 with roles *conj* and *coord* respectively. Example: *U.S. officials estimate that the move will result in $85 million in additional U. S. sales this year, + and + that such sales eventually could grow to $300 million annually*

(4) **(A -1 TO (INDEXED-NP -1 BEFORE -1) AS MOD)**
adds the top parse-stack parse entry into the last noun phrase to the before position -1 that has a matching index with a role of *mod*. Example: *The sales totaled* $< \$100 million >_1$, + [$< \$100 million >_1$] *compared with $90 million a year before*

Example (2) differs from the other examples in that the *PARTICLE* role triggers a special treatment. The particle (*out*) is merged with its verb (*taken*) at the deepest level, updating the verb concept from *I-EV-TAKE* to *I-EV-TAKE-OUT*.

5.3.5 Empty Category Instantiation

Empty categories (Chomsky, 1988; Lasnik & Uriagereka, 1988) are phonologically unrealized noun phrases including NP-traces such as *t* in "*John*$_i$ *was believed* t_i *to be clever.*", variables, such as the wh-trace *t* in "*What*$_i$ *are you looking at* t_i*?*" and PROs such as in "*He*$_i$ *wanted PRO*$_i$ *to win.*". Empty categories are useful tools to describe sentences where noun phrases have a semantic role in more than one clause (PRO) or where they are realized in a phrase other than the one where they have a semantic role (traces).

In order to have enough noun phrase representatives for noun phrases that play a role in more than one phrase, empty category tokens are created from noun phrase parse entries when necessary. The newly created parse entry is automatically co-indexed with the parse entry it is derived from. With several parse entries representing a noun phrase with a single phonological realization, these parse entries can now be assigned different roles in different phrases. E.g. in the sentence "*He*$_i$ *wanted PRO*$_i$ *to win.*", '*He*' will be assigned the roles *subject* and *experiencer*, whereas the corresponding '*PRO*' will be assigned the role *agent*. In a sentence like "*Sales*$_i$ *are expected* t_i *to increase.*", '*Sales*' will be assigned the roles *subject* and *dummy*, whereas 't_i' will be the *theme* in the embedded clause.

(1) **(EMPTY-CAT FROM ACTIVE-FILLER-1 AT 0)**
creates an empty-category parse entry from the last entry marked as an 'active filler', e.g. *where* in the following example, and places it at the top of the stack. Example: *He asked* + *where* + *I* + *came* + *from*

(2) **(EMPTY-CAT FROM NP-1 AT -1)**
creates an empty-category parse entry from the highest parse-stack parse entry that is a noun phrase and places it at position -1. Example: *Sales* + *are expected* + *to increase*

5.3.6 Co-Index

This action co-indexes the top parse entry on the parse-stack with the parse entry specified in the *WITH* clause. This is used particularly for co-indexing phonologically realized relative pronouns with their antecedents. In the representation the co-indexing is realized by annotating the primary parse entry with *(INDEX <index>)* and by annotating all other entries that refer to it with *(REF <index>)*, where <index> is a system generated integer. When an empty category parse entry is created, this is performed automatically. Non-relative pronoun resolution is currently done in a special phase at the end of parsing (see section 5.4).

(1) **(CO-INDEX WITH -2)**
co-indexes the top parse entry of the parse-stack with the parse entry to its left. Example: *someone* + *who*

(2) **(CO-INDEX WITH (SURF-NP -2 BEFORE -1))**
co-indexes the top parse entry of the parse-stack with the penultimate noun phrase on the surface to the left of position -1. Example: *The owner of the car,* + *who*, where *who* would be co-indexed with *the owner*.

5.3.7 Mark

The parse action **(M <path> <slot> <value>)** assigns slot <slot> of the parse entry specified by <path> the value <value>. This parse action is typically part of a complex parse action, i.e. mark actions typically occur in a list of two basic parse actions starting with a shift, reduce, or an empty category instantiation, followed by a mark action.

(1) **(M -1 GAP ACTIVE-FILLER)**
marks the top parse-stack parse entry as being an active gap. Example: *What * do you want ...* This mark-up can later be exploited for proper empty category instantiation (see subsection 5.3.5).

(2) **(M -1 IN-PARENTHESES TRUE)**
marks the top parse-stack parse entry as being being in parentheses.

5.3.8 Expand

The parse action **(EXPAND)** expands contractions of two or more words. An example in English is '*won't*', a contraction of *will* and *not*. Examples from other languages are *im* = *in* + *dem* in German, *aux* = *à* + *les* in French, *al* = *a* + *le* in Spanish and *nel* = *in* + *il* in Italian, all of which are contractions of a preposition with a definite article. In our system, the English verbal contractions are however already handled in the segmentation and morphology module, while the German contractions are expanded during parsing. The expand action does not take any arguments and always applies to the top parse-stack entry.

5.3.9 Done

The parameterless **(DONE)** parse 'action' signals that parsing is complete. Normally, this should only occur when there is a single item left on the parse stack and the input list is empty. That one parse entry is then returned as the result of the parse.

This parse action is also used in the case of serious problems during the parse, e.g. when strong symptoms of a potential endless loop are detected or when regularly proposed parse actions are overruled by the sanity checker, which tests whether a parse action can actually be performed in a specific context (more details in section 5.5). The DONE parse action is the only one that can always be used as a last resort, because it can be 'executed' in any parse state. When reaching such a pathological case, the system combines all remaining items on the parse stack and the input list into a new 'holding' parse entry.

5.3.10 Multiple Parse Actions

The parser decides what to do next by classifying the current partially parsed text. The classification is technically a list of parse actions. In most cases (99.8%), this parse action list contains only a single parse action, but sometimes multiple parse actions are used:

1. ((S S-ADV) (M -1 GAP ACTIVE-FILLER))
   Example: I know * *where* he is.
   Example: I know * *where* he is from.

2. ((EMPTY-CAT FROM NP-1 AT 0) (M -1 GAP ACTIVE-FILLER))
   Example: I know the *restaurant* * you were talking about.

3. ((EMPTY-CAT FROM ACTIVE-FILLER-1 AT 0) (M ACTIVE-FILLER-1 GAP SATURATED-FILLER))
   Example: I know the $restaurant_i$ PRO_i you were talking about *.

4. ((R 2 TO S-ADJP AS COMP PRED) (M -1 GRADE SUPERLATIVE))
   Example: the *most famous* * building

Multiple parse actions are used when shifting in interrogative pronouns or adverbs, as in example (1), in relative phrases without a phonologically realized relative pronoun, as in example (2), when resolving wh-traces, as in example (3), or when reducing periphrastic comparatives or superlatives, as in example (4). As usual, the asterisk (*) marks the current parse position.

Even though the parse action language allows an arbitrary number of elementary parse actions and does not have any restrictions on combinations, we only use up to two elementary parse action, and this only in a few cases (0.2%), all involving an 'empty-category' or 'mark' action. Mark actions often set parse entry slots that are only used in a few special circumstances. Theoretically the parser could of course learn to perform the *mark* action in a separate step. But since the *mark* actions are conceptually closely tied to their preceding action, it is simpler to learn the combination.

5.3.11 Triggered Computations

It is important that the supervisor does not get overwhelmed when providing the correct parse action. The actions should be as simple as possible. This is partly achieved by providing defaults, e.g. by letting a new parse entry inherit properties like its semantic class and its part of speech from its predicate sub-parse-entry.

In other cases, like compound verbs, the notion of inheritance is less useful. Consider for example the need to determine the form of a compound verb (e.g. "has" + "given"). The parser needs to know the tense, voice, aspect etc. of the compound. One way to gain this information would be to let the supervisor provide this information as part of the action of reducing an auxiliary and a main verb to a compound verb. A more elegant and less burdensome way for the supervisor is to let the reduction of an auxiliary and a main verb trigger the computation of the form of the complex verb. These computations don't depend on a lot of context and can therefore be performed using straightforward algorithms that don't involve any learning.

As another example, the reduction of the subject noun phrase and a verb phrase, e.g. "*The fish* + *were alive.*", triggers the computation of the form of a sentence (*third person plural*).

Currently, triggered computations are limited to determine the forms (e.g. tense or number) of new parse entries. Word sense disambiguation within a specific part of speech is currently successfully 'delegated' to the transfer process and anaphora resolution is performed at the end of core parsing. If, in a later extension, these tasks have to be moved up into core parsing, they could also be performed as a triggered computation, a solution superior to explicit parse actions, because the triggered computations would not only keep the parse action sequence simpler, but also keep parse action sequences impervious to lexicon additions that might turn a previously 'unambiguous' word like *pen* into an ambiguous one, necessitating a modification of all parse action sequences with such a 'newly ambiguous' word. Since such potential modifications of parse action sequences are highly undesirable, a triggered action is clearly preferable. The computations could be triggered by the evaluation of a feature that is used in the parse action decision making process and depends on the specific sense of a word or its antecedent.

Such additional triggered computations could principally be so complex that they could benefit from machine learning. However, this wasn't found to be necessary for the currently implemented triggered computations.

5.4 Anaphora Resolution

While the computations described in the last section can easily be linked to triggering parse actions, another computation, anaphora resolution, could be performed at several points during the parse process. For convenience's sake, we choose to perform it at the end of the parse action sequence.

Even in closely related languages such as English and German, which both differentiate pronouns with respect to number (singular/plural), person (1-3) and gender (masculine/feminine/neuter), pronouns don't match one to one. Compared to English, German has a much stronger notion of an independent grammatical gender, so that for example *der Löffel (the spoon)* is masculine, *die Gabel (the fork)* is feminine, and *das Messer (the knife)* is neuter. Now consider the translation of the English pronoun *it*. The gender of its equivalent German pronoun depends on the grammatical gender of its German antecedent, which is the translation of the antecedent of *it* in English. So, to pick up on the previous example, when translating the English sentence "*The spoon/fork/knife is expensive, because it is made out of silver.*", the English pronoun *it* would have to be translated as *er* (masc.), *sie* (fem.), or *es* (neuter), depending on what eating utensil it referred to. As this example shows, anaphora resolution is necessary for proper translation.

To identify the antecedent of an anaphor, the system first finds the syntactically permissible antecedent candidates, using for example number agreement and the linguistic relationship *c-command* (for details see for example (van Riemsdijk & Williams, 1986)). It then eliminates some candidates by using a few simple semantic restrictions, e.g. that a pronoun that fills the role of an agent must have an antecedent that semantically can be an agent. Consider for example the pronoun in the sentence "*The airline bought the plane because it had already decided to do so earlier*". Syntactically, both *The airline* and *the plane* qualify as an antecedent for *it*, but *the plane* is ruled out, because it can not be an agent as required for subjects of "to decide" (in the active voice). Finally, among any remaining candidates, the syntactically closest is picked.

Relatively simple heuristics were sufficient to cover all anaphora cases in the 48 training sentences. In a more advanced anaphora resolver, these heuristics have to be elaborated further. With enough complexity, machine learning might again prove useful in deciding which antecedent to pick. Previous work in this area includes (Aone & Bennett, 1995), which describes an anaphora resolution system trained on examples from Japanese newspaper articles.

5.5 Parsing Safeguards

The parser contains a few safeguards that suppress the attempt to execute unexecutable parse actions, detect potential endless loops and handle incomplete parses.

Before a proposed parse action is actually executed, the *sanity checker* tests whether this can actually be done. If it is impossible, e.g. in the case of a reduction of n elements on a parse stack that actually contains less than n elements or a shift-in on an empty input list, the sanity checker forces another action to be chosen. The sanity checker also suppresses a shift-in that would immediately follow a shift-out.

If the decision structure can naturally provide an alternative acceptable parse action, that action is chosen, otherwise, a new word is shifted in, or, if the input list is empty, the parse action 'done' is selected.

Another source of potential complications are endless loops. While the simple shift-in shift-out loop is already suppressed by the sanity-checker, other loops are still possible, e.g. a loop with the iteration *(M -1 GAP ACTIVE-FILLER) (EMPTY-CAT FROM NP-1 AT 0) (R 2 AS QUANT SAME)*.

To detect a loop, the parser limits the number of allowed remaining parse actions to four times the number of elements on the parse stack and the input list, about twice the amount of parse actions typically expected. Since the number of remaining parse actions allowed is limited, all actual loops are detected. All suspected loops actually turned out to be loops, as inspections showed. When a loop is detected, the parse is terminated by choosing the action 'done'. Fortunately, when the system is trained on at least a few dozen sentences, loops become quite rare.

If parsing is terminated before all words have been combined into a single parse entry, e.g. due to a 'done' parse action prescribed by the sanity checker, the endless loop detector or just an incorrect classification, all remaining elements on the parse stack and input list are lumped into a holding parse entry with uncommitted syntactic and semantic class (*S-SYNT-ELEM* and *I-EN-THING*) and uncommitted roles (*CONC*) for the elements. This operation allows the parsed sentences to be further processed. Obviously, we expect subsequent evaluations to yield substandard results, particularly in recall and translation, but it is important that parsing quality can also be measured for pathological cases.

5.6 Features

To make good parse decisions, a wide range of features at various degrees of abstraction have to be considered. To express such a wide range of features, we define a *feature language*. Given a particular parse state and a feature, the system can interpret the feature and compute its value for the given parse state, often using additional knowledge resources such as

1. a general *knowledge base* (KB), which currently consists of a directed acyclic graph of concepts, with currently 3608 *is-a*-relationship links, e.g. "$CAR_{NOUN-CONCEPT}$ is-a $VEHICLE_{NOUN-CONCEPT}$"; for more details see section 3.1
2. *subcategorization tables* that describe the syntactic and semantic role structure(s) for verbs and nouns with currently a total of close to 200 entries; for more details see section 3.5

The following examples, for easier understanding rendered in English and not in feature language syntax, illustrate the expressiveness of the feature language:

- the general syntactic class of the top element of the stack (e.g. adjective, noun phrase),
- the specific finite tense of the second stack element (e.g. present tense, past tense),
- whether or not some element could be a nominal degree adverb,
- whether or not some phrase already contains a subject,
- the semantic role of some noun phrase with respect to some verb phrase (e.g. agent, time; this involves pattern matching with corresponding entries in the verb subcategorization table),
- whether or not some noun and verb phrase agree.

Features can in principal refer to any element on the parse stack or input list, and any of their subelements, at any depth. Since all of the currently 205 features are supposed to bear some linguistic relevance, none of them actually refer to anything too far removed from the current focus of a parse state. A complete list of all 205 features can be found in appendix section B.1. The set of features is used for all parse examples for a specific language and can easily be extended when the need arises.

The current set of 205 features has been collected manually. The feature collection is basically independent from the supervised parse action acquisition. Before learning a decision structure for the first time, the supervisor has to provide an initial set of features that can be considered obviously relevant. During early development phases of our system, this set was increased whenever parse examples had identical values for all features (so far) but nevertheless demanded different parse actions. Given a specific conflict pair of partially parsed sentences, which is signaled during the machine learning process if it occurs, the supervisor would add a new feature that relevantly discriminates between the two examples. Such an addition requires fairly little supervisor effort. Relevant context features refer to parse entries that are shifted, reduced or otherwise directly participate in an operation, as well as parse entries that describe context in the narrow sense.

As explained further in the next section, parse examples are generated from parse states and parse actions. Given the state of a partially parsed sentences, the system computes the values for all features in the parsing feature list of a specific language. These values are then combined into a feature vector, a list of values, which, together with the parse action, form the core of a parse example. If the set of features changes, the parse examples have to be recomputed, because the feature vectors change. However, this can be done fully automatically since parse actions have been recorded in log files.

The feature sets for parsing in English and German are by and large the same. However, a few features, such as "*Is the second word on the parse stack 'there'?*", as it might be useful to characterize expressions like "there is"/"there are", are language specific. We expect the feature set to grow, possibly to 300 features, when many more training examples from the Wall Street Journal are added, and probably even more when expanding into new domains, but this does not appear to be very critical, because adding new features is easy and requires little time and because the subsequent parse action generation update is fully automatic.

The following subsections explain in detail how features can be defined.

5.6.1 General Feature Structure

The typical structure for a feature is

```
(<predicate> of <path> at <super-hierarchy-level>)
```

E.g. the feature *(tense of pred of -2 at f-finite-tense)* describes the tense of the sub-parse-entry with role *pred* of the second element of the parse stack at hierarchy-level *f-finite-tense.* Possible values for this feature are *f-pres-tense, f-past-tense, f-perf-tense, f-past-perf-tense, f-fut-tense* and *f-fut-perf-tense.*

Values are hierarchical. Consider a parse stack with a parse entry for "to read" at its top. While the value of feature *(synt of -1 at s-synt-elem)*[4] is *s-verb* in this example, the value for feature *(synt of -1 at s-verb)* would be *s-tr-verb*, meaning that the parse entry stands for a transitive verb. The 'super-hierarchy-level' field in the above feature template is hence used to specify the hierarchy level of the value. When evaluating a feature, the system will return the proper value directly below the 'super-hierarchy-level' node, if there is any, and, otherwise, the special value *unavail.*

For parsing, the reference point for the paths is the current position. Many paths just consist of a negative or positive integer, denoting a parse entry on the parse stack or the input list respectively. From any parse entry, one can further navigate into sub-parse-entries using roles, either syntactic or semantic. The feature *(class of pred-compl of co-theme of -1 at i-en-quantity)* for example explores the type of quantity of the predicate complement of the co-theme of the

[4] *s-synt-elem* denotes the top level of any syntactic category

top element on the parse stack. There is no set limit for the length of the path, but reasonable linguistically relevant features tend to have a limited path length, up to 3 in our current system.

Besides roles, the positional path elements *first, last, last-mod* and *last-coord* are available. E.g. *first* accesses the first sub-parse-entry of a parse entry, and *last-mod* accesses the last sub-entry with role *mod.*

The path element *parent* accesses the parent entry. It is useful for describing paths that have their reference points somewhere in a parse entry tree, as is the case for describing features in the transfer module (see chapter 7).

*PRED** designates the 'ultimate' PRED of a parse entry, i.e. the parse entry that is reached by repeatedly descending to sub-entry PRED until this is no longer possible. *npp-1* designates the last noun phrase or noun phrase within a prepositional phrase of a sentence.

Elements on the input list can have several alternatives, typically when a word has several parts of speech. Unless otherwise specified, paths always select the first alternative. Using path elements *alt2, alt3, alt-nom, alt-adv*, e.g. as in *(synt of alt2 of 1 at s-synt-elem)* or *(classp of i-en-agent of alt-nom of 2 at m-boolean)*, select the second, third, nominal or adverbial alternative.

Obviously, some paths don't lead to any parse entry. In that case, the system returns a value of *unavail* for the feature.

5.6.2 Syntactic and Semantic Category Features

*(***synt** *of <path> at <super-hierarchy-level>)* denotes the syntactic class of the parse entry accessed through <path> at <super-hierarchy-level>. *(synt of -1 at s-synt-elem)*, the general part of speech of the top parse stack element, is the most often feature.
*(***class** *of <path> at <super-hierarchy-level>)* provides the same for the semantic class.
*(***syntp** *of <syntactic class> of <path> at m-boolean)* is a boolean feature that holds iff the parse entry at the end of <path> is of the right <syntactic class>.[5]
*(***classp** *of <semantic class> of <path> at m-boolean)* is the corresponding semantic binary feature.

5.6.3 Syntactic and Semantic Role Features

The four features in this subsection draw on the background knowledge of the subcategorization table, as described in section 3.5.

*(***semrole** *of <position1> of <position2>)* is the feature to describe the semantic role of the parse entry at position 1 in a pattern best matched by the parse entry at position 2. If for example the two top parse stack entries are *(The student)* + *(read a book)*, the feature *(semrole of -2 of -1)* would have a value of *agent.*
The corresponding binary feature type *(***semrolep** *of <position1> of <position2>)* checks if the parse entry at position 1 has *some* role in a pattern best matched by the parse entry at position 2. If for example the three top parse stack elements are *(sent) (a book) (to New York)*, the feature *(semrolep of -1 of -3)* would be *true*, indicating that *(to New York)* can have a meaningful role in a sentence with the predicate *to send*, which in turn suggests that *(to New York)* should probably be attached to the verb and not to the preceding noun phrase.

[5]'m-boolean' is the direct super concept of 'true' and 'false'; the 'm' prefix ('mathematical concept') is used to distinguish it clearly from the concept representing the adjective 'boolean'.

(semrole of <syntactic role> of <position2>) describes the semantic role of the argument with <syntactic role> in the pattern best matched by the parse entry at <position2>. If for example the top element of the parse stack is *(ate potatoes)*, the feature *(semrole of subj of -1)* would be *AGENT*. *(semrolep of <syntactic role> of <position2>)* checks whether <syntactic role> is meaningful for the pattern best matched by position 2.

For verbal patterns, the parse entry at <position2> can be a verb, a verb phrase or a sentence. The verbal parse entry can be without any arguments, already have some attached, or even all attached. When assigning values to such role features, the system first compares the verb structure with the patterns in the verb subcategorization table and identifies the pattern that fits best, using a scoring system in which reward and penalty points are given for various types of matches and mismatches of individual components:

- -10,000 for each missing mandatory component
- -1,000 for each spurious primary (i.e. non-advp/pp) component
- +100 for each mandatory component covered before any spurious component
- +70 for each mandatory component covered after spurious advp/pp component
- +30 for each mandatory component covered after spurious primary component
- +10 for each optional component covered before any spurious component
- +7 for each optional component covered after spurious advp/pp component
- +3 for each optional component covered after spurious primary component
- -1 for each spurious app/pp component

The components of each pattern, whether mandatory or optional, have syntactic and semantic roles. A more detailed description of subcategorization tables can be found in section 3.5.

The same holds in principle for nominal patterns, but they are currently not used. "**syntrole**" and "**syntrolep**" are interpreted analogously.

5.6.4 Form Features

These features can be used to access the tense, aspect, person, number, case, gender, mode, voice and mood of a parse entry. The boolean features of examples 1-4 check whether a particular form value is compatible with the parse entry. Examples 5 and 6 can have vales *f-active* or *f-passive* and *f-masc*, *f-fem* or *f-neut* respectively.

1. (f-finite-tense of -1 at m-boolean)
2. (f-part of 1 at m-boolean)
3. (f-pres-part of 1 at m-boolean)
4. (f-third-p of -1 at m-boolean)
5. (voice of -1 at f-voice)
6. (gender of -2 at f-gender)

5.6.5 Other Unary Boolean Features

The following features check whether or not the parse entry referenced by <path> is an abbreviated word, whether it is indexed (antecedent) or a referent (anaphor), whether it is capitalized, or markedly capitalized (i.e. the capitalization is not due to the placement of the word at the beginning of a sentence), or whether or not the parse entry is lexical, i.e. represents a single word or another lexical unit such as *New York*.

1. (is-abbreviation of <path> at m-boolean)
2. (is-indexed of <path> at m-boolean)
3. (is-ref of <path> at m-boolean)
4. (capitalization of <path> at m-boolean)
5. (marked-capitalization of <path> at m-boolean)
6. (lexical of <path> at m-boolean)

5.6.6 Binary Boolean Features

The following features are special in that they involve two different parse entries. They check whether the parse entry pair described by <path1> and <path2> are compatible in terms of number, person and case as subject and verb phrase of a sentence (example 1), whether they are compatible to form a compound verb as auxiliary and main verb (example 2), and whether or not the concepts of the two parse entries are similar in that they both represent agents, places, temporal intervals, or compatible types of quantities (example 3).

1. (np-vp-match of <path1> with <path2> at m-boolean)
2. (compl-v-match of <path1> with <path2> at m-boolean)
3. (similar of <path1> with <path2> at m-boolean)

5.6.7 Other Features

The first two example features show how optional parse entry slots such as *gap* can be used. Notice the abstract path in example 2, which refers to the last parse stack entry that has an active filler. Since the filler status can only be *active-filler*, *saturated-filler* or *unavail*, example 2 is used to check whether there exists an active filler on the parse stack. The third example feature checks, whether an integer is more likely to designate the day of a month (1-31) or a year (e.g. 1997), which can be useful to distinguish between *July 4* and *July 1776*.

1. (gap of -2 at m-filler-status)
2. (gap of active-filler-1 at m-filler-status)
3. (day-or-year of -1 at i-enum-cardinal)

5.7 Training the Parser

The decision structure that is used to parse sentences by deciding what parse action(s) to take next is learnt from parse action examples. The acquisition of these examples is a central task and is done in interactive training with a supervisor.

For each training sentence, the system and the supervisor parse the sentence step by step, starting with segmented and morphologically analyzed words on the input list and an empty stack, e.g. "** The senators left .*"[6] At the beginning, when there are no parse action examples and thus no decision structure, the supervisor has to enter all parse actions by hand. In the given example, the first parse action would be *(S S-ART)*, which shifts *The* from the input list to the parse stack, represented by "*(The) * senators left .*" The representation of the elements on the parse stack uses brackets, because the parse entries can consist of complex parse entries that contain several words.

The complete parse action sequence for the given example is shown in table 5.1. A detailed

Parse state	parse action
* The senators left .	(S S-ART)
(The) * senators left .	(S S-NOUN)
(The) (senators) * left .	(R 1 TO S-NP AS PRED)
(The) (senators) * left .	(R 2 AS DET SAME)
(The senators) * left .	(S S-ADJ)
(The senators) (left) * .	(R 1 TO S-VP AS PRED)
(The senators) (left) * .	(R 2 TO S-SNT AS (SUBJ THEME) SAME)
(The senators left) * .	(S D-DELIMITER)
(The senators left) (.) *	(R 2 AS SAME DUMMY)
(The senators left.) *	(DONE)

Table 5.1: The complete parse action sequence for a very simple sentence. See page 50 for the resulting parse tree.

parse action sequence for a more interesting sentence is shown in appendix C.

The system records the parse action sequence for a trained sentence in a *log file*. Later, the system can automatically generate parse action examples from such log files. This separation is very useful, because it allows examples to be regenerated automatically if the set of features is changed.

After the system records the parse action, it interprets the command, executes it, and displays the resulting parse state. During the training phase, the parse states are displayed in short form, just as indicated in the table above. Internally of course, each element is a full fledged parse entry, or, on the input list, a list of alternative parse entries, e.g. the following for *left*:

[6] As usual, the asterisk * denotes the current position, between the parse stack and the input list.

```
(OR
 "left":
    synt:    S-ADJ
    class:   I-EADJ-LEFT
    forms:   (NIL)
    lex:     "left"
    props:   ((ADJ-TYPE S-NON-DEMONSTR-ADJ))
 "left":
    synt:    S-VERB
    class:   I-EV-LEAVE
    forms:   (((TENSE F-PAST-PART)) ((TENSE F-PAST-TENSE)))
    lex:     "leave"
```

The system repeats recording and interpreting parse actions until it reaches the parse action *done*. For the above example, at that point, the system contains this final parse result on the parse stack:

```
"The senators left.":
    synt:    S-SNT
    class:   I-EV-LEAVE
    forms:   (((PERSON F-THIRD-P) (NUMBER F-PLURAL)
               (CASE F-NOM) (TENSE F-PAST-TENSE)))
    lex:     "leave"
    subs:
    (SUBJ THEME)  "The senators":
        synt:    S-NP
        class:   I-EN-SENATOR
        forms:   (((NUMBER F-PLURAL) (PERSON F-THIRD-P)))
        lex:     "senator"
        subs:
        (DET)  "The":
            synt:    S-DEF-ART
            class:   I-EART-DEF-ART
            forms:   (NIL)
            lex:     "the"
            props:   ((CAPITALIZATION TRUE))
        (PRED)  "senators":
            synt:    S-COUNT-NOUN
            class:   I-EN-SENATOR
            forms:   (((NUMBER F-PLURAL) (PERSON F-THIRD-P)))
            lex:     "senator"
    (PRED)  "left":
        synt:    S-VERB
        class:   I-EV-LEAVE
        forms:   (((TENSE F-PAST-PART)) ((TENSE F-PAST-TENSE)))
        lex:     "leave"
    (DUMMY)  ".":
        synt:    D-PERIOD
        lex:     "."
```

To generate parse action examples from parse action sequences recorded in a log file, the system steps through the parse sequence of a sentence again. At each step, the system first computes the values for all parse features, as described in the previous section. The resulting feature vector plus the parse action provided by the supervisor form the core of the parse action example. Other parse action example components include an identifier, e.g. '*(LOBBY 3 20 "(The) * senators left .")*', which links the example back to a specific partially parsed sentence, e.g. sentence *3* from corpus *LOBBY* step 20.

As described in the next section, parse action examples can be used to build a decision structure, which can be used to predict the parse action for a given parse state. Even with only a few parse action examples, e.g. from a single phrase, such decision structures immensely support further parse action acquisition. For new training sentences, the decision structure proposes the parse action as a default, and for those that the system predicted right, the supervisor can just confirm the default by hitting the return key, thereby no longer having to type it in. The learning curve of the decision structure is initially so steep, that already for the second sentence, the supervisor can expect to enter more than half of the parse actions by confirming the system proposed parse action default.

The more parse action examples are acquired, the better the resulting decision structure will be, so that the supervisor has to overrule the system with decreasing frequency. Soon, the actual work of typing in parse action becomes almost negligible compared to the supervisor's effort to decide what the proper parse action of a sentence should be.

5.8 Learning Decision Structures

Traditional statistical techniques also use features, but often have to sharply limit their number (for trigram approaches to three fairly simple features) to avoid the loss of statistical significance.

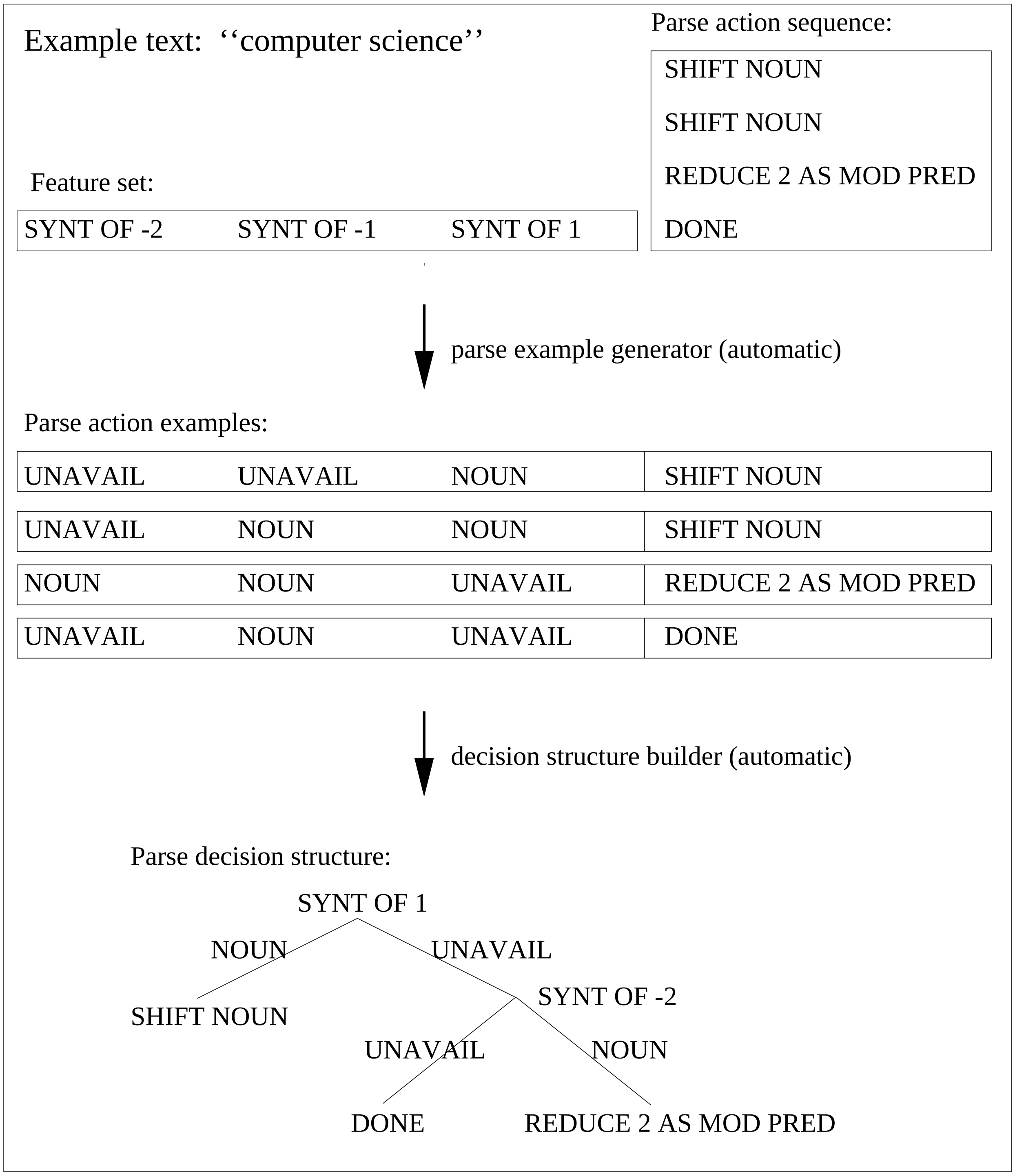
Given that we use more than 200 features, it is obviously very critical to choose a decision structure and corresponding construction algorithm that match the application well by exploiting domain knowledge to limit and/or bias the selection of discriminating features or rules.

In parsing, only a very small number of features are crucial over a wide range of examples, while most features are critical in only a few examples, being used to 'fine-tune' the decision structure for special cases.

In order to overcome the antagonism between the importance of having a large number of features and the need to control the number of examples required for learning, particularly when acquiring examples under supervision, we choose a decision-tree based learning algorithm, which recursively selects the most discriminating feature of the corresponding subset of training examples, eventually ignoring all locally irrelevant features, thereby tailoring the size of the final decision structure to the complexity of the training data. Our decision structure is a hybrid that combines elements of decision trees, decision lists and decision hierarchies.

5.8.1 Review of Decision Trees and Lists

Based on a training set of classified examples, algorithms like ID3 (Quinlan, 1986) and C4.5 (Quinlan, 1993) build so-called *decision trees* that can classify further (yet unclassified) cases. Consider


Example text: ‘‘computer science’’
Parse action sequence:
Feature set:
parse example generator (automatic)
Parse action examples:
decision structure builder (automatic)
Parse decision structure:
SYNT OF 1
NOUN
UNAVAIL
SHIFT NOUN
SYNT OF -2
UNAVAIL
NOUN
DONE
REDUCE 2 AS MOD PRED

Parse action sequence:

SHIFT NOUN
SHIFT NOUN
REDUCE 2 AS MOD PRED
DONE

Feature set:

SYNT OF -2	SYNT OF -1	SYNT OF 1

Parse action examples:

UNAVAIL	UNAVAIL	NOUN	SHIFT NOUN
UNAVAIL	NOUN	NOUN	SHIFT NOUN
NOUN	NOUN	UNAVAIL	REDUCE 2 AS MOD PRED
UNAVAIL	NOUN	UNAVAIL	DONE

Figure 5.4: An example for generating parse action examples and learning a parse decision structure.

the small training set with 14 examples in table 5.2. In each example, the weather is classified as "*Play*" or "*Don't Play*".

Outlook	Temp (°*F*)	Humidity (%)	Windy?	Class
sunny	75	70	true	Play
sunny	80	90	true	Don't Play
sunny	85	85	false	Don't Play
sunny	72	95	false	Don't Play
sunny	69	70	false	Play
overcast	69	90	true	Play
overcast	83	78	false	Play
overcast	64	65	true	Play
overcast	81	75	false	Play
rain	71	80	true	Don't Play
rain	65	70	true	Don't Play
rain	75	80	false	Play
rain	68	80	false	Play
rain	70	96	false	Play

Table 5.2: A small training set; from (Quinlan, 1993).

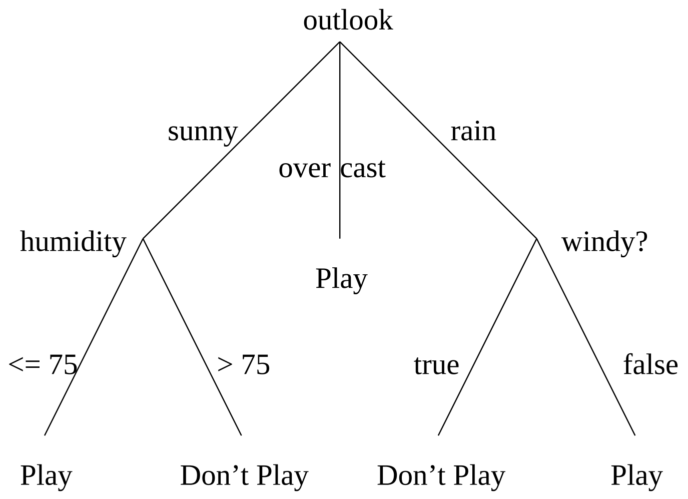


Figure 5.5: Decision tree built from training set shown in figure 5.2

Given the examples of table 5.2, the program C4.5 builds the decision tree shown in figure 5.5. This tree can not only be used to reproduce the classes of the examples used to build it, but also to classify new examples, e.g. (overcast/75°F/40% humidity/not windy) as "*Play*". The algorithm builds the decision tree recursively top down. At each node, it selects the feature with the highest *gain ratio*, and repeats the decision tree building for each value (or value interval) of the selected feature along with the corresponding example subset. The algorithm terminates when all remaining training examples share the same class.

The gain ratio, as defined in table 5.3 is a measure that describes how well a feature divides

$$gain\ ratio(X) = gain(X)/split\ info(X)$$

$$split\ info(X) = -\sum_{i=1}^{n} \frac{|T_i|}{|T|} * log_2 \frac{|T_i|}{|T|}$$

$$gain(X) = info(T) - info_X(T)$$

$$info_X(T) = \sum_{i=1}^{n} \frac{|T_i|}{|T|} * info(T_i)$$

$$info(T) = -\sum_{j=1}^{k} \frac{freq(C_j,T)}{|T|} * log_2 \frac{freq(C_j,T)}{|T|} bits$$

where $freq(C_j, T)$ is the frequency of the *j-th* class in example set T and the T_i's are the different example subsets of T according to partitioning test X.

Table 5.3: Definition of *gain ratio*. For a detailed motivation and explanation of the gain ratio, see (Quinlan, 1993), pp. 20 ff.

the example set into subsets that are to be as homogeneous as possible with respect to their classes. We chose to use the *gain ratio* and not the plain *gain*, because the gain tends to bias feature selection towards features with many values over those with only a few, and because the feature set of our system has features with two to over 30 values, indeed a considerable variation.

In our example, when computing the gain ratios for all four features (outlook, temperature, humidity, windy) at the top level, the highest gain ratio turns out to be for the 'outlook' feature, which partitions the example set into subsets for 'sunny', 'overcast', and 'rain'. All examples for 'overcast' are classified as 'Play', so the subtree for 'overcast' is a single node marked 'Play'. The other two example subsets have to be split one more time before all example subsets share the same class.

Alternatively, decision list algorithms (Rivest, 1987) generate a list of conjunctive rules, where rules are tested in order and the first one that matches an instance is used to classify it. Figure 5.6 shows the resulting decision list for the same weather training set.

5.8.2 Decision Hierarchies

Our first extension to the standard algorithms for decision trees and decision lists is the concept of a decision hierarchy. Recall that while parse actions might be complex for the action interpreter, they are atomic with respect to the decision structure learner; e.g. "(R 2 TO VP AS PRED (OBJ PAT))" would be such an atomic *classification*. However, different parse actions are often related and apply in very similar contexts. Consider for example the parse actions *(R 2 TO S-SNT AS (SUBJ AGENT) SAME)* and *(R 2 TO S-SNT AS (SUBJ THEME) SAME)*. Both parse actions are used in a very similar context: a noun phrase and a verb phrase are combined into a sentence,

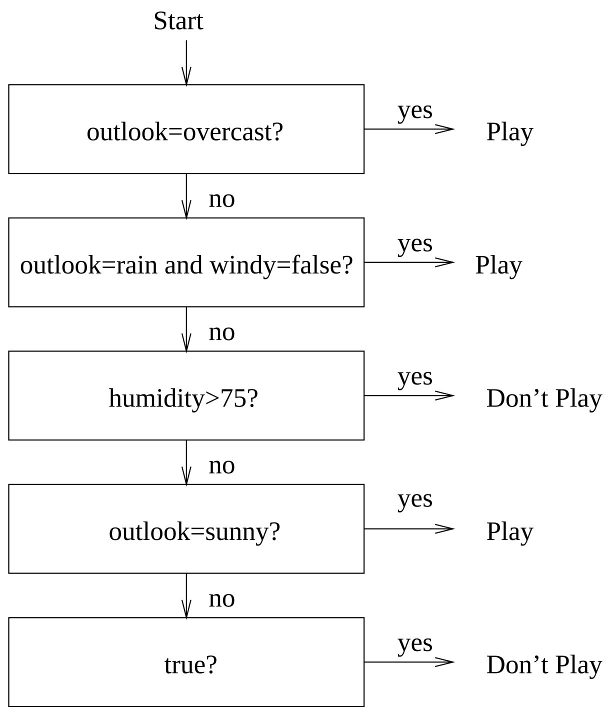


Figure 5.6: Decision list built from training set shown in figure 5.2

which then contains all the components of the verb the **same** way as in the verb phrase, plus the noun phrase as a subject. The only difference is that in the first case, the subject plays the semantic role of 'agent', whereas in the second case it plays the semantic role of 'theme'.

The knowledge of this type of similarity of parse actions can be exploited by defining sets of *similarity classes* for parse actions, e.g. the set of all parse actions that reduce a noun phrase and a verb phrase to a sentence with the noun phrase as the subject, regardless of the subject's semantic role. Decision structures can then be constructed in two steps. During the first step, a decision structure is built as usual except that all parse actions belonging to the same similarity class are treated as if the were the same. In a second step, a special decision structure is built for the parse actions within each similarity class.

This allows the decision structure to split up the classification process into a coarse classification and a fine classification step as illustrated in figure 5.7. The advantage of this approach is that the system can 'bundle' examples that are associated with different leaves of the coarse decision structure and thereby gain a stronger example base for the fine classification. In principal, a decision structure can be split up into an arbitrary number of levels of such decision hierarchies. The similarity classes are defined by predicates on parse actions; all parse actions for which the predicate holds are defined to be members of the corresponding similarity class. The predicates are defined manually. In our best-performing decision structure, we use two such decision hierarchy similarity classes: the one already mentioned in the preceding example and one which contains all parse actions that reduce a preposition and a noun phrase into a preposition phrase.

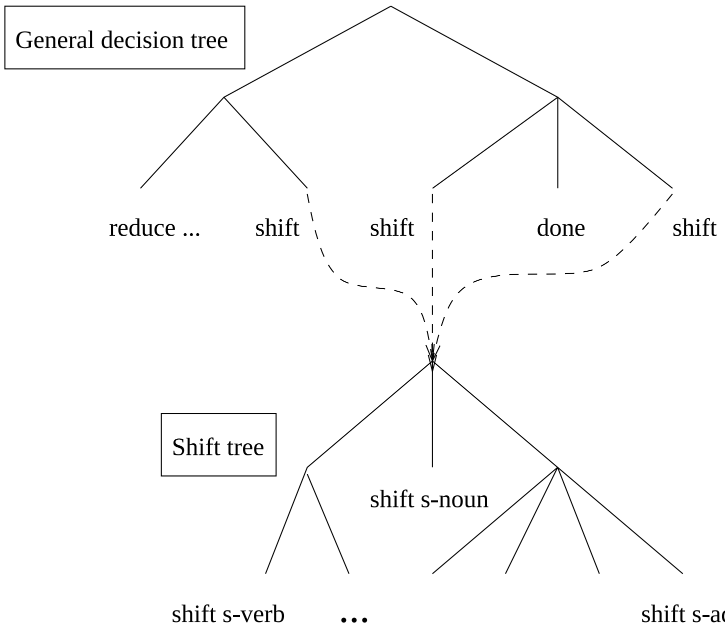


Figure 5.7: A two level decision hierarchy, where all shift parse actions have been grouped together.

5.8.3 Syntax of Decision Structure Definitions

It is not only possible to define decision trees of decision trees, as illustrated in figure 5.7, but in fact any decision structure (e.g. trees or lists) can have any type of sub-decision-structure, at any number of levels. In order to describe such a complex multi-level hybrid structure, we introduce a formal syntax:

Entity	Definition	Comment
PRED:	<a parse action predicate>	
DSTRUCT:	(PRED {tree *DSTRUCT**})	decision tree
	(PRED dlist *DSTRUCT**)	decision list
	(PRED slist *DSTRUCT*$^+$)	decision structure list

Table 5.4: Syntax of decision structures.

Parse action predicates ("PRED") are implemented functions that map parse actions to true or false. Examples are *any-operation-p*, which holds for all parse actions; *reduce-2-to-snt-operation-p*, which holds for all parse actions of the form *(R 2 TO S-SNT ...)*; and *reduce-2-to-lexical-operation-p*, which only holds for the single parse action *(R 2 TO LEXICAL)*.

A decision structure ("DSTRUCT") is a hybrid multi-layer structure composed of decision trees, lists, and structure lists (as further explained in the following subsection). Decision trees and lists can have any type of decision structure as a sub-component; for decision structure lists, the specification of such sub-components is required. The top level predicate (PRED) should be the all inclusive *any-operation-p*. Table 5.5 gives examples of decision structure definitions.

definition	description
(any-operation-p)	simple (non-hierarchical) decision tree
(any-operation-p tree)	same as above
(any-operation-p dlist)	simple decision list
(any-operation-p tree (reduce-operation-p) (shift-operation-p))	defines a hierarchical decision tree with sub-decision-trees for reduce and shift parse actions
(any-operation-p dlist (reduce-operation-p dlist) (shift-operation-p))	defines a hierarchical decision list with a sub-decision-list for reduce parse actions a sub-decision-tree for shift parse actions
(any-operation-p slist (done-operation-p) (reduce-operation-p) (shift-operation-p))	defines the decision structure list shown in figure 5.8

Table 5.5: Examples of decision structures. Note: 'any-operation-p', 'shift-operation-p', 'reduce-operation-p', and 'done-operation-p' are predefined predicates for similarity classes, containing all parse actions, all shift parse actions, all reduce parse actions, and all done parse actions respectively.

5.8.4 Decision Structure Lists

The final extension to decision trees and lists we introduce is the *decision structure list*. Recall that the major motivation for decision hierarchies was that we wanted to exploit our knowledge that some of the currently 265 different parse actions are quite related to each other; using decision hierarchies, we can split the decision structure learning into a coarse and a fine learning part with the advantage that the examples within a similarity class, no matter where in a coarser tree they might end up, are bundled back together, thereby providing a larger example base for 'fine-tuning'.

Another piece of linguistic background knowledge that we can exploit is the notion of exceptionality vs. generality of parse actions. Natural language exhibits a complex pattern of regularities, sub-regularities, pockets of exceptions, and idiosyncratic exceptions. Consider for example the noun phrase "New York exchanges". We don't want to treat "New" like a normal adjective, and in particular, we don't want to attach it as a standard modifier to the noun-compound "York exchanges". Instead, we want to take advantage of a special lexicon entry for "New York" and perform a special reduction ("R 2 TO LEXICAL") that is principally lexically motivated. Since parse actions often reflect this notion of degree of exceptionality, we can steer the more exceptional types of parse actions towards an early treatment, so that they later don't 'confuse' the more normal parse actions like shift and reduce. We achieve this by defining a list of similarity classes that we order such that more exceptional types of parse actions come first and more standard ones later. Figure 5.8 gives an example. The system first separates the relatively special 'done' operations from the rest of the flock, and then proceeds with 'reduce' operations before finally dealing with the fairly standard 'shift' operations. When building the decision structure, all 'done' examples are discarded when subsequent decision structure list sub-structures are built. We basically aim for a decrease of exceptional examples that are likely to 'pollute' the patterns of more regular types of parse actions.

Definition of Decision Structure Lists

More formally, a *decision structure list* is a list of element decision structures each of which classifies examples by assigning normal classes or the special class *OTHER*. If an element decision structure assigns a normal class, that class also becomes the class of the entire decision structure list; otherwise the example is classified by the next element decision structure in the decision structure list. This is repeated until the example is assigned a normal class. The last element decision structure should never assign a class *other*, because it would be undefined. As mentioned before, figure 5.8 gives an example of a decision structure list; it is generated from the definition (any-operation-p slist (done-operation-p) (reduce-operation-p) (shift-in-operation-p)).

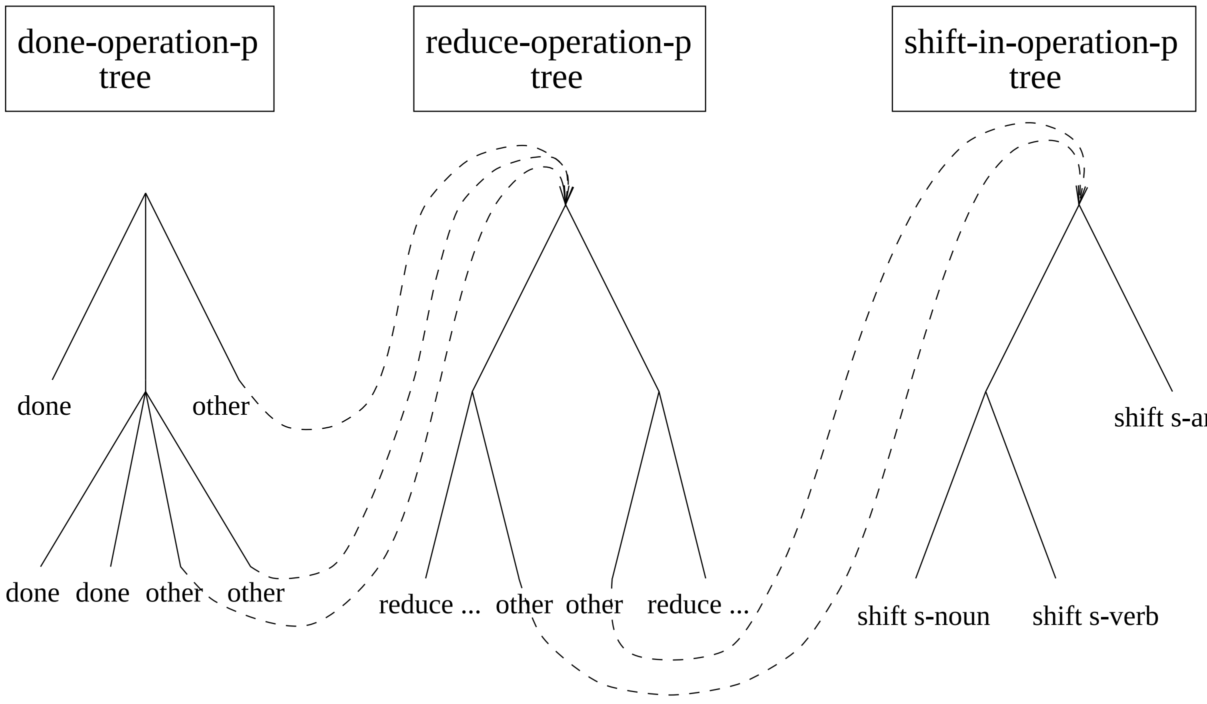


Figure 5.8: A decision structure list

A decision structure list is defined by a tuple (PRED slist $DSTRUCT_1$, $DSTRUCT_2$, ... $DSTRUCT_n$), where n must be at least one, but should be at least two to make any sense. Each sub-structure $DSTRUCT_i$ is headed by a predicate, which we shall refer to as $PRED_i$.

For each $PRED_i$, the decision structure list algorithm produces an element decision structure according to the type of $DSTRUCT_i$, but with the modification that all parse actions for which $PRED_i$ does not hold are classified as *OTHER*, a special class that is linked to the following element decision structure of the decision structure list.

Differences between Decision Structure Lists and Decision Hierarchies

The decision hierarchies and the decision structure lists share the concept of similarity classes and both build a larger acyclic graph out of smaller decision structures. But while decision hierarchies combine all classes *within* a similarity class, decision structure lists combine all classes *outside* the similarity class. And while the multiple similarity classes within a decision hierarchy will cause their corresponding sub-decision-structures to be arranged side by side, the various elements of a decision structure list are arranged in sequence.

Example of a Hybrid Decision Structure

Figure 5.9 shows a hybrid decision structure that is similar to the one depicted in figure 5.8. It introduces an additional level of decision hierarchy under the reduce-operation-p tree, reducing the number of different classifications in that tree to two, ‘reduce’ and ‘other’. Since there is only a single ‘done’ operation anyway, a similar sub-tree wouldn’t have made sense for the done-operation-p tree.

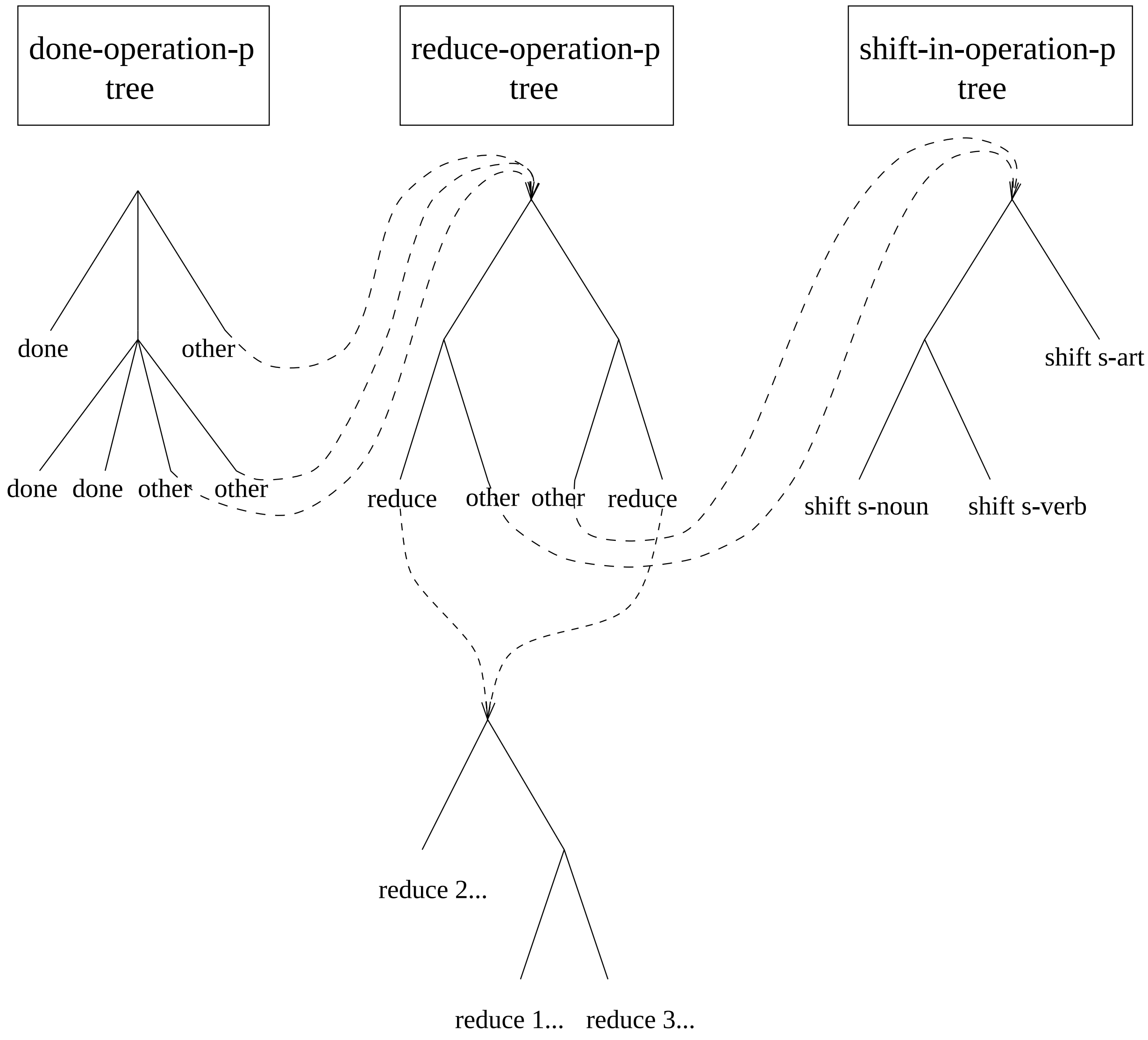


Figure 5.9: Example of a hybrid decision structure

The Hybrid Decision Structure Used For Parsing

The decision structure list shown in table 5.6 defines the hybrid decision structure used in our parsing experiments. The structure has been constructed manually, following two principles:

1. Group similar parse actions together, e.g. all 'co-index' parse actions or all those of the form "(r 2 to s-pp ...)".

2. Sort the resulting groups such that the more exceptional and specialized ones precede the more general groups.

Since the sub-decision-structures are built sequentially, with all examples covered by a previously listed predicate discarded, the more exceptional examples are no longer around when the more general sub-decision-structures are built.

```
(any-operation-p slist
   (reduce-2-to-lexical-operation-p)
   (reduce-3-to-lexical-operation-p)
   (reduce-5-to-lexical-operation-p)
   (done-operation-p)
   (expand-operation-p)
   (co-index-operation-p)
   (add-operation-p)
   (empty-cat-operation-p)
   (mark-operation-p)
   (shift-out-operation-p)
   (reduce-4-operation-p)
   (reduce--3--1-operation-p)
   (reduce--4--1-operation-p)
   (reduce-non-contiguous-operation-p)
   (reduce-3-as-dummy-pred-operation-p)
   (reduce-3-as-comp-conj-pred-operation-p)
   (reduce-3-as-same-operation-p)
   (reduce-3-operation-p)
   (reduce-1-operation-p)
   (reduce-2-as-pred-or-same-dummy-operation-p)
   (reduce-2-to-pp-operation-p tree
     (reduce-2-to-pp-operation-p))
   (reduce-2-to-snt-operation-p tree
     (reduce-2-to-snt-operation-p))
   (reduce-2-operation-p)
   (shift-in-operation-p))
```

Table 5.6: Definition of the hybrid decision structure used in parsing experiments.

Based on the decision structure list definition in table 5.6, the system first builds a decision tree (default) that classifies all examples as some *reduce-2-to-lexical-operation-p* parse action or

OTHER. Then, using all examples that have previously been classified as other, it builds the next decision tree and so forth. The union of decision structure list parse action classes should always be the full set of parse examples, but the classes are allowed to overlap.

Note that the predicates partitioning the examples are not necessarily disjoint. In particular, all examples covered by *reduce-3-as-comp-conj-pred-operation-p* are also covered by *reduce-3-operation-p*, but this is no problem, because all examples by the former are filtered out, so that the sub-decision-structure for *reduce-3-operation-p* is trained only on all remaining examples. It is important however that the disjunction of all sub-decision-structure predicates cover the entire example space.

Notice further the double occurrence of *reduce-2-to-pp-operation-p*. The first occurrence causes all **non**-*reduce-2-to-pp-operation-p* parse actions to be lumped together as *"other"*, whereas the second occurrence groups all *reduce-2-to-pp-operation-p* parse actions together. As a result, the corresponding decision tree has exactly two classifications: a token representing *reduce-2-to-pp-operation-p*, and *"other"*. A lower level decision tree then separates the different parse actions within *reduce-2-to-pp-operation-p*.

After initially obtaining a significant improvement using a decision structure list along the two principles outlined above, further variation did not yield significant additional improvement. This seems to indicate that the two basic principles above have merit, but that the results don't depend very much on the detailed configuration of the decision structure list.

5.8.5 Feature Selection Bias in Decision Trees

In parsing, certain features are much more specialized than others. For example, consider the feature *(CLASSP OF I-EN-INTERR-PRONOUN OF -2 AT M-BOOLEAN)*, which checks whether or not the second element on the parse stack is an interrogative pronoun, and *(SYNT OF -2 AT S-SYNT-ELEM)*, which checks the general part of speech of the same element. For better generalization when building decision trees and also for a potentially faster learning time, we want to restrict or discourage the selection of very specialized features before certain more general features have already been used higher up in the decision structure. For this we define a feature dependency graph, which for each feature specifies which other features should have been used before when building a sub-decision-structure.

In our system, the use of a feature is discouraged, if the feature(s) describing the general part(s) of speech of the one or more parse stack or input list elements that are accessed in the feature have not been used yet. The use of feature *(CLASSP OF I-EN-INTERR-PRONOUN OF -2 AT M-BOOLEAN)* for example would be discouraged until *(SYNT OF -2 AT S-SYNT-ELEM)* has been used.

Features checking the general part of speech are discouraged if there is still a feature for the part of speech of some element closer to position 0 that has not been used yet. This means that the only features that are always without any such bias are *(SYNT OF -1 AT S-SYNT-ELEM)* and *(SYNT OF 1 AT S-SYNT-ELEM)*. Features that have the same value for all remaining examples have an information gain of 0 and are considered to have been used, regardless of whether or not they actually have been.

Features are discouraged by dividing the "raw" *gain-ratio* by

$$1 + c_1 * d^{c_2}$$

where d is the degree of dependency violation (e.g. 2 for *(SYNT OF -4 AT S-SYNT-ELEM)* if (SYNT OF -1 AT S-SYNT-ELEM), but neither (SYNT OF -2 AT S-SYNT-ELEM) nor (SYNT OF -3 AT S-SYNT-ELEM) had been used before. c_1 and c_2 are constants.

Experiments varying the strength of this bias towards more general features by choosing different values for c_1 and c_2, from 'no bias' ($c_1 = 0$) over several intermediate values to 'very strong bias' (high values for both c_1 and c_2) have shown that stronger generalization biases produce better decision structures. In fact, the extreme bias of not considering any feature with any dependency violation leads to the best results. This allows the decision tree builder to ignore many (yet too) specialized features before even computing a gain ratio, thus considerably accelerating the decision tree building process.

5.8.6 Feature Selection Preferences

The supervisor can optionally mark a parse example with one or more features indicating that these are relevant for the specific example. Later on, this information can be used to bias the feature selection towards those features which have been marked as relevant in the example set that a remaining decision tree is built from. This bias is implemented by increasing the "raw" gain ratio in a way similar to the one we described in the last subsection.

To a degree this option is a remnant of our previous approach of collecting, from selected examples, features that were explicitly designated as relevant, instead of full feature vectors for all examples. Out of a total of 11,822 parse action examples, 702 or 5.9% contain such individual feature selection preference indications, with a total of 812 preference indications.

Simple decision trees, simple decision lists, hierarchical decision trees and hybrid decision structures containing hierarchical decision trees were all used in our parsing experiments. The results are compared in section 6.4.4.

5.9 Garden Paths

Garden path sentences are sentences that initially mislead the reader in their syntactic analysis. Classical examples (Marcus, 1980) are:

1. The horse raced past the barn fell.
2. Cotton clothing is made of grows in Mississippi.

Since local parsing preferences, e.g. the preference to interpret *raced* in example (1) as a finite verb or to interpret *Cotton clothing* in example (2) as a compound noun, aren't recognized as incorrect until much later in a left-to-right deterministic parse, these garden path sentences present a difficulty, which is not surprising, because the deterministic parser basically mimics a human reader in the way it processes sentences and human readers experience these problems as well.

With regard to attachment decisions, our parser follows a relatively 'conservative' policy and makes the attachments relatively late (because the supervisor trained it that way). For example, it postpones reducing subject and verb until the entire verb phrase has been processed. This allows it to circumnavigate the trap laid out by garden path sentence (1). In other garden path sentences, such as (2), where our system will interpret *Cotton clothing* as a complex noun, a locally preferred choice later turns out to be wrong.

Good writers can avoid garden path sentences. "The horse *which was* raced past the barn fell." and "Cotton *that* clothing is made of grows in Mississippi." read much more easily. Poor writing will never disappear, but garden path sentences are rare enough in practical environments that quantitatively their nuisance is minor compared to other parse errors.

Nevertheless there are solutions to cope with garden path sentences in a deterministic parsing paradigm. The system can learn to recognize *when* it has been misled, even if it is already too late for the 'proper' parse. Many garden path sentences are based on a few types of structural ambiguity, caused for example by phonologically unrealized relative pronouns. When training a parser on garden path traps, the parser could either be guided to directly repair the current partial path, which would typically have to include some sort of 'unreduce' operations, or it might somehow mark the trouble spot and use this mark when reaching the critical point of the sentence again after a parsing restart. This method certainly somewhat breaks out of the normal deterministic paradigm, but that should be acceptable, because human readers have to resort to this as well.

During the testing of parsing and translation of WSJ sentences, the parser in one case actually followed a wrong path: *The Federal Farm Credit Banks Funding Corp. plans to offer $1.7 billion of bonds Thursday.* It is questionable whether or not this is a true garden path sentence, since the local choice of whether *plans* is a noun or a verb might have a preference for a verbal interpretation anyway; however, even though the system recognizes a part of speech ambiguity for *plans*, it chooses the nominal interpretation, because none of the comparable training instances that the parser was trained on called for a verbal interpretation. (When training the system on 256 sentences, which includes 40 examples where a new word is shifted in that could be both a noun or a verb and that has a noun at position -1, i.e. at the top of the parse stack, all those 40 examples select *noun* as the proper part of speech.)

Chapter 6

Parsing Experiments

This chapter presents results on training and testing a prototype implementation of our system with sentences from the Wall Street Journal, a prominent corpus of 'real' text, as collected on the ACL-CD.

6.1 Corpus

In order to limit the size of the required lexicon, we work on a reduced corpus that includes all those sentences that are fully covered by the 3000 most frequently occurring words (ignoring numbers etc.) in the entire corpus. The lexically reduced corpus contains 105,356 sentences, a tenth of the full corpus. 3000 words is typically considered to be the size of the basic vocabulary of a language. The lexical size is small enough to build a lexicon and corresponding KB entries with moderate tools in reasonable time, while still allowing a sufficiently large number of sentences that still include a rich linguistic diversity.

For our training and testing we use the first 272 of the 105,356 sentences. They vary in length from 4 to 45 words, averaging at 17.1 words and 43.5 parse actions per sentence. These 272 sentences are from a section of the Wall Street Journal dated March 23/24, 1987. One of these sentence is "*Canadian manufacturers' new orders fell to $20.80 billion (Canadian) in January, down 4% from December's $21.67 billion on a seasonally adjusted basis, Statistics Canada, a federal agency, said.*". A complete listing of the 272 training and testing sentences can be found in appendix A.

6.2 Test Methodology and Evaluation Criteria

As a result of supervised acquisition, as described in section 5.7, the correct parse action sequences for the 272 WSJ sentences have been recorded. For the following parsing test series, the corpus of these 272 sentences is divided into 17 blocks of 16 sentences each. The 17 blocks are then consecutively used for testing. For each of the 17 sub-tests, a varying number of sentences from the *other* blocks is used for training the parse decision structure, so that within a sub-test, none of the training sentences are ever used as a test sentence. The results of the 17 sub-tests of each series are then averaged. Such a test is also referred to as a 17-fold cross-validation.

The following standard (Goodman, 1996) evaluation criteria are used:

Precision $= \frac{C_{system}}{N_{system}}$ **Labeled precision** $= \frac{L_{system}}{N_{system}}$

Recall $= \frac{C_{system}}{N_{logged}}$ **Labeled recall** $= \frac{L_{system}}{N_{logged}}$

where
N_{system} = number of constituents in system parse
N_{logged} = number of constituents in logged parse
C_{system} = number of correct constituents in system parse
L_{system} = number of correct constituents with correct syntactic label in system parse

Tagging accuracy: percentage of words with correct part of speech assignment.
Crossing brackets: number of constituents in system parse which violate constituent boundaries with a constituent in the logged parse.
Correct operations measures the number of correct operations during a parse that is continuously corrected based on the logged sequence. A sentence has a correct *operating sequence*, **OpSequence**, if the system fully predicts the logged parse action sequence, and a correct *structure and labeling*, **Struct&Label**, if the structure and syntactic labeling of the final system parse of a sentence is 100% correct, regardless of the operations leading to it.

6.3 Accuracy on Seen Sentences

The current set of 205 features was sufficient to always discriminate examples with different parse actions. This guarantees an 100% accuracy on sentences already seen during training. While that percentage is certainly less important than the accuracy figures for unseen sentences, it nevertheless represents an upper ceiling, which for many statistical systems lies significantly below 100%.

6.4 Testing on Unseen Sentences

The first test series was done with a varying number of training sentences. Here, as well as in all other test series, results are for training with all 205 features and using the hybrid decision structure list described in subsection 5.8.4.

In some test cases, parsing could not be fully completed. This can happen, when the decision structure prematurely selects *"done"* as the next parse action. This happens in particular when the system proposes a parse action that is actually undefined in the specific current parse state and therefore overruled by the sanity checker; if the decision structure can naturally provide an alternative legal parse action, e.g. in a decision list, by proceeding through the remainder of the list, that action is chosen; otherwise, the system resorts to *"done"* as a last resort. In that case, all elements on the parse stack and input list are lumped together under a new parse entry node with the generic syntactic label *S-SYNT-ELEM* and the generic semantic label *I-EN-THING* and with the unspecified roles *CONC* for its components. This way, there is at least formally a parse tree that can be evaluated or passed on to transfer for full translation.

The other anomaly is the 'endless' loop in which the system spins into a repetitive parse action sequence, in which the parser for example keeps inserting empty categories, or keeps reducing the same single parse entry over and over again. This occurs only very rarely when the system has been trained on sufficient data, an adequate feature set and an appropriate type of decision structure. Nevertheless, the parser is equipped with an 'endless' loop detector, that basically measures parsing progress in terms of the number of elements on the parse stack and input list, which gradually decreases, though not necessarily monotonically. By allowing up to four more parse steps for each remaining parse stack or input list element, the 'endless' loop detector was able to detect all loops fairly quickly and never terminated any parsing sequence that was in fact still promising. When an 'endless' loop is detected, the parsing is stopped by selecting the parse action *"done"* and by then proceeding as described in the previous paragraph.

These pathological cases are fully included in the following test data, not surprisingly with an overall negative impact. The number of test sentences caught in an 'endless' loop is explicitly included in the following tables. Prematurely terminated sentences obviously produce a lower recall score and have no chance at scoring as a fully correct operation sequence or even produce a parse tree with correct structure and labeling. However they can also lead to lower crossings, since the constituent structure is often still incomplete.

Table 6.1 clearly shows a continuous positive trend for all criteria. The accuracy for an individual

Number of training sentences	16	32	64	128	256
Precision	85.1%	86.6%	87.7%	90.4%	92.7%
Recall	82.8%	85.3%	87.7%	89.9%	92.8%
Labeled precision	77.2%	80.4%	82.5%	86.6%	89.8%
Labeled recall	75.0%	77.7%	81.6%	85.3%	89.6%
Tagging accuracy	96.6%	96.5%	97.1%	97.5%	98.4%
Crossings per sentence	2.5	2.1	1.9	1.3	1.0
Sent. with 0 crossings	27.6%	35.3%	35.7%	50.4%	56.3%
Sent. with up to 1 crossing	44.1%	50.7%	54.8%	68.4%	73.5%
Sent. with up to 2 crossings	61.4%	65.1%	66.9%	80.9%	84.9%
Sent. with up to 3 crossings	72.4%	77.2%	79.4%	87.1%	93.0%
Sent. with up to 4 crossings	84.9%	86.4%	89.7%	93.0%	94.9%
Correct operations	79.1%	82.9%	86.8%	89.1%	91.7%
Sent. with correct OpSequence	1.8%	4.4%	5.9%	10.7%	16.5%
Sent. with correct Struct&Label	5.5%	8.8%	10.3%	18.8%	26.8%
Sentences with endless loop	13	6	0	1	1

Table 6.1: Evaluation results with varying number of training sentences; with 205 features

parse action is 91.7% for 256 training sentences. Even though the number appears to be good at first glance, the fact that there are an average of 43.5 parse actions per sentence means, that, assuming independence of parse action errors, the probability of a completely correct operations sequence for an average length sentence is as low as $.917^{43.5} = 2.3\%$. The much higher percentages of test sentences with correct operation sequence or at least correct structure and labeling already show that this independence assumption fortunately does not hold. An individual analysis of parsing

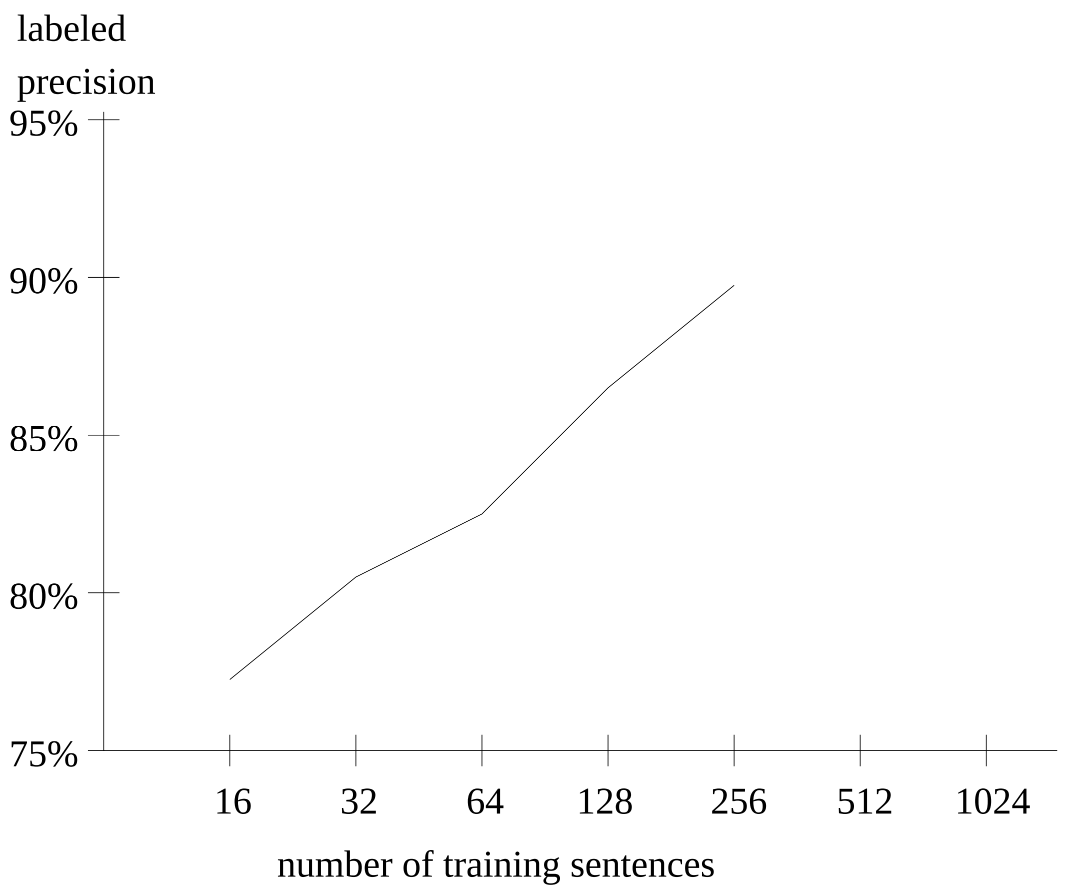


Figure 6.1: Learning curve for labeled precision (see table 6.1)

Number of training sentences	16	32	64	128	256
Precision	84.6%	86.6%	88.3%	90.5%	91.5%
Recall	82.3%	85.2%	88.1%	90.3%	91.6%
Labeled precision	76.5%	79.6%	82.9%	86.2%	87.9%
Labeled recall	74.0%	77.9%	82.2%	85.4%	87.7%
Tagging accuracy	96.5%	96.7%	97.0%	97.9%	98.2%
Crossings per sentence	2.6	2.1	1.7	1.4	1.2
Sent. with 0 crossings	26.1%	33.8%	39.3%	46.3%	53.3%
Sent. with up to 1 crossing	43.8%	50.7%	57.4%	67.7%	69.5%
Sent. with up to 2 crossings	60.7%	64.0%	72.4%	80.9%	81.3%
Sent. with up to 3 crossings	73.9%	76.9%	84.2%	87.9%	90.8%
Sent. with up to 4 crossings	80.9%	85.7%	90.8%	91.9%	94.1%
Correct operations	78.3%	81.9%	85.8%	88.5%	90.7%
Sent. with correct OpSequence	2.2%	1.5%	5.1%	10.3%	14.3%
Sent. with correct Struct&Label	3.7%	6.6%	12.1%	17.3%	22.1%
Sentences with endless loop	7	15	2	0	1

Table 6.2: Evaluation results without using relevant feature information from individual examples

errors shows that many mistakes are due to encountering constructions that just have not been seen before at all, typically causing several erroneous parse decisions in a row. This observation also supports our hope that with more training sentences, the accuracy for unseen sentences will still rise significantly.

6.4.1 Contribution of Individual Feature Selection Preferences

Recall that the supervisor can optionally mark a parse example with one or more features indicating that these are relevant for the specific example, as already described in subsection 5.8.6. Table 6.2 shows the relatively modest contribution of these individual feature selection preferences. Without these individual feature preferences, the parse action accuracy drops by about 1%, regardless of training size.

6.4.2 Contribution of the Subcategorization Table

Table 6.3 shows the contribution of the subcategorization table. The 10 features that access the

Number of training sentences	16	32	64	128	256
Precision	85.1%	86.4%	88.6%	90.6%	92.4%
Recall	83.1%	85.0%	88.2%	90.5%	92.4%
Labeled precision	78.1%	79.8%	83.5%	86.9%	89.8%
Labeled recall	75.2%	77.6%	82.1%	85.8%	89.3%
Tagging accuracy	96.4%	96.5%	97.1%	97.9%	98.3%
Crossings per sentence	2.6	2.2	1.8	1.3	1.1
Sent. with 0 crossings	28.7%	32.4%	37.9%	48.5%	57.0%
Sent. with up to 1 crossing	42.7%	49.3%	57.4%	67.7%	73.2%
Sent. with up to 2 crossings	60.3%	64.7%	70.2%	80.9%	84.6%
Sent. with up to 3 crossings	75.4%	77.9%	81.3%	87.9%	91.5%
Sent. with up to 4 crossings	83.1%	87.1%	89.0%	94.1%	95.2%
Correct operations	78.8%	82.0%	85.6%	87.6%	90.1%
Sent. with correct OpSequence	1.5%	3.3%	4.0%	7.7%	10.3%
Sent. with correct Struct&Label	5.5%	7.4%	10.7%	20.2%	27.9%
Sentences with endless loop	13	8	0	1	2

Table 6.3: Evaluation results without subcategorization features

subcategorization module have been discarded in this test series. While the figures for precision, recall, tagging accuracy and crossings hardly degrade at all, we observe a significant deterioration in correct operations, however not in structure and label correctness of the final parse tree. This means that so far the primary benefit from the subcategorization table lies in the assignment of proper roles to phrase components, which on one side is not a surprise, because the subcategorization table quite explicitly lists which semantic roles are to be assigned to various syntactic components. On the other side this means that the contribution towards the resolution of structural ambiguity does not depend as much on the subcategorization table as might have been expected.

6.4.3 Contribution of Rich Context

The following tables (6.4, 6.5, 6.6, 6.7, 6.8) show the impact of reducing the feature set to a set of n core features. We always chose those n features that appeared to be the most important n features, relying on older smaller versions of feature lists and general experience acquired during the development process. When 25 or fewer features are used, all of them are of a syntactic nature.

As table 6.4 show, the reduction of features to 100 has no significant impact on any criterion when

Number of training sentences	16	32	64	128	256
Precision	85.4%	86.4%	88.5%	90.1%	91.7%
Recall	83.1%	85.6%	88.5%	89.7%	91.7%
Labeled precision	77.7%	79.8%	83.4%	85.8%	88.6%
Labeled recall	75.1%	78.5%	82.3%	85.1%	88.1%
Tagging accuracy	96.5%	96.5%	96.9%	97.4%	98.2%
Crossings per sentence	2.5	2.3	1.8	1.4	1.1
Sent. with 0 crossings	26.8%	32.4%	38.6%	47.8%	54.0%
Sent. with up to 1 crossing	43.0%	51.1%	57.0%	68.4%	72.1%
Sent. with up to 2 crossings	62.5%	63.2%	71.7%	80.9%	84.2%
Sent. with up to 3 crossings	73.5%	75.0%	81.6%	87.5%	92.3%
Sent. with up to 4 crossings	84.2%	84.2%	89.3%	91.2%	94.5%
Correct operations	78.9%	82.8%	86.6%	88.7%	90.7%
Sent. with correct OpSequence	1.8%	4.8%	5.1%	8.1%	13.6%
Sent. with correct Struct&Label	5.9%	8.5%	9.6%	15.4%	23.5%
Sentences with endless loop	11	1	2	2	2

Table 6.4: Evaluation results using only 100 features

training with under 100 sentences. When training on 256 sentences, precision and recall drop 1% and compound test characteristics, i.e. the percentage of sentences with a totally correct operation sequence, or at least the correct final structure and labeling, decrease by about 3%.

Cutting the number of features in half again, table 6.5, basically continues this trend. Compound test characteristics start to deteriorate for medium numbers of training sentences and, for training on 256 sentences, are only little more than half of when using all features.

When using 25 (or fewer features), table 6.6, all remaining features are syntactic. Crossing brackets have increased significantly and now only little over 1% of the sentences achieve a perfect operation sequence.

With only 12 features, table 6.7, we notice a dramatic increase of sentences whose parsing has to be prematurely stopped because loops have been detected. This problem also causes a multitude of bad nodes, as particularly manifested in low labeled precisions; such bad nodes are often 'hordes' of incorrect empty category tokens etc.

Cutting the number of features one final time, table 6.8, produces relatively little further degradation. Let us now compare the results when using 6 features to those using the full complement of 205. For small training sizes, e.g. 6, we see only a very modest drop in precision (from 85.1% to 83.8%) and even a slight increase of labeled recall (from 75.0% to 77.4%), no loss of tagging accuracy (still 96.6%), a modestly higher number of crossings (from 2.5 per sentence to

Number of training sentences	16	32	64	128	256
Precision	84.6%	85.7%	87.9%	89.6%	90.8%
Recall	82.6%	85.2%	88.0%	89.4%	90.8%
Labeled precision	76.9%	79.0%	82.5%	84.7%	87.2%
Labeled recall	75.1%	78.0%	82.2%	84.2%	86.9%
Tagging accuracy	96.4%	96.4%	97.1%	97.3%	98.1%
Crossings per sentence	2.7	2.3	1.9	1.5	1.3
Sent. with 0 crossings	25.4%	30.9%	36.0%	46.0%	50.4%
Sent. with up to 1 crossing	42.3%	48.2%	56.6%	62.1%	70.6%
Sent. with up to 2 crossings	59.6%	65.8%	69.5%	78.7%	80.5%
Sent. with up to 3 crossings	72.8%	76.5%	80.9%	84.9%	88.6%
Sent. with up to 4 crossings	82.7%	83.5%	88.6%	91.5%	93.8%
Correct operations	78.5%	81.7%	85.3%	87.0%	88.9%
Sent. with correct OpSequence	2.2%	3.7%	4.4%	7.0%	8.8%
Sent. with correct Struct&Label	5.5%	7.4%	8.5%	12.9%	15.1%
Sentences with endless loop	4	0	0	2	2

Table 6.5: Evaluation results using only 50 features

Number of training sentences	16	32	64	128	256
Precision	84.7%	85.7%	86.8%	87.4%	88.7%
Recall	82.3%	84.3%	86.4%	87.4%	88.7%
Labeled precision	76.9%	78.5%	77.0%	82.4%	86.7%
Labeled recall	74.5%	76.9%	79.8%	81.8%	84.1%
Tagging accuracy	96.3%	96.4%	97.1%	97.5%	97.9%
Crossings per sentence	2.7	2.4	2.1	2.0	1.7
Sent. with 0 crossings	24.3%	29.4%	33.1%	36.0%	43.4%
Sent. with up to 1 crossing	43.4%	48.9%	52.6%	52.9%	59.6%
Sent. with up to 2 crossings	59.9%	61.8%	65.4%	70.6%	73.9%
Sent. with up to 3 crossings	72.4%	73.9%	77.2%	80.9%	84.9%
Sent. with up to 4 crossings	82.4%	83.5%	88.2%	88.2%	89.7%
Correct operations	74.8%	76.8%	79.2%	81.0%	81.9%
Sent. with correct OpSequence	0.0%	1.1%	1.1%	1.1%	1.5%
Sent. with correct Struct&Label	3.7%	5.5%	6.3%	7.0%	9.2%
Sentences with endless loop	5	12	8	3	2

Table 6.6: Evaluation results using only 25 features, all of them syntactic

Number of training sentences	16	32	64	128	256
Precision	84.0%	82.5%	85.4%	85.2%	86.4%
Recall	81.3%	83.6%	85.5%	86.3%	87.4%
Labeled precision	66.6%	17.9%	36.2%	26.0%	27.6%
Labeled recall	72.7%	75.2%	78.3%	79.9%	82.0%
Tagging accuracy	96.4%	96.5%	97.0%	97.5%	97.9%
Crossings per sentence	2.7	2.5	2.1	2.0	1.7
Sent. with 0 crossings	25.7%	26.8%	32.7%	33.8%	43.0%
Sent. with up to 1 crossing	43.0%	44.1%	52.9%	52.9%	58.8%
Sent. with up to 2 crossings	59.9%	58.8%	64.3%	68.4%	74.6%
Sent. with up to 3 crossings	69.5%	71.0%	77.2%	80.5%	84.6%
Sent. with up to 4 crossings	79.4%	82.7%	86.8%	87.5%	89.3%
Correct operations	74.2%	76.3%	78.5%	80.1%	80.9%
Sent. with correct OpSequence	1.1%	1.1%	1.1%	1.5%	1.5%
Sent. with correct Struct&Label	3.7%	4.0%	5.9%	7.0%	8.5%
Sentences with endless loop	16	17	20	17	11

Table 6.7: Evaluation results using only 12 features, all of them syntactic

Number of training sentences	16	32	64	128	256
Precision	83.8%	83.6%	85.1%	86.1%	88.0%
Recall	81.5%	83.5%	85.1%	85.9%	87.3%
Labeled precision	59.8%	24.6%	46.9%	69.9%	79.8%
Labeled recall	73.3%	75.5%	77.9%	79.5%	81.5%
Tagging accuracy	96.6%	96.7%	96.8%	97.2%	97.6%
Crossings per sentence	2.7	2.5	2.2	2.1	1.8
Sent. with 0 crossings	24.3%	26.1%	30.2%	32.4%	39.0%
Sent. with up to 1 crossing	43.0%	44.9%	48.5%	51.8%	57.4%
Sent. with up to 2 crossings	57.7%	60.3%	63.2%	65.8%	72.1%
Sent. with up to 3 crossings	72.4%	70.2%	77.9%	79.4%	82.7%
Sent. with up to 4 crossings	80.9%	83.1%	86.8%	86.0%	89.0%
Correct operations	73.7%	75.6%	78.0%	79.3%	80.6%
Sent. with correct OpSequence	1.1%	1.1%	1.1%	1.1%	2.6%
Sent. with correct Struct&Label	4.0%	4.0%	4.8%	5.9%	8.8%
Sentences with endless loop	19	14	15	14	8

Table 6.8: Evaluation results using only 6 features, all of them syntactic

2.7) and even for compound test characteristics a relatively moderate drop (e.g. from 5.5% to 4.0% for the percentage of sentences with perfect structure and labeling after parsing). For the full complement of 256 training sentences on the other hand, we observe a roughly 5% drop in labeled and unlabeled precision and recall, almost twice as many crossings, and a sharp decrease in compound test characteristics (from 16.7% to 2.6% totally correct parse sequences and from 26.8% to 8.8% for parse sequences leading to the proper structure and labeling).

So, while the loss of a few specialized features will not cause a major degradation, the relatively high number of features used in our system finds a clear justification when evaluating compound test characteristics. When comparing the decrease of parsing accuracy for the various training sizes, we observe that the accuracy loss is much more pronounced for higher number of training sentences. As one might have expected, this means that larger training corpora can exploit our rich feature set better. When the training size is increased beyond 256, we can therefore rightfully expect the advantage of a rich context to increase even further.

6.4.4 Contribution of Decision Structure Type

Type of decision structure	simple decision tree	simple decision list	hier. decision list	hybrid decision structure
Precision	87.6%	87.8%	91.0%	92.7%
Recall	89.7%	89.9%	88.2%	92.8%
Labeled precision	38.5%	28.6%	87.4%	89.8%
Labeled recall	85.6%	86.1%	84.7%	89.6%
Tagging accuracy	97.9%	97.9%	96.0%	98.4%
Crossings per sentence	1.3	1.2	1.3	1.0
Sent. with 0 crossings	51.5%	55.2%	52.9%	56.3%
Sent. with up to 1 crossing	65.8%	72.8%	71.0%	73.5%
Sent. with up to 2 crossings	81.6%	82.7%	82.7%	84.9%
Sent. with up to 3 crossings	90.1%	89.0%	89.0%	93.0%
Sent. with up to 4 crossings	93.4%	93.4%	93.4%	94.9%
Correct operations	90.2%	86.5%	90.3%	91.7%
Sent. with correct OpSequence	13.6%	12.9%	11.8%	16.5%
Sent. with correct Struct&Label	21.7%	22.4%	22.8%	26.8%
Sentences with endless loop	32	26	23	1

Table 6.9: Evaluation results comparing different types of decision structures

Table 6.9 compares four different machine learning variants: a plain decision tree, a plain decision list, a hierarchical decision list, and finally a hybrid decision structure, namely a decision list of hierarchical decision trees, as sketched in figure 5.9. The results show that extensions to the basic decision tree model can significantly improve learning results. The hybrid decision structure is superior to the other three models with respect to each and every criterion. While the three simpler decision structure have parsing loop problems for around 10% of the 272 sentences, the hybrid decision structure has only one such problem sentence. For compound test characteristics, sentences with the perfect operation sequence or at least the correct final structure and labeling, the decision structure list scores 3% and 4% respectively higher than any other decision structure.

Chapter 7

Transfer

The transfer module maps the final source language parse tree to a corresponding target language tree. Before the actual transfer, the parse tree is normalized and some language specific morphological and syntactic information is discarded. The normalization includes the addition of back-pointers to parent nodes and the backpropagation of forms. To understand what this backpropagation does, consider for example the sentence *"The deer are hungry."*, where the number of the noun phrase and the person and number of the verb are locally ambiguous. After constraint unification during parsing, form information such as number and person, unambiguous at the sentence level, is now propagated back to the individual components of the sentence, so that *the deer* is marked as plural and *are* as third person plural.

The other part of pre-transfer processing is the deletion of the inner structure of a compound verb, punctuation and other syntactic dummy components like for example the empty or 'prop' *it*[1] in English. The inner structure of verbs, e.g. [[AUX: [AUX: had] [PRED: been]] [PRED: dissatisfied]], is quite language specific and is therefore eliminated, leaving just the verb concept (I-EV-DISSATISFY) along with form information (past perfect, passive, etc.). Punctuation is also discarded. The main purpose of punctuation is to help structure sentences, something that can and should be exploited during parsing, but once the parse tree has been formed and the sentence has been assigned a structure, punctuation no longer carries any additional information, and since its usage is also language specific, it is not transferred to the target parse tree.

So the main information transferred to a target language tree is the structure of the sentence, its concepts and forms (such as tense), and roles of components in their respective phrases. The task of determining the correct word order, finding the specific surface words for the various concepts and punctuation is left to the generation module.

Bilingual dictionaries are fully symmetric and can be used for translations in either direction.

7.1 Simple Transfer

The structure of the parse tree and the roles are basically preserved, leaving the transfer of concepts as the core task. This concept transfer can happen for both individual concepts as well as concept clusters, representing phrases such as *"to reach [an] agreement"* or *"to comment on SOME-*

[1]as e.g. in *"It rained.", "What time is it?"*

THING_1". This is done using a *bilingual dictionary*, which was already described in section 3.4 in the chapter on background knowledge. Recall that the *surface* dictionary, which has a very user friendly and intuitive format similar to traditional (paper) dictionaries, is compiled into an *internal* dictionary, which links concepts (for individual words) or parse trees (for phrases) between languages. In the simplest case, an individual source language concept will correspond to exactly one target language concept, in which case that concept is selected.

One could ask, why in such a case the words from the respective languages aren't just linked to the same KB concept in the first place. The answer is that we don't want to certify that a specific word pair has a perfect conceptual match. While two words might appear to have a one-to-one correspondence, there might be an obscure case or context where this does not hold after all. We want to keep the information contained in the lexical entries micromodular and be able to deal with any unforeseen mismatches by just adding another entry to the bilingual dictionary.

7.2 Ambiguous Transfer

The much more interesting and fairly frequently occurring case is when words and phrases do not correspond one-to-one between languages. Consider for example the two dictionary entries for "know":

"know"	S-VERB
"kennen"	
"know"	S-VERB
"wissen"	

Depending on whether the object of "know" is a person, a place, etc., or a fact, "know" maps to "kennen" or "wissen" in German (and "conocer"/"saber" in Spanish; "connaître"/"savoir" in French). Lexical ambiguity can be divided into three levels: part of speech ambiguity, and heterogeneous and homogeneous ambiguity within the same part of speech. The transfer module assumes that the word has already been disambiguated with respect to the part of speech. A disambiguation of heterogeneous (homonymous) meanings (e.g. pen (for writing) vs. pen (to keep animals)) is currently not done during parsing, because it hasn't found to be necessary for our WSJ sentences. When such disambiguation should turn out to become necessary, e.g. to resolve structural ambiguity or case role assignment, it could be added, but so far in our research it has been sufficient to delay decisions on heterogeneous meanings until transfer. The partitioning of fairly homogeneous (polysemous) senses, like "know", is very language dependent though. It is hard, if not impossible, to do a proper partitioning without having a particular target language in mind. This type of disambiguation is clearly more appropriate to be resolved in the transfer module. So, ambiguous transfer currently has to resolve both heterogeneous and homogeneous ambiguity within the same part of speech. This however does not present any additional difficulty, because from the perspective of transfer, there is no substantive difference between heterogeneous and homogeneous ambiguity.

For choosing the appropriate target concept, we reuse the methods developed for parsing with only minor adaptations. Again, we provide examples and features, from which we compute complete examples that include a full feature vector plus classification, and feed these into our machine learning module. While there is one large example collection and one big feature set for parsing, there are separate sets of examples and features for each concept to be disambiguated. Consider the following three *transfer entries*:

```
(I-EV-KNOW
   ((SYNTP OF S-SUB-CLAUSE OF THEME OF PARENT AT M-BOOLEAN)
    (CLASSP OF I-EN-TANGIBLE-OBJECT OF THEME OF PARENT AT M-BOOLEAN))
   ((I-GV-KENNEN "I know the old man." (SHORT 5 1) 10)
    (I-GV-WISSEN "I know that the old man bought a car."
           (SHORT 7 1) 20)))

(I-EP-BY
   ((SYNT OF PARENT OF PARENT AT S-SYNT-ELEM) ;;; S-CLAUSE, S-NP
    (CLASSP OF I-EN-AGENT OF PRED-COMPL OF PARENT AT M-BOOLEAN)
    (CLASSP OF C-AT-TIME OF PARENT AT M-BOOLEAN))
   ((I-GP-VON "It will be sold by a shareholder." nil 10)
    (I-GP-BIS "The transaction will be finished by May 10." nil 20)
    (I-GP-DURCH "Sales will be hurt by the losses." nil 30)
    (I-GP-GEN-CASE-MARKER
           "The estimates by the bureau were published in May."
           nil 40)))

(I-EN-FUTURE
   ((NUMBER AT F-NUMBER))
   ((I-GN-ZUKUNFT "future" nil 10)
    (I-GN-TERMINGESCHAEFT "He bought futures." nil 20)))
```

Each transfer entry consists of the source concept, a list of features, and a set of examples. The transfer examples consist of a target concept, an example text, an optional example reference and an example identifier. The example text is often a sentence, but can basically be anything that provides enough context for the features. The optional reference, e.g. (SHORT 5 1), contains both a corpus reference, here sentence *5* of corpus *SHORT*, and an occurrence identifier, here *1*.

The transfer examples are processed by first parsing the example text. This is done using the same parser as for ordinary text. If a corpus reference is included and there is a log entry, i.e. a supervisor-provided parse action sequence, that parse action sequence is used, because it has already been certified as correct. Otherwise the parse is 'free', but since the examples are typically concise, the error potential is very small in the first place and only errors that would change any of the transfer entry features would be problematic. If example texts must approach or surpass the limits of the parser coverage, it is advised to store the text in some corpus and provide a log entry for it. The occurrence identifier, which has a default of 1, selects which of the potentially multiple occurrences of the source concept in the example text should serve as a reference point.

After the parse tree has been computed for the example text and the reference point is identified, the system computes values for all features listed in the transfer entry. The feature language is the same as for parsing. Note again though that the reference point here is not the current parse state with parse stack and input list, but a specific parse entry. Therefore the access paths of the transfer entry features (e.g. *THEME OF PARENT*) typically do not contain an integer, as the parsing features (e.g. *THEME OF -1*) typically do.

Once all examples in a transfer entry have been expanded to include their full feature-value vector, standard decision tree learning is applied. The number of examples and features tends to be fairly small per source concept, so compared to parsing, which uses a single huge decision structure, we use many (one for each ambiguous concept) small decision trees for ambiguous transfer.

Since, for each source concept, there are relatively few target concepts (classifications), examples, and features compared to parsing, a hand coded approach would be more feasible than in parsing. For several reasons we nevertheless believe that machine learning is better suited. Features would have to be provided for both learning and hand-coding, for learning though only once, whereas in hand-coding a feature might have to appear several times. While examples are explicitly required for learning they are practically also required for hand-coding, namely for testing and documentation purposes; the examples needed, typically short, unannotated sentences, can normally be provided fairly easily. With learning, the decision structures are built automatically; coverage of the examples is guaranteed unless there is an unresolvable example conflict, which the system signals automatically. "Testing the test sentences" is in a sense done implicitly. Hand-coding on the other side would require explicit rule construction and, in practise, repeated testing.

7.3 Complex Transfer

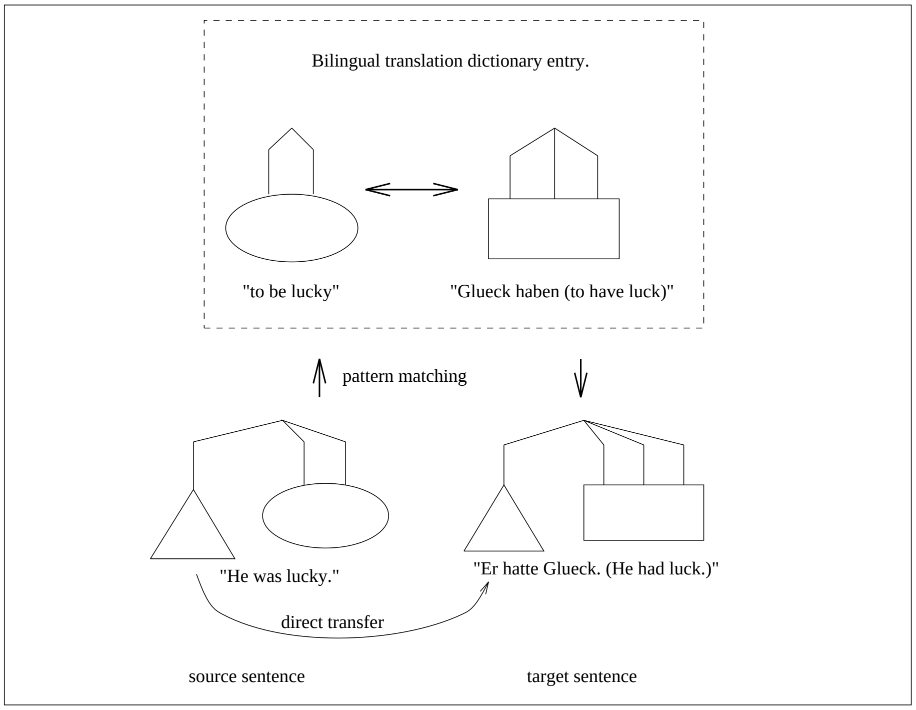

Bilingual translation dictionary entry.
"to be lucky"
"Glueck haben (to have luck)"
pattern matching
"He was lucky."
"Er hatte Glueck. (He had luck.)"
direct transfer
source sentence
target sentence

Figure 7.1: Complex transfer with pattern matching

"Structural mismatches" present another major issue. Often, the structures of the source and target language parse trees don't match. To handle such cases, the bilingual dictionary contains potentially complex phrases, represented as parse trees in the *internal* dictionary. During transfer, a pattern matching module tries to match source sentences subtrees to dictionary source trees and, if successful, maps the source sentence tree to a target sentence tree using the dictionary target tree linked to the successfully matched dictionary source tree as illustrated in figure 7.1.

The parse tree pair in the dashed box is an entry in the *internal* dictionary. To translate the English sentence *"He was lucky."*, we first parse it to get a parse tree as depicted in the lower left area of figure 7.1. The source sentence tree is transferred recursively. At each level we check whether the current subtree matches a pattern in the dictionary. For a valid match, all components of the dictionary source tree have to be covered, but the inverse does not need to hold. In our example for example the source sentence parse tree contains an additional subject component. Dictionary entries have been stripped of many of their forms, e.g. their infinitive tense, since such forms were meant to represent the word generically. As a consequence, the past tense form of *"He was lucky."* can be considered an additional component as well.

In case of a match, the target sentence tree is built based on the dictionary target tree. Additional components from the source sentence tree (here *He* and *past-tense*) are then transferred separately and the resulting target components are finally properly attached to the target sentence tree.

"to be lucky" "Glück haben"	S-VP
"downtown PLACENAME_1" "im Stadtzentrum von ORTSNAME_1"[2]	S-ADV S-PP
"it takes SOMEBODY_3 SOMETHING_1 TO_DO_SOMETHING_2" "JEMAND_3 braucht ETWAS_ACC_1, um ETWAS_ZU_MACHEN_2"	S-CLAUSE
"to reach SOME_1 agreement" "zu EINER_1 Übereinkunft kommen"	S-VP
"interest rate" "Zinssatz"	S-NOUN

The patterns can contain partially restricted variables, e.g. "PLACENAME_1" in the selection above.[3] Subtrees from the source sentence tree can match such variables if they fulfill the restrictions that are the lexicon associates with these variables, typically reflected in the name of the variable. In case of a match such subtrees are then transferred separately and the resulting target trees are finally properly attached to the target sentence tree at the place marked by the target variable with the same index.

The dummies of the form *SOME_i* are special 'collectors' which can be matched to a whole set of noun modifiers. This allows the tree for *"to reach SOME_1 agreement"* to match all of the following:

[3]This phrase literally means *in_the towncenter of PLACENAME_1*

- We reached an agreement.
- We didn't reach any agreement.
- No agreement was reached.
- We reached many new traffic agreements of great importance.
- Candidates Reached Agreement

A dictionary entry with *"to reach an agreement"* would only match in the first example.

The search for possibly matching patterns is sped up through the use of the head concept as an index and a quick pre-check, in which the cached set of all other obligatory leaf concepts in the dictionary source tree has to be covered by the source sentence tree.

If no complex match is found for a source sentence subtree, the various components of that subtree are transferred separately in further recursion and put together using the same structure as the source sentence tree.

Even if there is no structural mismatch, as for example in the last entry of the last table, where there is a complex noun consisting of two nouns in both English and German[4], complex transfer can be used as a simple tool for disambiguation, reducing the need for examples and features. In English, both *interest* and *rate* are semantically ambiguous, as also reflected by different German translations, e.g. *Interesse, Zins(en), Anteil* for *interest* and *Rate, Satz, Kurs* for *rate.* As a compound however, *interest rate* unambiguously maps to *Zinssatz.* Since complex transfer precludes any individual and possibly ambiguous transfer at a lower level, no ambiguous transfer examples and features are necessary for *interest* or *rate* when they occur in a joint compound.

7.4 Added Concepts

It can sometimes become necessary to add a concept to the target parse tree that is not based on any specific concept or group of concepts in the source parse tree. Examples for such added concepts are articles, pronouns or adverbs. Japanese, which does not have articles, for example does not mark its noun phrases as definite or indefinite. It also often omits pronouns that would be obligatory in English. When performing parse tree transfer to a language that requires overt components that don't have some overt corresponding concept in the source parse tree, we have to break through the basic transfer paradigm and basically have to create "something out of nothing". Certain source parse tree patterns have to trigger a context based decision making process that will add any appropriate concept to the target parse tree. Even though this process depends strongly on the target language and therefore might conceivably be included as an early step in generation, we deal with this issue at the end of transfer, because the work that is actually necessary also strongly depends on the specific source language.

English and German both require overt pronouns and generally share the same notion of definiteness, including the usage of definite and indefinite articles. The differences in article usage are however strong enough that they manifested themselves several time in the WSJ translation

[4]*Zinssatz* is a compound of *Zins* and *Satz*

development corpus. The noun phrase in "*Life can be difficult.*" for example must be definite in German and French.

In our system, at the end of the core transfer, we traverse the parse tree and determine for each noun phrase whether an additional definite article is necessary. While this is currently hard-coded, we could possibly see machine learning as a useful tool when further refining this process.

The reverse process, deleting concepts, can be handled more easily by using the marker "nil" in the bilingual dictionary entry. According to the following example, the English word "some" can be translated by "einige" or by nothing[5]. Unlike all other dictionary entries, which can be used in both directions, those dictionary entries that contain "nil" can only be used with "nil" as a target.

"some"	S-ADJ
"einige"	
"some"	S-ADJ
nil	

In English to German transfer, only few minor concepts have to be added to cover the 'nil to something' direction, but for other language pairs, in particular for less related languages, concepts might have to be added in more cases and also require more context.

[5]As in "Did you bring some records?" $\rightarrow$ "Hast Du Schallplatten mitgebracht?"

Chapter 8

Generation

Following parsing and transfer, generation completes the process of machine translation by ordering the components of phrases, adding appropriate punctuation, propagating morphologically relevant information, and finally generating the proper surface words and phrases in the target language.

In many natural language applications, generation can be quite formidable, because it involves tasks such as selecting the contents of a text and planning the discourse structure of the selected material. Such *strategic* generation, as it is often referred to, is however not necessary in machine translation, because the contents and overall discourse structure can be copied from the source text. This leaves the so called *tactical* generation, consisting of the above mentioned subtasks.[1]

Further reasons for the relative simplicity of tactical generation as we need it for our system are that

- ordering basically depends on only fairly local syntactic properties of the phrase components,
- the actual surface words basically depend on only fairly local morphological properties,
- a single good output is sufficient while the parser has to be able to cope with all good and sometimes even not so good formulation alternatives of a sentence,
- difficult decision issues, like for example idiomatic expressions, have already been handled in transfer.

Since the steps necessary for generation are already linguistically well researched and described in detail in grammar books, e.g. in (Lederer, 1969; Engel, 1988) for German, we don't use any learning for generation, but rather follow standard phrase constructions from literature.

The input to generation is a transfer tree, an integrated phrase-structure and case-frame tree, very similar to a parse tree, but typically not ordered, without punctuation, with less morphological information and no surface forms. The generation module then manipulates the transfer tree by reordering subtrees and propagating morphological information. It gradually adds surface forms, the target character string associated with each subtree. The tree that is finally formed after these

[1]When translating between very different languages, it might be necessary to precede the generation steps described in this chapter by an additional *operational* generation step to bridge general structural differences that can not be linked to specific lexical items in the mismatching phrases and therefore can not be covered by the transfer module.

manipulations then contains the translation result at the surface form slot of the top node. The following sections now describe the ordering and the morphological propagation and generation in more detail.

8.1 Ordering

For each type of phrase of a given language to be ordered, the system uses a specific and deterministic scheme. For German sentences or verbal clauses, the most complex case in that language, the system for example first determines whether the clause is finite, passive, a relative phrase, a main clause, and whether it needs a "prop" *es*[2] as a subject, needs to be enclosed by commas or needs to have the conjugated part of the verb split off.

Based on this information, it arranges the sentence in a specific order:

1. coordinate conjunction
2. subordinate conjunctions
3. topic
4. the conjugated part of the verb (if the verb has to be split)
5. interrogative pronoun
6. pronominal or definite subject
7. pronominal direct object
8. pronominal indirect object
9. (indefinite) subject
10. definite indirect object
11. definite direct object
12. pronominal genitive complement
13. assessorial adjuncts
14. situational adjuncts
15. negational adjunct
16. other quantifiers
17. (indefinite) indirect object
18. (indefinite) direct object
19. (definite or indefinite) genitive complement

[2]as in "*Es* regnete.", the English equivalent of "*It* rained."

20. misc. other modifiers

21. other complements

22. adverbial complement

23. adjectival complement

24. nominal complement*

25. predicate

26. severed ("heavy") relative clause

27. infinitival complement

28. particle phrase clauses*

29. other subclauses*

30. plus possibly enclosing commas[3]

Every specific clause contains of course only some of these components; so the preceding list only prescribes in which order the existing components are to be arranged with respect to each other.

The structure of the generated sentence thus follows a syntactical pattern that can be characterized as 'canonical', 'safe' and 'mainstream', but does not necessarily reflect all the variance that is acceptable. Similar schemes have be written for noun phrases, prepositional phrases, particle phrases, and complex nouns.

Figure 8.1 illustrates this ordering process. The target tree output from the transfer module still reflects the word order of the source sentence *"Yesterday, Thomas read a good book."*. As usual in German main clauses, the verb needs to split into the conjugated part and the remainder. This is done by creating a special verb-head unit that points to the full verb and symbolizes the conjugated part of the verb it points to; the full verb entry is marked as *head-referenced* to indicate that it is being pointed to by such a special verb-head unit.

(Severed ("heavy") relative clauses that have to separated from the noun phrases they belong to are treated the same way.)

The components in the bottom tree of figure 8.1 follow the order given in the above scheme: topic (time can qualify as such), the conjugated part of the verb, definite subject, (indefinite) direct object, and predicate. After the tree is ordered at the sentence level, the ordering recursively proceeds to the individual sentence components.

8.2 Morphological Propagation and Generation

During morphological propagation, the carriers of gender, number, case, tense etc. (nouns, prepositions, verbs etc.) propagate their forms to the constituents that they morphologically control.

[3]If a sentence element fits more than one of these categories, it is placed at the first applicable position, unless an applicable category is marked by star (*), in which case it is placed at the last such position.

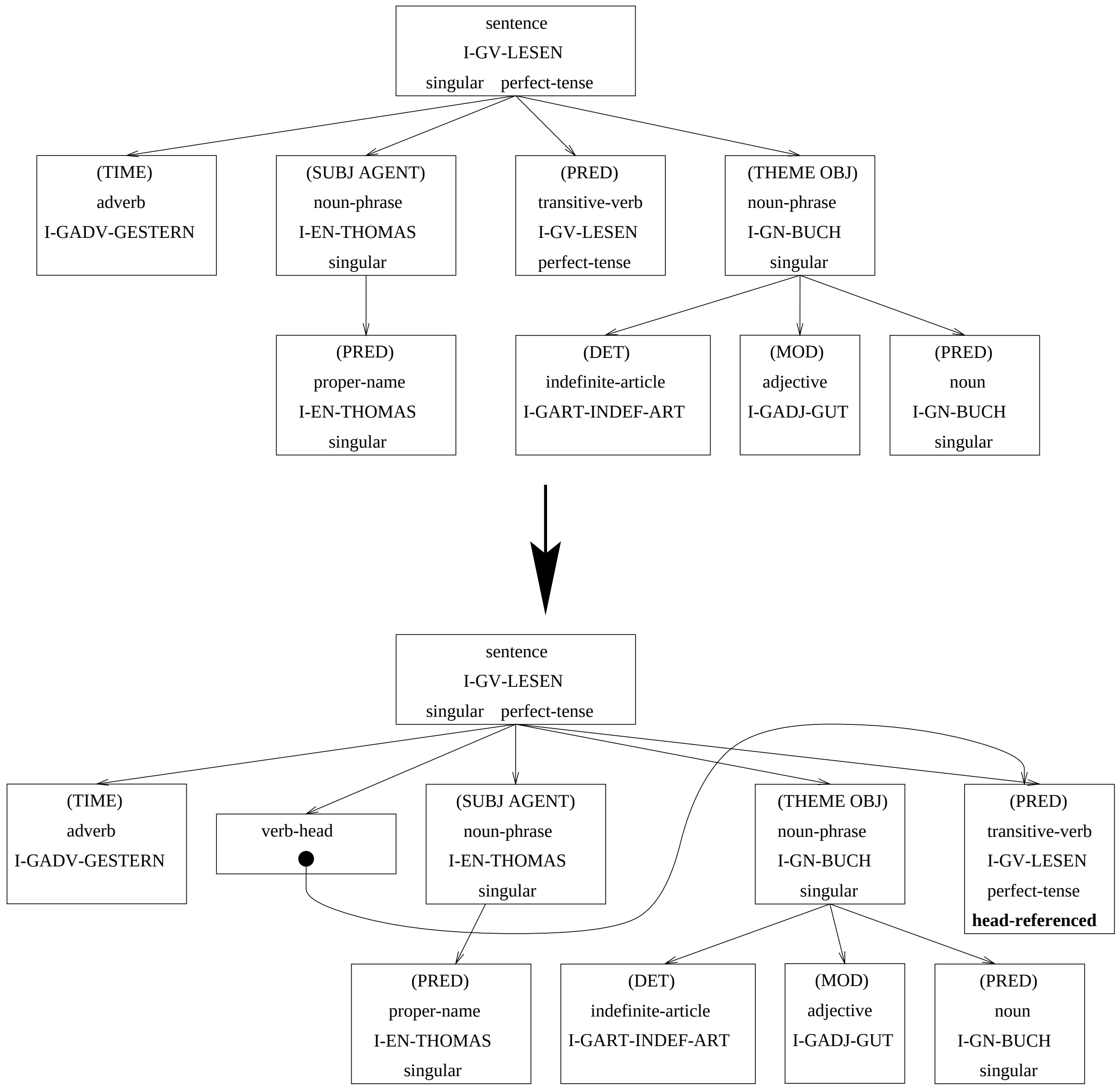


Figure 8.1: Ordering the target tree

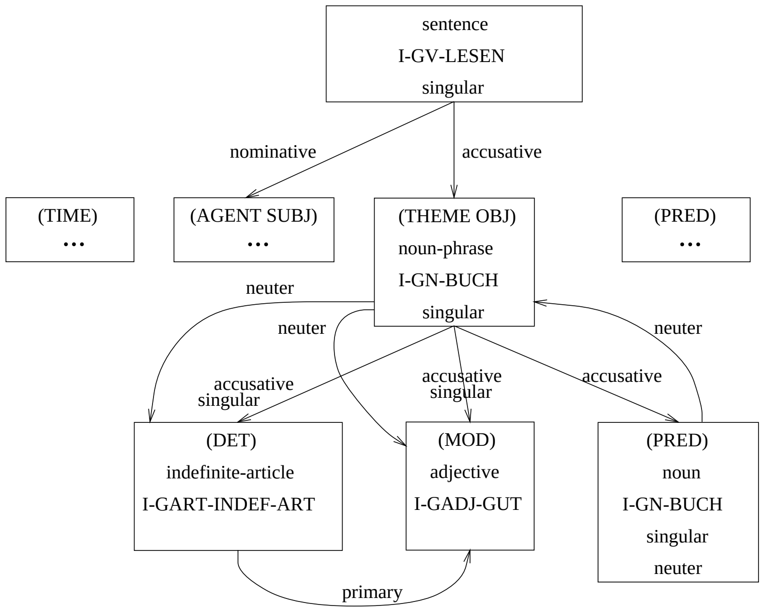
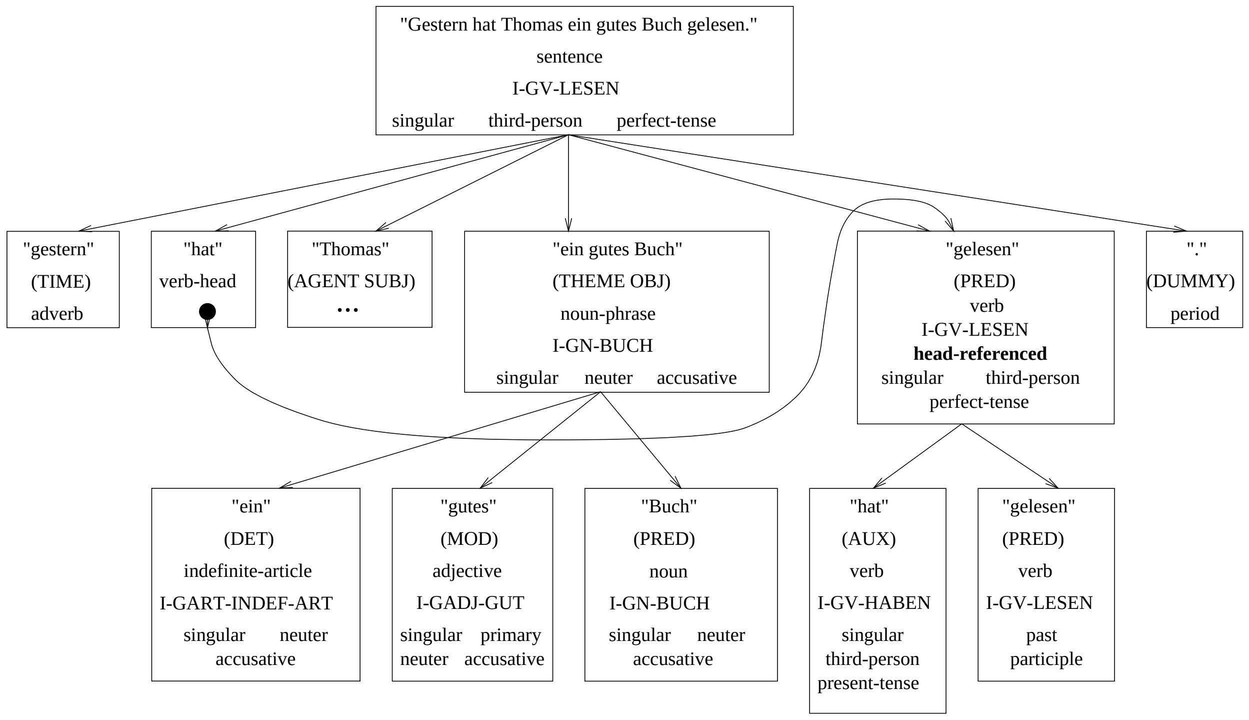


Figure 8.2: Morphological propagation

In figure 8.2, we show how this is done for a portion of the graph from figure 8.1. In order to form surface words, the tree nodes need to contain complete form information. In our example from German, the adjective I-GADJ-GUT for example needs case, number, gender and a primary/secondary indicator. Case for noun phrases is assigned by the verbs or prepositions that govern them, i.e. through the roles in the verbal clause they are a direct component of, or by the preposition with which they share a prepositional phrase. In German, the subject is assigned the nominative case and the (direct) object is assigned the accusative case. Many prepositions always assign the same specific case, e.g. "*wegen*" ("because of") should always assign genitive case, but some assign different cases, depending on the semantics of the prepositional phrase (e.g. *in* assigns accusative for C-TO-LOCATION and dative for C-AT-LOCATION). The case of a noun phrase is passed on to its components. Number is carried over from the source tree in the transfer process. It is propagated to all parts of the noun phrase, including adjectives, because in German, the number information is needed everywhere. All German nouns have a specific grammatical gender, which is listed in the (monolingual) lexicon. The German word *Buch* is neuter, and so this information gets passed up to the dominating noun phrase and then back down to the other components of the noun phrase. Finally, based on the article I-GART-INDEF-ART, the system signals the adjective to carry the primary ending, because this specific article does not have a primary ending itself[4].

Once all forms are propagated, the morphological module determines the surface forms of all words. Starting with the leaves of the tree, surface forms are computed using the morphology module. In a bottom up fashion, strings are then basically concatenated until the full target sentence is stored at the surface form slot of the top node. The string in that very slot is the final translation of the original source sentence.

[4]While definite articles always carry a primary ending, therefore causing the form *secondary* to be sent to any following adjectives, indefinite singular articles of certain cases and genders, incl. the one in our example, do not have such primary ending and therefore have the form *primary* sent to any following adjectives.



"Gestern hat Thomas ein gutes Buch gelesen."
sentence
I-GV-LESEN
singular third-person perfect-tense
"gestern" (TIME) adverb
"hat" verb-head
"Thomas" (AGENT SUBJ) ...
"ein gutes Buch" (THEME OBJ) noun-phrase I-GN-BUCH singular neuter accusative
"gelesen" (PRED) verb I-GV-LESEN head-referenced singular third-person perfect-tense
"." (DUMMY) period
"ein" (DET) indefinite-article I-GART-INDEF-ART singular neuter accusative
"gutes" (MOD) adjective I-GADJ-GUT singular primary neuter accusative
"Buch" (PRED) noun I-GN-BUCH singular neuter accusative
"hat" (AUX) verb I-GV-HABEN singular third-person present-tense
"gelesen" (PRED) verb I-GV-LESEN past participle

Figure 8.3: Generating the surface strings

As one can see in figure 8.3, disjoint components like split verbs need a little bit of a special treatment. The verb-head unit, pointing to a full verb complex, and representing the conjugated part of the verb, takes the surface string of the conjugated part of the verb, here "*hat*" of the German auxiliary verb "*haben*". In the main verb entry, the flag *head-referenced* indicates that the conjugated part of the verb is already been used by some verb-head unit, so that the surface string of that part does not get included in the surface string of the main verb component.

So, in contrast to parsing and transfer, the generation module in our system does not require any learning. Standard recursive techniques turned out to be sufficient, at least so far. During further refinement, it might turn out that for relatively small subtasks in generation, such as in the decision what case a specific preposition should assign to the noun phrase it governs, learning might be useful after all, but we expect that in the overall control of the generation module, learning will remain inexpedient.

Chapter 9

Translation Experiments

9.1 Test Methodology and Evaluation Criteria

For training and testing the transfer and generation modules for translations from English to German, we use sentences from the same Wall Street Journal corpus as for parsing.

The development of the transfer and generation modules including the selection of example sentences and features for ambiguous translation was largely corpus driven and based on the first 48 sentences (WSJ 0-47). The goal was to refine the transfer and generation modules to the point that the system could produce both syntactically and lexically good translations for all 48 "training" sentences. It is important to note that the expression "training sentence" here is not used in the strict sense of machine learning, but rather as a sub-corpus that provides guidance as to what linguistic phenomena need to be covered. Since the sub-corpus also already contains a number of words with ambiguous translation targets, it also drove the selection of succinct example sentences and corresponding features for the *transfer entries* that were used to build decision trees for ambiguous concept transfer.

Section 7.2 in the chapter on transfer already included examples of transfer entries. A list of the various features used in English to German translation disambiguation decision structures can be found in section B.2.

When the system, based on correct parse trees for the 48 "training sentences", eventually produced good translations, we then tested the entire system on the following 32 sentences (WSJ 48-79). The system input was an English sentence in the form of a simple character string, without any annotation. The output was a German sentence, again as a simple character string.

In order to better be able to evaluate the contribution of the parser on one side and the transfer and generation modules on the other side, we tested our system in two versions. In one version, the parser was trained on 256 sentences excluding the test sentences[1], thereby performing "full" test translations on sentences that had been used for neither training the parser nor any other module. In the other version, the parser was trained on the test sentences, meaning that the system by construction basically started with a correct parse, subjecting only the transfer and generation modules to sentences that those modules had not been exposed to before. Comparing the results of both versions helped us to better understand the contributions of the different phases

[1](training sentences WSJ 0-47 & 64-271 for test sentences WSJ 48-63 and training sentences WSJ 0-63 & 80-271 for test sentences WSJ 64-79)

of translation.

9.1.1 Comparative Translations

To obtain some interesting comparisons, we used the same 32 test sentences on three commercially available systems, Logos, SYSTRAN, and Globalink. The translation with these systems were made over the Internet, all in October 1996.

Globalink's Web site at http://www.globalink.com/ offers free translations for text of limited length from English to French, Spanish, German and Italian and vice versa. The source text is typed into a multiline text field form of the Web page and the translation results are sent back by email. We had to split up the 32 sentences into three subsets because our text was longer than the Web interface accepted at any one time.

SYSTRAN's Web site at http://systranmt.com/ offers free translation of Web pages. Given the fully qualified URL of a source page, the system allows translations from English to French, Spanish, German, Italian, Portuguese and vice versa as well as Russian to English. The translated Web page is displayed in a new browser window. We created a Web page with the 32 test sentences and had them translated to German.

Logos' Web site at http://www.logos-ca.com/ did not provide free translation to the general public, but the company was kind enough to provide a confidential Web site and let us use their system free of charge for our 32 test sentences. Logos offers to translate from English or German to German, French, Spanish, Italian or English. As an additional input parameter, the system asked for a domain choice, offering *General, Computers, Telecommunication* and *Business* as alternatives. We chose the last option as the clearly most appropriate for our Wall Street Journal sentences.

Two details about these three comparative translations:

- Since in testing our system, we translated one sentence at a time, we ensured that the commercial systems were at no disadvantage in this respect by separating input sentences by blank lines, if needed.
- The results from SYSTRAN and Logos were returned so promptly that any human post editing can be excluded just based on time. Based on the quality of the results from Globalink, plus given that the translations were free of charge, human post editing is also most unlikely for that system.

Recall that the subcorpus we use is lexically limited to the 3000 words most frequently occurring in the Wall Street Journal. This allowed us to have a complete monolingual lexicon for parsing and bilingual dictionary for transfer, so that in the tests, our system was not confronted with unknown words. The commercial translation systems were designed for a wider range of domains and they also have significantly larger lexicons and dictionaries. The only English words in the 32 test sentences that apparently were unknown were *seasonally* (Logos) and *nationwide* (Logos and Globalink). Globalink also 'translated' "holiday schedule" as "holidayschedule", "currency exchange" as "currencyexchange", "economic policy" as "economicpolicy", and "[over] the counter" as "[über] thecounter", but since for example "exchange" was actually translated in another sentence, we conclude that there are entries for presumably at least most of the individual words in the lexicon and that these translation failures are primarily due to some shortcoming in the Globalink translation algorithm. SYSTRAN, a veteran of machine translation systems, did not exhibit any lexical gaps. So, with an average of one unknown word for all 32 test sentences, the commercial systems were at no significant disadvantage with respect to unknown words.

As an additional comparison, the author of this dissertation, with a minor degree in business and previous experience as a professional translator from English to German, translated the 32 test sentences "by hand".

9.1.2 Evaluators and Evaluation Criteria

The translations were evaluated by ten bilingual graduate students at the University of Texas at Austin. Seven of the volunteers were native speakers of German, two were native speakers of English teaching German at the university and one was a native speakers of English who has lived in Germany for many years, speaks German without an accent and has at least the German proficiency of the two teaching German. Professional translations and proofreading of translations is normally performed by native speakers of the target language, in our case German. This certainly justifies the dominance of native German speakers in our group of evaluators.

Each of the 10 evaluators was given a list of the 32 original sentences (in English) along with up to six German translations for each original sentence. The translations always included those from the two versions of our system and the three commercial systems. The human translations were given only to half of the evaluators. This was done to control any possible influence of that translation on the evaluation of the machine translations. When two or more translations were identical, the translation was presented only once. The translations were listed in randomized order, separately randomized for each WSJ sentence, and without identification.

For each of the translations, the evaluators were asked to assign two grades, one for the grammatical correctness and the other for meaning preservation. Table 9.1 and the following example were given as a guideline to ensure a relative uniform standard.

(EXAMPLE)
Yesterday, I ate a red apple.

(a)	Gästern, ich haben essen Apfl-rot.	*Grammar:_5_ Meaning:_2_*
(b)	Meine roten Äpfel haben viel gegessen.	*Grammar:_1_ Meaning:_6_*

The scale is like the one used in the German education system: 1 = sehr gut *(excellent)*; 2 = gut *(good)*; 3 = befriedigend *(satisfactory)*; 4 = ausreichend *(passing)*; 5 = mangelhaft *(poor)*; 6 = ungenügend *(unsatisfactory)*.

The following is one of the 32 translation evaluation blocks of the questionnaire, incidently the one with the funniest[2] machine translations:

[2]Back translations: (a) He said that Pan Am currently has " over $150 million " in the cash. *Note: 'Million' should have been 'Millionen' (plural).* (b) He said Pan Am currently has over 150 million dollars in cash. (c) He said that frying-pan-in-the-morning currently has "more than 150 million dollars" in cash. (d) He said that frying-pan is, has currently-" in surplus of $150 million" in in cash. *Note: 'million' should have been 'Millionen' (capitalized and plural).*

Grammar (syntax and morphology)

Grade	*Usage*
1	Correct grammar, including word order, word endings; the sentence reads fluently.
2	Basically correct grammar, but not very fluent.
3	Mostly correct grammar, but with significant shortcomings.
4	The grammar is acceptable only in parts of the sentence.
5	The grammar is generally so bad that the entire sentence becomes very hard to read.
6	The grammar is so bad that the sentence becomes totally incomprehensible.

Meaning (semantics)

Grade	*Usage*
1	The meaning is fully preserved and can easily be understood.
2	The meaning is mostly preserved and can be understood fairly well.
3	The general idea of the sentence is preserved.
4	Contains some useful information from the original sentence.
5	A reader of the translated sentence can guess what the sentence is about, but the sentence provides hardly any useful information.
6	The sentence is totally incomprehensible or totally misleading.

Table 9.1: Grading guidelines for the syntax and semantics

(WSJ 75)
He said Pan Am currently has ”in excess of $150 million” in cash.

(a)	Er sagte, daß Pan Am z.Z. ” über $150 Million ” im Bargeld hat.	*Grammar:*____ *Meaning:*____
(b)	Er sagte, Pan Am habe gegenwärtig über 150 Mio. Dollar in Bargeld.	*Grammar:*____ *Meaning:*____
(c)	Er sagte, daß Pfannenvormittags zur Zeit ”mehr als 150 Millionen Dollar” in Bargeld hat.	*Grammar:*____ *Meaning:*____
(d)	Er sagte, daß Pfanne ist, hat gegenwärtig-” in Überschuß von $150 million” in in bar.	*Grammar:*____ *Meaning:*____

The questionnaires handed out for translation evaluation were hardcopies of the two Web pages http://www.cs.utexas.edu/users/ulf/eval_tegm.html, the version that always includes human translations, and http://www.cs.utexas.edu/users/ulf/ eval_tego.html, the version that does not include human translations, unless it happens to match one of the machine translations. Appendix D contains the questionnaire including human translations in dissertation format.

9.2 Test Results

9.2.1 Overview

Table 9.2 summarizes the evaluation results of translating 32 sentences from the Wall Street Journal from English to German.

Our system performed significantly better than the commercial systems, but this has to

System	Syntax	Semantics
Human translation	1.18	1.41
CONTEX on correct parse	2.20	2.19
CONTEX (full translation)	2.36	2.38
Logos	2.57	3.24
SYSTRAN	2.68	3.35
Globalink	3.30	3.83

Table 9.2: Summary of translation evaluation results (best possible = 1.00, worst possible = 6.00)

be interpreted with caution, since our system was trained and tested on sentences from the same lexically limited corpus (but of course without overlap), whereas the other systems were developed on and for texts from a larger variety of domains, making lexical choices more difficult in particular. Additionally, the syntactic style varies from to domain to domain, but we believe that this is less critical for this translation quality evaluation, because the Wall Street Journal already covers a wide range of syntactic constructs.

Note that the translation results using our parser are fairly close to those starting with a correct parse. This means that the errors made by the parser have had a relatively moderate impact on translation quality. The transfer and generation modules were developed and trained based on only 48 sentences, so we expect a significant translation quality improvement by further development of those modules.

Recall that Logos required a translation domain as an extra input parameter. To get an idea of the influence of this parameter, we repeated the translation with a domain choice of 'general' as opposed to 'business'. In 32 sentences, only four words/expressions were translated differently, one equally bad, one better for 'business' and two better for 'general'. The difference is statistically insignificant so that for our test sentences, the domain parameter apparently did not play a crucial role.

The impact of parse errors on the final translation quality depends of course greatly on the specific language pair. Germanic and Romance languages like English, German, French and Spanish are not only linked by springing from the same Western branch within the Indo-European language family; due to intensive political, commercial and cultural contacts of the native speakers of these languages throughout history, the languages continuously kept influencing each other.

Many ambiguities, like for example "with"-clauses, which in the instrumental case ("eat pasta with a fork") attach to the verb and which in the complementary case ("eat pasta with sauce") attach to the preceding noun phrase, are lexically and structurally homomorphic among these languages, so that an incorrect attachment would not be visible in the final translation, whereas other languages, as for example Japanese, make a clear lexical distinction in this case.[3] Therefore we would clearly expect a higher impact of parse errors on translation quality for an English to Japanese translation. On the other hand, translations between more closely related languages, e.g. between Swedish, Norwegian and Danish, would suffer even less from parse errors.

Despite the relative proximity of English and German, the impact of parse errors in our

[3]The complementary *with* translates as *"to"*, whereas the instrumental *with* translates as *"de"*.

system on translations is certainly smaller than had been expected. This seems to indicate that our parser is already fairly robust where it counts.

9.2.2 Variation Analysis

In a more detailed analysis of the evaluation results, we computed standard deviations for the various grades and grade differences as shown in table 9.3.

System	Syntax		Semantics	
	average	stand.dev.	average	stand.dev.
Human translation	1.18	0.29	1.41	0.31
difference	1.02	0.41	0.78	0.39
CONTEX starting on correct parse	2.20	0.45	2.19	0.46
difference	0.16	0.08	0.19	0.09
CONTEX (full translation)	2.36	0.43	2.38	0.47
difference	0.21	0.25	0.86	0.72
Logos	2.57	0.60	3.24	0.82
difference	0.11	0.21	0.12	0.17
SYSTRAN	2.68	0.52	3.35	0.86
difference	0.62	0.13	0.48	0.25
Globalink	3.30	0.60	3.83	0.73

Table 9.3: Standard deviation analysis of translation evaluation results

For machine translations, we find standard deviations in the range of 0.43 to 0.60 for syntax and 0.46 to 0.86 for semantics. However, the standard deviations for *differences* between adjacently ranking machine translations are much lower (0.08 to 0.25 for syntax and 0.09 to 0.72 for semantics). This means that while there are considerable differences in grading from one evaluator to another, the relative grading differences between systems are relatively small across different evaluators. So, while some evaluators graded more strictly than others, they all came up with fairly similar relative ratings.

Note in particular that while the difference in both syntax and semantics between the two versions of our system is quite small (0.16 and 0.19 respectively), the corresponding standard deviations are even smaller (0.08 and 0.09) indicating that while the evaluators rank the translations based on a correct parse only a little better than the full translations, this difference is quite consistent; in fact all evaluators graded the CONTEX translations starting on a correct parse to be better for both syntax and semantics.

Note that on the other hand the small differences between Logos and Systran (0.11 and 0.12) are accompanied by relatively big standard deviations (0.21 and 0.17); this is reflected by the fact that a minority of evaluators ranked SYSTRAN better than Logos. By and large the results from native English and German speaking evaluators are similar. However, the Americans tended to give worse grades. The most striking difference is for the syntax of the manual translations, which, relative to the best possible grade (1.00), are almost four times as negative for the native English speakers than for the countrymen of the translator. However, the data has to be interpreted very

Evaluators	American (3)		German (7)	
System	Syntax	Semantics	Syntax	Semantics
Human translation	1.39	1.54	1.10	1.36
CONTEX starting on correct parse	2.30	2.34	2.16	2.13
CONTEX (full translation)	2.44	2.55	2.33	2.30
Logos	2.83	3.32	2.45	3.20
SYSTRAN	2.71	3.36	2.67	3.35
Globalink	3.30	3.75	3.30	3.87

Table 9.4: Evaluation results differentiated by nationality of evaluator

cautiously, because the sub-sample sizes are quite small.

Another somewhat unexpected result was the grade for meaning preservation for the human control translation (1.41). A closer analysis revealed that more than a third of the difference from the optimal 1.00 resulted from disagreement over a single sentence (WSJ 53). All except one evaluator interpreted the sentence "*The St. Louis-based bank holding company previously traded on the American Stock Exchange.*" as meaning that the bank holding company was the agent of the trading, and therefore gave extremely bad grades to the human control translation which interpreted the sentence such that stocks of the bank holding company were traded on the American Stock Exchange. Monolingual domain experts (incl. from the University of Texas Business School), who were given this sentence without context, however confirm the passive reading of *traded*. The passive reading is further corroborated by the context of the WSJ article, which, of course, had not been available to the translation programs, the translator or the evaluators.

While such a misunderstanding has occurred in only one sentence to such an extent, and to a lesser degree also in a second sentence, and the overall consequences appear to be fairly limited, the evaluation of the control translations nevertheless demonstrates that translations can be quite hard to evaluate, even for bilingual speakers. Evaluators sometimes don't agree on substantial semantic issues, which results in evaluation differences.[4] Optimally, translators and evaluators should have top competency not only in the languages involved, but also in the translation subject domain area. Such a strong triple competency requirement greatly reduces the number of well qualified professional translators and reviewers, which, incidentally, is one of the main reasons why translation support by computers is so important: it can free the relatively scarce expert translators from the routine aspects of translation and thereby make them more efficient.

9.2.3 Correlation Analysis

Given evaluations on both parsing and full translation, an interesting question is whether the parsing evaluation criteria we used are a useful predictor for actual translation quality. To investigate this issue, we computed a matrix of correlation coefficients between our parsing evaluation criteria and the syntactic and semantic grades of the final (full) translation and show the results in table 9.5.

[4]However in the case just discussed, all machine translation programs selected the active reading for "traded" and therefore did not benefit or suffer from the controversial evaluations relative to each other.

	pr	rec	l_pr	l_rec	t_acc	cross	ops	opseq	match	str_l	synt	sem
pr	1.00	0.91	0.84	0.88	0.48	-0.87	0.42	0.49	0.94	0.69	-0.63	-0.63
rec	0.91	1.00	0.86	0.95	0.39	-0.84	0.53	0.57	0.97	0.80	-0.64	-0.66
l_pr	0.84	0.86	1.00	0.92	0.64	-0.66	0.56	0.57	0.95	0.80	-0.75	-0.78
l_rec	0.88	0.95	0.92	1.00	0.54	-0.71	0.54	0.55	0.98	0.78	-0.65	-0.65
t_acc	0.48	0.39	0.64	0.54	1.00	-0.25	0.21	0.20	0.54	0.28	-0.66	-0.56
cross	-0.87	-0.84	-0.66	-0.71	-0.25	1.00	-0.42	-0.47	-0.79	-0.66	0.58	0.54
ops	0.42	0.53	0.56	0.54	0.21	-0.42	1.00	0.77	0.54	0.72	-0.45	-0.41
opseq	0.49	0.57	0.57	0.55	0.20	-0.47	0.77	1.00	0.57	0.71	-0.39	-0.36
match	0.94	0.97	0.95	0.98	0.54	-0.79	0.54	0.57	1.00	0.80	-0.70	-0.71
str_l	0.69	0.80	0.80	0.78	0.28	-0.66	0.72	0.71	0.80	1.00	-0.62	-0.62
synt	-0.63	-0.64	-0.75	-0.65	-0.66	0.58	-0.45	-0.39	-0.70	-0.62	1.00	0.67
sem	-0.63	-0.66	-0.78	-0.65	-0.56	0.54	-0.41	-0.36	-0.71	-0.62	0.67	1.00

Table 9.5: Correlation matrix between various parsing and translation metrics. (pr = precision, rec = recall, l_pr = labeled precision, l_rec = labeled recall, t_acc = tagging accuracy, cross = number of crossing constituents, ops = number of correct operations, opseq = whether or not the entire operation sequence is correct, match = mixed index including precision, recall, labeled precision and labeled recall, str_l = whether or not the parse tree has the proper structure and labeling, synt = syntactic grade of the translated sentence, sem = semantic grade of the translated sentence)

Assuming an approximate linear correlation between two variables x and y, the *correlation coefficient* c indicates the strength of the correlation between the two variables. $|c|$ near 1.00 indicates a very strong correlation, whereas $|c|$ near 0.00 indicates a weak or no correlation. A *positive* c indicates that y tends to increase with an increasing x whereas a *negative* c indicates that y tends to decrease with an increasing x.

The correlation coefficient of a variable with itself is always 1.00. Correlation coefficients are symmetrical with respect to their two variables. An example for a high correlation is the one between *match* and *precision, recall, labeled precision* and *labeled recall* (0.94, 0.97, 0.95, and 0.98 respectively), which is no big surprise because *match* is an index composed of just those criteria. An example for a low correlation is between tagging accuracy and the number of crossing constituents, -0.25 (not because of the minus sign, but because of the low absolute value).

As it turns out, the best predictor for translation quality is labeled precision, with correlation coefficients of -0.75 and -0.78 for syntax and semantics respectively. Recall that on the German grading scale, 1 is the best possible and 6 the worst possible grade. Therefore a negative correlation coefficient here means that the better the labeled precision, the better the expected syntactic and semantic quality of the translation. A more detailed analysis has shown that labeled precision is also more strongly correlated to translation quality than any combination of labeled and unlabeled precision. The various parsing metrics have about the same correlation with the syntactic and semantic grades. Only tagging accuracy has a noticeably larger impact on syntax than on semantics (-0.66 and -0.56 respectively).

Figures 9.1 and 9.2 show the detailed distribution of labeled precision and the syntactic and semantic grades. The correlations are clearly visible, but by no means overwhelming, which

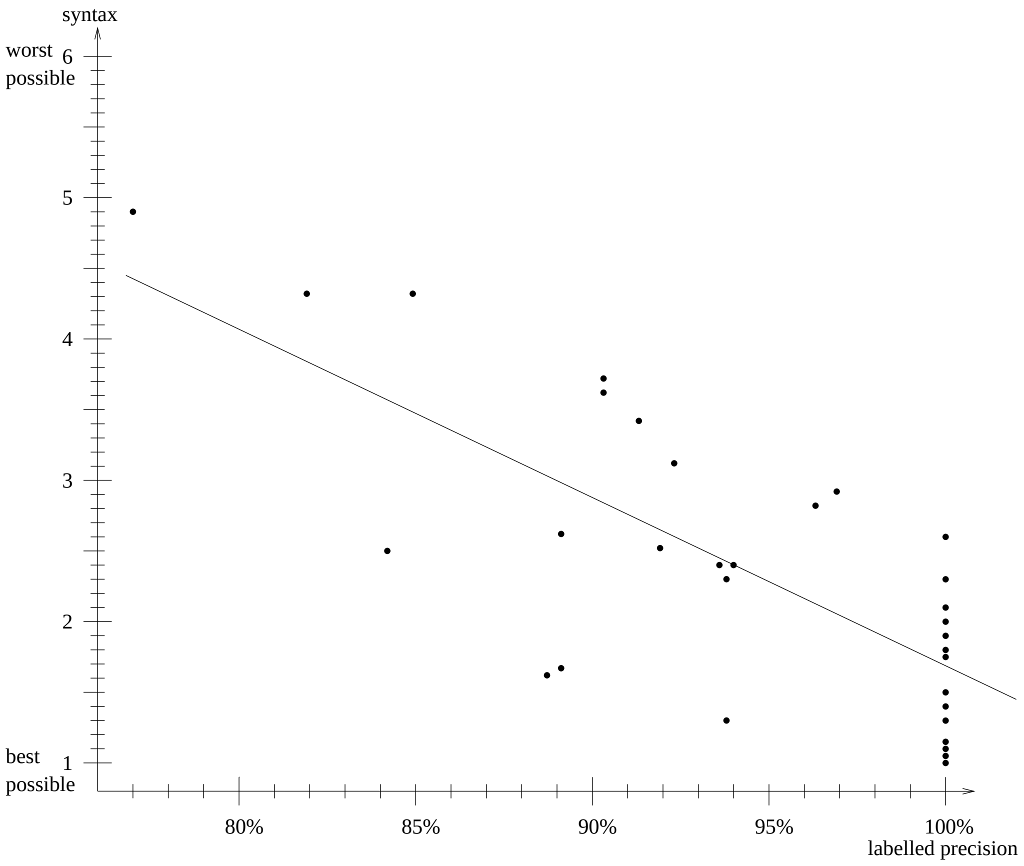


Figure 9.1: Correlation between labeled precision and syntax; coefficient = -0.75

concurs with our previous observation that the impact of parse errors on translation quality in our system is only limited.

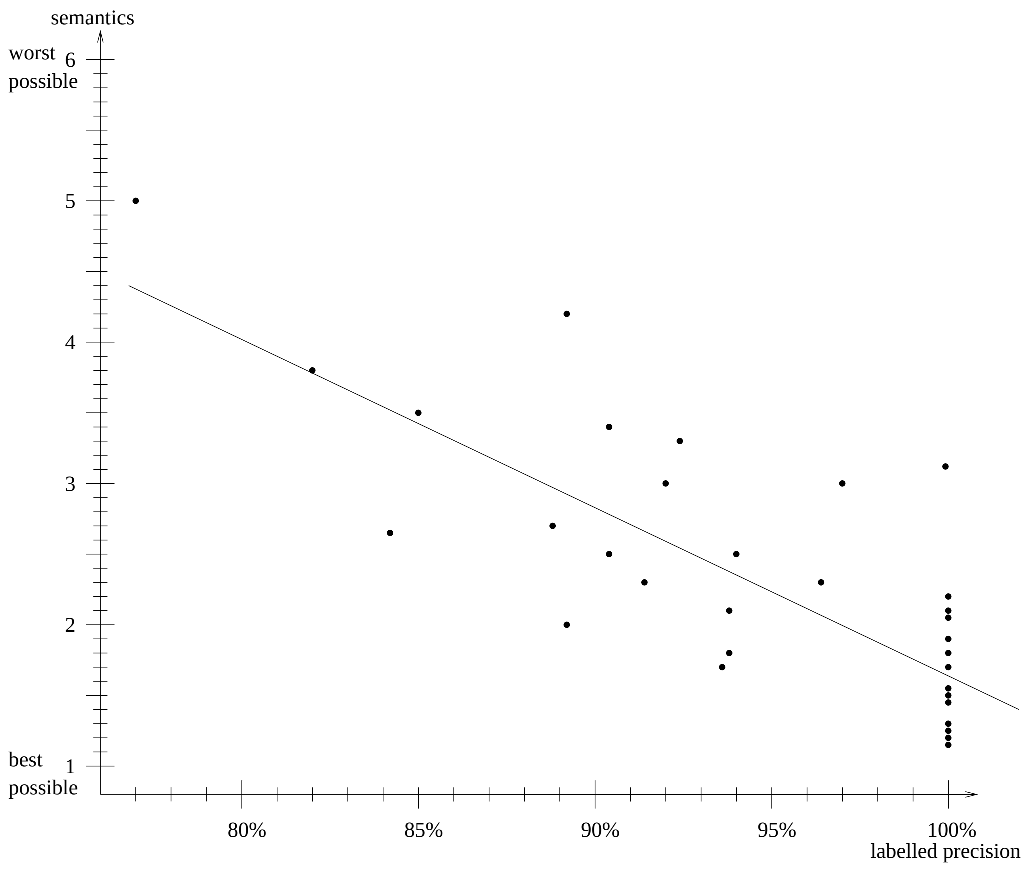


Figure 9.2: Correlation between labeled precision and semantics; coefficient = -0.78

Chapter 10

Related Work

In the introductory chapter, we already mentioned a number of papers representing the traditional approach to parsing with hand-crafted rules, as well as a number of references to learning-oriented approaches of 'sub-parsing' tasks. We now describe related empirical work on parsing.

10.1 Simmons and Yu

Our basic deterministic parsing and interactive training paradigm is based on (Simmons & Yu, 1992). Their "context-dependent grammar" (CDG) has the following characteristics:

- It uses 10 features, the five syntactic categories to the left and the five to the right of the current parse position.
- It has two parse actions, "*shift*" and "*reduce <new syntactic category>*".
- Words are pre-tagged, based on a *context-sensitive dictionary*, which examines a context of plus and minus three words for part of speech assignment.
- Output is a binary syntactic phrase structure; a separate system produces case structures.
- For a given partially parsed state, the system proposes the next parse action based on the 'closest' example, namely the one that maximizes

  $$\sum_{\text{f} \in \text{feature set}} \begin{cases} weight(f) & \text{if examples have matching values for feature f} \\ 0 & \text{otherwise} \end{cases}$$

  where *weight(f)* is is the distance of syntactic element referred to by feature *f* from the context window, i.e. 5 for the top element of the stack and the next input list element, and 1 less for each position that an element is further removed from the 'current position'.[1]

We have extended their work by

- significantly increasing the expressiveness of the feature language, in particular by moving far beyond the 10 simple features that were limited to syntax only;

[1]Since the examples are stored in a hash table using the top two parse stack elements as a key, examples that don't match these two features are actually not considered at all.

- significantly increasing the expressiveness of the parse action language by adding six new types of operations, adding (further) arguments to the shift and reduce operations, thereby, among other things, allowing discontinuous constituents and 'empty categories';
- adding background knowledge, in particular a KB and a subcategorization table;
- building a much richer internal structure, including both phrase-structure and case-frame information;
- providing morphological pre-processing;
- introducing a sophisticated machine learning component.

10.2 Parsers Learning From Treebank Examples

Several researchers have now used the Penn Treebank (Marcus et al., 1993) to develop parsers. The Penn Treebank is a corpus of over 4.5 million words of American English that has been annotated with part of speech. In addition, over half of it has been annotated for skeletal syntactic structure. For a better comparison of the systems, we first describe some representational differences between the Penn Treebank and our system, argue that parse action sequences carry more information, and finally discuss and compare specific treebank based parsers.

10.2.1 Comparison between Penn Treebank and CONTEX Parse Trees

The Penn Treebank provides only a skeletal syntactic structure, in which, most prominently, all internal structure of the noun phrase up through the head and including any single-word post-head modifiers is left unannotated. Experiments at the University of Pennsylvania have shown that this format increases the productivity (per word) of human annotators, a crucial aspect when annotating such a large corpus.

The output of CONTEX on the other hand provides a much more detailed integrated phrase-structure and case-frame tree that contains both syntactic and semantic classes and roles for the various constituents. Consider for example the representations of the noun phrase "*the October 1987 stock market crash*" as shown in table 10.1.

In the CONTEX representation, there is full syntactic and semantic class information plus morphological, lexical and other applicable values at all levels. (However, in these examples, only a reduced level of detail is actually displayed at lower levels to keep the outputs from quantitatively overwhelming the reader.) Appendix section C.3 gives an example for a complete parse tree.

The second example (see tables 10.2 and 10.3/10.4) shows the different representations for a full sentence ("*She wanted to avoid the morale-damaging public disclosure that a trial would bring.*"). The asterisk (*) in the Penn Treebank representation stands for an "understood" subject, and corresponds to the "<REF>2" in the CONTEX representation. Notice that the CONTEX representation provides syntactic and semantic roles, semantic classes, co-indexing and miscellaneous morphological and lexical information, but most important of all, once again, a more detailed structure.

Penn Treebank:

```
(NP the October 1987 stock market crash)
```

CONTEX:

```
"the October 1987 stock market crash":
   synt:    S-NP
   class:   I-EN-CRASH
   forms:   (((NUMBER F-SING) (PERSON F-THIRD-P)))
   lex:     "crash"
   subs:
   (DET)  "the":
       synt/class: S-DEF-ART/I-EART-DEF-ART
       forms: unrestricted
   (TIME)  "October 1987":
       synt/class: S-NP/C-AT-TIME
       forms: (SING 3P)
       (PRED) "October 1987":
           (PRED)  "October"
           (MOD)  "1987"
   (PRED)  "stock market crash":
       synt/class: S-COUNT-NOUN/I-EN-CRASH
       forms: (SING 3P)
       (MOD) "stock market":
           (MOD)  "stock"
           (PRED)  "market"
       (PRED)  "crash"
```

Table 10.1: Comparison of the syntactic structure of a complex NP as provided by the Penn Treebank and CONTEX.

Penn Treebank:

```
(S (NP she)
   (VP wanted
       (S (NP *)
          to
          (VP avoid
              (NP (NP the morale-damaging public disclosure)
                  (SBAR that
                        (S (NP a trial)
                           would
                           (VP bring)))))))
```

Table 10.2: Penn Treebank representation of the sentence "*She wanted to avoid the morale-damaging public disclosure that a trial would bring.*".

Contex:

```
"<She>2 wanted <REF>2 to avoid <the morale-damaging public disclosure
 <that>1 a trial would bring>1.":
   synt:    S-SNT
   class:   I-EV-WANT
   forms:   (((GENDER F-FEM) (PERSON F-THIRD-P) (NUMBER F-SING)
              (CASE F-NOM) (TENSE F-PAST-TENSE)))
   lex:     "want"
   subs:
   (SUBJ EXP)  <>2 "<She>2":
       synt/class: S-NP/I-EN-PERSONAL-PRONOUN
       forms: (SING 3P NOM FEM)
       props:   ((INDEX 2) (ORIG-SURF "She") (INDEXED TRUE))
       (PRED)  "She"
   (PRED)  "wanted":
       synt/class: S-VERB/I-EV-WANT
       forms: (PAST or PAST-PART)
   (INF-COMPL
    THEME)  "<REF>2 to avoid <the morale-damaging public disclosure
             <that>1 a trial would bring>1":
       synt/class: S-SNT/I-EV-AVOID
       forms: (TO-INF SING 3P NOM FEM)
       (SUBJ AGENT)  <REF>2"<REF>2":
       (PRED) "to avoid":
           (AUX)  "to"
           (PRED)  "avoid"
       (OBJ
        THEME)  <>1"<the morale-damaging public disclosure <that>1 a
                    trial would bring>1":
           (DET)  "the"
           (MOD) "morale-damaging":
               (MOD)  "morale"
               (DUMMY)  "-"
               (PRED)  "damaging"
           (MOD) "public":
               (PRED)  "public"
           (PRED)  "disclosure"
```

Continued in table 10.4

Table 10.3: Contex representation of the sentence "*She wanted to avoid the morale-damaging public disclosure that a trial would bring.*".

Continued from table 10.3

```
          (PRED)  "disclosure"
          (MOD) "<that>1 a trial would bring":
              (OBJ THEME) "<that>1":
                  (PRED)  <>1 "<that>1"
              (SUBJ INSTR) "a trial":
                  (DET)  "a"
                  (PRED)  "trial"
              (PRED) "would bring":
                  (AUX)  "would"
                  (PRED)  "bring"
(DUMMY)  ".":
    synt: D-PERIOD
```

Table 10.4: CONTEX representation, continued from table 10.3

In neither system are the parse trees actually entered from scratch. For the Penn Treebank sentences, a team of humans corrected automatically computed trees, while for CONTEX, a human corrected automatically computed parse actions[2] based on which the correct parse tree was continuously and automatically constructed.

10.2.2 Declarative vs. Operational Parse Information

This brings us to another essential difference with respect to parse information. The Penn Treebank is actually a collection of trees, whereas our system is based on parse action sequences. Since we can easily and automatically derive a parse tree from a parse action sequence, the later obviously carries at least as much 'linguistic' information. We believe that it carries in fact more information and is much better suited for training a parser, which has to break down the parsing of a sentence into smaller pieces anyway. Consider the example in table 10.5 with the sentence pair ("I ate the pasta with the expensive cheese/fork.") as represented in Penn Treebank format.

Despite the superficially similar surface structure of the sentences, the prepositional phrases have very different functions and accordingly attach differently: the complementational PP is an adjunct to the preceding noun phrase, whereas the instrumental PP is an adjunct to the verb.

Deriving the proper parse action sequences from a parse tree can now be quite tricky. If the parse action sequence just follows the bottom-up, left-to-right order of the parse tree, we can encounter problems as illustrated in the following example[3]:

1. I + ate + the pasta * with + the + expensive + cheese + .
2. I + ate + the pasta * with + the + expensive + fork + .

Following the parse tree order, the parser should, in the first example, shift in the preposition, because the prepositional phrase is part of the pasta noun phrase, whereas in the second case, the

[2]Except for the very first sentence, for which a complete parse action sequence had to be entered by hand; for all other sentences, a bootstrapping approach limited the manual entry to corrections; see section 5.7 for more details.

[3]As usual, the asterisk (*) indicates the current position, the pluses (+) separate the current partially parsed components.

```
(S (NP I)
   (VP ate
       (NP the pasta
           (PP with
               (NP the expensive cheese)))))
(S (NP I)
   (VP ate
       (NP the pasta)
       (PP with
           (NP the expensive fork))))
```

Table 10.5: Penn Treebank representation of "I ate the pasta with the expensive cheese/fork."

pasta noun phrase should now be combined with the verb. Assuming that the differentiating word (*cheese/fork*) is still out of the context window, the contexts of the two examples are identical and the parser must process both cases equally. This does not only pose a problem for a deterministic parser, but also for a statistical model, which would check both alternatives, noting the probabilities associated with each choice. But since the rest of the parse is unambiguous, there won't be any more probabilities below 1, resulting in total parse probabilities equal to the probabilities at the critical point, and thus attaching the PP the same way in both cases.

In our system on the other hand, we first parse the PP in both cases and then make an attachment decision when the parse is advanced to "*I + ate + the pasta + with the expensive cheese/fork * .*" Then the parser decides whether or not to attach the PP to the preceding noun phrase. If not, the PP is shifted back out onto the input list, the object is combined with the verb, the PP is shifted back in and combined with the verb phrase. This type of strategy is not included in a parse tree, but is automatically part of a parse action sequence, just by virtue of the supervisor guiding the parser that way during training.

10.2.3 SPATTER

Magerman (1994, 1995) uses a statistical decision tree model, training his system SPATTER with parse action sequences for 40,000 Wall Street Journal sentences derived from the Penn Treebank. Questioning the traditional n-grams, Magerman already advocates a heavier reliance on contextual information. Going beyond Magerman's still relatively rigid set of a little over 36 features, we propose a yet richer, basically unlimited feature language set. Our parse action sequences are too complex to be derived from a treebank like Penn's. Not only do our parse trees contain semantic annotations, roles and more syntactic detail, we also rely on the more informative parse action sequence. While this necessitates the involvement of a parsing supervisor for training, we are able to perform deterministic parsing and get already very good test results for only 256 training sentences.

10.2.4 IBM Language Modeling Group

Magerman's work was significantly influenced by the IBM Language Modeling Group. Black, Lafferty, and Roukos (1992) show how, using a treebank, the various parse rules of a hand-coded broad-coverage context-free phrase-structure grammar can automatically be assigned probabilities in order to better identify the parse tree that is most likely correct. Their "history-based grammar" (HBG) system (Black, Jelineck, Lafferty, Magerman, Mercer, & Roukos, 1993) enhances the probabilistic context-free grammar approach by providing more detailed (incl. semantic) linguistic information to resolve ambiguity.

Finally, the Candide system (Brown & et al., 1990; Berger & al., 1994) presents a statistical approach to machine translation. A target sentence is built word by word from left to right. The system keeps a hypothesis set of ranked partial target sentences. A partial target sentence (hypothesis) is extended by adding one word, and then re-ranked by estimating the probability that the new partial target sentence corresponds to the source sentence and that the new partial target sentence would occur like that in the target language. Recent additions to the system include (1) pre-processing of the source text, including part-of-speech tagging, morphological analysis, special treatment for numbers and names, and some local word reordering and normalizations, (2) a reverse post-processing of the target text, and (3) the introduction of context sensitivity to translation probabilities.

There is however no structural analysis of the source sentence. It seems that as a consequence, Candide works relatively well with short sentences[4], but it is not clear how much it can be improved to produce good translations for longer sentences without incorporating at least a shallow structural analysis. Reported results from an ARPA evaluation show that Candide's fluency of the target language was better than Systran's, that its adequacy was remarkable (.67 on a 0 to 1 scale compared to .743 for Systran) and that Candide could at least serve as a time-saving tool to produce translations that can be manually post-edited faster than a manual translation from scratch would take.

10.2.5 Bigram Lexical Dependencies

Another treebank based statistical parser, the bigram lexical dependencies (BLD) system (Collins, 1996), is based on probabilities between head-words in the parse tree, enhanced by additional distance features that check for order and adjacency of dependency candidates as well as intervening verbs, and intervening or surrounding commas. Trained on the same 40,000 sentences as SPATTER, it relies on a much more limited type of context than our system and needs little background knowledge.

10.2.6 Comparison of Results

The results in table 10.6 have to be interpreted cautiously since they are not based on the exact same sentences and detail of bracketing. Due to lexical restrictions, our average sentence length (17.1) is below the one used in SPATTER and BLD for a similar comparable sentence length range. While the Penn Treebank leaves many phrases such as "the New York Stock Exchange"

[4]In (Berger & al., 1994), the authors report that their "in-house evaluation methodology consists of fully-automatic translation of 100 sentences of 15 words or less".

System	Spatter	Spatter	BLD	CONTEX	CONTEX
Training Sentences	40,000	40,000	40,000	**64**	**256**
Background Knowledge	little	little	little	much	much
Test Sentence Length Range	**4-25**	**4-40**	≤ 40	4-45	4-45
Test Sentence Length Average	16.8	22.3	≈ 22	17.1	17.1
Labeled precision	88.1%	84.5%	86.3%	82.5%	89.8%
Labeled recall	87.6%	84.0%	85.8%	81.6%	89.6%
Crossings per sentence	0.63	1.33	1.14	1.87	1.02
Sent. with 0 crossings	69.8%	55.4%	59.9%	35.7%	56.3%
Sent. with ≤ 2 crossings	92.1%	80.2%	83.6%	66.9%	84.9%

Table 10.6: Comparing CONTEX with Magerman's SPATTER and Collins' BLD, all trained and tested on sentences from the Wall Street Journal.

without internal structure, our system performs a complete bracketing, thereby increasing the risk of crossing brackets.

The labeled precision and recall for CONTEX (trained on 256 sentences) surpass the rates for Spatter and BLD, even when compared to Spatter limited on sentences up to 25 words. This is partly due to the finer granularity of our parse trees, because precision and recall rates tend to be higher within the core[5] noun phrases. With respect to the three crossing criteria, CONTEX fares better when compared with longer sentences, and worse when compared with the shorter sentences. While the average sentence length for the shorter sentences (16.8) is only slightly shorter then our average of 17.1, the finer granularity that we use is now a disadvantage, because additional crossings are possible within core noun phrases.

However, the results are very encouraging, in particular since we can still expect significant improvement by increasing the number of our training sentences.

[5]By *core* noun phrases, we here mean those noun phrases that the Penn Treebank does not sub-structure any further.

Chapter 11

Future Work

The ways to extend our system are numerous and promising. Future work could further scale up the system, deal with incomplete knowledge, move beyond 'English to German', and improve the run time of the system.

11.1 Scaling Up Further

The analysis of the evaluation results revealed that the grades for translations that started with a correct parse are only very moderately better than those for a fully automated translation. As a consequence, the largest potential for improvement in translation quality lies in the stages beyond parsing. An investigation of the type of translation errors that occur when the system starts on a correct parse shows that it is clearly the generation module that is most responsible. Mistakes included wrong case assignment in prepositional phrases governed by prepositions that can assign both the dative and accusative case, and incorrect word order within a participle phrase and its place in the superior phrase. There are many other types of errors and it seems that the generation exhibits the need for modest fine-tuning in a multitude of places, rather than the resolution of a few principle problems. An example of a 'small', yet quite damaging shortcoming was the translation of mixed fractions, as often used in stock market notations. The parser correctly analyzed the four tokens '3', '1', '/', and '4' as the number representing $3 + \frac{1}{4}$, which it stored as the value of the number. In generation, this mixed fraction was however printed as '13/4', the default format of the programming language used in the implementation (Lisp). Even though $3\frac{1}{4} = \frac{13}{4}$, most evaluators counted the unified fraction as a very serious semantic error. One extra line of code can solve this problem, but it becomes clear that generation has to consider a host of smaller issues. Fortunately, the resolutions of the various problems seem to be relatively independent. Overall, the shortcomings of the generation module are actually quite understandable, given that it was developed based on only 48 sentences.

Considering that the parser was trained on 256 sentences, it already exhibits an amazing degree of robustness. The learning 'curve' of parsing quality, based on the number of training sentences, as shown in table 6.1, strongly suggests that more training examples will most likely still significantly improve parsing quality. This estimate is corroborated by the observation that many parse errors are due to obviously novel syntactic constructions that have not occurred in the training set.

11.1.1 New Domains

Another important avenue of future work is the extension of the current Wall Street Journal domain to others, in particular to text based on spoken language, which often includes quite peculiar or even technically ill-formed constructions.

We have used our parser to process a few sentences that other authors have used to make a point about their systems and found our system to be quite competitive. Consider for example the parse tree and translation of the ill-formed sentence shown in table 11.1. Even though such anecdotal evidence already suggests a degree of robustness in parsing when moving to sentences that are ill-formed or from other domains, adding additional examples from other domains might be quite beneficial, in particular when text other than well-formed complete sentences is used. Along with more examples, we might also add more features, particularly when moving to a new domain. We expect the number of 'specialized' features to be relatively limited though. In our current feature set, the most business oriented features are *(classp of i-en-currency-unit of -1 at m-boolean)*[1] and *(classp of i-en-monetarily-quantifiable-abstract of* -n *at m-boolean)*[2] for n = 1, 2, 3. It might be beneficial to have features indicating the domain, which might improve especially the quality of choice of alternative transfer concepts. The domain can probably be identified automatically by matching the vocabulary of the text to be translated with a typical domain vocabulary profile. Eventually, subcategorization tables might be added for nouns or even prepositions.

11.1.2 Lexical Expansion

Even though the amount of work to increase the coverage of lexicon and KB is quite small per word, the task to increase the currently slightly over 3000 entries of the English lexicon by one order of magnitude for an advanced vocabulary or two orders of magnitude to match Webster's Encyclopedic Unabridged Dictionary would have good use for further automation.

Riloff (1996) describes research in dictionary construction automation. The techniques described for information extraction applications can be adapted for our system. A key idea of successful information mining is to look for patterns that are typical for certain word classes and then have a supervisor confirm or reject likely candidates.

The bilingual dictionary is a prime candidate for automated acquisition. Bilingual corpora like the Canadian Hansards (parliamentary proceedings) or articles from the *Scientific American* along with their German translations in *Spektrum der Wissenschaft* can be exploited to identify co-occurrences of sentences (Gale & Church, 1991), words and expressions. Kay and Roescheisen (1993) even have been able to extract a word alignment table containing German/English word pairs based solely on the internal evidence of the bilingual corpus. As for the monolingual lexicon, alignment programs could propose likely candidates to a supervisor for approval or rejection.

The Champollion program (Smadja, McKeown, & Hatzivassiloglou, 1996) for example, using aligned text from the Canadian Hansards and given a multi-word English collocation, can identify the equivalent collocation in French with a precision of up to 78%. Champollion is limited to a statistical analysis of sets of words. The use of parsers and moderate background knowledge can certainly further improve the accuracy. Wu and Xia (1995) use similar statistical techniques to

[1] "Is the top element on the stack a currency unit?"
[2] "Is the *nth* element on the stack a monetarily quantifiable abstract?"

Parse tree for "*On February 27 I I in school.*":

```
"On February 27 I I in school.":
   synt:    S-SYNT-ELEM
   class:   I-EN-THING
   subs:
   (CONC)  "On February 27 I I":
       synt/class/roles: S-PP/C-AT-TIME/(CONC)
       forms: unrestricted
       (PRED)  "On"
       (PRED-COMPL) "February 27":
           (PRED) "February 27":
               (MOD)  "February"
               (PRED)  "27"
       (PRED-COMPL) "I I":
           (MOD) "I":
               (PRED)  "I"
           (PRED)  "I"
   (CONC)  "in school.":
       synt/class/roles: S-PP/C-IN-PP/(CONC)
       forms: unrestricted
       (PRED)  "in"
       (PRED-COMPL) "school":
           (PRED)  "school"
       (DUMMY)  "."
```

Subsequent translation to German, based on above parse tree:

```
"Am 27. Februar ich ich in der Schule."
```

Table 11.1: Parse tree and subsequent translation of an ill-formed sentence.
The sentence is a 'noisy' variation of "On February 27 I was in school.", both based on a German sentence pair from (Wermter & Weber, 1997). As usual, the parse tree is printed with decreasing detail at lower levels.
The parse tree is somewhat pathological (see section 5.5 for an explanation of *CONC* etc.), but adequately preserves the "healthy" parts of the input sentence. The German translation (which is based on the parse tree) precisely reflects the English original. It is encouraging to find this robustness even though CONTEX was never trained on any ill-formed sentences.

automatically extract an English-Chinese dictionary with 6429 English entries and report precision rates from 86% to 96%.

Finally, online dictionaries and other resources like lists of proper names can be used for lexical expansion. Even if the knowledge encoded in these resources is not deep enough, it is probably better than having no entry at all for a word or expression. The following section describes how such incomplete knowledge might be handled.

11.1.3 Considering Context Beyond the Sentence Boundary

An extension of the feature-encoded context beyond the sentence boundary is simpler than it might seem. The parse structure would just need additional slots for previous and following sentences and the feature language could easily be extended to access these slots. This might be useful for anaphora resolution, which sometimes depends on previous sentences, and for a better identification of the general type of text, e.g. business, science, sports, weather. Such topic features could then be used to make better decisions for ambiguous transfer. The relative ease of such an extension shows how useful it is that CONTEX very naturally integrates features of very different types and that it can automatically recompute the parse action examples and the decision structure to incorporate new features.

11.2 Incomplete Knowledge

We have used a lexically limited corpus for our research. While the restriction to 3000 words still allowed sentences with a very diverse cross-section of text, representing the enormous complexity of natural language, the problem of incomplete knowledge, like lexically unknown words, and corresponding missing entries in the subcategorization tables and the knowledge base, has been eliminated. For truly free text, any assumption of complete knowledge will always be unrealistic. We believe however that our paradigm of learning from examples lends itself particularly well for handling incomplete knowledge.

11.2.1 Unknown Words

Webster's Encyclopedic Unabridged Dictionary of the English Language (Webster's, 1994) contains over 250,000 lexical entries. Nevertheless, many technical terms like "*to disambiguate*" as well as most proper names are not included. Even if unknown words are unavoidable in free text, we nevertheless want to proceed with parsing, so that, for example, a sentence could still be translated, even if some individual words have to be left in the source language. To do proper parsing, we need to assign a part of speech to the unknown word. Since this is just another classification problem, machine learning seems to be the natural solution again. The context of the current parse state before shifting[3] in the unknown word can provide features, just like for 'normal' parsing itself. Additional features that access word endings or try to decompose a word into suffixes and known words will probably be useful. The ending "*-ated*" as in "*disambiguated*" for example is a typical verbal ending (past tense or past participle), and "*unrealistic*" could be identified as an adjective

[3]Recall that the shift-in operation includes an argument designating the part of speech; a shift-in operation therefore performs the so-called tagging.

composed of the prefix "*un-*" and the lexically known adjective "*realistic*". The syntactic context will certainly help as well to identify the proper part of speech of the unknown word. Endings might also be used to identify forms, such as tense for verbs and number for nouns. Weischedel and al. (1993) used features like these in a probabilistic model and were able to achieve a tagging accuracy rate of 85% for unknown words.

Examples for unknown word classification can be obtained by temporarily disabling some entries in the lexicon and thus rendering previously known words in already acquired parse action examples 'unknown'. Disabling entries that were added to the lexicon more recently is probably better than disabling lexicon entries at random, because the more recent lexicon additions typically present a better cross-section of words that have not been covered yet. While the ten most frequently occurring words of the full WSJ corpus (the, of, to, a, in, and, that, for, is, said) contain mostly words from closed word classes, i.e. word classes with a limited, often fairly small number of members, like for example articles, prepositions and conjunctions, later additions will at some point exclusively belong to open word classes like nouns (in particular proper names), verbs and adjectives. The 10 least frequent words of the 3000 most frequent WSJ words for example are *sluggish, repeatedly, Lewis, Caesars, quotas, Dominion, Lake, F, volatile* and *Wells*. Currently, the system just assumes that any unknown word is a noun. To obtain these additional examples, we let our parser run over the logged training sentences with part of the lexicon disabled and collect only those examples where some feature(s) in the current feature set accesses the unknown word; all other examples have already been collected during the regular automatic full parse example generation process (as described in section 5.7).

11.2.2 Incomplete Subcategorization Table

Subcategorization entries can be missing for some verbs that are in the lexicon and presumably just about all verbs that are not in the lexicon. In analogy to lexically unknown words, we can temporarily disable some subcategorization table entries and gain examples that lack any match in the subcategorization table. It is however important to distinguish this case from a situation where a match has been found, but no role is available for a specific potential phrase component. Using techniques described in (Manning, 1993), we could at least partially automate the acquisition of subcategorization entries from corpora.

11.2.3 Incomplete Knowledge Base

When unknown words get assigned a part of speech, they also need to be assigned a semantic concept from the knowledge base. Unless the context can provide some more specific semantic restrictions, this will have to be one of the very generic concepts, e.g. 'I-EADJ-ADJECTIVE' (some adjective), 'I-EV-PROCESS-STATE' (some process or state, for verbs), 'I-EN-THING' (some thing, for nouns). The entry for the unknown word should also be marked as semantically un(der)specified, to distinguish cases of lexically covered and uncovered words. Consider for example a test in which we check whether or not an entity can be an agent. While a negative result for the lexically covered word 'table' should, using the closed world assumption, be treated at face value, an equally negative test result for an unknown word should probably yield a special value like '*don't know*'. A feature value '*don't know*' can then be used like any other value in the decision structure building process.

11.3 Moving Beyond 'English to German'

In terms of functionality, our system could be expanded in several ways. Besides adding the capability to also translate from German to English, other languages could be added and our system might be used for other tasks such as a grammar checking.

11.3.1 German to English

For a translation in the opposite direction, much of the current system could be reused.

The only essential component that is totally missing for translation from German to English is the English generation module. It would not be very hard and time consuming to add such a module, because the basic program structure would be exactly the same. Starting with the duplicate of the German generation module, we would have to make a number of modifications, particularly with respect to the order of components in sentences and noun phrases; in the morphological propagation part, most of the changes would be deletions, since English is morphologically much simpler than German.

The transfer component is bidirectional and basically does not require any extensions. The only exception is the set of the ambiguous transfer examples, because transfer ambiguity is not symmetric. The current English to German ambiguous transfer example file with only 62 lines is short enough to justify our estimate that the corresponding German to English example file could be developed in a relatively short period of time.

A German parser already exists. It was trained to parse the German entries of the bilingual dictionary. Since the phrases in dictionary entries typically exhibit significantly less ambiguity and variation, fewer training examples were needed. In fact, the log file for the German examples is about 5% of the size of the log file containing the parse action sequences for the 272 Wall Street Journal sentences. For parsing quality comparable to the current English parser, the number of training examples would have to be increased to a similar level as for English. This would most likely be the most labor intensive aspect of an extension to translate from German to English.

Furthermore the German lexicon currently has fewer entries than the English lexicon (1039 as opposed to 3015) and there aren't any subcategorization tables for German yet. The morphological processing is fully bidirectional and thus does not need any further work.

11.3.2 Adding More Languages

For each additional language, a new morphological processor, a lexicon, a parse example training set, and a generation module would be required. Our dictionaries are **bi**lingual, but we don't necessarily have to develop a full matrix of bilingual dictionaries from scratch. Obviously, we need at least a set of dictionaries that link the languages that we want to translate to and from. For English/German/French/Spanish, we might for example have an German/English, an English/French, and a French/Spanish dictionary. When translating, we can then parse the source text using the source language parser, then perform a *series* of tree transfers, based on the dictionary 'graph', and then compute the final output using the target language generation module. This is far better than making full translations to intermediate languages, because we save both generation and parsing for each intermediate language, steps that can introduce a host of errors.

A more efficient approach though might be the 'compilation' of a dictionary for languages A and B and one for languages B and C into a dictionary for languages A and C. For each pair of matching entry pairs A_i/B_j and B_j/C_k, one could create an entry pair A_i/C_k in the compiled dictionary. For example, based on the entries "EINEN_1 Kompromiß erreichen"/"to reach SOME_1 compromise" and "to reach SOME_1 compromise"/"aboutir à UN_1 compromis", we would add "EINEN_1 Kompromiß erreichen"/"aboutir à UN_1 compromis" to the German/French dictionary. This can lead to overgeneration for ambiguous entries. Given the dictionary pairs "wissen"/"to know", "kennen"/"to know", "to know"/"savoir", and "to know"/"connaître", we would get four word pairs for German/French, even though only "wissen"/"savoir" and "kennen"/"connaître" are actually valid. These spurious dictionary entries can be deleted manually, or, based on ambiguous transfer decision structures from English to German and English to French, automatically.

Finally, the corresponding ambiguous transfer examples should be compiled into direct transfer examples, using the ambiguous transfer decision structures, and combining the sets of relevant features. To minimize these ambiguity mismatches, it is apparently wise to choose the original dictionary language pairs giving preference to related languages. E.g. given that Danish and Norwegian are very close to each other, Danish/Norwegian and Norwegian/Japanese are better than Danish/Japanese and Japanese/Norwegian when compiling a dictionary for the remaining language pair.

11.3.3 Grammar Checking

For any type of automatic grammar checker, it is hard to distinguish between the insufficiencies of the grammar checker and the text to be checked. For our system, the robustness of our parser, a benefit when it comes to machine translation, can present an additional disadvantage when it comes to grammar checking. In a sentence like "A dogs eats a bone." for example, our parser would probably not even check the number agreement within the subject or between subject and verb, because it does not help the system in making parse decisions. Both the general and the specific problem can be handled by actively training the system to find grammatical errors by also providing grammatically incorrect examples along with the error classification.

Two variants on how to do this come to mind:

1. The sentence is first parsed as usual and then the 'error examples' classify nodes of the parse tree. This resembles the decision making in transfer, which also operates on parse tree nodes.

2. 'Error examples' operate on parse states representing partially parsed sentences. This resembles the decision making in parsing, which also operates on parse states.

It seems that the first alternative is conceptually somewhat simpler, because it might be hard to link an error to a specific parse state, but the second alternative could possibly interact better with the actual parsing, by not only signaling a grammar problem, but also passing this information on to the 'running' parser, which might in some 'obvious' cases then automatically correct the error and thereby avoid follow-up problems.

11.4 Speed-Ups

Due to the deterministic nature of the parser, which keeps its time complexity **linear** with respect to the length of the sentence, the system is already very fast. On the hardware we use (a 70 Mhz SparcStation 5 with 32 MB of memory), our current system needs about 2.4 sec in total CPU time (2.9 sec real time) to translate an average sentence with 17 words from English to German (incl. segmentation, morphological analysis, parsing, transfer and generation). To train our parser on 11,822 examples, acquired under supervision from 272 sentences from the Wall Street Journal, the system needs 20 minutes in total CPU time (4.3 hours of real time).

We estimate[4] that the system can be sped up by a factor of 10 to 100 by measures including the following steps:

- reimplementation of the Lisp system in C++
- caching of information that is used repeatedly
- a more compressed format of examples
- adding a new parse mode that would allow parse actions to be destructive, i.e. don't force the parser to preserve all intermediate parse states; during supervised example acquisition it can be useful to 'backstep' to a previous parse state, but in actual applications of parsing such 'backstepping' is not needed anyway.

These speed-ups would result in virtually instantaneous translations of sentences and short texts.

[4]This estimate is based on previous experience of the author of this dissertation from work in the Knowledge Based Natural Language (KBNL) group at the Microelectronics and Computer Technology Corporation (MCC) in Austin, Texas, where the author reimplemented major parts of a Lisp parser in C++, achieving a speed-up of one to two orders of magnitude, and speeding up the JUMAN Japanese language segmenter (Matsumoto et al., 1994) by a factor of 60, mostly by caching information that was used repeatedly.

Chapter 12

Conclusions

Guided by the goal to advance the technology of robust and efficient parsers and machine translation systems for free text, we try to bridge the gap between the typically hard-to-scale hand-crafted approach and the typically large-scale but context-poor probabilistic approach.

Using

- a rich and unified context with 205 features,
- a complex parse action language that allows integrated part of speech tagging and syntactic and semantic processing,
- a sophisticated decision structure that generalizes traditional decision trees and lists,
- a balanced use of micromodular background knowledge and machine learning,
- a modest number of interactively acquired examples from the Wall Street Journal,

our system Contex

- computes parse trees and translations fast, because it uses a deterministic single-pass parser,
- shows good robustness when encountering novel constructions,
- produces good parsing results comparable to those of the leading probabilistic methods, and
- delivers competitive results for machine translations.

Finally, given that so far we trained our parser on examples from only 256 sentences, and developed our generation module based on only 48 sentences, we can still expect to improve our results very significantly, because the learning curve hasn't flattened out yet and adding substantially more examples is still very feasible. Besides scaling up further, extensions in other directions, such as more languages and grammar checking, are numerous and very promising.

Question: So when will the general purpose machine translation system
with 100% accuracy be ready?
Short answer: Never.

As many scientists have already pointed out, natural language processing in general and machine learning in particular are "AI-complete", i.e. NLP and MT can not be fully solved until artificial intelligence with all its other subdisciplines like knowledge representation and problem solving have been solved as well — until the computer has become omniscient. We recommend that you don't hold your breath for that.

The good news is that the relation between knowledge and accuracy is not linear. We believe that it is safe to assume that with less than "1% knowledge" we can achieve more than "99% accuracy" and that even within that fraction of one percent, a minute sub-fraction will yield a "90% accuracy". We believe that this strongly unbalanced nature of knowledge suggests a hybrid approach to machine translation. Since the "core knowledge" is relatively small yet critical for accuracy, it seems reasonable to acquire it under very careful human supervision or even provide it manually. The more we move away from the "core knowledge", the lower the rate of return will be. With increasingly larger quantities of additional knowledge necessary to reduce the error rate by a specific factor, it becomes more and more acceptable to allow rough approximations of knowledge. Probabilistic methods that can operate on large corpora of texts with little or no human intervention at all become more and more attractive. Even if the knowledge they build is not as "perfect" as if built under careful human supervision, it will still be useful, given that very large scale manual knowledge support becomes prohibitive and assuming that the final accuracy is still better than without such machine generated knowledge. The natural combination of more manually acquired "core knowledge" and more automatically acquired vast "outer knowledge" manifests itself in a tendency of significant MT systems to move towards more hybrid solutions as the systems are developed. While for example the originally extremely probabilistic Candide system (Brown & et al., 1990; Berger & al., 1994) has been augmented by a number of manual heuristics, the originally fairly manual-knowledge-based PANGLOSS Mark I/II/III system (Frederking & al., 1994) has incorporated less dogmatic example-based and transfer-based approaches.

Following the same trend, our system CONTEX, when scaled up further, will have to include ever more automatically acquired knowledge, but we believe that CONTEX already incorporates many characteristics of the successful hybrid system of the future: its machine learning can provide critical robustness, "real corpus"-based development can provide the necessary focus on critical knowledge, and a unified context of diverse features easily integrates knowledge from different sources.

Appendix A

WSJ Corpus

The WSJ corpus used in this work is a subset of the sentences from Wall Street Journal articles from 1987, as provided on the ACL data-disc. The WSJ corpus consists of the 272 first sentences that are fully covered by the 3000 most frequently occurring words of the entire corpus. These sentences are also available on the WWW at http://www.cs.utexas.edu/users/ulf/diss/wsj_corpus272.html .

(WSJ 0) Industry sources put the value of the proposed acquisition at more than $100 million.
(WSJ 1) A seat on the Chicago Board Options Exchange was sold for $340,000, up $5,000 from the previous sale March 11.
(WSJ 2) Of the total, 15 million shares will be sold by the company and the rest by a shareholder.
(WSJ 3) The securities are convertible at a rate of $27.61 of debentures for each common share.
(WSJ 4) The utility said it expects an additional $109 million (Canadian) in 1987 revenue from the rate increase.
(WSJ 5) It was cause to wonder.
(WSJ 6) The gold mines rose 12 points to 362.4.
(WSJ 7) The airline said that it would report a loss for the first quarter but that it would be "substantially less" than the $118.4 million deficit it posted in the first quarter of 1986.
(WSJ 8) In 1986, while general inflation was only 2%, hospital expenses grew 8%.
(WSJ 9) Banks will be offered a slightly higher interest-rate margin if they accept the notes than if they take a cash payment.
(WSJ 10) "It's a real high," said one.
(WSJ 11) "You can't get out of a falling market if you're in an index fund."
(WSJ 12) We're part of our own problem.
(WSJ 13) The average increase is about 1.8%, Toyota said.
(WSJ 14) This gave Japanese manufacturers a huge cost advantage over U.S. companies.
(WSJ 15) He said that this was the first he had heard about any such deal.
(WSJ 16) No one is particularly happy about the situation.
(WSJ 17) South Korea posted a surplus on its current account of $419 million in February, in contrast to a deficit of $112 million a year earlier, the government said.
(WSJ 18) Seats currently are quoted at $330,000 bid, $345,000 asked.
(WSJ 19) Sales are expected to increase about 10% to $40 million from $36.6 million in 1986.
(WSJ 20) In over-the-counter trading Friday, On-Line common closed at $22.25, down 50 cents.

(WSJ 21) The issue is one of Germany's biggest.
(WSJ 22) He said no.
(WSJ 23) The stock exchange index rose 2.39 to 1860.70.
(WSJ 24) For all of 1986, Pan Am reported a loss of $462.8 million, compared with 1985 net income of $51.8 million, or 45 cents a share.
(WSJ 25) The for-profit hospital, in contrast, has a powerful social voice in the form of equity markets.
(WSJ 26) Major U.S. accounting firms are split on how to value the notes.
(WSJ 27) "It's like you're king of the hill."
(WSJ 28) That makes index funds equal to 15% of institutional equity holdings, or about 7% of the entire stock market.
(WSJ 29) American industry can't just keep running to government for relief.
(WSJ 30) Toyota said the prices of 16 1987 models weren't increased.
(WSJ 31) But by the end of January, Japanese compensation costs had risen to 79% of the U.S. level, according to estimates by the Bureau of Labor Statistics.
(WSJ 32) The statement reported total assets of $40 million, net worth of $39 million and annual income of $908,000.
(WSJ 33) There's nothing he can say.
(WSJ 34) Industrial production in Italy declined 3.4% in January from a year earlier, the government said.
(WSJ 35) The record price of $342,000 for a full membership on the exchange was set Feb. 27.
(WSJ 36) The company said it doesn't expect the ruling to have a major impact on earnings because it had already set aside about $14 million in reserves to cover the judgment and reached an agreement for a bank loan to pay the balance.
(WSJ 37) The previous terms weren't ever disclosed.
(WSJ 38) That would have valued the holding at between $1.49 billion and $1.71 billion.
(WSJ 39) Others in Washington don't much care where new revenues come from.
(WSJ 40) Volume was about 1.5 billion shares, up from 1.4 billion Thursday.
(WSJ 41) Revenue declined 13% to $3.04 billion from $3.47 billion.
(WSJ 42) They represent only 17% of U.S. hospitals, and some of them are run on a for-profit basis.
(WSJ 43) But he declined to comment on the issue of the notes.
(WSJ 44) The S&P-500 stock index finished up 4.09 at 298.17 and the New York Stock Exchange composite index gained 2.09 to 169.37.
(WSJ 45) By 10 a.m., he was done.
(WSJ 46) It's a chance to get some improvement here.
(WSJ 47) "What would it take to get people to pick this up?"
(WSJ 48) Largely because of the falling dollar, West German labor costs rose to 120% of those for U.S. production workers from 75% in 1985.
(WSJ 49) In high school, he was a member of the speech team.
(WSJ 50) But the immediate concern is the short-term credits.
(WSJ 51) Canadian manufacturers' new orders fell to $20.80 billion (Canadian) in January, down 4% from December's $21.67 billion on a seasonally adjusted basis, Statistics Canada, a federal

agency, said.
(WSJ 52) The Federal Farm Credit Banks Funding Corp. plans to offer $1.7 billion of bonds Thursday.
(WSJ 53) The St. Louis-based bank holding company previously traded on the American Stock Exchange.
(WSJ 54) The transaction is expected to be completed by May 1.
(WSJ 55) A successor for him hasn't been named.
(WSJ 56) The president's news conference was a much-needed step in that direction.
(WSJ 57) The Tokyo exchange was closed Saturday as part of its regular holiday schedule.
(WSJ 58) Pan Am said its full-year results were hurt by foreign currency exchange losses of $46.8 million, primarily related to Japanese yen debt, compared with $11.1 million in 1985.
(WSJ 59) The American hospital sector spent $181 billion in 1986.
(WSJ 60) But the company inquiry, which began a few months ago, changed everything.
(WSJ 61) Texas Instruments rose 3 1/4 to 175 1/2.
(WSJ 62) If futures fell more than 0.20 point below the stocks, he would buy futures and sell stocks instead.
(WSJ 63) "I'm willing to work for a foreign company," says the father of two.
(WSJ 64) It estimated that sales totaled $217 million, compared with the year-earlier $257 million.
(WSJ 65) With the falling dollar, U.S. manufacturers are in a pretty good position to compete on world markets.
(WSJ 66) You go on to Bank B and get another loan, using some of Bank B's proceeds to pay back some of your debt to Bank A.
(WSJ 67) The critical point is to know when Brazil will produce an economic policy to generate foreign exchange to pay its interest.
(WSJ 68) Manufacturers' shipments followed the same trend, falling 1.5% in January to $21.08 billion, after a 2.8% increase the previous month.
(WSJ 69) The offerings will be made through the corporation and a nationwide group of securities dealers and dealer banks.
(WSJ 70) The real estate services company formerly traded over the counter.
(WSJ 71) The shares are convertible at a rate of $9.50 of preferred for each common share.
(WSJ 72) He continues as president of the company's corporate division.
(WSJ 73) In tests it has worked for many heart-attack victims.
(WSJ 74) The percentage change is since year-end.
(WSJ 75) He said Pan Am currently has "in excess of $150 million" in cash.
(WSJ 76) These revenues could then have been used for many purposes, such as funding those people without medical insurance.
(WSJ 77) The case continues in U.S. Bankruptcy Court in St. Louis.
(WSJ 78) American Express fell 1 1/2 to 77 1/4 on more than 2.1 million shares.
(WSJ 79) His typical trade involved $30 million.
(WSJ 80) Other voters reflect the divisions between their two senators.
(WSJ 81) However, the indicated fourth-quarter net loss is about $10 million.
(WSJ 82) In fact, unit labor costs have risen less in the U.S. than in other industrial countries.

(WSJ 83) First Interstate Bank of Denver – $5 million.
(WSJ 84) "He knows when to act and when to do nothing."
(WSJ 85) The company said it isn't aware of any takeover interest.
(WSJ 86) Earnings included a $615,000 write-down of an investment.
(WSJ 87) The $1.4375 Series A preferred stock is convertible at the rate of 1.0593 common shares for each preferred share.
(WSJ 88) In fact, no investigation was completed, but one is under way.
(WSJ 89) His approach never seems to fail.
(WSJ 90) It should be approved.
(WSJ 91) The card will cost $15 a year, in addition to the $45 that American Express charges for its regular card.
(WSJ 92) At the end of 1986, long-term debt totaled $830 million.
(WSJ 93) Construction is to begin immediately, with completion expected in late 1989.
(WSJ 94) Whatever the case, the three executives left within two days of one another.
(WSJ 95) The company filed an offering of 15.7 million shares.
(WSJ 96) You have to look for the hand at the other end.
(WSJ 97) But other voters think the issue is more complex.
(WSJ 98) The credit of about $44 million made net income $38.2 million, or $5.26 a share.
(WSJ 99) A look at Japan shows why.
(WSJ 100) That, too, was just fine with Mr. Clark, because Hutton also had an office in the building.
(WSJ 101) "They've clearly put the company up for sale."
(WSJ 102) Terms weren't disclosed, but the industry sources said the price was about $2.5 million.
(WSJ 103) The company expects the sale to be completed later this month, but said it still requires a definitive agreement.
(WSJ 104) It has 5,225,000 units outstanding.
(WSJ 105) In fiscal 1986, it earned a record $9.9 million, or 89 cents a share, on record revenue of $138.5 million.
(WSJ 106) Financial regulators closed several thrifts and banks over the weekend, including a Texas savings and loan association with $1.4 billion in assets.
(WSJ 107) That was two years ago.
(WSJ 108) But it will extend 13.5% credit.
(WSJ 109) Pan Am's unions, meanwhile, are trying to find another merger partner for the company.
(WSJ 110) "We needed someone who would be in it for the long run."
(WSJ 111) Has there been any improvement since then?
(WSJ 112) Pan Am was unchanged at 4 3/8 in active trading.
(WSJ 113) "It isn't an emergency situation where people have to trade at all costs," he says.
(WSJ 114) "It could help steel, but it would hurt coal because they have to sell out of the country."
(WSJ 115) The company currently has 10 million shares outstanding.
(WSJ 116) But when the yen wages are converted to dollars to show the impact of exchange rates,

compensation rose 47% to the equivalent of $9.50 in 1986 from $6.45 in 1985.
(WSJ 117) "Banks are highly competitive in trying to place good loans," he says.
(WSJ 118) He declined to elaborate.
(WSJ 119) The plant will be used by Texas Instruments' Defense Systems and Electronics Group to produce electronic equipment.
(WSJ 120) Canada's seasonally adjusted consumer price index rose 0.3% in February, Statistics Canada, a federal agency, said.
(WSJ 121) It said then that it would make a final decision by late March.
(WSJ 122) There is quite a difference.
(WSJ 123) It had assets of $115 million.
(WSJ 124) Yet last year Congress approved $900 million in grant military aid, the highest level since the end of the Vietnam War.
(WSJ 125) American Express says only a limited number of existing customers will be offered the new card.
(WSJ 126) They have made their own cost-cutting proposals, but the two sides haven't reached any agreements.
(WSJ 127) In December, the company sold six radio stations for $65.5 million.
(WSJ 128) Right on target, for 1979 or 1981.
(WSJ 129) A bankruptcy judge approved the company's $100 million financing agreement, an important step in its bankruptcy reorganization.
(WSJ 130) Growth in such investing was slow in the 1970s.
(WSJ 131) The only thing all voters seem to agree on is that something must be done.
(WSJ 132) It didn't give a reason for the sales.
(WSJ 133) Although the dollar began to fall two years ago, U.S. trade performance is just beginning to turn around.
(WSJ 134) The banking system, he says, "is built on trust."
(WSJ 135) He suggested that the two executives' interests might not reflect shareholders' interests.
(WSJ 136) But the company wouldn't elaborate.
(WSJ 137) The February rise followed increases of 0.2% in January and 0.4% in December.
(WSJ 138) It has 5.8 million units outstanding.
(WSJ 139) For the fourth quarter, the insurance-brokerage company posted a loss of $26.3 million, compared with a loss of $87.4 million in the year-earlier quarter.
(WSJ 140) First American will assume deposits of about $37 million in 7,200 accounts.
(WSJ 141) Of the administration's proposed $20 billion foreign-affairs budget for 1988, $15 billion would go to foreign aid.
(WSJ 142) Banks also believe that the American Express estimates are too modest, and some fear a plastic rate war.
(WSJ 143) We all had to take out second mortgages to help finance our investments in Stanley.
(WSJ 144) Per-share earnings were adjusted to reflect a 2-for-1 stock split paid in January.
(WSJ 145) "Reform" is not "revolution".
(WSJ 146) The next few days may determine whether the small, closely held airline has finally run out of fuel.

(WSJ 147) That's when they really decided to let it grow.
(WSJ 148) "There's no guarantee for anything in global competition."
(WSJ 149) Allegheny International makes consumer and industrial products.
(WSJ 150) About $184 million of debt is affected.
(WSJ 151) And a person intent on bank fraud "is going to get it done."
(WSJ 152) They did not give us any response to our offer.
(WSJ 153) Terms of the contract haven't been disclosed.
(WSJ 154) First Federal, which has about $270 million in assets, currently has 566,100 common shares outstanding.
(WSJ 155) Details about the proceeds weren't immediately available.
(WSJ 156) Revenue rose 11% to $99.3 million from $89.4 million.
(WSJ 157) New City had assets of about $21 million.
(WSJ 158) Of that $15 billion, 38% is for development aid, up from the administration's request of 33% last year.
(WSJ 159) Card rates haven't fallen nearly as sharply as other interest rates since 1980.
(WSJ 160) This came to a head during an early planning session when a production manager asked what the company policy was on a particular matter.
(WSJ 161) Industry officials said any combination probably would be the first step toward a merger of all five New York exchanges.
(WSJ 162) He had led her to believe it was worth more than $1 million.
(WSJ 163) Air Atlanta officials said the company may be able to get through the week with cash on hand.
(WSJ 164) It's expensive.
(WSJ 165) But it is likely that its economy won't expand this year at even the modest 2% rate forecast only a few months ago – and that it will actually decline in the current quarter.
(WSJ 166) "This means that, to a significant degree, we're using up the inventory found prior to 1970 at an increasing rate," he said.
(WSJ 167) S&P cited "expectations of continued profit pressures" for the company.
(WSJ 168) Mr. Lewis couldn't be reached for comment.
(WSJ 169) The truck maker said the group agreed not to buy Clark voting stock for 10 years.
(WSJ 170) Merrill Lynch also has an option to extend the agreement through February 1993.
(WSJ 171) "We now have a relatively small presence there."
(WSJ 172) As it happened, Far Eastern interest was limited.
(WSJ 173) Latest results included $24.2 million in losses from discontinued operations.
(WSJ 174) Hardly any of the participants in the tax debate, however, are taking account of the positive revenue effects of lower tax rates.
(WSJ 175) The total cost of those proposed issues to shareholders will be about the equivalent of $897.3 million.
(WSJ 176) In a Securities and Exchange Commission filing, the group including the New York investment company said it holds 555,057 shares, including 59,800 purchased Jan. 23 through Feb. 6 for $14 to $14.37 each.
(WSJ 177) "I don't know," I said.
(WSJ 178) Moreover, any merger would take time; industry executives suggested it would take

more than two years to work out.
(WSJ 179) Then the rally will continue, he said.
(WSJ 180) Despite the airline's problems, company officials say they believe Air Atlanta will get the capital needed to put it on its feet.
(WSJ 181) "It doesn't take much to realize that we have a very big problem."
(WSJ 182) Germany's inflation-adjusted growth of 2.4% last year was its worst showing since the economy climbed out of recession in 1983.
(WSJ 183) The coming period is a very critical economic period.
(WSJ 184) Data General couldn't be reached for comment.
(WSJ 185) Mr. Smith, who is a former partner of Bear, Stearns & Co., didn't return phone calls.
(WSJ 186) In composite trading on the Big Board, Clark common shares closed at $25, down $1.875.
(WSJ 187) The suit seeks class-action status on behalf of other company shareholders.
(WSJ 188) Ohio Power Co., a unit of American Electric Power Co., said it will redeem $34.4 million of its first mortgage bonds June 1.
(WSJ 189) The securities are convertible at a rate of $44.25 of debentures for each common share.
(WSJ 190) Revenue rose 8.4% to $390.9 million from $360.5 million in 1985.
(WSJ 191) Recent events have served to focus the debate.
(WSJ 192) London shares advanced to a record close Friday in moderate trading.
(WSJ 193) The company's shares rose 12.5 cents a share, to $14.125, in American Stock Exchange composite trading Friday.
(WSJ 194) "It's our company now; we have to create the policy."
(WSJ 195) But Sen. Dole announced Friday that he decided against offering the amendment.
(WSJ 196) In Chicago, the June contract on the Standard & Poor's 500-stock index soared to a record high of 299.80, up 3.80 for the day.
(WSJ 197) One reason the company is negotiating with potential investors is to provide money to lease additional aircraft.
(WSJ 198) It's a price we had to charge to stay in business.
(WSJ 199) That happened, to some extent.
(WSJ 200) It's an improvement over the first proposal, all right, but that's not saying very much at all.
(WSJ 201) The issue next goes to the cabinet-level Economic Policy Council, which has scheduled a meeting for mid-week.
(WSJ 202) The company previously declared a 3-for-2 split in November 1986.
(WSJ 203) In late New York trading Friday, the pound stood at $1.6042, up from $1.6003 Thursday, but eased to 2.9333 marks, from 2.9338.
(WSJ 204) The transaction is subject to regulatory approval.
(WSJ 205) Ohio Edison owns about 30% of the unit.
(WSJ 206) In composite trading on the New York Stock Exchange Friday, the company's common shares closed at $35.125, down 25 cents.
(WSJ 207) The deficit in January was $2.17 billion.
(WSJ 208) No doubt taxes are easier to reform in times of high economic growth.
(WSJ 209) The Financial Times industrial-share index rose 17.3 to 1598.9.

(WSJ 210) Pan Am Corp. reported a \$197.5 million fourth-quarter loss, worse than it had predicted, and said it expects to post a deficit in the first quarter.
(WSJ 211) Soon, they were no longer talking about what they had to have, but about what they no longer needed.
(WSJ 212) President Reagan suggested Friday that he would accept the earlier Senate version.
(WSJ 213) Stock prices started to rise as trading got under way Friday.
(WSJ 214) "What looks to the public like we were acting late was actually a situation in which we thought we were acting fast," the executive said.
(WSJ 215) The government has to take a lead role.
(WSJ 216) Export volume grew less than 1% last year, compared with a 6% increase in 1985.
(WSJ 217) Currently, there's still some air around Wright's museum.
(WSJ 218) Current practices require U.S. officials to draw up a list of potential targets and allow affected parties here to comment on their likely impact.
(WSJ 219) Attorneys for Mr. Boesky declined to comment.
(WSJ 220) Gold was quoted at \$406 an ounce in early trading Monday in Hong Kong.
(WSJ 221) The financial services company said about 100 employees of American Health will lose their jobs after the sale is completed in August.
(WSJ 222) Ohio Edison said the investors will pay it \$509 million for the stake, then lease it back for 29 years at a rate estimated at 8.5% to 9% of the cash payment.
(WSJ 223) Both the government and the opposition have refused to compromise on election reform.
(WSJ 224) The Reagan administration's most recent estimate for the fiscal 1987 deficit is \$143.91 billion.
(WSJ 225) The next move is up to the opposition Liberal-National coalition, which has still not announced its tax policy.
(WSJ 226) They expect retail issues to benefit from the budget's personal-tax cuts and the drop in British banks' lending rates, a trader said.
(WSJ 227) In the 1985 fourth quarter, Pan Am had net income of \$241.4 million, or \$1.79 a share, which included a \$341 million gain from the Pacific division sale.
(WSJ 228) The bank holding company is expected to make the announcement today.
(WSJ 229) "I am in full support of reasonable funding levels for these programs similar to the legislation passed by the Senate," he said in a statement put out by the White House.
(WSJ 230) Airline, drug, oil, technology, and some brokerage firm stocks took off.
(WSJ 231) He predicted that the break-even level will be reached in 1987's second half.
(WSJ 232) We have a lot of poor patients, and I would like to see them on the drug.
(WSJ 233) Capital spending by German firms this year is expected to run only slightly ahead of 1986 levels, after gains of 9% in 1985 and 5% last year.
(WSJ 234) The addition doesn't, of course, need to be this high.
(WSJ 235) U.S. officials estimate that the move will result in \$85 million in additional U.S. sales this year, and that such sales eventually could grow to \$300 million annually.
(WSJ 236) Competitors have recently supported the company's efforts to get regulation removed.
(WSJ 237) Every one-cent rise in the gasoline tax would raise \$900 million to \$1 billion a year.
(WSJ 238) Substantially all the delayed shipments should be made in the second quarter, the

company said.
(WSJ 239) Ohio Edison said the transaction will allow the equity investors to take advantage of federal tax benefits.
(WSJ 240) The company had 32.9 million shares outstanding at Dec. 31.
(WSJ 241) The deficit for all of fiscal 1986 was $220.7 billion.
(WSJ 242) We view it in one way.
(WSJ 243) And the benefits of lower mortgage costs, resulting from falling interest rates, a drop in money supply and stable inflation, will be "larger than the gains" from tax cuts, the trader said.
(WSJ 244) Without that gain, Pan Am would have reported a 1985 fourth-quarter loss of about $100 million.
(WSJ 245) Approval of the agreement between the Dallas-based aerospace, energy and steel concern and its 22-member bank group had been delayed since January.
(WSJ 246) It also raises the accounting question of how banks will value such notes.
(WSJ 247) Morgan Stanley & Co. reported shortly after 3 p.m. that it had blue-chip stock buy orders totaling $1.1 billion.
(WSJ 248) Now managing more than $60 billion for clients, mainly big pension funds, the firm has become the biggest single investor in the stock market.
(WSJ 249) We expect that they will.
(WSJ 250) The bureau said the number of housing units in the country will reach 100 million by the end of March.
(WSJ 251) We do this kind of work every day.
(WSJ 252) It also called on Japan to allow U.S. producers a larger share of Japan's own market.
(WSJ 253) Brazil's short-term financing from foreign banks could soon be cut by as much as $3 billion, according to bankers in the U.S. and Brazil.
(WSJ 254) The tax boost would be part of a $36 billion deficit-reduction package the House Democratic leadership seeks for the coming fiscal year.
(WSJ 255) Under the plan, the holder of each common share will receive on March 31 the right to buy one additional share for $3.
(WSJ 256) The following issues recently were filed with the Securities and Exchange Commission:
(WSJ 257) Holders are to vote at the April 23 annual meeting.
(WSJ 258) The government paid $13.7 billion in interest on the federal debt in February, up from $13.49 billion in January.
(WSJ 259) We're sure we can reach a compromise.
(WSJ 260) Foreign demand, Wall Street's performance Thursday and buying ahead of the new trading quarter, which starts today, helped as well, traders said.
(WSJ 261) In the 1986 fourth quarter, revenue declined 12%, to $797.3 million from $906.7 million in the year-earlier quarter.
(WSJ 262) No one at the unit could be reached for comment.
(WSJ 263) Talks on those issues will resume later this week.
(WSJ 264) It's not going to be up that much.
(WSJ 265) In contrast, conventional managers usually hold some cash reserves, which have hurt their performance recently.
(WSJ 266) Why didn't you set it at $100,000?

(WSJ 267) "There is a lot of housing out there," Mr. Young said.
(WSJ 268) Let's keep it the way Wright built it.
(WSJ 269) It raised more money from British institutions and took out its first major bank loans.
(WSJ 270) About $10 billion of trade credits and $5 billion of money-market deposits fall due March 31.
(WSJ 271) An additional $18 billion would come from spending cuts.

Appendix B

Features

For an explanation of the structure of the following 205 features, please refer to section 5.6.

B.1 Features Used for Parsing English

[1] **(synt of -5 at s-synt-elem)**
[2] **(synt of -4 at s-synt-elem)**
[3] **(synt of -3 at s-synt-elem)**
[4] **(synt of -2 at s-synt-elem)**
[5] **(synt of -1 at s-synt-elem)**
[6] **(synt of 1 at s-synt-elem)**
[7] **(synt of 2 at s-synt-elem)**
[8] **(synt of 3 at s-synt-elem)**
[9] **(similar of -3 with -1 at m-boolean)**
[10] **(class of -2 at c-app)**
[11] **(class of -1 at c-app)**
[12] **(class of 1 at c-app)**
[13] **(class of -2 at c-clause)**
[14] **(class of -1 at c-clause)**
[15] **(synt of -3 at d-other-delimiter)**
[16] **(synt of -2 at d-other-delimiter)**
[17] **(synt of last of -2 at d-dividing-delimiter)**
[18] **(synt of det of -1 at s-synt-elem)**
[19] **(synt of time of -2 at s-synt-elem)**
[20] **(synt of -1 at d-dividing-delimiter)**
[21] **(synt of -1 at d-other-delimiter)**
[22] **(synt of 1 at d-dividing-delimiter)**
[23] **(synt of 1 at d-other-delimiter)**
[24] **(f-finite-tense of -1 at m-boolean)**
[25] **(f-non-finite-tense of -1 at m-boolean)**
[26] **(f-part of -1 at m-boolean)**
[27] **(f-part of 1 at m-boolean)**

[28] (f-pres-part of -1 at m-boolean)
[29] (f-pres-part of 1 at m-boolean)
[30] (f-inf of -1 at m-boolean)
[31] (f-to-inf of -1 at m-boolean)
[32] (f-passive of -1 at m-boolean)
[33] (class of -2 at i-ec-conjunction)
[34] (class of 1 at i-en-agent)
[35] (class of subj of -1 at i-en-interrogative)
[36] (class of -2 at i-en-quantity)
[37] (class of -1 at i-en-quantity)
[38] (class of pred-compl of co-theme of -1 at i-en-quantity)
[39] (classp of i-ec-conjunction-introducing-correlative-apposition of -2 at m-boolean)
[40] (classp of i-eart-indef-art of -2 at m-boolean)
[41] (classp of i-en-prename-title of -1 at m-boolean)
[42] (classp of i-en-prename-title of -2 at m-boolean)
[43] (classp of i-eadv-verbal-degree-adverb of -2 at m-boolean)
[44] (classp of i-eadv-adverbial-degree-adverb of -2 at m-boolean)
[45] (classp of i-eadv-adjectival-degree-adverb of -2 at m-boolean)
[46] (classp of i-eadv-nominal-degree-adverb of -2 at m-boolean)
[47] (classp of i-eadv-verbal-degree-adverb of -1 at m-boolean)
[48] (classp of i-eadv-adverbial-degree-adverb of -1 at m-boolean)
[49] (classp of i-eadv-adjectival-degree-adverb of -1 at m-boolean)
[50] (classp of i-eadv-adjectival-degree-adverb of alt-adv of 1 at m-boolean)
[51] (classp of i-eadv-nominal-degree-adverb of -1 at m-boolean)
[52] (classp of i-en-temporal-unit of alt-nom of 2 at m-boolean)
[53] (classp of i-en-temporal-concept of -1 at m-boolean)
[54] (classp of i-en-temporal-interval of -1 at m-boolean)
[55] (classp of i-en-temporal-interval of -2 at m-boolean)
[56] (classp of i-en-process of -1 at m-boolean)
[57] (class of pred-compl of -1 at i-en-process)
[58] (classp of i-e-process of -2 at m-boolean)
[59] (classp of i-en-reason of -1 at m-boolean)
[60] (class of -1 at i-en-unit)
[61] (day-or-year of -1 at i-enum-cardinal)
[62] (day-or-year of 1 at i-enum-cardinal)
[63] (classp of i-en-month-of-the-year of -1 at m-boolean)
[64] (class of -4 at i-ep-preposition)
[65] (class of -3 at i-ep-preposition)
[66] (class of -2 at i-ep-preposition)
[67] (class of 2 at i-ep-preposition)
[68] (class of 1 at i-ev-process-state)
[69] (is-abbreviation of -1 at m-boolean)
[70] (is-indexed of -1 at m-boolean)

[71] (capitalization of pred* of -2 at m-boolean)
[72] (capitalization of pred* of -1 at m-boolean)
[73] (capitalization of pred* of 1 at m-boolean)
[74] (marked-capitalization of pred* of -1 at m-boolean)
[75] (is-abbreviation of -2 at m-boolean)
[76] (is-ref of -1 at m-boolean)
[77] (is-ref of subj of -1 at m-boolean)
[78] (is-ref of subj of inf-compl of -1 at m-boolean)
[79] (is-ref of -2 at m-boolean)
[80] (np-vp-match of -2 with -1 at m-boolean)
[81] (np-vp-match of -1 with 1 at m-boolean)
[82] (compl-v-match of -2 with -1 at m-boolean)
[83] (compl-v-match of -1 with 1 at m-boolean)
[84] (marked-capitalization of 1 at m-boolean)
[85] (syntp of unavail of lexical of 5 at m-boolean)
[86] (syntp of unavail of lexical of 3 at m-boolean)
[87] (syntp of unavail of lexical of 2 at m-boolean)
[88] (syntp of unavail of lexical of ((pred* of -2) -1) at m-boolean)
[89] (syntp of unavail of lexical of (-1 1) at m-boolean)
[90] (syntp of unavail of lexical of (-1 1 2) at m-boolean)
[91] (syntp of unavail of lexical of (-2 -1 1) at m-boolean)
[92] (semrole of -1 of -2)
[93] (semrole of -1 of -4)
[94] (semrole of 1 of -1)
[95] (semrole of -1 of -3)
[96] (semrolep of -1 of -3)
[97] (semrole of -2 of -3)
[98] (semrolep of -2 of -3)
[99] (semrolep of obj of -1)
[100] (semrole of subj of -1)
[101] (syntrole of -2 of -1)
[102] (synt of -1 at s-clause)
[103] (synt of -2 at s-clause)
[104] (classp of i-en-interr-pronoun-whatever of -2 at m-boolean)
[105] (classp of i-eprt-adv-particle of -1 at m-boolean)
[106] (classp of i-en-letter-character of -1 at m-boolean)
[107] (classp of i-en-day-of-the-week of -1 at m-boolean)
[108] (classp of i-en-day-of-the-week of alt-nom of 1 at m-boolean)
[109] (classp of i-eadj-able of -1 at m-boolean)
[110] (classp of i-eadj-able of 1 at m-boolean)
[111] (synt of -1 at s-particle)
[112] (synt of 1 at s-particle)
[113] (syntp of unavail of vp-1 at m-boolean)

[114] **(syntp of unavail of npp-1 of -2 at m-boolean)**
[115] **(syntp of unavail of npp-1 of -3 at m-boolean)**
[116] **(syntp of unavail of conj of -1 at m-boolean)**
[117] **(syntp of unavail of pred of -1 at m-boolean)**
[118] **(syntp of unavail of pred of -2 at m-boolean)**
[119] **(syntp of unavail of from-quant of -2 at m-boolean)**
[120] **(syntp of unavail of pred-compl of -1 at m-boolean)**
[121] **(syntp of unavail of time of pred-compl of -1 at m-boolean)**
[122] **(syntp of d-dividing-delimiter of last of -1 at m-boolean)**
[123] **(syntp of unavail of obj of -1 at m-boolean)**
[124] **(syntp of unavail of inf-compl of -1 at m-boolean)**
[125] **(synt of alt2 of 1 at s-synt-elem)**
[126] **(synt of alt3 of 1 at s-synt-elem)**
[127] **(synt of -3 at s-verb)**
[128] **(syntp of s-aux of -1 at m-boolean)**
[129] **(classp of i-en-point of -1 at m-boolean)**
[130] **(classp of c-at-time of -1 at m-boolean)**
[131] **(classp of c-at-time of -2 at m-boolean)**
[132] **(classp of i-eadv-there of -2 at m-boolean)**
[133] **(classp of i-ev-be of -1 at m-boolean)**
[134] **(classp of i-ep-to of -1 at m-boolean)**
[135] **(classp of i-ep-to of 1 at m-boolean)**
[136] **(classp of i-en-agent of -5 at m-boolean)**
[137] **(classp of i-en-place of -5 at m-boolean)**
[138] **(classp of i-en-quantity of -5 at m-boolean)**
[139] **(classp of i-en-quantity of -4 at m-boolean)**
[140] **(classp of i-en-agent of -3 at m-boolean)**
[141] **(classp of i-en-place of -3 at m-boolean)**
[142] **(classp of i-en-quantity of -3 at m-boolean)**
[143] **(classp of i-en-tangible-object of -3 at m-boolean)**
[144] **(classp of i-en-pronoun of -3 at m-boolean)**
[145] **(classp of i-en-pronoun of -2 at m-boolean)**
[146] **(classp of i-en-interr-pronoun of -2 at m-boolean)**
[147] **(classp of i-en-agent of -2 at m-boolean)**
[148] **(classp of i-en-proper-place of -2 at m-boolean)**
[149] **(classp of i-en-place of -2 at m-boolean)**
[150] **(classp of i-en-quantity of -2 at m-boolean)**
[151] **(classp of i-en-temporal-entity of -2 at m-boolean)**
[152] **(classp of i-en-currency-unit of -1 at m-boolean)**
[153] **(classp of i-en-agent of -1 at m-boolean)**
[154] **(classp of i-en-place of -1 at m-boolean)**
[155] **(classp of i-en-percent of -1 at m-boolean)**
[156] **(classp of i-en-percentage of -1 at m-boolean)**

[157] (classp of i-en-pronoun of -1 at m-boolean)
[158] (classp of i-en-personal-pronoun of -1 at m-boolean)
[159] (classp of i-en-personal-pronoun of -2 at m-boolean)
[160] (classp of i-en-monetarily-quantifiable-abstract of -1 at m-boolean)
[161] (classp of i-en-monetarily-quantifiable-abstract of -2 at m-boolean)
[162] (classp of i-eadv-quantifier of -2 at m-boolean)
[163] (classp of i-en-mod-abstract of -3 at m-boolean)
[164] (classp of i-en-monetarily-quantifiable-abstract of -3 at m-boolean)
[165] (classp of i-en-quantifying-abstract of -3 at m-boolean)
[166] (syntp of s-indef-pron of pred* of -3 at m-boolean)
[167] (syntp of d-delimiter of last of -3 at m-boolean)
[168] (synt of -2 at s-conj)
[169] (classp of i-eadv-temporal-quantifier of -1 at m-boolean)
[170] (classp of i-en-tangible-object of -1 at m-boolean)
[171] (classp of i-en-quantity of -1 at m-boolean)
[172] (classp of i-en-temporal-entity of -1 at m-boolean)
[173] (classp of i-en-unit of -1 at m-boolean)
[174] (classp of i-en-agent of subj of -1 at m-boolean)
[175] (classp of i-en-place of subj of -1 at m-boolean)
[176] (classp of i-en-tangible-object of alt-nom of 1 at m-boolean)
[177] (classp of i-en-agent of alt-nom of 1 at m-boolean)
[178] (classp of i-en-place of alt-nom of 1 at m-boolean)
[179] (classp of i-en-temporal-entity of 1 at m-boolean)
[180] (classp of i-en-unit of alt-nom of 1 at m-boolean)
[181] (classp of i-en-agent of alt-nom of 2 at m-boolean)
[182] (classp of i-en-place of alt-nom of 2 at m-boolean)
[183] (classp of i-en-tangible-object of alt-nom of 2 at m-boolean)
[184] (classp of i-en-tangible-object of -2 at m-boolean)
[185] (classp of i-en-temporal-entity of 2 at m-boolean)
[186] (classp of i-en-unit of alt-nom of 2 at m-boolean)
[187] (classp of i-en-interrogative of active-filler-1 at m-boolean)
[188] (classp of i-en-interrogative of -2 at m-boolean)
[189] (classp of i-en-interrogative of -1 at m-boolean)
[190] (classp of i-en-interr-pronoun of -1 at m-boolean)
[191] (classp of i-en-interrogative of 1 at m-boolean)
[192] (classp of i-eadv-at-location of -1 at m-boolean)
[193] (classp of i-eadv-at-location of alt-adv of 1 at m-boolean)
[194] (classp of i-eadv-to-location of alt-adv of 1 at m-boolean)
[195] (gap of active-filler-1 at m-filler-status)
[196] (gap of -2 at m-filler-status)
[197] (synt of -2 at s-app)
[198] (synt of -1 at s-app)
[199] (synt of 1 at s-app)

[200] **(synt of -3 at s-numeral)**
[201] **(synt of -2 at s-numeral)**
[202] **(synt of -1 at s-numeral)**
[203] **(synt of 1 at s-numeral)**
[204] **(synt of 1 at s-conj)**
[205] **(classp of i-en-demonstr-pronoun of -2 at m-boolean)**

B.2 Features Used in English to German Translation Disambiguation Decision Structures

The following is a list of features used for English to German translation disambiguation. Recall that the transfer examples and features are organized in *transfer entries*, each of which contains the example phrases and features for a specific source concept. For each individual transfer entry, only a small subset of the following features is used.

[1] **(class of parent of parent at c-app)**
[2] **(classp of c-at-time of parent at m-boolean)**
[3] **(classp of c-to-quant of parent at m-boolean)**
[4] **(classp of i-eadv-neg-adverb of quantifier of parent at m-boolean)**
[5] **(classp of i-en-agent of pred-compl of parent at m-boolean)**
[6] **(classp of i-en-agent of theme of parent at m-boolean)**
[7] **(classp of i-en-exchange-boerse of pred-compl of parent at m-boolean)**
[8] **(classp of i-en-proper-place of mod of parent at m-boolean)**
[9] **(classp of i-en-proper-place of pred-compl of parent at m-boolean)**
[10] **(classp of i-en-tangible-object of theme of parent at m-boolean)**
[11] **(number at f-number)**
[12] **(synt of agent of parent at s-synt-elem)**
[13] **(synt of parent of parent at s-synt-elem)**
[14] **(syntp of s-np of theme of parent at m-boolean)**
[15] **(syntp of s-sub-clause of theme of parent at m-boolean)**
[16] **(syntp of unavail of det of pred-compl of parent at m-boolean)**
[17] **(syntp of unavail of obj of parent at m-boolean)**
[18] **(voice of parent at f-voice)**

Appendix C

Parse Example

The following sections show the sentence "*I like the house Mr. Miller built.*" after segmentation and morphological analysis, the complete sequence of partial parse states and parse actions, and finally the resulting parse tree.

C.1 After Segmentation and Morphological Processing

```
 "I":
    synt:    S-PRON
    class:   I-EN-PERSONAL-PRONOUN
    forms:   (((NUMBER F-SING) (PERSON F-FIRST-P) (CASE F-NOM)))
    lex:     "PRON"
    props:   ((CAPITALIZATION TRUE))
(OR
 "like":
    synt:    S-PREP
    class:   I-EP-LIKE
    forms:   (NIL)
    lex:     "like"
 "like":
    synt:    S-VERB
    class:   I-EV-LIKE
    forms:   (((PERSON F-SECOND-P) (NUMBER F-SING)
               (TENSE F-PRES-TENSE))
              ((PERSON F-FIRST-P) (NUMBER F-SING) (TENSE F-PRES-TENSE))
              ((NUMBER F-PLURAL) (TENSE F-PRES-TENSE))
              ((TENSE F-PRES-INF)))
    lex:     "like")
 "the":
    synt:    S-DEF-ART
    class:   I-EART-DEF-ART
    forms:   (NIL)
    lex:     "the"
 "new":
    synt:    S-ADJ
    class:   I-EADJ-NEW
    forms:   (NIL)
    lex:     "new"
```

```
   props:   ((ADJ-TYPE S-NON-DEMONSTR-ADJ))
"house":
   synt:    S-COUNT-NOUN
   class:   I-EN-HOUSE
   forms:   (((NUMBER F-SING) (PERSON F-THIRD-P)))
   lex:     "house"
"Mr":
   synt:    S-COUNT-NOUN
   class:   I-EN-MISTER-TITLE
   forms:   (((NUMBER F-SING) (PERSON F-THIRD-P)))
   lex:     "Mr"
   props:   ((CAPITALIZATION TRUE) (IS-ABBREVIATION TRUE))
".":
   synt:    D-PERIOD
   lex:     "."
"Miller":
   synt:    S-PROPER-NAME
   class:   I-EN-MILLER
   forms:   (((NUMBER F-SING) (PERSON F-THIRD-P)))
   lex:     "Miller"
   props:   ((CAPITALIZATION TRUE))
"Miller":
   synt:    S-PROPER-NAME
   class:   I-EN-MILLER
   forms:   (((NUMBER F-SING) (PERSON F-THIRD-P)))
   lex:     "Miller"
   props:   ((CAPITALIZATION TRUE))
"built":
   synt:    S-VERB
   class:   I-EV-BUILD
   forms:   (((TENSE F-PAST-PART)) ((TENSE F-PAST-TENSE)))
   lex:     "build"
".":
   synt:    D-PERIOD
   lex:     "."
```

C.2 Parse State and Actions Step by Step

```
* I like the new house Mr . Miller built .
   (S S-PRON)
(I) * like the new house Mr . Miller built .
   (R 1 TO S-NP AS PRED)
(I) * like the new house Mr . Miller built .
   (S S-VERB)
(I) (like) * the new house Mr . Miller built .
   (R 1 TO S-VP AS PRED)
(I) (like) * the new house Mr . Miller built .
   (S S-ART)
(I) (like) (the) * new house Mr . Miller built .
   (S S-ADJ)
(I) (like) (the) (new) * house Mr . Miller built .
   (R 1 TO S-ADJP AS PRED)
(I) (like) (the) (new) * house Mr . Miller built .
```

```
   (S S-NOUN)
(I) (like) (the) (new) (house) * Mr . Miller built .
   (R 1 TO S-NP AS PRED)
(I) (like) (the) (new) (house) * Mr . Miller built .
   (R 2 AS MOD SAME
(I) (like) (the) (new house) * Mr . Miller built .
   (R 2 AS DET SAME)
(I) (like) (the new house) * Mr . Miller built .
   (EMPTY-CAT FROM NP-1 AT 0) (M -1 GAP ACTIVE-FILLER)
(I) (like) (<the new house>1) (<REF>1) * Mr . Miller built .
   (S S-NOUN)
(I) (like) (<the new house>1) (<REF>1) (Mr) * . Miller built .
   (S D-DELIMITER)
(I) (like) (<the new house>1) (<REF>1) (Mr) (.) * Miller built .
   (R 2 AS PRED DUMMY)
(I) (like) (<the new house>1) (<REF>1) (Mr.) * Miller built .
   (S S-NOUN)
(I) (like) (<the new house>1) (<REF>1) (Mr.) (Miller) * built .
   (R 2 AS MOD PRED)
(I) (like) (<the new house>1) (<REF>1) (Mr. Miller) * built .
   (R 1 TO S-NP AS PRED)
(I) (like) (<the new house>1) (<REF>1) (Mr. Miller) * built .
   (S S-VERB)
(I) (like) (<the new house>1) (<REF>1) (Mr. Miller) (built) * .
   (R 1 TO S-VP AS PRED)
(I) (like) (<the new house>1) (<REF>1) (Mr. Miller) (built) * .
   (R 2 TO S-SNT AS (SUBJ AGENT) SAME)
(I) (like) (<the new house>1) (<REF>1) (Mr. Miller built) * .
   (R 2 TO S-REL-CLAUSE AS (OBJ THEME) SAME)
(I) (like) (<the new house>1) (<REF>1 Mr. Miller built) * .
   (R 2 AS SAME MOD)
(I) (like) (<the new house <REF>1 Mr. Miller built>1) * .
   (R 2 AS SAME (OBJ THEME))
(I) (like <the new house <REF>1 Mr. Miller built>1) * .
   (R 2 TO S-SNT AS (SUBJ EXP) SAME)
(I like <the new house <REF>1 Mr. Miller built>1) * .
   (S D-DELIMITER)
(I like <the new house <REF>1 Mr. Miller built>1) (.) *
   (R 2 AS SAME DUMMY)
(I like <the new house <REF>1 Mr. Miller built>1.) *
   (DONE)
```

C.3 Resulting Parse Tree

```
"I like <the new house <REF>1 Mr. Miller built>1.":
    synt:    S-SNT
    class:   I-EV-LIKE
    forms:   (((CASE F-NOM) (TENSE F-PRES-TENSE) (PERSON F-FIRST-P)
               (NUMBER F-SING)))
    lex:     "like"
    props:   ((CACHED-BEST-PATTERN 25
               (((SUBJ EXP (I-EN-AGENT)) (OBJ THEME (I-EN-THING)))
                (( <>1 "<the new house <REF>1 Mr. Miller built>1" 1)
            ( "I" 0)))))
subs:
(SUBJ EXP)  "I":
    synt:    S-NP
    class:   I-EN-PERSONAL-PRONOUN
    forms:   (((NUMBER F-SING) (PERSON F-FIRST-P) (CASE F-NOM)))
    lex:     "PRON"
    subs:
    (PRED)  "I":
        synt:    S-PRON
        class:   I-EN-PERSONAL-PRONOUN
        forms:   (((NUMBER F-SING) (PERSON F-FIRST-P) (CASE F-NOM)))
        lex:     "PRON"
        props:   ((CAPITALIZATION TRUE))
(PRED)  "like":
    synt:    S-VERB
    class:   I-EV-LIKE
    forms:   (((PERSON F-SECOND-P) (NUMBER F-SING)
               (TENSE F-PRES-TENSE))
              ((PERSON F-FIRST-P) (NUMBER F-SING)
               (TENSE F-PRES-TENSE))
              ((NUMBER F-PLURAL) (TENSE F-PRES-TENSE))
              ((TENSE F-PRES-INF)))
    lex:     "like"
(OBJ THEME)  "<the new house <REF>1 Mr. Miller built>1":
    synt:    S-NP
    class:   I-EN-HOUSE
    forms:   (((NUMBER F-SING) (PERSON F-THIRD-P)))
    lex:     "house"
    props:   ((INDEX 1) (ORIG-SURF "the new house") (INDEXED TRUE))
    subs:
    (DET)  "the":
        synt:    S-DEF-ART
        class:   I-EART-DEF-ART
        forms:   (NIL)
        lex:     "the"
    (MOD)  "new":
        synt:    S-ADJP
        class:   I-EADJ-NEW
        forms:   (NIL)
        lex:     "new"
        subs:
        (PRED)  "new":
            synt:    S-ADJ
            class:   I-EADJ-NEW
            forms:   (NIL)
            lex:     "new"
            props:   ((ADJ-TYPE S-NON-DEMONSTR-ADJ))
    (PRED)  "house":
        synt:    S-COUNT-NOUN
        class:   I-EN-HOUSE
        forms:   (((NUMBER F-SING) (PERSON F-THIRD-P)))
        lex:     "house"
```

```
(MOD)  "<REF>1 Mr. Miller built":
    synt:    S-REL-CLAUSE
    class:   I-EV-BUILD
    forms:   (((PERSON F-THIRD-P) (NUMBER F-SING) (CASE F-NOM)
               (TENSE F-PAST-TENSE)))
    lex:     "build"
    subs:
    (OBJ THEME)  "<REF>1":
        synt:    S-NP
        class:   I-EN-HOUSE
        forms:   (((NUMBER F-SING) (PERSON F-THIRD-P)))
        lex:     "house"
        props:   ((GAP ACTIVE-FILLER) (INDEXED TRUE) (REF 1)
                   (ORIG-SURF "the new house"))
        subs:
        (DET)  "the":
            synt:    S-DEF-ART
            class:   I-EART-DEF-ART
            forms:   (NIL)
            lex:     "the"
        (MOD)  "new":
            synt:    S-ADJP
            class:   I-EADJ-NEW
            forms:   (NIL)
            lex:     "new"
            subs:
            (PRED)  "new":
                synt:    S-ADJ
                class:   I-EADJ-NEW
                forms:   (NIL)
                lex:     "new"
                props:   ((ADJ-TYPE S-NON-DEMONSTR-ADJ))
        (PRED)  "house":
            synt:    S-COUNT-NOUN
            class:   I-EN-HOUSE
            forms:   (((NUMBER F-SING) (PERSON F-THIRD-P)))
            lex:     "house"
    (SUBJ AGENT)  "Mr. Miller":
        synt:    S-NP
        class:   I-EN-MILLER
        forms:   (((NUMBER F-SING) (PERSON F-THIRD-P)))
        lex:     "Miller"
        subs:
        (PRED)  "Mr. Miller":
            synt:    S-PROPER-NAME
            class:   I-EN-MILLER
            forms:   (((NUMBER F-SING) (PERSON F-THIRD-P)))
            lex:     "Miller"
            subs:
            (MOD)  "Mr.":
                synt:    S-COUNT-NOUN
                class:   I-EN-MISTER-TITLE
                forms:   (((NUMBER F-SING) (PERSON F-THIRD-P)))
```

```
                    lex:     "Mr"
                    subs:
                    (PRED)  "Mr":
                        synt:    S-COUNT-NOUN
                        class:   I-EN-MISTER-TITLE
                        forms:   (((NUMBER F-SING)
                                   (PERSON F-THIRD-P)))
                        lex:     "Mr"
                        props:   ((CAPITALIZATION TRUE)
                                  (IS-ABBREVIATION TRUE))
                    (DUMMY)  ".":
                        synt:    D-PERIOD
                        lex:     "."
                (PRED)  "Miller":
                    synt:    S-PROPER-NAME
                    class:   I-EN-MILLER
                    forms:   (((NUMBER F-SING) (PERSON F-THIRD-P)))
                    lex:     "Miller"
                    props:   ((CAPITALIZATION TRUE))
        (PRED)  "built":
            synt:    S-VERB
            class:   I-EV-BUILD
            forms:   (((TENSE F-PAST-PART)) ((TENSE F-PAST-TENSE)))
            lex:     "build"
(DUMMY)  ".":
    synt:    D-PERIOD
    lex:     "."
```

Appendix D

Translation Evaluation Questionnaire and Key

D.1 Translation Evaluation Questionnaire

The questionnaires handed out for translation evaluation were hardcopies of the two Web pages `http://www.cs.utexas.edu/users/ulf/diss/eval_tegm.html`, the version that always includes human translations, and `http://www.cs.utexas.edu/users/ulf/diss/eval_tego.html`, the version that does not include human translations, unless it happens to match any of the machine translations.

The questionnaire starting on the next page is the dissertation format version of `http://www.cs.utexas.edu/users/ulf/diss/eval_tegm.html` .

Translation Evaluation of Wall Street Journal sentences 48-79

October 1996

http://www.cs.utexas.edu/users/ulf/diss/eval_tegm.html

Hello! This evaluation will help the research of a dissertation in the area of natural language processing. The following includes the results of computer programs that translated sentences from the Wall Street Journal from English to German.
Please evaluate the following translations by **assigning grades** for both grammatical correctness and meaning preservation using the following tables as a guideline. Note that the scale is like the one used in the German education system (1 = sehr gut; 2 = gut; 3 = befriedigend; 4 = ausreichend; 5 = mangelhaft; 6 = ungenügend).

Grammar (syntax and morphology)

Grade	*Usage*
1	Correct grammar, including word order, word endings; the sentence reads fluently.
2	Basically correct grammar, but not very fluent.
3	Mostly correct grammar, but with significant shortcomings.
4	The grammar is acceptable only in parts of the sentence.
5	The grammar is generally so bad that the entire sentence becomes very hard to read.
6	The grammar is so bad that the sentence becomes totally incomprehensible.

Meaning (semantics)

Grade	*Usage*
1	The meaning is fully preserved and can easily be understood.
2	The meaning is mostly preserved and can be understood fairly well.
3	The general idea of the sentence is preserved.
4	Contains some useful information from the original sentence.
5	A reader of the translated sentence can guess what the sentence is about, but the sentence provides hardly any useful information.
6	The sentence is totally incomprehensible or totally misleading.

(EXAMPLE)

Yesterday, I ate a red apple.

(a) Gästern, ich haben essen Apfl-rot. *Grammar:_5_ Meaning:_2_*

(b) Meine roten Äpfel haben viel gegessen. *Grammar:_1_ Meaning:_6_*

The evaluation of the translations will take you **about 45-60 minutes**, or about 1-3 minutes for each of the 32 English sentences and their German translations. If you have any questions, please don't hesitate to contact Ulf at

- +1 (512) 320-0650 (home; voice & fax)
- +1 (512) 471-9777 (office)
- ulf@cs.utexas.edu

Please return the evaluations to:

- *Campus mail:* Ulf Hermjakob - Dept. of Computer Sciences - Mail code C0500
- *Postal address:* Ulf Hermjakob - 600 W26th St #A308 - Austin, TX 78705
- *Office location:* Taylor Hall 150B (with 24 hour accessible mailbox)
- or call or email Ulf for pick-up

(WSJ 48)

Largely because of the falling dollar, West German labor costs rose to 120% of those for U.S. production workers from 75% in 1985.

(a) Im großen und ganzen wegen des fallenden Dollars stiegen Westliche deutsche Arbeitskosten 1985 zu 120% von denen für die Produktionsarbeiter der USA von 75%. *Grammar:*____ *Meaning:*____

(b) Groß wegen des fallenden Dollars, stiegen westdeutsche Lohnkosten zu 120% von denen für US-Produktionsarbeiter von 75% 1985. *Grammar:*____ *Meaning:*____

(c) Zum größten Teil wegen des fallenden Dollars stiegen westdeutsche Arbeitskosten auf 120% von denen für US Produktionsarbeiter von 75% 1985. *Grammar:*____ *Meaning:*____

(d) Zum größten Teil wegen des fallend Dollar, Westdeutscher wirtschaftliche Preise Rosen to120% von jenen für AMERIKANISCHE Produktion-Arbeiter von 75% in 1985. *Grammar:*____ *Meaning:*____

(e) Zum größten Teil wegen des fallenden Dollars stiegen westdeutsche Lohnkosten auf 120% von denen für US-Industriearbeiter von 75% im Jahre 1985. *Grammar:*____ *Meaning:*____

(f) Zum größten Teil wegen des fallenden Dollars stiegen westdeutsche Arbeitskosten für US Produktionsarbeiter auf 120% von denen von 75% 1985. *Grammar:*____ *Meaning:*____

(WSJ 49)

In high school, he was a member of the speech team.

(a) Im Gymnasium war er ein Mitglied der Redemannschaft. *Grammar:*____ *Meaning:*____

(b) In Gymnasium war er ein Mitglied der Rede-Mannschaft. *Grammar:*____ *Meaning:*____

(c) Auf dem Gymnasium war er ein Mitglied der Redemannschaft. *Grammar:*____ *Meaning:*____

(d) In der hohen Schule war er ein Mitglied der Redemannschaft. *Grammar:*____ *Meaning:*____

(e) In High School war er ein Mitglied des Redeteams. *Grammar:*____ *Meaning:*____

(WSJ 50)

But the immediate concern is the short-term credits.

(a) Aber das unmittelbare Anliegen ist die kurzfristigen Kredite. *Grammar:*____ *Meaning:*____

(b) Aber das unmittelbare Anliegen sind die kurzfristigen Kredite. *Grammar:*____ *Meaning:*____

(c) Aber die unmittelbare Angelegenheit ist die kurzfristigen Kredite. *Grammar:*____ *Meaning:*____

(d) Aber das sofortige Interesse ist die kurzfristigen Gutschriften. *Grammar:*____ *Meaning:*____

(e) Aber die unmittelbare Sorge ist die kurzfristigen Kredite. *Grammar:*____ *Meaning:*____

(WSJ 51)

Canadian manufacturers' new orders fell to $20.80 billion (Canadian) in January, down 4% from December's $21.67 billion on a seasonally adjusted basis, Statistics Canada, a federal agency, said.

(a) Neue Aufträge der kanadischen Hersteller fielen auf $20,80 Milliarde (Kanadier) im Januar, unten 4% von Dezembers $21,67 Milliarde auf einer jahreszeitlichen Grundlage, Statistiken Kanada, ein Bundesamt, gesagt. *Grammar:*____ *Meaning:*____

(b) Kanadischer Hersteller neue Aufträge seien im Januar auf 20.8 Mrd. (kanadische) Dollar gefallen, ein Minus von 4% von Dezembers 21.67 Mrd. Dollar auf einer saisonal bereinige Basis, sagte Statistik-Kanada eine Bundesbehörde. *Grammar:*____ *Meaning:*____

(c) Neuen Aufträge kanadischer Hersteller fällten zu 20,80 Milliarden Dollar (Kanadier) in Januar, herunter 4% vom $. Dezember 21,67 Milliarde auf einer von seasonally eingestellten Grundlage, statistische Angaben Kanada, eine Bundesagentur sagte. *Grammar:*____ *Meaning:*____

(d) Die neuen Disziplinen kanadischer Hersteller fielen zu $20.80 milliard (kanadisch) in Januar, besiegen Sie 4% von Dezembers $21.67 milliard auf einer saisongemäß eingestellt Basis, Statistiken, die Kanada, eine Bundes Agentur, sagte. *Grammar:*____ *Meaning:*____

(e) Neue Aufträge kanadischer Hersteller seien im Januar auf 20.8 Mrd. (kanadische) Dollar gefallen, ein Minus von 4% im Vergleich zu 21.67 Mrd. Dollar im Dezember auf einer saisonal bereinigten Basis, sagte Statistik-Kanada, eine Bundesbehörde. *Grammar:*____ *Meaning:*____

(f) Kanadischer Hersteller neue Aufträge seien im Januar auf 20.8 Mrd. Dollar (kanadisch) gefallen, ein Minus von 4% aus Dezember 21.67 Mrd. Dollar auf einer saisonal bereinige Basis, sagte Statistik-Kanada eine Bundesbehörde. *Grammar:*____ *Meaning:*____

(WSJ 52)

The Federal Farm Credit Banks Funding Corp. plans to offer $1.7 billion of bonds Thursday.

(a) Federal Farm Credit Banks Funding Corp. plant, $1,7 Milliarde von Bindungen Donnerstag anzubieten. *Grammar:*____ *Meaning:*____

(b) Die Bundes Bauernhof-Kredit-Banken, die finanzieren, AG plant, $1.7 milliard von Banden Donnerstag anzubieten. *Grammar:*____ *Meaning:*____

(c) Die Federal Farm Credit Banks Funding Corp. plant, Donnerstag 1.7 Mrd. Dollar in Obligationen anzubieten. *Grammar:*____ *Meaning:*____

(d) Die Bundesstaatlichen Bauernhofkreditbankfinanzierungs Corp. Pläne zu Angebot 1,7 Milliarden Dollar Übereinkommen-Donnerstags. *Grammar:*____ *Meaning:*____

(e) Die Bundesbauernhofkreditbankenfinanzgesellschaftpläne, Donnerstag 1.7 Mrd. Dollar in Obligationen anzubieten. *Grammar:*____ *Meaning:*____

(f) Die Bundesbauernhofkreditbankenfinanzgesellschaft plant, Donnerstag 1.7 Mrd. Dollar in Obligationen anzubieten. *Grammar:*____ *Meaning:*____

(WSJ 53)

The St. Louis-based bank holding company previously traded on the American Stock Exchange.

(a) Die Str. Louis-gestützte Bankholdinggesellschaft nützte vorher die amerikanische Börse aus. *Grammar:*____ *Meaning:*____

(b) Die der Sankt Louis basierte Bankholdinggesellschaft handelte früher an die amerikanische Börse. *Grammar:*____ *Meaning:*____

(c) Die in St. Louis basierte Bankholdinggesellschaft wurde zuvor an der amerikanischen Börse gehandelt. *Grammar:*____ *Meaning:*____

(d) Die St. Louis basierte Bankholdinggesellschaft handelte früher an die amerikanische Börse. *Grammar:*____ *Meaning:*____

(e) Die Str. Louis-louis-based Bank, die Firma tauschte hält vorher, auf der amerikanischen Börse. *Grammar:*____ *Meaning:*____

(f) Die Str. Louis-based, die Bank, die Gesellschaft vorher hält, auf der amerikanischen Börse tauschte. *Grammar:*____ *Meaning:*____

(WSJ 54)

The transaction is expected to be completed by May 1.

(a) Die Verhandlung wird erwartet, durch den 1. Mai vervollständigt zu werden. *Grammar:*____ *Meaning:*____

(b) Die Verhandlung wird erwartet, bis Mai 1 durchgeführt zu werden. *Grammar:*____ *Meaning:*____

(c) Es wird erwartet, daß die Transaktion bis zum erstem Mai abgeschlossen wird. *Grammar:*____ *Meaning:*____

(d) Es wird bis zum erstem Mai erwartet, daß die Transaktion abgeschlossen ist. *Grammar:*____ *Meaning:*____

(e) Die Transaktion soll im 1. Mai beendet werden. *Grammar:*____ *Meaning:*____

(f) Es wird erwartet, daß die Transaktion bis zum 1. Mai abgeschlossen wird. *Grammar:*____ *Meaning:*____

(WSJ 55)

A successor for him hasn't been named.

(a) Ein Nachfolger für ihn ist nicht genannt worden. *Grammar:*____ *Meaning:*____

(WSJ 56)

The president's news conference was a much-needed step in that direction.

(a) Die Nachrichtenkonferenz des Präsidenten war ein sehr notwendig Schritt in jener Richtung. *Grammar:*____ *Meaning:*____

(b) Die Nachricht-Konferenz des Präsidenten war ein viel-gebraucht Schritt in jenem direction. *Grammar:*____ *Meaning:*____

(c) Des Präsidenten Nachrichtenkonferenz war ein dringend notwendiger Schritt in dieser Richtung. *Grammar:*____ *Meaning:*____

(d) Die Nachrichtenkonferenz des Präsidenten war ein dringend notwendiger Schritt in diese Richtung. *Grammar:*____ *Meaning:*____

(e) Die Nachrichtenkonferenz des Präsidenten war ein dringend benötigter Schritt in dieser Richtung. *Grammar:*____ *Meaning:*____

(WSJ 57)

The Tokyo exchange was closed Saturday as part of its regular holiday schedule.

(a) Die Tokyo-Börse wurde Sonnabend als Teil seines normalen Feiertagzeitplans geschlossen. *Grammar:*____ *Meaning:*____

(b) Der Tokyoaustausch war geschlossener Samstag als Teil seines regelmäßigen Feiertagzeitplanes. *Grammar:*____ *Meaning:*____

(c) Der Tokio Austausch wurde Samstag als Teil seines regelmäßigen Feiertagplanes geschlossen. *Grammar:*____ *Meaning:*____

(d) The, Tokyo Tausch wurde Samstag als Teil seines regulären holidayschedule geschlossen. *Grammar:*____ *Meaning:*____

(e) Die Tokyoter Börse wurde Sonnabend als Teil ihres normalen Feiertagszeitplans geschlossen. *Grammar:*____ *Meaning:*____

(WSJ 58) **Pan Am said its full-year results were hurt by foreign currency exchange losses of \$46.8 million, primarily related to Japanese yen debt, compared with \$11.1 million in 1985.**

(a) Pan Am sagte, seine Ganzjahresresultate seien durch 46.8 Mio. Dollar, verglichen mit 11.1 Mio. Dollar 1985, in ausländischen Währungswechselverlusten in erster Linie zur japanischer Yenschuld bezogen geschadet worden. *Grammar:*____ *Meaning:*____

(b) Pan Am, die seine Volljahr-Resultate besagt ist, wurden durch die Austauschverluste der ausländischen Währung von \$46,8 Million verletzt, hauptsächlich bezogen auf der japanischen Yenschuld, verglichen mit \$11,1 Million 1985. *Grammar:*____ *Meaning:*____

(c) Pfanne wird gesagt, daß seine voll-Jahr-Ergebnisse durch fremde currencyexchange-Verluste von \$46.8 million verletzt wurden, hauptsächlich erzählt zu Japanisch Yen-Schuld, die mit \$11.1 million in 1985 verglichen wurde. *Grammar:*____ *Meaning:*____

(d) Pan Am sagte, seine vollen Jahrresultate seien durch 46.8 Mio. Dollar, verglichen 1985 mit 11.1 Mio. Dollar, in ausländischen Währungswechselverlusten in erster Linie zur japanischer Yenschuld bezogen geschadet worden. *Grammar:*____ *Meaning:*____

(e) Pfanne wird gesagt, daß seine voll-Jahr-Ergebnisse von Devisen-Austausch-Verlusten von 46,8 Millionen Dollar weh getan wurde, vorwiegend mit japanischer Yenschuld zusammenhing, verglichen 1985 mit 11,1 Millionen Dollar. *Grammar:*____ *Meaning:*____

(f) Pan Am sagte, seine Ganzjahresresultate seien durch ausländische Währungswechselverluste in Höhe von 46.8 Mio. Dollar beeinträchtigt worden, insbesondere durch Schulden in japanischen Yen, verglichen mit 11.1 Mio. Dollar im Jahre 1985. *Grammar:*____ *Meaning:*____

(WSJ 59)

The American hospital sector spent $181 billion in 1986.

(a)	Der amerikanische Krankenhaussektor gab 1986 181 Mrd. Dollar aus.	*Grammar:*____ *Meaning:*____
(b)	Der amerikanische Krankenhaussektor gab 181 Milliarden Dollar 1986 aus.	*Grammar:*____ *Meaning:*____
(c)	Der amerikanische Krankenhaussektor wendete $181 Milliarde 1986 auf.	*Grammar:*____ *Meaning:*____
(d)	Der amerikanische Krankenhaus-Sektor gab $181 milliard in 1986 aus.	*Grammar:*____ *Meaning:*____

(WSJ 60)

But the company inquiry, which began a few months ago, changed everything.

(a)	Aber die Firmenanfrage, die vor einigem Monaten anfing, änderte alles.	*Grammar:*____ *Meaning:*____
(b)	Aber die Gesellschaftsanfrage, es einem wenigen Monat vor begann, änderte alles.	*Grammar:*____ *Meaning:*____
(c)	Aber die Firmenuntersuchung, die vor wenigen Monaten begann, änderte alles.	*Grammar:*____ *Meaning:*____
(d)	Aber die Gesellschaftsanfrage, die es einem wenigen Monat vor begann, änderte alles.	*Grammar:*____ *Meaning:*____
(e)	Aber die Firmaanfrage, die vor einigen Monaten anfing, änderte alles.	*Grammar:*____ *Meaning:*____
(f)	Aber die Gesellschaft-Anfrage, die vor einigen Monaten anfing, veränderte alles.	*Grammar:*____ *Meaning:*____

(WSJ 61)

Texas Instruments rose 3 1/4 to 175 1/2.

(a)	Texas Instruments stieg um 13/4 auf 351/2.	*Grammar:*____ *Meaning:*____
(b)	Texas Instrumente standen 3 1/4 bis 175 1/2 auf.	*Grammar:*____ *Meaning:*____
(c)	Texas Instruments stieg 3 1/4 bis 175 1/2.	*Grammar:*____ *Meaning:*____
(d)	Texas Instruments stiegen um 13/4 auf 351/2.	*Grammar:*____ *Meaning:*____
(e)	Texas Instruments stieg um 3 1/4 auf 175 1/2.	*Grammar:*____ *Meaning:*____
(f)	Texas Instrumente, die rosarot sind, 3 1/4 bis 175 1/2.	*Grammar:*____ *Meaning:*____

(WSJ 62)

If futures fell more than 0.20 point below the stocks, he would buy futures and sell stocks instead.

(a) Wenn Zukunft mehr als 0.20 Punkt unter den Aktien fällten, würdete er kaufen Sie Zukunft und verkaufen Sie Aktien stattdessen. *Grammar:*____ *Meaning:*____

(b) Wenn Terminpapiere mehr als 0.2 Punkte unter die Aktien fielen, würde er Terminpapiere kaufen, und Aktien stattdessen verkaufen. *Grammar:*____ *Meaning:*____

(c) Wenn Zukunft mehr als 0,20 Punkt unter die Vorräte fiel, würde er Zukunft kaufen und Vorräte anstatt verkaufen. *Grammar:*____ *Meaning:*____

(d) Wenn Termingeschäfte mehr als 0.2 Punkt unter die Aktien fielen, würde er Termingeschäfte kaufen, und Aktien stattdessen verkaufen. *Grammar:*____ *Meaning:*____

(e) Wenn Termingeschäfte mehr als 0.2 Punkt unter den Aktien fielen, würde er Termingeschäfte kaufen, und Aktien stattdessen verkaufen. *Grammar:*____ *Meaning:*____

(f) Wenn Zukunft mehr fällt als 0,20 unter den Anteilen richten, würde er Zukunft kaufen und Anteile stattdessen verkaufen. *Grammar:*____ *Meaning:*____

(WSJ 63)

"I'm willing to work for a foreign company," says the father of two.

(a) "Ich bin bereit, für eine fremde Firma zu arbeiten," sage den Vater von zwei. *Grammar:*____ *Meaning:*____

(b) Ich bin bereit zu arbeiten für eine ausländische Gesellschaft sagt der Vater von 2. *Grammar:*____ *Meaning:*____

(c) Ich sei bereit, für eine ausländische Gesellschaft zu arbeiten, sagt der Vater vom 2. *Grammar:*____ *Meaning:*____

(d) "Ich, der wollen arbeiten für eine fremde Gesellschaft," sagt den Vater von zwei. *Grammar:*____ *Meaning:*____

(e) " ich bin bereit, für eine Auslandsgesellschaft zu arbeiten, " sagt den Vater von zwei. *Grammar:*____ *Meaning:*____

(f) "Ich bin bereit, für eine ausländische Gesellschaft zu arbeiten", sagt der Vater von zweien. *Grammar:*____ *Meaning:*____

(WSJ 64)

It estimated that sales totaled $217 million, compared with the year-earlier $257 million.

(a) Es schätzte ab, daß Verkäufe zusammen 217 Millionen Dollar zählten, verglichen 257 Million mit den jahresfrüheren $. *Grammar:*____ *Meaning:*____

(b) Es schätzte, daß Verkäufe $217 million zusammenzählten, verglich mit das Jahr-früher $257 million. *Grammar:*____ *Meaning:*____

(c) Es schätzte, daß Verkäufe 217 Mio. Dollar, verglichen mit dem 257 Mio. Dollar im Jahr zuvor, betrugen. *Grammar:*____ *Meaning:*____

(d) Sie schätzte, daß Verkäufe $217 Million zusammenzählten, mit dem Jahrfrüh $257 Million verglichen. *Grammar:*____ *Meaning:*____

(e) Es schätzte, daß Verkäufe insgesamt 217 Mio. Dollar betrugen, verglichen mit 257 Mio. Dollar im Jahr zuvor. *Grammar:*____ *Meaning:*____

(WSJ 65)

With the falling dollar, U.S. manufacturers are in a pretty good position to compete on world markets.

(a) Mit dem fallenden Dollar sind US Hersteller in einer ziemlich guten Position, um auf Weltmärkte zu konkurrieren. *Grammar:*_____ *Meaning:*_____

(b) Mit dem fallenden Dollar sind die Hersteller der USA in der ziemlich guten Lage, auf Weltmärkten zu konkurrieren. *Grammar:*_____ *Meaning:*_____

(c) Mit dem fallend Dollar sind AMERIKANISCHE Hersteller in einer ganz guten Position, auf Welt-Märkten zu konkurrieren. *Grammar:*_____ *Meaning:*_____

(d) Mit dem fallenden Dollar sind– US-Hersteller in einer hübschen guten Position zum Konkurrieren in den Weltmärkten. *Grammar:*_____ *Meaning:*_____

(e) Mit dem fallenden Dollar sind US-Hersteller in einer ziemlich guten Position, auf Weltmärkten zu konkurrieren. *Grammar:*_____ *Meaning:*_____

(f) Mit dem fallenden Dollar sind US Hersteller in einer ziemlich guten Position, auf Weltmärkte zu konkurrieren. *Grammar:*_____ *Meaning:*_____

(WSJ 66)

You go on to Bank B and get another loan, using some of Bank B's proceeds to pay back some of your debt to Bank A.

(a) Sie gehen zu Bank B über und bekommen ein anderes Darlehen, was einige von Bank B Erlösen benutzt, um etwas von Ihrer Schuld zu Bank A zurückzuzahlen. *Grammar:*_____ *Meaning:*_____

(b) Sie gehen weiter, B zu überhöhen und noch einen Kredit zu bekommen, das Benutzen von einigen von Bank B's geht weiter, einige Ihrer Schuld zurückzuzahlen, um A. zu überhöhen *Grammar:*_____ *Meaning:*_____

(c) **abend** Sie fortfahren on zu haben b und erhalten ein Darlehen, mit einig von Ertrag der Bank b zu zahlen zurück etwas von Ihr Schuld zu Bank a. *Grammar:*_____ *Meaning:*_____

(d) Man fährt auf Bank-b fort, und bekommt ein weiter Kredit einige von Bank-bs Erträge, einige von deiner Schuld zur zurückzuzahlen, benutzend. *Grammar:*_____ *Meaning:*_____

(e) Man fährt zur Bank b fort, und bekommt ein weiter Kredit, um einige von deiner Schuld zur Bank a zurückzuzahlen, einige von Bank-bs Erträge benutzend. *Grammar:*_____ *Meaning:*_____

(f) Man wendet sich dann an Bank B und bekommt dort einen weiteren Kredit, den man dann teilweise dazu benutzt, einen Teil der Schulden bei Bank A zurückzuzahlen. *Grammar:*_____ *Meaning:*_____

(WSJ 67)

The critical point is to know when Brazil will produce an economic policy to generate foreign exchange to pay its interest.

(a) Der kritische Punkt soll wissen, wann Brasilien eine Wirtschaftspolitik produziert, um Devisenkurs zu erzeugen, um sein Interesse zu zahlen. *Grammar:* ____ *Meaning:* ____

(b) Der kritische Punkt ist, um zu kennen, wann Brasilien eine Wirtschaftspolitik produzieren wird, um ausländischen Wechsel zu erzeugen, um seinen Zins zu zahlen. *Grammar:* ____ *Meaning:* ____

(c) Der kritische Punkt sollte wissen, wenn Brasilien einen economicpolicy produzieren wird, um fremden Tausch zu erzeugen, um sein Interesse zu zahlen. *Grammar:* ____ *Meaning:* ____

(d) Der kritische Punkt ist, zu wissen, wann Brazil, um, um seinen Zins zu zahlen, ausländischen Wechsel zu erzeugen, eine Wirtschaftspolitik produzieren wird. *Grammar:* ____ *Meaning:* ____

(e) Der kritische Punkt ist zu wissen, wann Brasilien eine Wirtschaftspolitik aufstellen wird, um Außenhandel zu erzeugen, um seine Zinsen zu zahlen. *Grammar:* ____ *Meaning:* ____

(f) Der kritische Punkt ist, zu wissen, wann Brasilien eine wirtschaftliche Politik produziert, um Devisen zu generieren, um seine Zinsen zu bezahlen. *Grammar:* ____ *Meaning:* ____

(WSJ 68)

Manufacturers' shipments followed the same trend, falling 1.5% in January to $21.08 billion, after a 2.8% increase the previous month.

(a) Die Sendungen von Herstellern folgten der gleichen Tendenz, fallenden 1,5% im Januar zu 21,08 Milliarden Dollar, nachdem ein 2,8% den vorhergehenden Monat vergrößert. *Grammar:* ____ *Meaning:* ____

(b) Herstellersendungen folgten dem gleichen Trend und fielen im Januar um 1.5% auf 21.08 Mrd. Dollar nach einer 2.8% Zunahme im vorherigen Monat. *Grammar:* ____ *Meaning:* ____

(c) Die Sendungen Hersteller folgten dem gleichen Trend, beim Fallen, 1.5% in Januar zu $21.08 milliard, nach einem 2.8% Zunahme der vorausgehende Monat. *Grammar:* ____ *Meaning:* ____

(d) Hersteller Sendungen folgten den gleichen Trend in Januar auf 21.08 Mrd. Dollar 1.5% fallend nach einer 2.8% Zunahme dem vorherig Monat. *Grammar:* ____ *Meaning:* ____

(e) Hersteller Sendungen im Januar 1.5% auf 21.08 Mrd. Dollar fallend folgten den gleichen Trend nach einer 2.8% Zunahme der vorherige Monat. *Grammar:* ____ *Meaning:* ____

(f) Versand der Hersteller folgte der gleichen Tendenz, fallenden 1,5% im Januar bis $21,08 Milliarde, nachdem eine Zunahme 2,8% der vorhergehende Monat. *Grammar:* ____ *Meaning:* ____

(WSJ 69)

The offerings will be made through the corporation and a nationwide group of securities dealers and dealer banks.

(a) Die Angebote werden durch die Gesellschaft und eine landesweit Gruppe von Wertpapierhändlern und Händler Banken gemacht werden. *Grammar:*____ *Meaning:*____

(b) Die Gaben werden durch die Firma gemacht werden, und anationwide gruppieren von Sicherheiten-Händlern und Händler-Banken. *Grammar:*____ *Meaning:*____

(c) Zu den Angeboten wird durch das Unternehmen und eine nationwide Gruppe von Wertpapierhändlern und Händlerbanken gemacht. *Grammar:*____ *Meaning:*____

(d) Die Angebote werden durch die Gesellschaft und eine landesweite Gruppe von Wertpapierhändlern und Händlerbanken gemacht werden. *Grammar:*____ *Meaning:*____

(e) Die Opfer werden durch die Corporation und eine allgemein Gruppe Wertpapierhändler und Händlerbanken gebildet. *Grammar:*____ *Meaning:*____

(WSJ 70)

The real estate services company formerly traded over the counter.

(a) Die wirkliche Gut-Dienste-Gesellschaft tauschte über thecounter ehemals. *Grammar:*____ *Meaning:*____

(b) Die Immobilienservicegesellschaft handelte früher im Freiverkehr. *Grammar:*____ *Meaning:*____

(c) Die Immobilienservicegesellschaft wurde früher im Freiverkehr gehandelt. *Grammar:*____ *Meaning:*____

(d) Die Immobilien warten Firma, die früher über dem Zähler getauscht war. *Grammar:*____ *Meaning:*____

(e) Die Immobilienservice-Firma tauschte früher über dem Kostenzähler. *Grammar:*____ *Meaning:*____

(WSJ 71)

The shares are convertible at a rate of $9.50 of preferred for each common share.

(a) Die Anteile sind mit einer Rate von $9,50 von bevorzugt für jeden allgemeinen Anteil umwandelbar. *Grammar:*____ *Meaning:*____

(b) Die Anteile sind umwandelbar bei einer Rate von $9.50 von, zog vor für jeden gewöhnlichen Anteil. *Grammar:*____ *Meaning:*____

(c) Die Aktien sind zu einem Kurs von 9.5 Dollar in bevorzugter pro Stammaktie konvertierbar. *Grammar:*____ *Meaning:*____

(d) Die Aktien sind zu einem Kurs von 9.50 Dollar in Vorzügen pro Stammaktie konvertierbar. *Grammar:*____ *Meaning:*____

(e) Die Anteile sind bei einer Geschwindigkeit von 9,50 Dollar für jeden weitverbreiteten Anteil konvertibel von vorgezogen. *Grammar:*____ *Meaning:*____

(f) Die Aktien sind konvertierbar zu einem Kurs von 9.5 Dollar von bevorzugt pro Stammaktie. *Grammar:*____ *Meaning:*____

(WSJ 72)

He continues as president of the company's corporate division.

(a) Er setzt als Präsident von der korporativen Teilung der Gesellschaft fort. *Grammar:*____
*Meaning:*____

(b) Er macht als Präsident der Gesellschaft Firmenabteilung weiter. *Grammar:*____
*Meaning:*____

(c) Er macht als Präsident der körperschaftlichen Division der Firma weiter. *Grammar:*____
*Meaning:*____

(d) Er macht als Präsident der Firmenabteilung des Unternehmens weiter. *Grammar:*____
*Meaning:*____

(e) Er fährt als Präsident der korporativen Abteilung der Firma fort. *Grammar:*____
*Meaning:*____

(WSJ 73)

In tests it has worked for many heart-attack victims.

(a) In den Tests, die es für viele funktioniert hat, Opfer Herz-angreifen. *Grammar:*____
*Meaning:*____

(b) In Prüfungen hat es für viele Herz-Angriffs-Opfer gearbeitet. *Grammar:*____
*Meaning:*____

(c) In Tests hat es für viele Herzinfarktopfer funktioniert. *Grammar:*____
*Meaning:*____

(d) In Prüfungen hat es für viele Herz-Angriff-Opfer gearbeitet. *Grammar:*____
*Meaning:*____

(e) In Tests hat es für viele Herzinfarktopfer gearbeitet. *Grammar:*____
*Meaning:*____

(WSJ 74)

The percentage change is since year-end.

(a) Die ProzentsatzÄnderung ist seit Jahresende. *Grammar:*____
*Meaning:*____

(b) Die Prozentsatzänderung ist seit Jahresende. *Grammar:*____
*Meaning:*____

(c) Die Prozent Änderung ist seit Jahr-Spitze. *Grammar:*____
*Meaning:*____

(d) Die Prozentsatzänderung ist seit year-end. *Grammar:*____
*Meaning:*____

(e) Die Prozentsatzänderung ist seit Jahres-Ende. *Grammar:*____
*Meaning:*____

(WSJ 75)

He said Pan Am currently has "in excess of $150 million" in cash.

(a)	Er sagte, daß Pan Am z.Z. " über $150 Million " im Bargeld hat.	*Grammar:*____ *Meaning:*____
(b)	Er sagte, Pan Am habe gegenwärtig über 150 Mio. Dollar in Bargeld.	*Grammar:*____ *Meaning:*____
(c)	Er sagte, daß Pfannenvormittags zur Zeit "mehr als 150 Millionen Dollar" in Bargeld hat.	*Grammar:*____ *Meaning:*____
(d)	Er sagte, daß Pfanne ist, hat gegenwärtig-" in Überschuß von $150 million" in in bar.	*Grammar:*____ *Meaning:*____

(WSJ 76)

These revenues could then have been used for many purposes, such as funding those people without medical insurance.

(a)	Diese Einkommen hätten dann für viele Zwecke benutzt werden können, wie die Finanzierung jener Menschen ohne medizinische Versicherung.	*Grammar:*____ *Meaning:*____
(b)	Diese Einkommen konnten für viele Zwecke, wie Finanzierung jener Leute ohne medizinische Versicherung dann benutzt worden sein.	*Grammar:*____ *Meaning:*____
(c)	Diese Einnahmen hätten dann für viele Zwecke benutzt werden können, wie z.B. der Bezuschussung von Leuten ohne Krankenversicherung.	*Grammar:*____ *Meaning:*____
(d)	Diese Einnahmen gekonnt werden dann für viele Zwecke benutzt werden wie Finanzen diese Leute ohne die Krankenversicherung.	*Grammar:*____ *Meaning:*____
(e)	Diese Einnahmen gekonnt werden dann für viele Zwecke wie finanzierend diese Leute ohne Krankenversicherung benutzt werden.	*Grammar:*____ *Meaning:*____
(f)	Diese Einnahmen könnten für viel purposes,such als das Finanzieren jener Leute ohne medizinische Versicherung dann benutzt worden sein.	*Grammar:*____ *Meaning:*____

(WSJ 77)

The case continues in U.S. Bankruptcy Court in St. Louis.

(a)	Der Fall macht im US Konkursgericht in St. Louis weiter.	*Grammar:*____ *Meaning:*____
(b)	Der Fall macht in USA weiter. Konkurs-Hof in Str. Louis.	*Grammar:*____ *Meaning:*____
(c)	Der Fall macht in das US Konkursgericht in St. Louis weiter.	*Grammar:*____ *Meaning:*____
(d)	Der Fall wird am Konkursgericht in St. Louis fortgesetzt.	*Grammar:*____ *Meaning:*____
(e)	Der Fall setzt in AMERIKANISCHEM Bankrott-Gericht in Str. Louis fort.	*Grammar:*____ *Meaning:*____
(f)	Der Fall fährt in US fort. Konkursgericht in St. Louis.	*Grammar:*____ *Meaning:*____

(WSJ 78)

American Express fell 1 1/2 to 77 1/4 on more than 2.1 million shares.

(a)	Amerikanischer Expreß fällt 1 1/2 bis 77 1/4 auf mehr als 2,1 Millionen Anteilen.	*Grammar:*____ *Meaning:*____
(b)	Amerikanischer Bestimmter kahle Berg 1 1/2 bis 77 1/4 auf mehr als 2.1 millionshares.	*Grammar:*____ *Meaning:*____
(c)	American Express fiel 3/2 auf 309/4 auf mehr als 2.1 Mio. Aktien.	*Grammar:*____ *Meaning:*____
(d)	Der ausdrückliche Amerikaner fiel 1 1/2 bis 77 1/4 auf mehr als 2,1 Million Anteilen.	*Grammar:*____ *Meaning:*____
(e)	American Express fiel 3/2 auf mehr als 2.1 Mio. Aktien auf 309/4.	*Grammar:*____ *Meaning:*____
(f)	American Express fiel um 1 1/2 auf 77 1/4 bei mehr als 2.1 Mio. Aktien.	*Grammar:*____ *Meaning:*____

(WSJ 79)

His typical trade involved $30 million.

(a)	Sein typischer Handel umfaßte 30 Mio. Dollar.	*Grammar:*____ *Meaning:*____
(b)	Sein typischer Handel bezog $30 Million mit ein.	*Grammar:*____ *Meaning:*____
(c)	Sein typischer Beruf brachte $30 million mit sich.	*Grammar:*____ *Meaning:*____
(d)	Sein typischer Handel betraf 30 Millionen Dollar.	*Grammar:*____ *Meaning:*____

Thank you!

D.2 Translation Evaluation Key

The two numbers in parentheses associated with each translation are the averages of the syntactic and semantic grades that the evaluators have assigned to that specific translation.

(WSJ 48)

(a): Logos (2.1, 3.3)
(b): SYSTRAN (2.8, 2.9)
(c): CONTEX on correct parse (2.0, 2.1)
(d): Globalink (4.6, 4.4)
(e): human translation (1.5, 1.2)
(f): CONTEX (full translation) (1.7, 4.2)

(WSJ 49)

(a): CONTEX (full translation), CONTEX on correct parse (1.0, 1.5)
(b): Globalink (2.1, 1.4)
(c): human translation (1.0, 1.0)
(d): SYSTRAN (1.1, 2.8)
(e): Logos (2.1, 1.9)

(WSJ 50)

(a): CONTEX (full translation), CONTEX on correct parse (2.3, 1.9)
(b): human translation (1.0, 1.5)
(c): Logos (2.4, 2.9)
(d): SYSTRAN (2.3, 4.5)
(e): Globalink (2.3, 2.2)

(WSJ 51)

(a): SYSTRAN (3.7, 3.1)
(b): CONTEX on correct parse (3.4, 2.3)
(c): Logos (4.1, 3.5)
(d): Globalink (4.5, 4.9)
(e): human translation (1.5, 1.2)
(f): CONTEX (full translation) (3.4, 2.3)

(WSJ 52)

(a): SYSTRAN (2.3, 3.4)
(b): Globalink (4.0, 5.3)
(c): human translation (1.7, 1.5)
(d): Logos (5.3, 5.5)
(e): CONTEX (full translation) (4.3, 3.5)
(f): CONTEX on correct parse (1.7, 2.3)

(WSJ 53)

(a): Logos (1.8, 5.3)
(b): CONTEX (full translation) (3.1, 3.3)
(c): human translation (1.2, 4.5)
(d): CONTEX on correct parse (2.5, 2.3)
(e): SYSTRAN (5.0, 5.2)
(f): Globalink (4.8, 5.0)

(WSJ 54)

(a): Globalink (2.3, 4.4)
(b): SYSTRAN (2.5, 3.1)
(c): CONTEX on correct parse (1.5, 1.2)
(d): CONTEX (full translation) (1.6, 2.7)
(e): Logos (2.2, 2.2)
(f): human translation (1.0, 1.0)

(WSJ 55)

(a): CONTEX (full translation), CONTEX on correct parse, Logos, SYSTRAN, Globalink, human translation (1.4, 1.5)

(WSJ 56)

(a): Logos (2.2, 1.5)
(b): Globalink (3.8, 2.9)
(c): CONTEX (full translation), CONTEX on correct parse (2.0, 1.8)
(d): human translation (1.0, 1.2)
(e): SYSTRAN (1.3, 1.7)

(WSJ 57)

(a): CONTEX (full translation), CONTEX on correct parse (1.8, 1.7)
(b): SYSTRAN (2.4, 4.1)
(c): Logos (1.4, 2.9)
(d): Globalink (3.7, 3.6)
(e): human translation (1.2, 1.0)

(WSJ 58)

(a): CONTEX on correct parse (2.9, 2.9)
(b): SYSTRAN (3.6, 3.7)
(c): Globalink (4.0, 5.3)
(d): CONTEX (full translation) (2.9, 3.0)
(e): Logos (3.8, 4.9)
(f): human translation (1.6, 1.7)

(WSJ 59)

- (a): Contex (full translation), Contex on correct parse, human translation (1.1, 1.3)
- (b): Logos (1.9, 1.4)
- (c): Systran (2.0, 2.1)
- (d): Globalink (3.0, 1.8)

(WSJ 60)

- (a): Logos (2.1, 1.9)
- (b): Contex on correct parse (4.1, 3.5)
- (c): human translation (1.0, 1.2)
- (d): Contex (full translation) (3.7, 3.4)
- (e): Systran (1.5, 2.0)
- (f): Globalink (1.6, 2.5)

(WSJ 61)

- (a): Contex on correct parse (1.1, 3.1)
- (b): Globalink (2.8, 4.8)
- (c): Systran (2.3, 2.4)
- (d): Contex (full translation) (1.5, 3.1)
- (e): human translation (1.0, 1.0)
- (f): Logos (3.9, 5.8)

(WSJ 62)

- (a): Globalink (4.5, 4.7)
- (b): human translation (1.5, 1.5)
- (c): Systran (2.9, 4.7)
- (d): Contex on correct parse (2.0, 1.7)
- (e): Contex (full translation) (2.4, 1.7)
- (f): Logos (3.6, 4.8)

(WSJ 63)

- (a): Logos (2.3, 2.5)
- (b): Contex (full translation) (2.6, 2.0)
- (c): Contex on correct parse (2.5, 3.2)
- (d): Globalink (4.0, 3.9)
- (e): Systran (2.6, 2.2)
- (f): human translation (1.0, 1.2)

(WSJ 64)

- (a): Logos (3.3, 3.5)
- (b): Globalink (3.5, 2.7)
- (c): Contex (full translation), Contex on correct parse (2.6, 2.1)

(d): Systran (3.6, 3.3)
(e): human translation (1.0, 1.0)

(WSJ 65)
(a): Contex (full translation) (2.0, 1.2)
(b): Logos (1.5, 2.1)
(c): Globalink (2.8, 1.8)
(d): Systran (2.9, 2.4)
(e): human translation (1.2, 1.0)
(f): Contex on correct parse (1.8, 1.7)

(WSJ 66)
(a): Logos (2.8, 2.5)
(b): Globalink (3.7, 5.3)
(c): Systran (5.3, 4.8)
(d): Contex (full translation) (4.9, 5.0)
(e): Contex on correct parse (3.8, 3.2)
(f): human translation (1.0, 1.2)

(WSJ 67)
(a): Systran (2.5, 4.3)
(b): Contex (full translation) (2.5, 3.0)
(c): Globalink (3.2, 4.3)
(d): Contex on correct parse (3.3, 3.6)
(e): human translation (1.2, 1.5)
(f): Logos (2.1, 2.8)

(WSJ 68)
(a): Logos (3.4, 3.0)
(b): human translation (1.3, 1.0)
(c): Globalink (3.8, 3.0)
(d): Contex (full translation) (3.6, 2.5)
(e): Contex on correct parse (3.6, 3.0)
(f): Systran (3.6, 3.0)

(WSJ 69)
(a): Contex (full translation), Contex on correct parse (2.3, 1.8)
(b): Globalink (3.8, 4.5)
(c): Logos (3.6, 4.5)
(d): human translation (1.0, 1.0)
(e): Systran (3.1, 5.2)

(WSJ 70)

(a): Globalink (3.4, 4.5)
(b): Contex (full translation), Contex on correct parse (1.3, 2.1)
(c): human translation (1.0, 3.7)
(d): Logos (3.9, 5.6)
(e): Systran (1.8, 4.9)

(WSJ 71)

(a): Systran (2.8, 3.2)
(b): Globalink (3.7, 4.3)
(c): Contex on correct parse (2.3, 2.3)
(d): human translation (1.0, 1.0)
(e): Logos (2.8, 4.6)
(f): Contex (full translation) (2.5, 2.7)

(WSJ 72)

(a): Globalink (1.9, 4.1)
(b): Contex (full translation), Contex on correct parse (1.9, 2.2)
(c): Logos (1.4, 2.7)
(d): human translation (1.0, 1.7)
(e): Systran (1.3, 3.2)

(WSJ 73)

(a): Systran (4.3, 5.0)
(b): Logos (2.1, 3.1)
(c): Contex on correct parse, human translation (1.1, 1.1)
(d): Globalink (1.8, 2.9)
(e): Contex (full translation) (1.1, 2.1)

(WSJ 74)

(a): Contex (full translation), Contex on correct parse (2.1, 1.2)
(b): human translation (1.0, 1.0)
(c): Globalink (2.8, 4.2)
(d): Systran (2.4, 2.9)
(e): Logos (1.7, 1.2)

(WSJ 75)

(a): Systran (2.0, 1.6)
(b): Contex (full translation), Contex on correct parse, human translation (1.3, 1.2)
(c): Logos (1.8, 5.2)
(d): Globalink (4.4, 5.2)

(WSJ 76)

- (a): Logos (1.3, 1.5)
- (b): SYSTRAN (2.8, 2.7)
- (c): human translation (1.2, 1.6)
- (d): CONTEX (full translation) (4.3, 3.8)
- (e): CONTEX on correct parse (4.5, 3.7)
- (f): Globalink (3.9, 3.6)

(WSJ 77)

- (a): CONTEX on correct parse (1.5, 2.1)
- (b): Logos (3.8, 4.1)
- (c): CONTEX (full translation) (2.8, 2.3)
- (d): human translation (1.0, 1.0)
- (e): Globalink (3.3, 3.2)
- (f): SYSTRAN (3.6, 3.5)

(WSJ 78)

- (a): Logos (2.7, 2.8)
- (b): Globalink (5.0, 5.7)
- (c): CONTEX (full translation) (2.4, 2.5)
- (d): SYSTRAN (2.7, 5.1)
- (e): CONTEX on correct parse (2.6, 2.9)
- (f): human translation (1.2, 1.2)

(WSJ 79)

- (a): CONTEX (full translation), CONTEX on correct parse, human translation (1.1, 1.5)
- (b): SYSTRAN (1.3, 2.7)
- (c): Globalink (1.4, 4.7)
- (d): Logos (1.3, 2.5)

Appendix E

Abbreviations

A	add (operation)
ACL	Association for Computational Linguistics
ADJ	adjective
ADJP	adjective phrase
ADV	adverb
ADVP	adverbial phrase
AI	artificial intelligence
AMTA	Association for Machine Translation in the Americas
APP	adverbial or prepositional phrase
ARPA	Advanced Research Projects Agency (a United States Department of Defense agency for advanced technology research)
ART	article
AUX	auxiliary (verb)
BEN	beneficiary
CAT	category
COMPL	complement
CONC	concatenate, concatenated element
CONJ	conjunction
D-...	delimiter
DEF	definite
...-E...	English
ELEM	element
F-...	form
FEM	feminine
...-G...	German
I-...	internal concept
INDEF	indefinite
INF	infinitive/infinitival
INFL	inflection
INTR	intransitive
IOBJ	indirect object

IRR	irregular
KB	knowledge base
KBMT	knowledge based machine translation
LEX	lexicon/lexical
M-...	mathematical (concept)
MASC	masculine
MT	machine translation
N	noun
nc	count noun
NEUT	neuter
NL	natural language
NLP	natural language processing
NP	noun phrase
OBJ	object
P	person
PART	participle
pat	patient (a semantic role)
pn	proper name
PP	prepositional phrase
PRED	predicate
PREP	preposition
PRES	present (tense)
R	reduce (operation)
R-...	role
QUANT	quantity
S	shift (operation)
S-...	syntactic
SEM	semantic
SEMROLE	semantic role
SING	singular
SUBJ	subject
SYNT	syntactic
SYNTROLE	syntactic role
TR	transitive
UNAVAIL	unavailable
URL	universal resource locator (World Wide Web address)
V	verb
VP	verb phrase
WSJ	Wall Street Journal

Bibliography

Allen, J. F. (1995). *Natural Language Understanding (2nd Ed.)*. Benjamin/Cummings, Menlo Park, CA.

Aone, C., & Bennett, S. W. (1995). Evaluating automated and manual acquisition of anaphora resolution strategies. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pp. 122–129 Cambridge, MA.

Bates, M. (1978). The theory and practise of augmented transition networks. In Bloc, L. (Ed.), *Natural Language Communication with Computers*. Springer Verlag, New York.

Berger, A., & al. (1994). The Candide system for machine translation. In *Proceedings of ARPA Workshop on Human Language Technologies*.

Black, E., Jelineck, F., Lafferty, J., Magerman, D., Mercer, R., & Roukos, S. (1993). Towards history-based grammars: Using richer models for probabilistic parsing. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, pp. 31–37 Columbus, Ohio.

Black, E., Lafferty, J., & Roukos, S. (1992). Development and evaluation of a broad-coverage probabilistic grammar of English-language computer manuals. In *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics*, pp. 185–192 Newark, Delaware.

Brill, E. (1995). Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics*, *21*(4), 543–565.

Brown, P., & et al. (1990). A statistical approach to machine translation. *Computational Linguistics*, *16*(2), 79–85.

Cardie, C. (1992). Learning to disambiguate relative pronouns. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pp. 38–43 San Jose, CA.

Cardie, C. (1993). Using decision trees to improve case-based learning. In *Proceedings of the Tenth International Conference on Machine Learning*, pp. 25–32 San Mateo, California. Morgan Kaufmann.

Cardie, C. (1996). Automating feature set selection for case-based learning of linguistic knowledge. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 113–126.

Chomsky, N. (1988). *Lectures on Government and Binding, the Pisa lectures.* Floris Publications, Dordrecht, The Netherlands and Providence, Rhode Island.

Collins, M. J. (1996). A new statistical parser based on bigram lexical dependencies. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pp. 184–191 Santa Cruz, CA.

Daelemans, W., Zavrel, J., Berck, P., & Gillis, S. (1996). MBT: A memory-based part of speech tagger-generator. In *Proceedings of the Fourth Workshop on Very Large Corpora.*

Engel, U. (1988). *Deutsche Grammatik.* Julius Groos Verlag, Heidelberg.

Francis, W., & Kucera, H. (1982). *Frequency Analysis of English Usage: Lexicon and Grammar.* Houghton Mifflin, Boston.

Frederking, R., & al. (1994). The PANGLOSS Mark III machine translation system. In *Proceedings of the 1st AMTA Conference.*

Gale, W. A., & Church, K. W. (1991). A program for aligning sentences in bilingual corpora. In *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics*, pp. 177–184.

Gazdar, G., Klein, E., Pullum, G. K., & Sag, I. (1982). *Generalized Phrase Structure Grammar.* Harvard University Press, Cambridge, MA.

Goodman, J. (1996). Parsing algorithms and metrics. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pp. 177–183 Santa Cruz, CA.

Hermjakob, U., & Mooney, R. J. (1997). Learning parse and translation decisions from examples with rich context. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics* Madrid, Spain.

Hornby, A. S. (1974). *Oxford Advanced Learner's Dictionary of Current English.* Oxford University Press, Oxford, England.

Joshi, A. (1985). Tree-adjoining grammars: How much context sensitivity is required to provide reasonable structural descriptions. In Dowty, D. R., Karttunen, L., & Zwicky, A. (Eds.), *Natural Language Parsing.* Cambridge University Press, New York.

Kaplan, R. M., & Bresnan, J. (1982). Lexical-functional grammar. In Bresnan, J. (Ed.), *The Mental Representation of Grammatical Relations.* MIT Press, Cambridge, MA.

Kay, M. (1982). Parsing in functional unification grammar. In Dowty, D. R., Karttunen, L., & Zwicky, A. (Eds.), *Natural Language Parsing*, pp. 251–278. Cambridge University Press, New York.

Kay, M., & Roescheisen, M. (1993). Text-translation alignment. *Computational Linguistics*, *19*(1), 121–142.

Lasnik, H., & Uriagereka, J. (1988). *A Course in GB Syntax, Lectures on Binding and Empty Categories.* MIT Press, Cambridge, Massachusetts and London, England.

Lederer, H. (1969). *Reference Grammar of the German Language.* Charles Scribner's Sons, New York.

Litman, D. J. (1996). Cue phrase classification using machine learning. *Journal of Artificial Intelligence Research*, *5*, 53–95.

Magerman, D. M. (1995). Statistical decision-tree models for parsing. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pp. 276–283 Cambridge, MA.

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

Manning, C. D. (1993). Automatic acquisition of a large subcategorization dictionary from corpora. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, pp. 235–242 Columbus, Ohio.

Marcus, M. (1980). *A Theory of Syntactic Recognition for Natural Language.* MIT Press, Cambridge, MA.

Marcus, M., Santorini, B., & Marcinkiewicz, M. (1993). Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics*, *19*(2), 313–330.

Matsumoto, Y., Kurohashi, S., Utsuro, T., Myoki, Y., & Nagao, M. (1994). Japanese morphological analysis system JUMAN manual, version 2.0 (in Japanese). Technical report NAIST-IS-TR94025, Nara Institute of Science and Technology, Nara, Japan.

Mooney, R. J. (1996). Comparative experiments on disambiguating word senses: An illustration of the role of bias in machine learning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 82–91 Philadelphia, PA.

Ng, H. T., & Lee, H. B. (1996). Integrating multiple knowledge sources to disambiguate word sense: An exemplar-based approach. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pp. 40–47 Santa Cruz, CA.

Nirenburg, S., Carbonell, J., Tomita, M., & Goodman, K. (1992). *Machine Translation: A Knowledge-Based Approach.* Morgan Kaufmann, San Mateo, CA.

Nirenburg, S. (Ed.). (1987). *Machine Translation: Theoretical and Methodological Issues.* Cambridge University Press, Cambridge, England.

Pereira, F. C. N., & Warren, D. H. D. (1980). Definite clause grammars for language analysis - a survey of the formalisms and a comparison with augmented transition networks. *Artificial Intelligence*, *13, 3*, 231–278.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, *1*(1), 81–106.

Quinlan, J. R. (1987). Generating production rules from decision trees. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pp. 304–307 Milan, Italy.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning.* Morgan Kaufmann, San Mateo,CA.

Quirk, R., Greenbaum, S., Leech, G., & Svartvik, J. (1985). *A Comprehensive Grammar of the English Language.* Longman, London and New York.

Radford, A. (1988). *Transformational Grammar.* Cambridge University Press, Cambridge, England.

Riloff, E. (1996). An empirical study of automated dictionary construction for information extraction in three domains. *Artificial Intelligence*, *85*, 101–134.

Rivest, R. L. (1987). Learning decision lists. *Machine Learning*, *2*, 229–246.

Siegel, E. V., & McKeown, K. R. (1994). Emergent linguistic rules from inducing decision trees: Disambiguating discourse clue words. In *Proceedings of the Twelfth National Conference on Artificial Intelligence.*

Simmons, R. F., & Yu, Y. (1992). The acquisition and use of context dependent grammars for English. *Computational Linguistics*, *18*(4), 391–418.

Slocum, J. (Ed.). (1988). *Machine Translation Systems.* Cambridge University Press, Cambridge, England.

Smadja, F., McKeown, K. R., & Hatzivassiloglou, V. (1996). Translating collocations for bilingual lexicons: A statistical approach. *Computational Linguistics*, *22*(1), 1–38.

Somers, H. S. (1993). Current research in machine translation. *Machine Translation*, *7*(4), 231–246.

Tanenhaus, M. K., & al. (1996). Eye movements and spoken language comprehension. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pp. 48–54.

Tomita, M. (1986). *Efficient Parsing for Natural Language.* Kluwer Academic Publishers, Boston.

van Riemsdijk, H. C., & Williams, E. (1986). *Introduction to the Theory of Grammar.* MIT Press, Cambridge, Massachusetts and London, England.

Webster's (1994). *Webster's Encyclopedic Unabridged Dictionary of the English Language.* Gramercy Books, New York/Avenel.

Weischedel, R., & al. (1993). Coping with ambiguity and unknown words through probabilistic models. *Computational Linguistics*, *19*(2), 359–382.

Wermter, S., & Weber, V. (1997). Screen: Learning a flat syntactic and semantic spoken language analysis using artificial neural networks. *Journal of Artificial Intelligence Reserach*, *6*, 35–85.

Wu, D., & Xia, X. (1995). Large-scale automatic extraction of an English-Chinese translation lexicon. *Machine Translation*, *9*(3-4), 285–313.

Yarowsky, D. (1992). Word-sense disambiguation using statistical methods of Roget's categories trained on large corpora. In *Proceedings of the Fourteenth International Conference on Computational Linguistics*, pp. 454–460.

Yarowsky, D. (1994). Decision lists for lexical ambiguity resolution: Application to accent restoration in Spanish and French. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, pp. 88–95 Las Cruces, NM.

Yarowsky, D. (1995). Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pp. 189–196.

Zelle, J. M. (1995). *Using Inductive Logic Programming to Automate the Construction of Natural Language Parsers.* Ph.D. thesis, University of Texas, Austin, TX. Also appears as Artificial Intelligence Laboratory Technical Report AI 96-249.

Zelle, J. M., & Mooney, R. J. (1994). Combining top-down and bottom-up methods in inductive logic programming. In *Proceedings of the Eleventh International Conference on Machine Learning*, pp. 343–351 New Brunswick, NJ.

Vita

Ulf Hermjakob was born in Meppen, Germany, on November 26, 1962. After a few years in Basel, Switzerland, and Bad Oldesloe, Germany, his family moved to Bünde, Germany, where he graduated from the Freiherr-vom-Stein Gymnasium in 1981. After spending an academic year at the University of Georgia at Athens on a Georgia Rotary Student Program scholarship, he studied Computer Science with minors in business and (German) civil law at the University of Karlsruhe, Germany, where he received his Vordiplom in 1984 and his Hauptdiplom in 1988. After briefly working for Siemens in Paris, France, he entered the Computer Science Ph.D. program of the University of Texas at Austin on a one-year scholarship from the "Studienstiftung des deutschen Volkes" (German National Scholarship Foundation). After two years as Teaching Assistant for Computer Graphics, he joined the Knowledge Based Natural Language group at the Microelectronics and Computer Technology Corporation (MCC) in Austin, Texas, where he worked from 1991-1995.

Email: ulf@cs.utexas.edu
URL: http://www.cs.utexas.edu/users/ulf/
Permanent Address: Moltkestr. 40a
32257 Bünde
Germany

This dissertation was typeset with LaTeX 2_ε[1] by the author.

[1] LaTeX 2_ε is an extension of LaTeX. LaTeX is a collection of macros for TeX. TeX is a trademark of the American Mathematical Society. The macros used in formatting this dissertation were written by Dinesh Das, Department of Computer Sciences, The University of Texas at Austin.