Automatic Treebank Annotation for the Acquisition of LFG Resources

Michael Burke

A dissertation submitted in partial fulfilment of the
requirements for the award of

Doctor of Philosophy

to the

Dublin City University

School of Computing

Supervisors: Prof. Josef van Genabith
Dr. Andy Way

February 2006

Declaration

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Doctor of Philosophy is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work

Signed Michael Burke

(Michael Burke)

Student ID 52178781

Date February 2006

Contents

Abstract

Traditionally, rich, constraint-based grammatical resources have been hand-coded. Scaling wide-coverage, deep, constraint-based grammars such as Lexical-Functional Grammars from fragments to naturally occurring unrestricted text is knowledge-intensive, time-consuming and (often prohibitively) expensive.

Based on earlier work by McCarthy (2003), this thesis presents the development and evaluation of an automatic LFG f-structure annotation algorithm which is the core component in a larger project on rapid, wide-coverage, deep, constraint-based, multilingual grammar acquisition, addressing the knowledge acquisition bottleneck familiar from traditional rule-based approaches to NLP and AI. The algorithm annotates the Penn-II treebank with LFG f-structure information. Grammars and lexical resources are then extracted from the f-structure annotated treebank. Extensive evaluation of the annotation algorithm against independently constructed gold-standards (PARC 700 Dependency Bank and Propbank) shows the quality of the f-structures acquired.

The methodology developed in this thesis has been deployed for multilingual, rapid grammar development: grammars and lexical resources for Mandarin Chinese were acquired from the Penn Chinese Treebank (CTB) using a generic version of the annotation algorithm, seeded with linguistic generalisations for Mandarin Chinese.

Acknowledgements

I am deeply indebted to my supervisors Prof. Josef van Genabith and Dr. Andy Way for their guidance and support over the past few years. I am very grateful to them for giving me the opportunity to join their research group. Their advice and encouragement has helped me through times when things seemed impossible.

I enjoyed working with Aoife Cahill from the early days when she was patiently answering my daily questions via email from Stuttgart to the last year when she has been like a third supervisor to me. We achieved a lot together and I had a lot of fun.

Our research group has given me opportunities to meet people and experience places beyond my wildest dreams. I will be eternally grateful to everyone who made it possible for me to snowboard in New Zealand, trek in the Fijian jungle, get stuck in a lift in Barcelona, dance on podia in Vienna, eat elk burgers in Bergen, sleep in a Japanese volcano caldera and see the sun rise in Hong Kong before heading back into the nightclub.

Thanks to everyone in CAPG for creating a fun place to work. I am grateful to Dr. Wu Hai for the transliteration of Chinese sentences into English. The encouragement of Stephen O'Driscoll, Leo McNeir and all his characters is greatly appreciated. Thanks to Maribel, Nell and Elena for some fun nights in the most relaxed house in the city.

Séamus Murphy's rise from young manager of the month to manager of the year was an inspiration. I look forward to meeting his entourage again and finally presenting his award. I really appreciate the efforts Michelle Tooher made to introduce me to everyone when I started in DCU. She has been badly missed since she left for TCD. I hope Eoin and Sylvia are well.

Not going for coffee with Ruth O'Donovan every day and talking about whatever it is we have managed to talk about for the last few years (and nine short hours in Nadi airport) will take a lot of getting used to. She has been a true friend through good and bad times.

Finally, thanks to my family for their support, for always having the unbelievable belief in me and for managing to get the balance between concern and complete apathy just right. I am looking forward to getting home more often.

List of Tables

List of Figures

Chapter 1

Introduction

Deep grammars map strings to "meaning" representations in the form of dependency structures, predicate-argument structures or simple logical forms. In order to construct accurate and complete predicate-argument (dependency) structures (or logical forms), deep grammars resolve long-distance dependencies (LDDs). Traditionally, rich, constraint-based grammars, such as Lexical-Functional Grammars (LFG) (Kaplan and Bresnan, 1982; Bresnan, 2001; Dalrymple, 2001) or Head-Driven Phrase Structure Grammars (HPSG) (Pollard and Sag, 1994), have been hand-crafted. In this rule-based, *rationalist* approach, scaling grammars from small fragments to model naturally occurring unrestricted text is knowledge-intensive, time-consuming and (often prohibitively) expensive.

The availability of treebank resources has facilitated a new *empirical* approach to grammar development: the automatic acquisition of probabilistic context-free grammars (PCFGs) or history-based, generative parsers from treebanks (Charniak, 1996; Johnson, 1999; Charniak, 2000; Klein and Manning, 2003). While this quick, inexpensive process produces wide coverage, robust grammars, these grammars are usually "shallow". Most of the automatically acquired grammars do not map strings to "meaning" representations or attempt to resolve LDDs. Although there are some notable exceptions (Collins, 1999; Johnson, 2002; Hockenmaier, 2003), automatically acquired grammars are substantially less detailed than current unification- or constraint-based grammars such as LFG or HPSG. This poses a research question:

Is it possible to automatically acquire wide-coverage, deep, constraint-based,

grammatical resources from treebanks?

It *is* possible to automatically acquire wide-coverage, deep, constraint-based, grammatical resources from treebanks if an f-structure annotated treebank is available. This thesis presents a wide-coverage automatic f-structure annotation algorithm for the Penn-II treebank. The annotation algorithm is a core component of a larger project for the rapid automatic acquisition of wide-coverage, probabilistic LFG resources (Burke et al., 2004b). This leads to a second research question:

> Can we demonstrate that the resources automatically acquired using the annotation algorithm are of a high quality?

This thesis presents an extensive evaluation of the annotation algorithm against the DCU 105 gold standard, the PARC 700 Dependency Bank (King et al., 2003) and PropBank (Kingsbury and Palmer, 2002) data for Penn-II WSJ Section 23, demonstrating the high quality of the f-structures produced by the annotation algorithm. O'Donovan et al. (2004, 2005a) and Cahill et al. (2004b) present further extensive evaluation of the lexicon and grammar resources. Given that these high quality resources can be acquired for English, a third research question is:

> Can the automatic f-structure annotation algorithm and LFG resource acquisition methodologies be applied to other languages?

This thesis shows that the annotation algorithm and lexicon and grammar acquisition methodologies *can* be applied in a multilingual scenario by successfully porting the generic annotation algorithm to the Penn Chinese Treebank (CTB) for Mandarin Chinese (Xue et al., 2002), collaborative work which was published as Burke et al. (2004c).

The f-structure annotation algorithm consists of four modules: Left-Right Context Annotation, Co-ordination, Traces and Catch-All and Clean-Up. The Left-Right Context Annotation module identifies the head of each local subtree using a modified version of the head-lexicalisation rules of Magerman (1994). Annotations are provided for non-head nodes lying in either the left or right context of the head using annotation matrices which encode linguistic generalisations. Co-ordinate structures are annotated in a separate

module as the relatively flat Penn-II analysis of co-ordinate structures would significantly complicate the annotation matrices, making them harder to maintain and extend. The Traces module captures LDDs by using the null elements and co-indexation in Penn-II trees to produce corresponding re-entrancies at f-structure level. The Catch-All and Clean-Up module attempts to systematically correct some over-generalisations made in the earlier modules and to provide default annotations for remaining unannotated nodes.

McCarthy (2003) presents the linguistic basis for an early version of the f-structure annotation algorithm which provides basic f-structures with almost complete coverage of Penn-II. The present thesis describes an extensive overhaul, further development, extension and evaluation of the automatic f-structure annotation algorithm. I rewrote core components of the algorithm of McCarthy (2003) to overcome inefficiencies in the original implementation of the algorithm which significantly slowed development, extension, testing and evaluation cycles of the algorithm and negatively impacted on the performance of the parsing technology which incorporates the algorithm (Cahill et al., 2004a,b; Cahill, 2004). I corrected and extended the original annotation algorithm modules to provide a more fine-grained and standardised f-structure analysis. I reviewed the Penn-II annotation guidelines (Bies et al., 1995) to identify linguistic information encoded in the treebank trees which is not harnessed by the original algorithm. We[1] conducted a complete review of the DCU 105 gold standard.

Annotation quality is extremely important as the annotation algorithm and the f-structures it acquires from Penn-II are the basis for the automatic acquisition of wide-coverage and robust probabilistic approximations of LFG grammars and the induction of probabilistic lexical resources. I performed a quantitative and qualitative evaluation of the f-structures produced by the annotation algorithm. The algorithm produces a single covering and connected f-structure for 99.8% of all Penn-II trees. The algorithm achieves an f-score of 96.93% for all grammatical functions and 94.28% for preds-only against the DCU 105 gold standard using the evaluation methodology and software presented by Crouch et al. (2002) and Riezler et al. (2002). There are a number of problems with

[1]This thesis is part of a larger collaborative research project on the automatic acquisition from treebanks of probabilistic lexical (O'Donovan, 2006) and parsing (Cahill, 2004) LFG resources. In this thesis I will consistently use "I" to refer to work where I have been the lead researcher or first author of a publication and "we" for other collaborative work within our research team.

evaluating against a gold standard of this size, most notably that of over-fitting. There is a risk of assuming that the gold standard is a complete and balanced representation of the linguistic phenomena in a language and of basing design decisions on this. It is therefore preferable to evaluate against more extensive, independently constructed resources.

I evaluate the annotation algorithm against the PARC 700. The task of evaluating the automatically acquired f-structures against the PARC 700 is non-trivial and time-consuming due to the systematic differences in linguistic analysis, feature geometry and nomenclature between the PARC 700 and the automatically acquired f-structure representations. In order to achieve a fair evaluation against the PARC 700, I designed conversion software to overcome these systematic differences. The automatically acquired and mapped f-structures achieve an f-score of 87.33% against the PARC 700 test set for the feature set of Kaplan et al. (2004). Most of this work is published by Burke et al. (2004a).

I also evaluate the annotation algorithm against PropBank. Evaluating against PropBank provides a semantic evaluation of the automatically acquired f-structures, in contrast to the syntax-based DCU 105 and PARC 700. I converted the semantic role-based PropBank annotations (ARG0, ..., ARGM) into a dependency format (triples) to form the gold standard for evaluation using the software of Crouch et al. (2002) and Riezler et al. (2002). I converted the automatically acquired f-structures into LFG grammatical function-based triples (SUBJ, OBJ, ...). I developed conversion software to systematically map these triples to the PropBank semantic role-based triples encoding. Using the Penn-II WSJ section 24 as the development set, the mapped output of the annotation algorithm achieves an f-score of 76.58% against PropBank for the WSJ section 23 test set. Most of this work is published by Burke et al. (2005).

The f-structure annotation algorithm underpins the automatic acquisition of high-quality, wide-coverage LFG resources from treebanks. The methodology developed in this thesis has been deployed for multilingual, rapid grammar development. In collaboration with a research team from the University of Hong Kong (Adams Bodomo, Olivia Lam and Rowena Chan), I explore the application of our technology to the Penn Chinese Treebank (CTB) for Mandarin Chinese, acquiring grammars and lexical resources for Mandarin Chinese using a generic version of the annotation algorithm, seeded with

linguistic generalisations for Mandarin Chinese. For 95.123% of the CTB training set trees, the annotation algorithm generates a single covering and connected f-structure. A total of 2,510 verbal semantic form tokens with 26 distinct frame types are extracted from the annotated treebank. The best-performing grammar (PCFG-P-A) achieves a labelled f-score of 81.77% in the tree-based evaluation, outperforming the previous best reported labelled f-score of 79.9% by Chiang and Bikel (2002). Most of this work was published by Burke et al. (2004c).

Applications of the work presented in this thesis include the automatic acquisition of lexicon and grammar resources. In O'Donovan et al. (2004) and O'Donovan et al. (2005a) we show how subcategorisation frames and their associated probabilities are extracted from the automatically f-structure-annotated Penn-II and Penn-III treebanks. An evaluation of the acquired probabilistic lexical resources is performed against COMLEX (Macleod et al., 1994). The extracted subcategorisation frames with OBL arguments, but without specific prepositions and particles, achieve f-scores of 63.6% and 62.2% with thresholds of 1% and 5% respectively. In Cahill et al. (2004b) we show how the annotation algorithm underpins wide-coverage, robust treebank-based probabilistic LFG approximations to parse raw text into f-structures. The f-structures resulting from parsing raw text achieve an f-score of 83.08% when evaluated against the PARC 700 Dependency Bank (King et al., 2003) using the conversion software presented in this thesis.

This thesis is structured as follows:

Chapter 2 reviews previous research into the treebank-based approach to grammar development and motivates the need for the improvement and extension of this research. Important avenues for the correction, extension and evaluation of the annotation algorithm of McCarthy (2003) are outlined.

Chapter 3 presents an extensive overhaul, further development, extension and evaluation of the automatic f-structure annotation algorithm.

Chapter 4 evaluates the automatic f-structure annotation algorithm against the PARC 700 using conversion software to overcome systematic differences between the automatically acquired f-structure and the PARC 700 dependency representations.

Chapter 5 evaluates the annotation algorithm against PropBank.

Chapter 6 applies our technology to the Penn Chinese Treebank (CTB) for Mandarin Chinese.

Chapter 7 concludes and outlines some applications of the annotation algorithm and areas for future work.

Chapter 2

Background and Motivation

2.1 Introduction

Deep grammars map strings to "meaning" representations in the form of dependency structures, predicate-argument structures or simple logical forms. Traditionally, deep unification- or constraint-based grammars have been manually constructed. This is time-consuming and expensive and rarely achieves wide coverage on unrestricted text. The availability of treebank resources has facilitated a new approach to grammar development: the automatic extraction of probabilistic context-free grammars (PCFGs) from treebanks. While this quick, inexpensive process produces wide coverage grammars, these grammars are usually "shallow". They do not map strings to "meaning" representations and very few attempt long distance dependency resolution.

This thesis presents an automatic f-structure annotation algorithm for the annotation of the Penn-II treebank (Marcus et al., 1994) with Lexical Functional Grammar (LFG) (Kaplan and Bresnan, 1982; Bresnan, 2001; Dalrymple, 2001) f-structure information. From the f-structure-annotated treebank, probabilistic constraint-based LFG resources are automatically extracted. This approach, like previous shallow automatic grammar acquisition paradigms, is quick, inexpensive and achieves wide coverage. However, the automatically acquired LFG resources are deep, mapping strings to dependency structures, and capture long distance dependencies.

This chapter reviews previous research into transfer-based approaches to LFG grammar

development and motivates the need for the improvement and extension of this research. Section 2.2 introduces LFG and motivates the choice of this formalism for treebank annotation. Section 2.3 describes previous small-scale and proof of concept approaches to automatic f-structure annotation by Sadler et al. (2000) and Frank (2000). Section 2.4 summarises McCarthy (2003) which reports on the development of a basic large-scale automatic f-structure annotation algorithm prior to my thesis research. Section 2.5 motivates the need for further research into the automatic f-structure annotation of Penn-II. Important avenues for the correction, extension and evaluation of the annotation algorithm are outlined, motivating the research presented in Chapters 3 to 5. Section 2.6 summarises the chapter.

2.2 Lexical Functional Grammar

Lexical Functional Grammar (LFG) (Kaplan and Bresnan, 1982; Bresnan, 2001; Dalrymple, 2001) is a unification- or constraint-based grammar formalism. C(onstituent)-structure and f(unctional)-structure are the two levels of LFG representation most relevant to this thesis. Language-specific surface grammatical configurations are represented as syntax trees at c-structure level. Abstract syntactic functions, e.g. SUBJ(ect), are represented at f-structure level in the form of attribute-value matrices (AVMs). Figure 2.1 provides a c-structure for the sentence *John saw Mary* annotated with f-structure equations. The up-arrow meta-variable (↑) denotes information associated with the f-structure of the parent node, while the down-arrow meta-variable (↓) associates information with the local node. Each instance of a meta-variable is instantiated using a unique identifier associated with the node to which the meta-variable refers, which allows a set of equations (f-descriptions) to be created from the annotated c-structure. For example, the annotations on the subtree NP → *John* in Figure 2.1 would include the equations F_1(SUBJ) = F_2, F_2(PRED) = *John*, F_2(NUM) = *sg* and F_2(PERS) = *3*. An f-structure is formed if the f-descriptions of an annotated c-structure can be resolved (Figure 2.1).

While the principles underpinning the annotation algorithm are independent of linguistic formalisms, the algorithm is implemented using LFG for the following reasons:

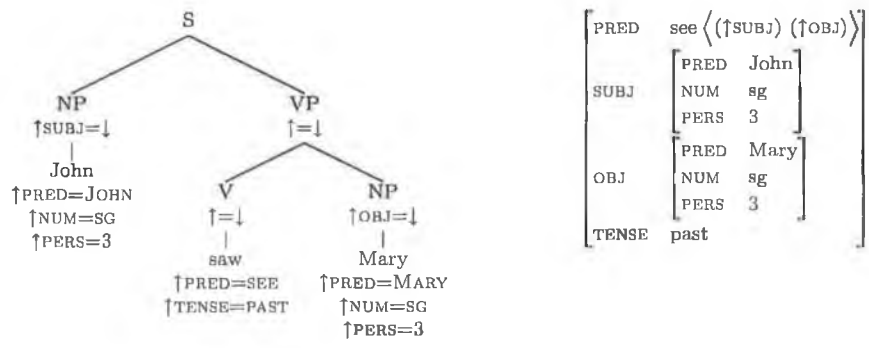


Figure 2.1: C- and f-structures for the sentence *John saw Mary*

- LFG was designed from the outset to be used in computational systems and provides a platform for the concise declaration of linguistic generalisations required by the annotation algorithm.
- LFG f-structures provide an abstract syntactic representation which (to a certain extent) is independent of language-specific surface configurations. For example, the f-structure (Figure 2.2) for *Chonaic Seán Máire* — the Irish translation of *John saw Mary* — is isomorphic with the English equivalent (Figure 2.1) despite the widely differing c-structures. This characteristic has benefits for the application of the algorithm and acquired f-structures for machine translation and multilingual grammar development purposes.
- A body of previous research into automatic LFG f-structure annotation was available (Section 2.3).

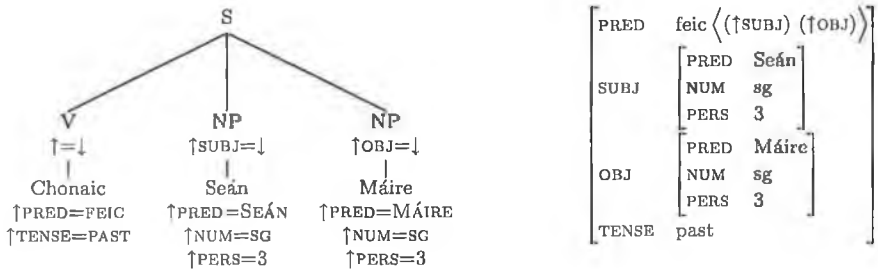


Figure 2.2: C- and f-structures for the sentence *Chonaic Seán Máire*

2.3 Automatic F-Structure Acquisition Techniques (Sadler et al., 2000), (Frank, 2000)

Sadler et al. (2000) describes a regular expression-based approach to the automatic f-structure annotation of treebank trees, experimenting with the 100-sentence publicly available subset of the AP treebank (Leech and Garside, 1991). A context-free grammar (CFG) is extracted from this treebank subset. F-description templates in the form of regular expressions capturing linguistic generalisations are created and applied to the extracted CFG rules. Re-applying the annotated CFG rules to the original treebank trees produces f-structure-annotated c-structures from which f-structures can be generated. The number of f-description templates required was significantly lower than the total number of CFG rule types extracted.

Frank (2000) presents a flat, set-based tree description rewriting methodology for the acquisition of f-structures from treebank trees, experimenting with 166 sentences of the Susanne corpus (Sampson, 1995). Instead of encoding linguistic generalisations for the annotation of c-structures with f-structure equations, a tree description language is used to represent the c-structures as a flat set description. Annotation principles are then applied to this more abstract representation of the treebank tree using a re-writing system originally designed for use in transfer-based machine translation architectures (Kay et al., 1994). F-structures are then generated from the annotated flat representation.

To date, both of these 'proof-of-concept' approaches to f-structure acquisition have only been applied to small treebank subsets. The automatic f-structure annotation algorithm presented in this thesis is scaled to provide near complete coverage of the one million word WSJ section of Penn-II.

2.4 Annotation Algorithm (McCarthy, 2003)

2.4.1 Introduction

Cahill et al. (2002a,b,c) report on initial research into the development of an automatic f-structure annotation algorithm for Penn-II. At that stage, the algorithm provided a

coarse-grained linguistic analysis producing *proto* f-structures which interpret linguistic information in the treebank trees locally and do not resolve long distance dependencies (LDDs). McCarthy (2003) extends this research by providing a more fine-grained linguistic analysis and producing *proper* f-structures which capture LDDs. The resulting algorithm provides almost complete coverage of the WSJ section of Penn-II and was evaluated against the DCU 105, a publicly available[1] gold standard consisting of f-structures for 105 randomly selected trees from WSJ Section 23 of Penn-II. To create the gold standard f-structures, the trees were first automatically annotated and then the annotations were manually extended and corrected. McCarthy (2003) reports on the linguistic information encoded in the annotation algorithm, while the implementation was carried out by Cahill (2004). This section summarises the status of the automatic f-structure annotation algorithm prior to my thesis research.

2.4.2 Algorithm Overview

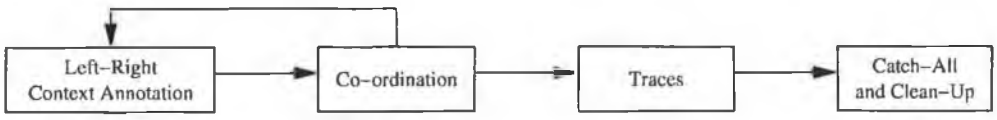


Figure 2.3: Annotation Algorithm Modules

The annotation algorithm consists of four modules: Left-Right Context Annotation, Co-ordination, Traces and Catch-All and Clean-Up (Figure 2.3). The Left-Right Context Annotation module identifies the head of each local subtree using a modified version of the head-lexicalisation rules of Magerman (1994). This creates a bi-partition of the local subtree, with non-head nodes lying in either the left or right context of the head. This module provides annotations for nodes in both contexts using a set of annotation matrices. The procedure for constructing the annotation matrices and the left-right context annotation process are outlined in Section 2.4.3. Section 2.4.4 describes the annotation of co-ordinate structures by the Co-ordination module. Co-ordination is annotated in this separate module as its relatively flat analysis in Penn-II would complicate the left-right context annotation matrices, making them harder to maintain and extend. Section 2.4.5

[1]Available from `http://www.computing.dcu.ie/research/nclt/gold105.txt`

outlines the annotation of LDDs by the Traces module, which uses the null elements and co-indexation in Penn-II trees to produce corresponding re-entrancies at f-structure level and allow *proper* f-structures to be generated. A degree of over-generalisation in the first three modules allows a clearer statement of linguistic generalisations. Section 2.4.6 provides an overview of the Catch-All and Clean-Up module which attempts to systematically correct some over-generalisations made in the earlier modules. The results of the evaluation performed by McCarthy (2003) are provided in Section 2.4.7.

2.4.3 Left-Right Context Annotation Module

2.4.3.1 Head-Lexicalisation

The annotation of a subtree begins with the identification of the head node. Originally, the annotation algorithm used the head-lexicalisation rules of Collins (1996), but better results were achieved using Magerman's (1994) rules with some amendments. For each Penn-II parent category, the rules list the most likely head categories in rank order and indicate the direction from which the search for the head category should begin. Figure 2.1 provides Magerman's (1994) head rules for the Penn-II S and VP categories. These rules indicate that the head of an S subtree is identified by traversing the daughter nodes from right to left and that VP is the most likely head. The annotation algorithm marks the rightmost VP in an S subtree as head using the f-structure equation $\uparrow=\downarrow$. If the S subtree does not contain a VP node, it is searched from right to left for the next most likely head candidate (SBAR). In the unlikely event that none of the listed candidates occur in the subtree, the rightmost non-punctuation node is marked as head. Similarly, the process of identifying the head of a VP subtree begins by searching from left to right for a VBD node.

Parent Category	Direction	Ranked head categories
S	right	VP SBAR ADJP UCP NP
VP	left	VBD VBN MD VBZ TO VB VP VBG VBP ADJP NP

Table 2.1: Magerman's Head Rules for S and VP

McCarthy (2003) lists some amendments which were made to the head-lexicalisation rules of Magerman (1994) for use in the annotation algorithm. The most noticeable change is the elevation of MD (modal verb) to be the highest-ranking head candidate for VPs. Another important change is the addition of all nominal POS tags and phrasal categories to the head candidate list for WHNP as the highest-ranking categories.

2.4.3.2 Annotation Matrix Construction

The head-lexicalisation process creates a bi-partition of the local subtree, with non-head nodes lying in the left or right context of the head node. Annotation matrices were constructed for each Penn-II parent category to provide f-structure annotations for nodes in the left and right contexts. The linguistic generalisations encoded in these annotation matrices allowed automatic f-structure annotation to be scaled up from the small treebank subsets of previous approaches (Section 2.3) to provide near complete coverage of Penn-II. The matrix construction process leveraged the Zipfian distribution in Penn-II of CFG rule tokens over rule types, whereby a small fraction of rule types account for the majority of token CFG rule occurrences. The most frequently occurring rule types providing combined coverage of at least 85% of rule tokens for each Penn-II parent category were extracted. These subsets of rule types were analysed and manually annotated with LFG f-structure equations which were then used to create the linguistic generalisations encoded in the annotation matrices.

Extracting the most frequently occurring rule tokens in this manner greatly reduced the task of linguistic analysis. Although there is an element of over-generalisation in the resulting annotations, the benefits of this approach far outweigh this consideration. For example, only the most frequently occurring 102 Penn-II NP rule types were analysed as they provided combined coverage of over 85% of all 6,595 NP types.

Table 2.2 contains a sample of the annotations provided by the algorithm of McCarthy (2003) in the NP annotation matrix, which amongst other things indicate that a DT node occurring to the left of an NP's head node should be annotated $\uparrow$SPEC:DET=$\downarrow$. Similarly, a PP occurring to the right of the head should be annotated as an adjunct.

Left context	Head	Right context
DT: ↑SPEC:DET=↓ ADJP, JJ, N, NN, NNS: ↓∈↑ADJUNCT CD: ↑SPEC:QUANT=↓	N, NN, NNS: ↑=↓	RRC, SBAR: ↑RELMOD=↓ PP: ↓∈↑ADJUNCT

Table 2.2: Sample from NP Annotation Matrix

2.4.3.3 Left-Right Context Annotation Process

Left-right context annotation proceeds in a top-down, left to right manner. The head node of each local subtree is identified using the modified version of Magerman's (1994) rules and annotated ↑=↓. The remaining nodes in the local subtree lie in the left or right context of this head node and are annotated using the annotation matrices. Figure 2.4 provides the automatically annotated Penn-II tree for the NP *the gloomy forecast.* The NN node is annotated ↑=↓, as the NP head rules indicate that the rightmost nominal node is the head. The nodes DT and JJ lie in the left context. Consulting the NP annotation matrix (Table 2.2) provides the annotations ↑SPEC:DET=↓ and ↓∈↑ADJUNCT for DT and JJ, respectively.

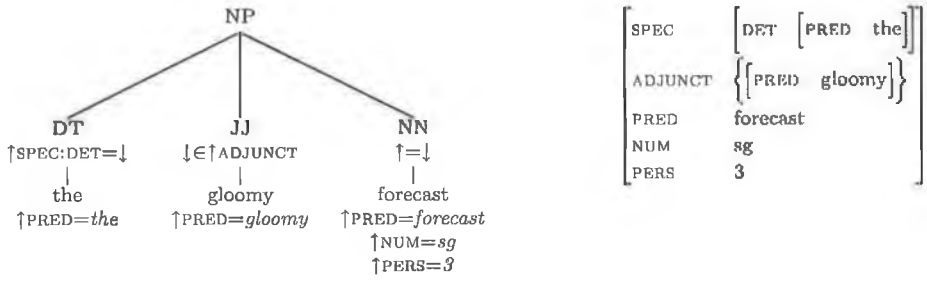


Figure 2.4: Automatically annotated Penn-II tree and resulting f-structure for *the gloomy forecast*

Lexical macros for each Penn-II POS tag provide annotations for word nodes. Verbal categories are annotated with TENSE features while nouns receive number and person features (Figure 2.4). The surface word forms are lemmatised using the XLE morphological component (Maxwell and Kaplan, 1993) to provide the PRED values.

2.4.4 Co-ordination Module

2.4.4.1 Introduction

Penn-II provides a deliberately flat analysis for co-ordinate structures. The annotation algorithm processes these structures in a separate module, as annotating co-ordination using the Left-Right Context Annotation module would complicate the annotation matrices significantly. Penn-II distinguishes between *like* and *unlike* co-ordinate structures. In most cases, co-ordinated elements belong to the same or similar categories. The parents of such *like* co-ordinate structures are tagged to indicate the type of the co-ordinated elements, e.g. the parent of a phrase containing co-ordinated singular (NN) and plural (NNS) nouns is tagged NP. When the co-ordinated elements belong to different categories, the parent is tagged with the UCP (Unlike Co-ordinated Phrase) category, e.g. the parent of co-ordinated nouns and adjectives. This subsection summarises the handling of *like* co-ordination by the annotation algorithm. *Unlike* co-ordination occurs relatively infrequently and is covered in some detail by McCarthy (2003).

2.4.4.2 NP Co-ordination

When handling co-ordination within NPs, the annotation algorithm first identifies whether the co-ordinate structure forms the head of the NP or an adjunct of the head. Secondly, the number of consecutive nominal nodes to the right of the CC is counted.[2] If more than one consecutive nominal node is found, the CC is annotated as an adjunct and the head-lexicalisation rules are used to find the head of the NP. If only one nominal node is found, the CC is annotated as the head. The procedure for annotating both cases follows.

Co-ordinated adjuncts within NPs The NP *futures and options trading firms* is an example of an NP containing co-ordinated adjuncts (Figure 2.5). The head-lexicalisation rules identify the rightmost nominal as the NP's head ($\uparrow=\downarrow$). The co-ordinate structure must be annotated as an adjunct, but this is not possible using only the up- and down-arrow meta-variables due to the flat Penn-II analysis within the co-ordinated NP. To ensure that the co-ordinate structure is grouped as a single adjunct at f-structure level, a unique

[2]Multi-word co-ordinating conjuncts are grouped in Penn-II under the CONJP tag. Single word conjuncts are tagged as CCs. For simplicity, both are referred to as CC throughout this thesis.

variable (X in this case) is created to identify the CC (X=↓) and to annotate it as an adjunct of the head (X∈↑ADJUNCT). The nominals to the immediate left and right of the CC are annotated as elements of the co-ordination set linked to the CC (↓∈XCONJ). The Left-Right Context Annotation module is invoked to annotate all remaining unannotated nodes, providing ↓∈↑ADJUNCT as the annotation for the NN *trading*.

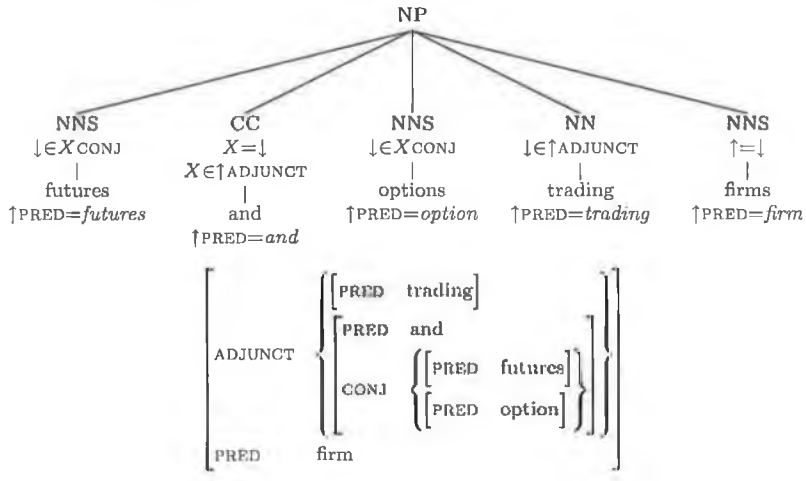


Figure 2.5: Automatically annotated Penn-II tree and resulting f-structure for *futures and options trading firms.* (NUM and PERS features are omitted.)

NP with conjunct as head The up- and down-arrow meta-variables are sufficient for the annotation of NPs headed by co-ordinate structures. The automatically annotated tree and resulting f-structure for the NP *The energy and ambitions* is provided in Figure 2.6. The CC is annotated as head and the nominals to its immediate left and right are annotated as elements of the co-ordination set. Any remaining unannotated nodes are annotated using the Left-Right Context module, which provides ↑SPEC:DET=↓ as the annotation for DT.

The Co-ordination module extends to handle lists of co-ordinated nominals separated by commas, e.g. *the shopping, laundry and cooking.* The nodes preceding the nominal immediately to the left of the CC are recursively examined. Nominals occurring in pairs of nominal and comma nodes are added to the co-ordination set.

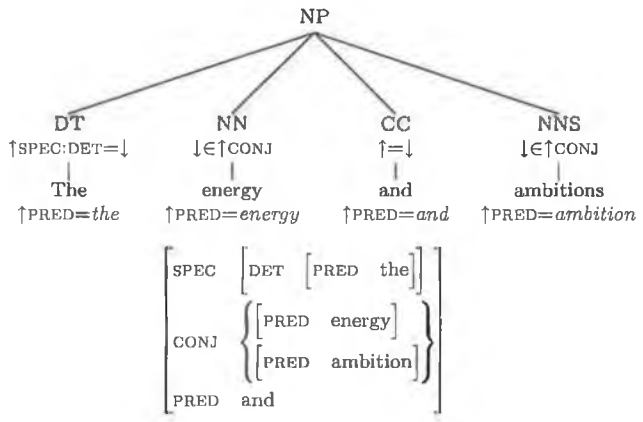


Figure 2.6: Automatically annotated Penn-II tree and resulting f-structure for *The energy and ambitions.* (NUM and PERS features are omitted.)

2.4.5 Traces Module

2.4.5.1 Passivisation

The standard Penn-II treatment of passivisation is to insert a null NP node as the object of the passivised verb, co-indexed with the constituent in subject position. Figure 2.7 provides the automatically annotated Penn-II tree and the resulting f-structure for the sentence *A successor wasn't named.* The null NP node (NP → -NONE- → *-1) follows the passivised verb and is co-indexed with the surface subject (NP-SBJ-1 → *A successor*). The null NP node in Figure 2.7 triggers the annotation algorithm to provide passive annotations in a two-step process. First, the VP parent of the null NP node is annotated ↑PASSIVE=+. Second, the tree is traversed upwards with the equations ↑PASSIVE=+ and ↑XCOMP:PASSIVE=+ added to all VP nodes until a non-VP node is met.

All f-structure equations on the annotated tree are provided by the Left-Right Context Annotation module and the above procedure for annotating passivisation. The equations resolve to the f-structure provided (Figure 2.7), which shows that the two occurrences of the equation ↑PASSIVE=+ unify. As a result, the three passive equations on the annotated tree resolve to two passive feature-value pairs in the f-structure.

The annotation algorithm has two further methods for annotating passive voice. When the logical subject of a passive verb is realised in a sentence, Penn-II annotates it with the

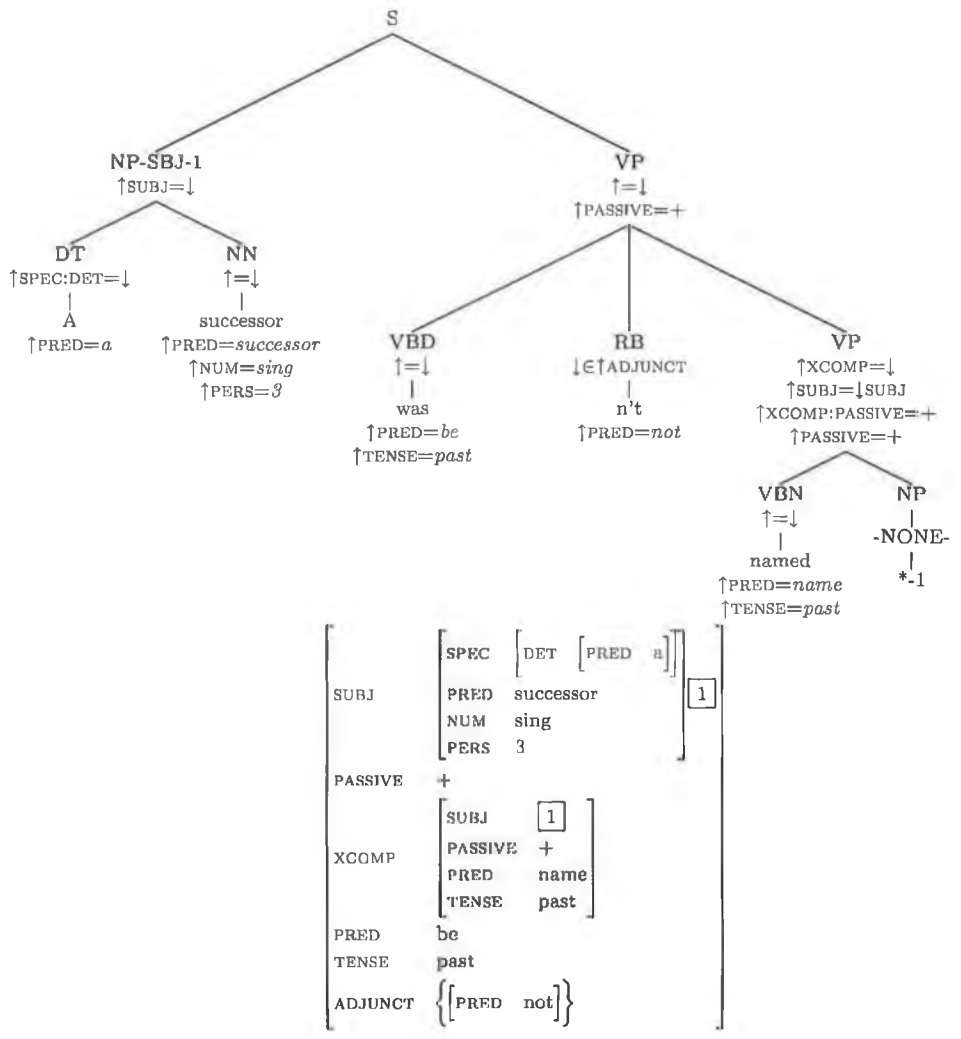


Figure 2.7: Automatically annotated Penn-II tree and resulting f-structure for *A successor wasn't named.*

functional tag -LGS. This tag triggers the algorithm to provide passive annotations in the manner outlined above. Passive annotations are added only once for each verb, so if the algorithm has already found a null NP node and provided passive annotations as a result, these annotations will not be duplicated if an -LGS tag is found. In practice, -LGS tags rarely occur in Penn-II without a null NP node also being present, so this second passive annotation method is rarely invoked.

The annotation algorithm is an important component of the parsing technology outlined in Chapter 6. In the pipeline parsing model, the annotation algorithm is used to

annotate automatically generated parse trees which do not contain Penn-II null nodes or co-indexation. Most parsers do not produce Penn-II functional tags either, so the algorithm requires a more general case in order to annotate passivisation in these parse trees. The algorithm annotates passivisation in a VP using this more general case if the following conditions are met:

1. The VP is headed by any form of *be* or *get*.
2. There is a VP to the right of this verb.
3. This second VP is headed by a past participle (VBN).

These criteria match most VPs which will already have been annotated as passivised by the previous two methods which are triggered by null NP nodes and -LGS tags, e.g. this third more general case describes the VP *wasn't named* in Figure 2.7. In this way, passive voice can be annotated relatively accurately in trees produced by parsers which do not contain null elements or Penn-II functional tags.

2.4.5.2 Topicalisation

Penn-II employs the functional tag -TPC and the co-indexed null element *T* to represent the LDD between a topicalised constituent and that argument's canonical location relative to the subcategorising verb. The fronted element is annotated with the -TPC tag and is given an identity index. The null element *T* is given a referential index to match the fronted element. The annotation algorithm uses this co-indexation to capture the LDD as a re-entrancy at f-structure level.

Figure 2.8 provides the automatically annotated Penn-II-style tree and the resulting f-structure for the sentence *An excellent actor he is.* The fronted element *An excellent actor* is tagged -TPC and has the identity index 1. The node NP-PRD governs the null element *T*-1, which is co-indexed with the fronted element and represents its canonical location. The algorithm uses the -TPC tag to annotate the fronted element with the equation $\uparrow$TOPIC=$\downarrow$. The Traces module invokes the Left-Right Context Annotation module to annotate the NP-PRD node as the object of the verb. On finding a *T*-1 null element, the tree is traversed in a top-down left to right manner to locate the co-indexed node

and to provide an annotation to capture the LDD as an f-structure re-entrancy. Given the nature of LDDs this annotation is not possible using only the up- and down-arrow meta-variables.

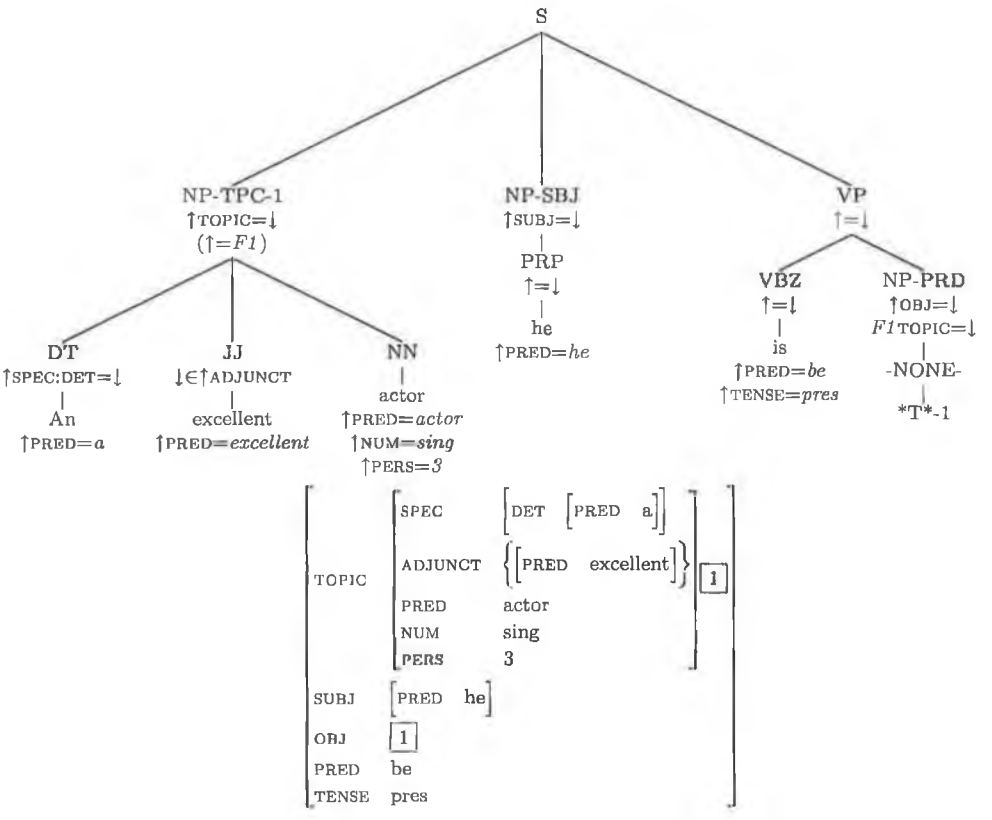


Figure 2.8: Automatically annotated Penn-II-style tree and resulting f-structure for the *An excellent actor he is.*

The process of resolving the functional equations of an annotated tree to form an f-structure involves instantiating each occurrence of a meta-variable with a unique identifier associated with the node to which the meta-variable refers (cf. Section 2.2, pp. 8). The annotation algorithm uses these unique identifiers to capture LDD re-entrancies. In Figure 2.8, the node NP-TPC-1 representing the fronted element is annotated ↑TOPIC=↓. The re-entrancy is captured by instantiating the up-arrow meta-variable in this equation with a unique identifier for the node to which it refers, e.g. ↑=*F1*. The unique identifier in this case is *F1* which refers to the S node. Instantiating the up-arrow meta-variable produces the equation *F1*TOPIC=↓. This new equation is placed on the NP-PRD node which governs

the null *T*-1 element. This equation states that the f-structure information associated with the TOPIC of the S node is now also associated with the NP-PRD node. The process of unification ensures that the equations *F1*TOPIC=↓ and ↑OBJ=↓ on the NP-PRD node resolve to provide the re-entrancy between TOPIC and OBJ in the resulting f-structure.

2.4.5.3 Relative Clauses

Relative clauses are typically grouped in Penn-II trees as SBARs and attached as phrasal post-modifiers. The SBAR has two daughter nodes: an identity indexed WH-phrase (e.g. WHNP-1) governing the relative pronoun and an S clause which contains a *T* null element co-indexed with the WH-phrase. The Left-Right Context Annotation module annotates the SBAR and WH-phrase with the equations ↑RELMOD=↓ and ↑TOPICREL=↓, respectively. The Traces module must invoke the Left-Right Context Annotation module to annotate the node governing the *T* null element, as annotations are not provided initially for nodes governing null elements. The re-entrancy between the null element and the WH-phrase governing the relative pronoun is captured using the same procedure as described in Section 2.4.5.2 for topicalisation.

Figure 2.9 provides the automatically annotated Penn-II-style tree and resulting f-structure for the phrase *firm, which tracks earnings.* The NP *firm* is post-modified by an SBAR which consists of a WH-phrase (WHNP-1) and an S clause. The S clause contains the co-indexed *T*-1 null element which occurs in subject position. All annotations are provided by the Left-Right Context Annotation module, except for those on the NP-SBJ node governing the null element. The Traces module invokes the Left-Right Context Annotation module to annotate this node with the equation ↑SUBJ=↓. The tree is then traversed in a top-down left to right manner to find the node co-indexed with the null element. Having located the co-indexed WHNP-1 node, the Traces module captures the re-entrancy by annotating the NP-SBJ node with the equation *F5*TOPICREL=↓. The variable *F5* uniquely identifies the SBAR node. The equations *F5*TOPICREL=↓ and ↑SUBJ=↓ on the NP-SBJ node indicate that the f-structure information associated with the WHNP-1 node should be interpreted as both SUBJ and TOPICREL of the verb *track.*

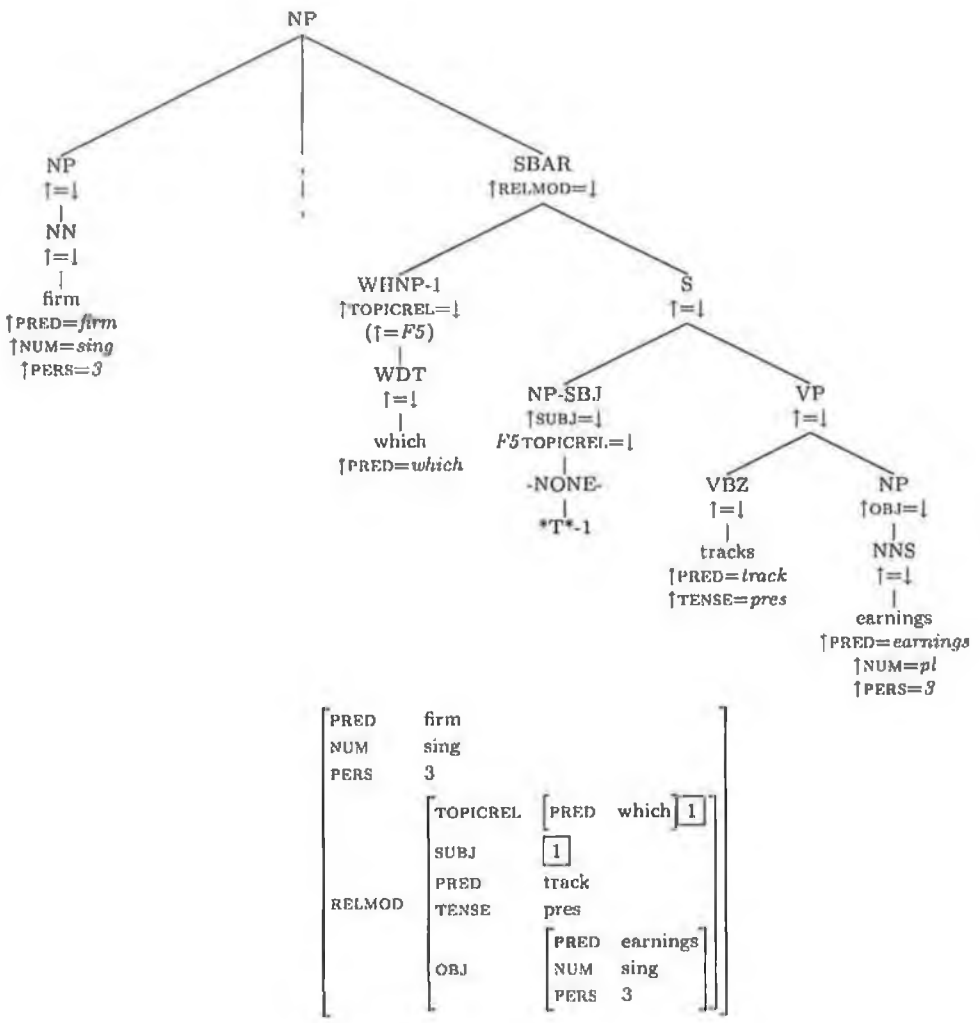


Figure 2.9: Automatically annotated Penn-II-style tree and resulting f-structure for *firm, which tracks earnings*

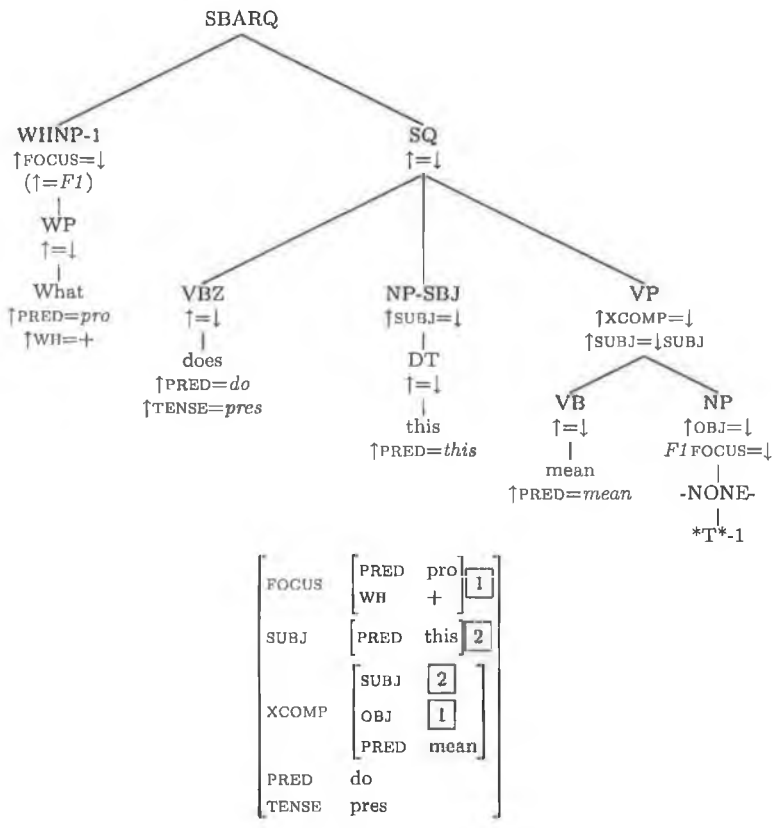


Figure 2.10: Automatically annotated Penn-II-style tree and resulting f-structure for the wh-question *What does this mean?*

2.4.5.4 *Wh*-questions

The Penn-II analysis of *wh*-questions is very similar to the treatment of relative clauses. Relative clauses are grouped under SBAR nodes, while *wh*-questions are governed by SBARQ nodes. The daughter nodes of SBARQ are a WH-phrase and an SQ clause. As with relative clauses, the WH-phrase has an identity index. The SQ clause contains a null *T* element which is co-indexed with the WH-phrase. The annotation algorithm must capture the re-entrancy at f-structure level between the fronted WH-phrase and the location of its interpretation in the SQ clause using the discourse function FOCUS. The same procedure as with relative clauses and topicalisation is used to locate the co-indexed WH-phrase and to provide the re-entrancy annotation.

Figure 2.10 provides the automatically annotated Penn-II-style tree and resulting f-structure for the sentence *What does this mean?* All annotations are provided by the

Left-Right Context Annotation module except for those on the NP governing the *T*-1 null element. The Traces module invokes the Left-Right Context Annotation module to annotate this node as the object of *mean*. The tree is traversed in a top-down left to right manner to locate the node WHNP-1 which is co-indexed with the *T*-1 null element. *F1* uniquely identifies the SBARQ node. The equation *F1*FOCUS=↓ is added by the Trace modules to capture the FOCUS re-entrancy.

2.4.6 Catch-All and Clean-Up

Catch-All and Clean-Up, the final module of the annotation algorithm, attempts to correct errors which may have been caused by over-generalisation in the previous three modules. Penn-II functional tags are used to insert missing annotations or to correct existing ones. Table 2.3 lists the default annotations provided by the *Catch-All* phase of this module for Penn-II functional tags. Any remaining unannotated nodes occurring with any Penn-II functional tag are annotated as adjuncts. Attempts are made to avoid feature clashes in the *Clean-Up* phase of this module by identifying when two occurrences of OBL, OBJ or XCOMP features are annotated for a single verb. In such cases, the second occurrence of these features is renamed OBL2, OBJ2 or XCOMP2 as appropriate.[3] The Catch-All and Clean-Up module also contains preliminary, unsuccessful attempts at annotating apposition.

Penn-II functional tag	Catch-All annotation
-BNF	↑OBL=↓
-CLR	↑OBL=↓
-DTV	↑OBL=↓
-PUT	↑PART=↓
-SBJ	↑SUBJ=↓

Table 2.3: Default annotations provided for Penn-II functional tags

[3]Functions such as OBL2 and XCOMP2 are not proper LFG grammatical functions but more a "robustness" feature of McCarthy's (2003) system. They are used in rare cases to ensure that f-structures are generated (rather than producing no output due to feature clashes).

2.4.7 Evaluation

All Penn-II trees (excluding trees with FRAG and X nodes[4]) were annotated by the algorithm. The f-structure equations were resolved and the quantity and quality of the resulting f-structures were evaluated by McCarthy (2003). The annotation algorithm provided a single covering and connected f-structure for 99.41% of all Penn-II trees (Table 2.4). Despite the attempts made in the Catch-All and Clean-Up module, feature clashes in 0.47% of trees resulted in no f-structures being produced for those trees. Unannotated nodes resulted in two separate f-structure fragments being generated for 58 trees (0.12%).

# F-structure fragments	# Trees	% Treebank
0	226	0.47
1	48140	99.41
2	58	0.12

Table 2.4: Quantitative F-Structure Evaluation

Annotation quality was evaluated against the DCU 105 in terms of precision, recall and f-score[5] using the methodology and software of Crouch et al. (2002) and Riezler et al. (2002). The 105 Penn-II trees of the gold standard were automatically annotated and the resulting f-structures were evaluated against the gold standard f-structures. Results are provided for all grammatical functions and preds-only[6] f-structures (Table 2.5). The annotation algorithm achieves an f-score of 94.11% for all grammatical functions and 90.86% for preds-only f-structures.

	All grammatical functions	Preds-only
Precision (%)	93.53	90.46
Recall (%)	94.69	91.26
F-score (%)	94.11	90.86

Table 2.5: Qualitative F-Structure Evaluation

[4]FRAG(ment) marks clauses whose exact structure cannot be determined. X is used to mark ungrammatical strings.

[5]Precision, recall and f-score were calculated according to the following equations:

$$precision = \frac{\#\ of\ correct\ feature{-}value\ pairs\ in\ the\ automatically\ generated\ f{-}structure}{\#\ of\ feature{-}value\ pairs\ in\ the\ automatically\ generated\ f{-}structure}$$

$$recall = \frac{\#\ of\ correct\ feature{-}value\ pairs\ in\ automatically\ generated\ f{-}structure}{\#\ of\ feature{-}value\ pairs\ in\ the\ gold\ standard\ f{-}structure}$$

$$f - score = \frac{2 \times precision \times recall}{precision + recall}$$

[6]Preds-only f-structures consider only paths in f-structures ending in a PRED feature-value pair.

2.5 Overhaul, Further Development, Extension, Correction and Evaluation of the Annotation Algorithm

2.5.1 Practical Considerations

The original annotation algorithm of McCarthy (2003) was slow, taking over 30 minutes for the annotation of Penn-II and the generation of f-structure equations for resolution. This hindered the process of developing the algorithm with unacceptably long development, testing and evaluation turnaround cycles for any modifications of the annotation algorithm and impacted on the performance of the parsing technology (Cahill et al., 2004b) which incorporates the algorithm. The generation of f-structure equations from f-structure annotations was the main source of inefficiency as a result of extensive string manipulation in Java in McCarthy (2003). Significant improvements were required to speed up the algorithm including the removal of redundant code in the annotation process and the development of a new method for computing f-structure equations. Section 3.2 reports on work I have carried out to improve the annotation algorithm with respect to these practical issues.

2.5.2 Improvements to Existing Procedures

The Left-Right Context Annotation matrices had to be extended to improve the coverage of the annotation algorithm. The annotation matrices of McCarthy (2003) failed to provide annotations for certain parent/daughter combinations in one or both contexts. 58 Penn-II trees received 2 f-structure fragments due to unannotated nodes (Table 2.4). Analysis of f-structures produced by the parsing technology which incorporates the annotation algorithm (Chapter 6) highlighted further missing annotations. The main changes which were required for the Co-ordination module were the simplification of the co-ordination rule implementation. The algorithm code had to be aligned with the Co-ordination module as described by McCarthy (2003) and redundant code had to be removed. The implementation of the Traces module had to be simplified. Important extensions to this module necessitated an extensive review of the Penn-II treebank annotation guidelines (Bies et al., 1995) (Section 2.5.4). A more fine-grained analysis was required to provide re-entrancies

for *wh*-less relative clauses. Missing re-entrancies into XCOMPs had to be examined and appropriate changes made. A more standardised annotation of oblique agents was needed. One area of improvement for the Catch-All and Clean-Up module was the completion of the preliminary attempts at the annotation of apposition. Section 3.3 presents improvements to the existing procedures of the annotation algorithm which I have carried out.

2.5.3 Review of DCU 105

The DCU 105 gold standard consists of f-structures for 105 randomly selected trees from WSJ Section 23 of Penn-II. To create the gold standard f-structures, the trees were first automatically annotated and then the annotations were manually extended and corrected. A complete review of the McCarthy (2003) DCU 105 was required to correct errors which were missed in the original manual correction phase; to standardise the treatment of some grammatical functions, in particular relative clauses and to provide more fine-grained analysis for a number of important phenomena (such as relative clauses). Idiosyncratic feature and value names had to be standardised to more widely accepted terminology. In addition to improving and extending the DCU 105, the review process informed annotation algorithm design decisions and extensions. We performed a manual review of the DCU 105 which is presented in Section 3.5.

2.5.4 Review of Penn-II Annotation Guidelines

A review of the Penn-II annotation guidelines (Bies et al., 1995) was required to allow the linguistic generalisations of the annotation algorithm to be optimised. Information encoded in the treebank trees which was not being harnessed by the algorithm (McCarthy, 2003) had to be identified and appropriate extensions made, e.g. using the level of SBAR attachment within NPs to disambiguate between relative clauses (↑RELMOD=↓) and clausal complements (↑COMP=↓). The full inventory of Penn-II null elements had to be reviewed and the Traces module extended, e.g. *ICH* (Interpret Constituent Here) and *RNR* (Right Node Raising) nodes were ignored by the annotation algorithm of McCarthy (2003). Section 3.4 presents a review of the annotation guidelines and the changes I made to the

annotation algorithm as a result of this review.

2.5.5 Evaluation

The annotation algorithm of McCarthy (2003) generated basic f-structures for core phenomena and achieved near complete coverage of Penn-II. It is important to maintain a high standard of annotation quality as well as coverage while correcting and extending the annotation algorithm to produce a more fine-grained f-structure analysis. Annotation quality was evaluated against the DCU 105 by McCarthy (2003). There are a number of problems with evaluating against a gold standard of this size, most notably that of overfitting. There is a risk of assuming that the gold standard is a complete and balanced representation of the linguistic phenomena in a language and of basing design decisions on this. It is preferable to evaluate against more extensive, independently constructed gold standards. Although the DCU 105 is publicly available, larger well-established external gold standards provide more widely recognised benchmarks against which annotation quality can be evaluated. Chapters 4 and 5 evaluate the improved and extended annotation algorithm against the PARC 700 and PropBank, respectively.

2.6 Summary

This chapter has introduced LFG and outlined some previous approaches to the automatic acquisition of LFG f-structures from treebank trees. McCarthy (2003) describes the linguistic basis for an automatic f-structure annotation algorithm which scales to provide basic f-structures with almost complete coverage of Penn-II. A review of the algorithm was provided and important areas for the further development, correction and extension of the annotation algorithm were identified in Section 2.5. Chapter 3 describes the extension and correction of the annotation algorithm in line with the requirements listed in Sections 2.5.1 to 2.5.4. Chapters 4 and 5 pursue the external evaluation motivated in Section 2.5.5.

Chapter 3

Extended and Revised Automatic F-Structure Annotation Algorithm

3.1 Introduction

This thesis presents an automatic f-structure annotation algorithm which is a core component of a larger project (Burke et al., 2004b) for the automatic acquisition of high quality LFG lexicon and grammar resources. Chapter 2 describes the original, basic annotation algorithm of McCarthy (2003). This chapter describes an extensive overhaul, further development, extension and evaluation of this annotation algorithm. The corrections and extensions presented in this chapter improve the quality of the lexicon and grammar resources acquired using the annotation algorithm.

McCarthy (2003) describes the linguistic basis for an early version of the algorithm to provide basic f-structures with almost complete coverage of Penn-II. However, the original implementation of the algorithm is inefficient which significantly slows the further development, extension, testing and evaluation of the algorithm and negatively impacts on the performance of the parsing technology (Cahill et al., 2004b) which incorporates the algorithm. In order to improve on this situation, core components were re-written. In addition, the original annotation algorithm modules was corrected and extended to provide a more fine-grained and standardised f-structure analysis. A review of the Penn-II annotation guidelines (Bies et al., 1995) was performed to identify linguistic information

encoded in the treebank trees which is not being harnessed by the original algorithm. A complete review of the DCU 105 gold standard was performed which informed design decisions in the algorithm development process.

Section 3.2 describes practical changes to the implementation of the annotation algorithm which significantly improve performance and allow for efficient development, testing and evaluation cycles, as well as improved parse speed for parsers employing the annotation algorithm in the pipeline architecture of Cahill et al. (2004b) (Chapter 6). Section 3.3 outlines corrections and extensions to the original procedures encoded in the algorithm modules of McCarthy (2003). Section 3.5 reports on the extensive review of the DCU 105 gold standard. Section 3.4 details amendments to the algorithm resulting from a review of the Penn-II annotation guidelines (Bies et al., 1995). Section 3.6 provides a quantitative and qualitative evaluation of the f-structures produced by the annotation algorithm. The algorithm achieves an f-score of 96.93% for all grammatical functions and 94.28% for preds-only against the DCU 105 gold standard. Section 3.7 summarises the chapter.

3.2 Practical Considerations

The original annotation algorithm of McCarthy (2003) was very slow, taking over 30 minutes for the annotation of Penn-II and the generation of f-structure equations to be passed to the constraint solver. In order to achieve acceptable development and testing turnaround cycles, the implementation of the algorithm was made significantly more efficient. This allowed the necessary corrections and extensions to be made and resulted in high quality, more fine-grained f-structures being produced. Improvements in efficiency also provide better performance from the parsing technology incorporating the annotation algorithm (Cahill et al., 2004b).

3.2.1 Separating Data from Processing Procedures

The annotation algorithm consists of linguistic data (left-right context annotation matrices, head-lexicalisation rules, lexical macros, etc.) and annotation procedures which employ this linguistic data and other information (e.g. Penn-II null elements). In the new version of the algorithm, the linguistic data has been separated from the annotation

procedures to provide greater modularity and crucially to ensure that the linguistic data is only loaded into memory once for each annotation session. In the original algorithm of McCarthy (2003) some of the linguistic data was being unnecessarily re-loaded for each tree.

3.2.2 Processing F-Descriptions

Implementation efficiency was further improved by changing the procedure for producing f-descriptions for resolution from annotated trees. When the annotation of a tree is fragmented, the algorithm must generate separate f-descriptions for each fragment. Unannotated nodes result in multiple f-structure fragments for one Penn-II tree. For each tree, the algorithm of McCarthy (2003) produced one string containing all the f-descriptions for that tree. Square brackets were placed around the f-descriptions representing the annotations on descendants of unannotated nodes. The string of f-descriptions was then post-processed to check for square brackets and to isolate each embedded f-description. The post-processed string contained a series of separated f-descriptions for each fragment.

Instead of creating one single string of embedded f-description fragments for each tree and then post-processing this string to separate each f-description fragment, the new algorithm keeps a list of all unannotated nodes in each tree as they are met. After the main f-description string is generated, separate strings of f-description fragments are created as necessary for each unannotated node. This removes a large amount of unnecessary string processing, a task which is relatively inefficient in Java. Furthermore, a redundant phase of numerically ordering the equations within each f-description was removed from the original algorithm.

3.2.3 Speed-Up

Further minor optimisations were made throughout the code to improve efficiency and to remove redundant code. Overall, the changes reduced the processing time from over 30 minutes to less than 5 minutes for the annotation of Penn-II.

The availability of more powerful computers with greater memory capacity has since reduced the processing time for the annotation of Penn-II by the original algorithm of

McCarthy (2003) to 10 minutes. The subsequent sections of this chapter describe several extensions to the algorithm which have added to the processing required for the annotation of each tree. Despite this increased complexity, the annotation of Penn-II has been reduced to 2.5 minutes due to the additional memory capacity and processing speed, amounting to a 4-fold reduction in processing time compared to the original, less complex annotation algorithm.

3.3 Improving Existing Annotation Procedures

The original algorithm of McCarthy (2003) required several corrections and extensions to allow more fine-grained and standardised f-structures to be produced. This section outlines the most significant changes:

- Annotating subjects of co-ordinated verb phrases.
- Capturing SUBJ re-entrancies into XCOMPs for infinitival clauses.
- Introducing a more standardised treatment of oblique agents.
- Extending the TOPICREL analysis to *wh*-less relative clauses.
- Implementing an effective analysis of apposition.

3.3.1 Subjects of Co-ordinated VPs

Co-ordinated verb phrases are annotated as elements of a COORD set. The orignal Co-ordination module annotates the shared subject locally and does not percolate it into the co-ordinated elements. This is particularly problematic for the extraction of probabilistic lexical resources (O'Donovan et al., 2004, 2005a) as incorrect subcategorisation frames will be extracted for verbs occurring in co-ordinate structures. Therefore, the annotation algorithm has been extended to overcome this problem. All verbal co-ordinated elements, i.e. nodes with verbal Penn-II categories and the annotation $\downarrow \in \uparrow$COORD, are given the annotation $\uparrow$SUBJ=$\downarrow$SUBJ which percolates the shared subject into each of the co-ordinated elements.

Figure 3.1 provides the Penn-II-style tree for the sentence *Sony learned lessons and fired him* annotated using the extended annotation algorithm. The VP nodes representing the phrases *learned lessons* and *fired him* are identified as co-ordinated elements by the Co-ordination module. The extended algorithm adds the annotation ↑SUBJ=↓SUBJ to both nodes. The f-structures produced by the algorithm of McCarthy (2003) and the extended algorithm are provided. The extended algorithm allows the subcategorisation frame extraction algorithm (O'Donovan et al., 2004, 2005a) to produce the frame *learn*<SUBJ, OBJ> instead of *learn*<OBJ> which is incorrect for this sentence.

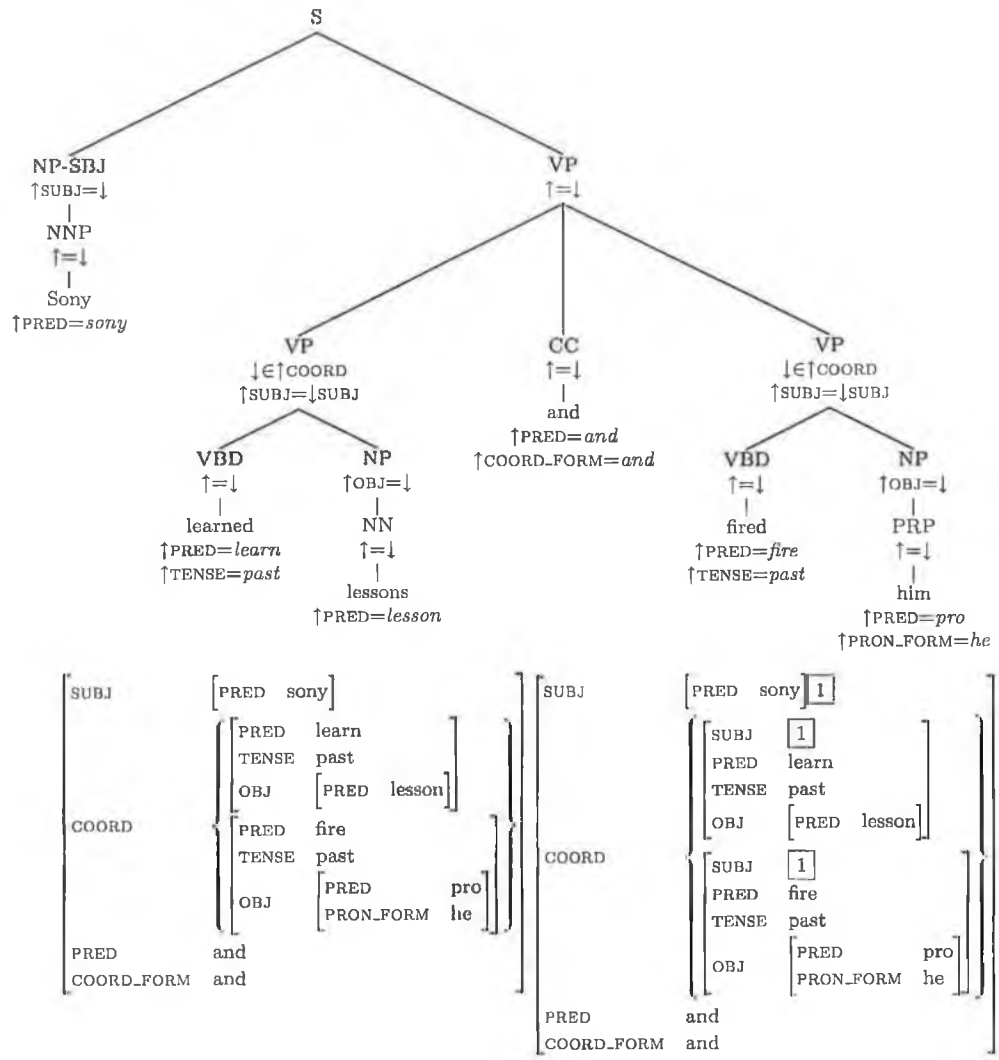


Figure 3.1: Automatically annotated Penn-II-style tree and f-structures for *Sony learned lessons and fired him*

Even though this is an improvement compared with the original annotation algorithm, it is not yet a "perfect" analysis: first, it still leaves a non-subcategorised SUBJ at the outermost level of the resulting f-structure and second, certain quantified (or rather indefinite) NPs should not be distributed into elements of a COORD set. Compare *A manager learned the lesson and fired him* ≠ *A manager learned the lesson and a manager fired him.* The first problem (the remaining unsubcategorised SUBJ at the outermost f-structure level) could be addressed in terms of an architecture which explicitly distinguishes between "distributable" and "non-distributable" grammatical functions and distributes the former into COORD set elements and removes them from their original position in the f-structure in a post-processing step (rather than in terms of explicit equations as is done in the current approach). The second problem is beyond the scope of the current dissertation. Our current solution (with the unsubcategorised SUBJ) is punished in the gold standard evaluations in terms of reduced precision scores. Furthermore, our current approach needs to be extended to non-SUBJ arguments and non-VP co-ordination. This is handled to some degree by extensions to the algorithm to process Right Node Raising (Section 3.4.3, pp. 43).

3.3.2 Re-entrant XCOMP Subjects

The annotation algorithm of McCarthy (2003) captured the re-entrant subjects of XCOMPs in the Left-Right Context Annotation module by annotating VP complements within VPs with the equations ↑XCOMP=↓ and ↑SUBJ=↓SUBJ. However, some complements within VPs (e.g. infinitival clauses) are tagged S. Penn-II tags the subjects of these clauses using null NP-SBJ nodes which are co-indexed with the subject or object of the matrix clause. The information available to the Left-Right Context Annotation module for the annotation of a node (node category, parent node category and context of the node relative to the head daughter) is insufficient to correctly capture the subject re-entrancy into these S clauses. It is impossible to tell whether the null subject of the complement is re-entrant with the subject or object of the matrix clause. Therefore, S complements within VPs are annotated ↑XCOMP=↓ with no attempt made to annotate the subject from within the Left-Right Context Annotation module. The Traces module should, but does not in the

original algorithm, attempt to capture the re-entrancy. The null NP-SBJ node of the S complement is annotated ↑SUBJ=↓ by default due to the -SBJ functional tag. This SUBJ annotation is not instantiated as there is no other f-structure information associated with the null NP-SBJ node.

Figure 3.2 provides a Penn-II-style tree and corresponding f-structure for *you have to recognize* annotated using the algorithm of McCarthy (2003). The S complement is annotated ↑XCOMP=↓ by the Left-Right Context Annotation module. The null NP-SBJ node is annotated ↑SUBJ=↓ by the default annotations for the -SBJ functional tag. No attempt is made to capture the re-entrancy indicated by the co-indexation between the nodes NP-SBJ-1 and *-1. As there is no further f-structure information associated with the null NP-SBJ node or its descendants, the value of the SUBJ function is not instantiated at f-structure level. Note that, again, failure to generate an f-structure with the required re-entrancies leads to incorrect subcategorisation frame extraction results for the predicates in question.

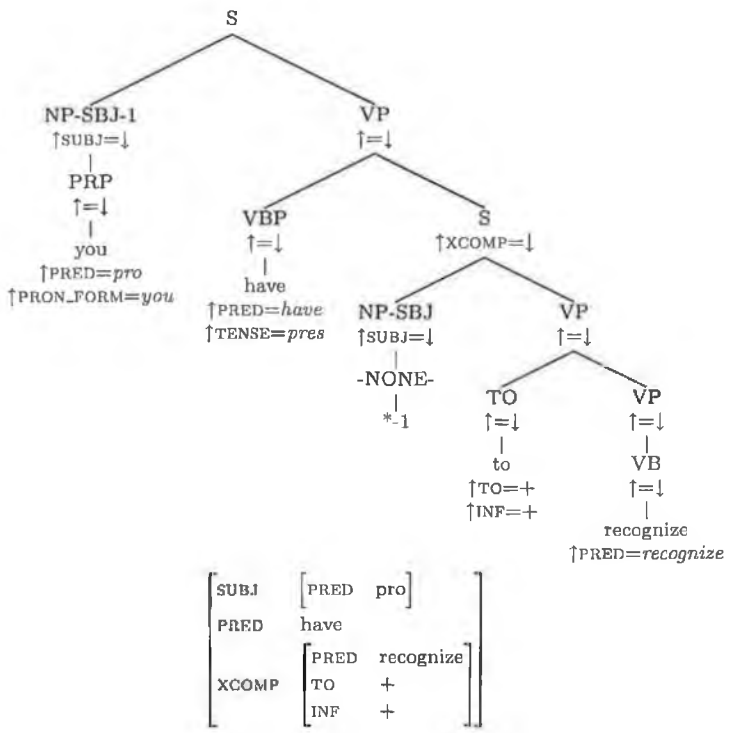


Figure 3.2: Automatically annotated Penn-II-style tree and f-structure for *you have to recognize* (McCarthy, 2003)

In order to capture the missing re-entrancy, I extended the original Traces module to properly capture re-entrancies into XCOMPs for S complements within VPs. The null NP-SBJ node is annotated with the f-structure information of the co-indexed subject or object of the matrix clause. This is achieved by unifying a variable associated with the co-indexed node in the matrix clause (e.g. *F2*) with the null NP-SBJ node using the equation $\downarrow$=*F2* on the NP-SBJ node.

Figure 3.3 provides a Penn-II-style tree for *ordered him to pay* annotated using the extended algorithm. The co-indexation indicates that the object of the matrix clause is re-entrant with the subject of the XCOMP clause. The variable associated with the NP-1 node is *F4* and is indicated in Figure 3.3 by the equation $\downarrow$=*F4*. The extended algorithm annotates the null NP-SBJ node with this equation to achieve the re-entrancy indicated in the resulting f-structure. The same procedure is applied to the tree of Figure 3.2. The null NP-SBJ node is annotated to unify with it the f-structure information associated with the subject of the matrix clause as required.

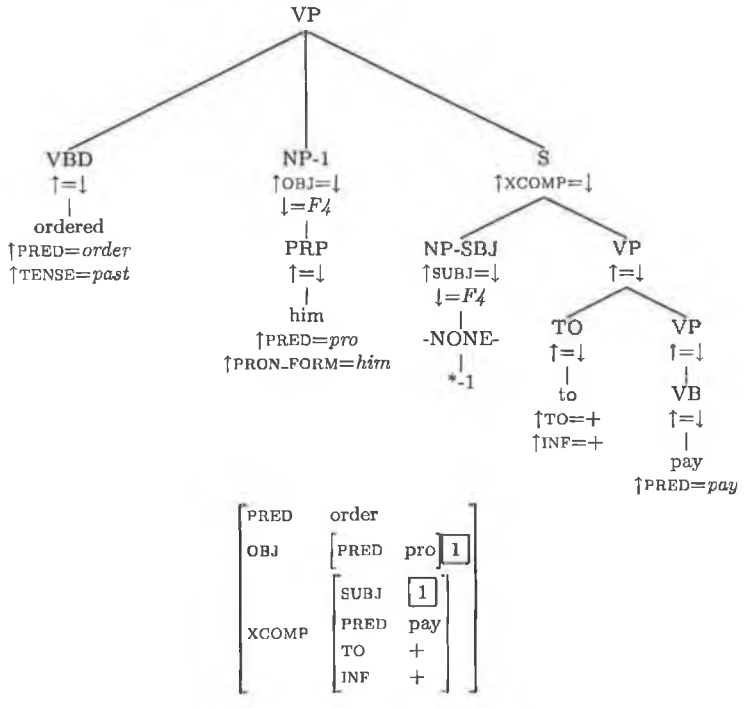


Figure 3.3: Automatically annotated Penn-II-style tree and f-structure for *ordered him to pay* (revised annotation algorithm)

3.3.3 Apposition

The head-lexicalisation rules of Magerman (1994) indicate that the most likely head candidate of an NP is the rightmost nominal. This procedure would be incorrect in cases of apposition, e.g. the NP *director* would wrongly be marked as the head of the phrase *Gerry Purdy, director* (Figure 3.4). McCarthy (2003) extends the head-lexicalisation process for NPs to mark the rightmost nominal *not immediately preceded by a comma* as the head. This identifies *Purdy* as the head of the example NP. However, previous preliminary attempts by McCarthy (2003) to annotate apposition do not succeed and the NP *director* is simply annotated as an element of the adjunct set as shown in the f-structure to the left of Figure 3.4. The simplest approach to the proper annotation of apposition is through the left-right context annotation matrices. The matrices were changed so that NPs occurring to the right of the head within an NP are annotated ↓∈↑APP. In the example of Figure 3.4, the head-lexicalisation rules correctly identify the first NP as the head (because the second NP is preceded by a comma). Therefore, the second NP lies to the right of the head and is annotated ↓∈↑APP using the corrected Left-Right Context Annotation module.

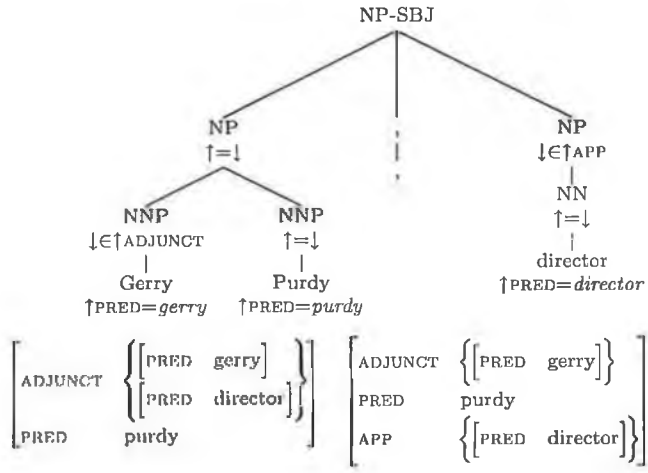


Figure 3.4: Automatically annotated Penn-II-style tree and f-structures for *Gerry Purdy, director of marketing*

3.3.4 *Wh*-less relative clauses

The Traces module handles the annotation of relative clauses and other LDDs. However, relative clauses introduced by null complementizers (*wh*-less relative clauses) are not properly annotated by the algorithm of McCarthy (2003). The algorithm should produce a TOPICREL annotation which is re-entrant with another grammatical function within the relative clause, but both functions are absent from the resulting f-structures as in the original algorithm no f-structure information is provided for the null complementizer. In order to address this problem, the Traces module has been extended to annotate null complementizers with the equations ↓PRED=*pro* and ↓PRON_FORM=*null*. This extension captures the LDD producing the desired f-structure.

Figure 3.5 provides the Penn-II-style tree for *smelters the company operated* annotated using the extended algorithm. The null complementizer (WHNP-1 → -NONE- → 0) is annotated with the equations ↓PRED=*pro* and ↓PRON_FORM=*null* as described above. The f-structures produced by the algorithm of McCarthy (2003) and the extended algorithm are provided. The extended algorithm correctly produces TOPICREL and OBJ at the RELMOD f-structure level. As with the percolation of subjects into co-ordinated VP (Section 3.3.1), this extension improves the subcategorisation frames extracted by O'Donovan et al. (2004, 2005a).

3.3.5 Oblique Agents

Oblique agents were annotated by the algorithm of McCarthy (2003) as adjuncts. Noun phrases representing logical subjects are tagged in Penn-II with the -LGS functional tag. McCarthy (2003) annotated the embedded NP representing the logical subject with the equation ↑LGS=+. This annotation was the only means of identifying the adjunct as the logical subject. A more standardised oblique agent analysis is now used. Figure 3.6 provides a Penn-II-style tree for the phrase *made by Rowe* with the f-structures produced by the algorithm of McCarthy (2003) and the new analysis. The OBL_AG feature has been introduced to represent the logical subject. The non-standard LGS feature has been removed from the noun phrase.

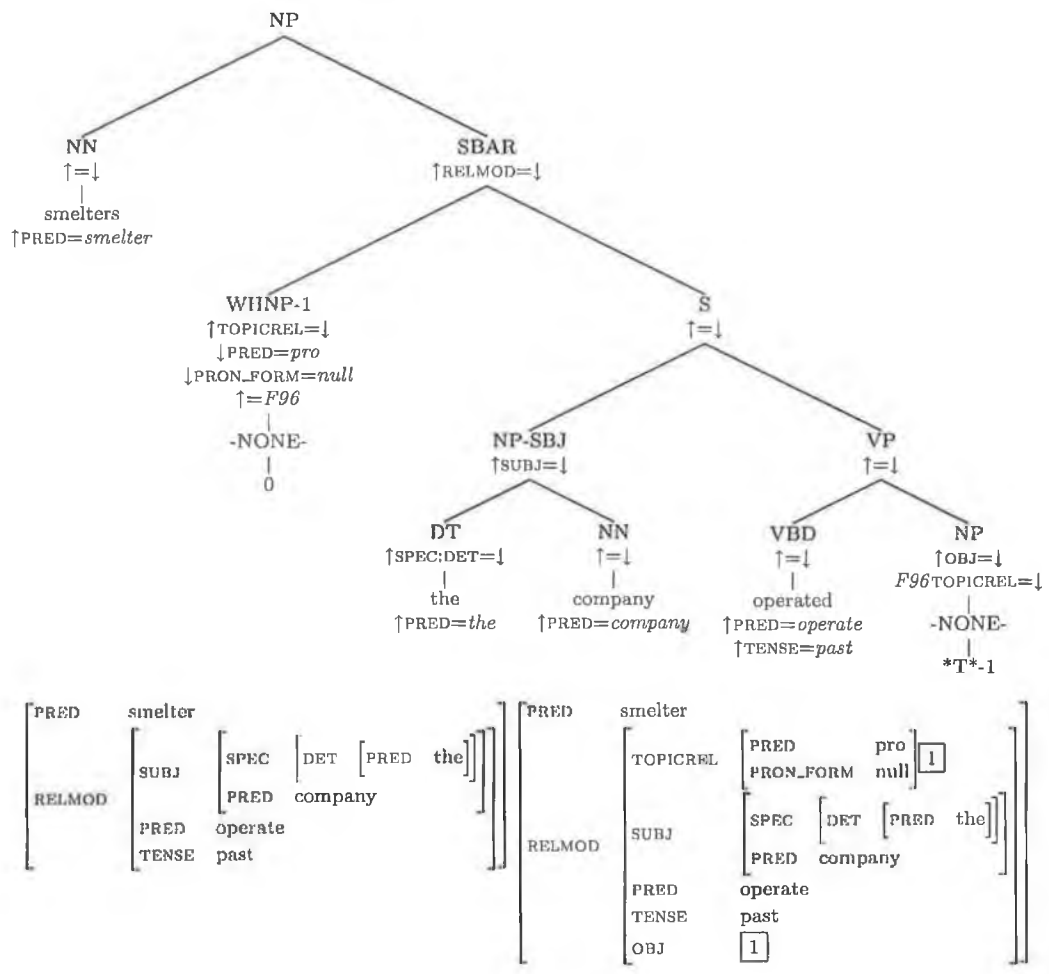


Figure 3.5: Automatically annotated Penn-II-style tree and f-structures for *smelters the company operated*

3.4 Review of Annotation Guidelines (Bies et al., 1995)

Bies et al. (1995) provide extensive guidelines and policies which underpin the bracketing and annotation of Penn-II. I have reviewed these guidelines to verify that the existing annotation algorithm modules are implemented in accordance with Penn-II conventions and also to identify any linguistic information encoded in Penn-II which is not being leveraged by the original algorithm of McCarthy (2003). This section describes the most important changes made to the annotation algorithm as a result of this review.

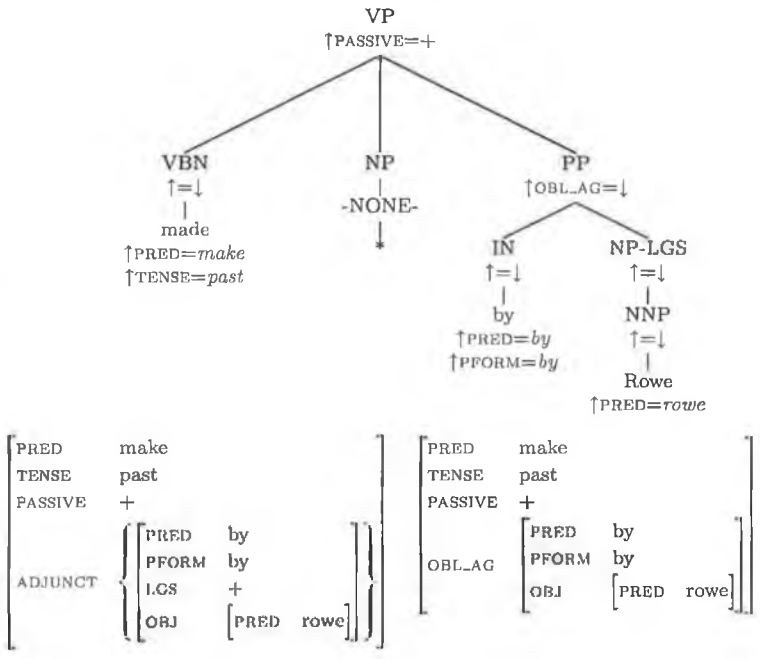


Figure 3.6: Automatically annotated Penn-II-style trees and f-structures for *made by Rowe*

3.4.1 Clausal complements of NPs

In Penn-II, relative clauses and clausal complements are both tagged as SBARs occurring to the right of the head within NPs. Bies et al. (1995) indicate that the level of SBAR attachment is used to distinguish between these NP modifiers. SBARs representing clausal complements are attached at the same level as the POS tag of the head noun. NPs containing relative clauses group the head noun and any pre-modifiers as an embedded NP. The SBAR representing the relative clause is then attached at the same level as the embedded NP. The annotation algorithm of McCarthy (2003) does not use this level of attachment information. The Left-Right Context Annotation module annotates all SBARs occurring to the right of the head within NPs with the equation ↑RELMOD=↓. This annotation is correct for relative clauses but is incorrect for clausal complements which should be annotated ↑COMP=↓.

I have incorporated attachment information from Bies et al. (1995) to correctly annotate clausal complements. The level of attachment of a node is not available to the Left-Right Context Annotation module, so the necessary changes were made in the Catch-All

and Clean-Up module. If the head of the NP is a preterminal POS tag, then the default ↑RELMOD=↓ annotation provided by the Left-Right Context Annotation module is changed to ↑COMP=↓. Otherwise, the relative clause analysis is left unchanged.

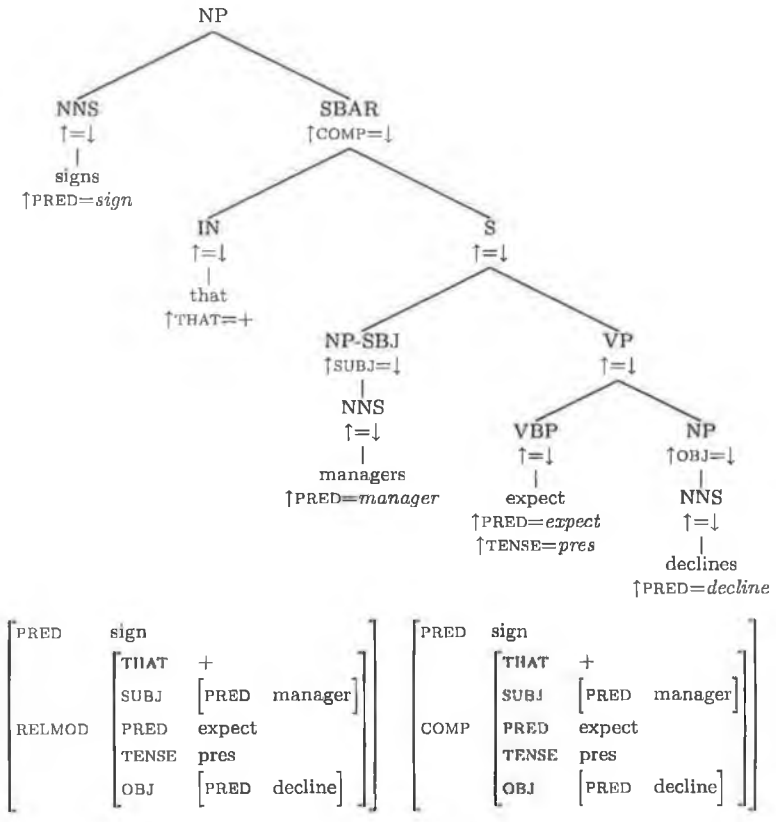


Figure 3.7: Automatically annotated Penn-II-style tree and f-structures for *signs that managers expect declines*

Figure 3.7 provides the Penn-II-style tree for *signs that managers expect declines* annotated using the corrected algorithm. The default ↑RELMOD=↓ annotation provided by the Left-Right Context Annotation module is changed to ↑COMP=↓ as the head of the NP is a POS tag (NNS) and not an embedded phrasal NP. The incorrect RELMOD analysis is produced by the algorithm of McCarthy (2003) as indicated by the f-structure on the left in Figure 3.7. The f-structure generated by the revised algorithm with the correct COMP analysis is provided on the right of Figure 3.7. Figure 3.8 provides the Penn-II-style tree for *houses that use coal.* The head of the NP is an embedded NP which indicates that the SBAR is a relative clause. Therefore, the default annotation is left unchanged.

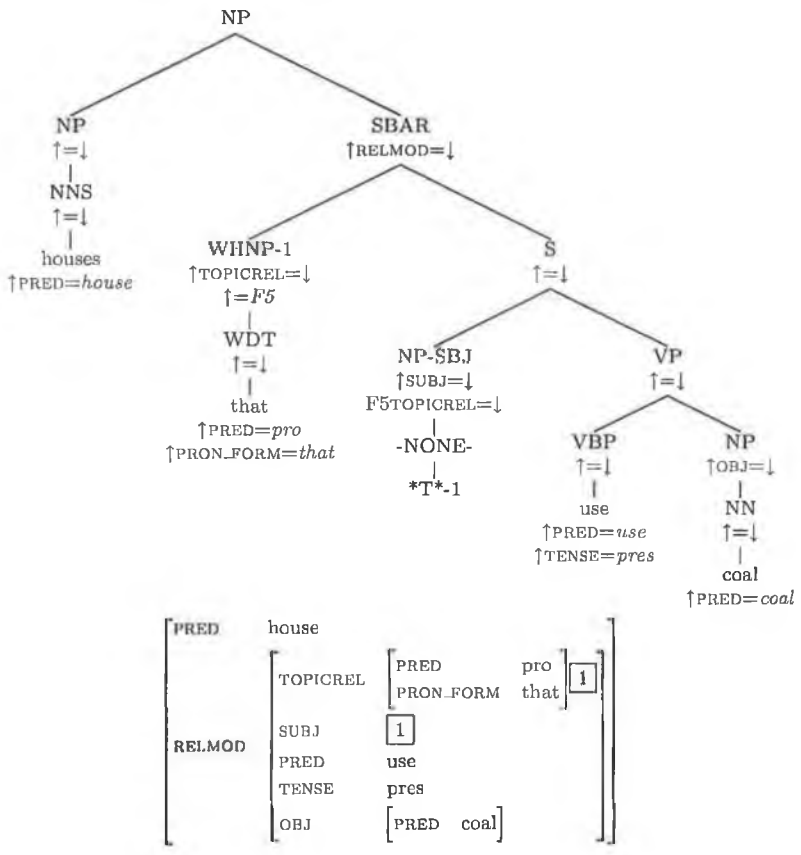


Figure 3.8: Automatically annotated Penn-II-style tree and f-structure for ***houses that use coal***

3.4.2 *ICH* - Interpret Constituent Here

Bies et al. (1995) provide an inventory of Penn-II null elements which includes *ICH* - "Interpret Constituent Here". Co-indexed *ICH* nodes are used when intervening material splits a constituent into two parts. The Traces module of the annotation algorithm should use this information to reconstruct the split constituent at f-structure level. The algorithm of McCarthy (2003) ignores *ICH* null elements and both parts of the constituent are interpreted locally resulting in an incorrect split analysis at f-structure level. I have extended the Traces module to handle *ICH* nodes correctly.

Lexical macros for each POS tag provide f-structure information for each non-null tree node. Null elements have no POS tag, so the lexical macros cannot provide local f-structure information for those nodes. Therefore, the Left-Right Context Annotation

module does not annotate the parent or grandparent nodes of null elements as there is no local f-structure information to instantiate the ↓ meta-variable of any annotation it could provide. The Traces module annotates the grandparent or parent nodes of a null element if it can unify the ↓ meta-variable with the f-structure information of another, usually co-indexed, node. The extended Traces module determines the index of the *ICH* node and locates the co-indexed second part of the split constituent. This node will have an annotation previously provided by the Left-Right Context Annotation module which must be deleted. Instead, the node is annotated to identify it with a new variable, e.g. $X1{=}\downarrow$. Then, the Left-Right Context Annotation module is invoked to provide an annotation for the grandparent of the *ICH* node. The ↓ meta-variable in this annotation is replaced with the new variable identifying the second part of the split constituent. This process reconstructs the split constituent at f-structure level.

Figure 3.9 provides the Penn-II-style tree for *heard testimony today about Jones* annotated using the extended algorithm. The *ICH* node is indexed 1. The extended algorithm locates PP-1, the second part of the split constituent. The Left-Right Context Annotation module initially provides the annotation $\downarrow{\in}\uparrow$ADJUNCT for this node and no annotation for PP, the grandparent of the *ICH* node. The algorithm of McCarthy (2003) ignores *ICH* nodes so these annotations remain producing the f-structure provided on the left of Figure 3.9. This incorrectly attaches *about Jones* as an adjunct of *hear* instead of *testimony* as both parts of the split constituent are interpreted locally. The extended algorithm deletes the annotation provided by the Left-Right Context Annotation module for PP-1 and replaces it with the equation $X1{=}\downarrow$, which identifies a new variable with that node. The Left-Right Context Annotation module is invoked providing the annotation $\downarrow{\in}\uparrow$ADJUNCT for the PP dominating *ICH*-1. The ↓ meta-variable in this annotation is replaced with the new variable *X1*. This reconstructs the split constituent as shown in the resulting f-structure on the right on Figure 3.9.

3.4.3 *RNR* - Right Node Raising

The inventory of Penn-II null elements provided by Bies et al. (1995) also includes *RNR* - "Right Node Raising". *RNR* nodes occur in pairs and indicate that a co-indexed non-

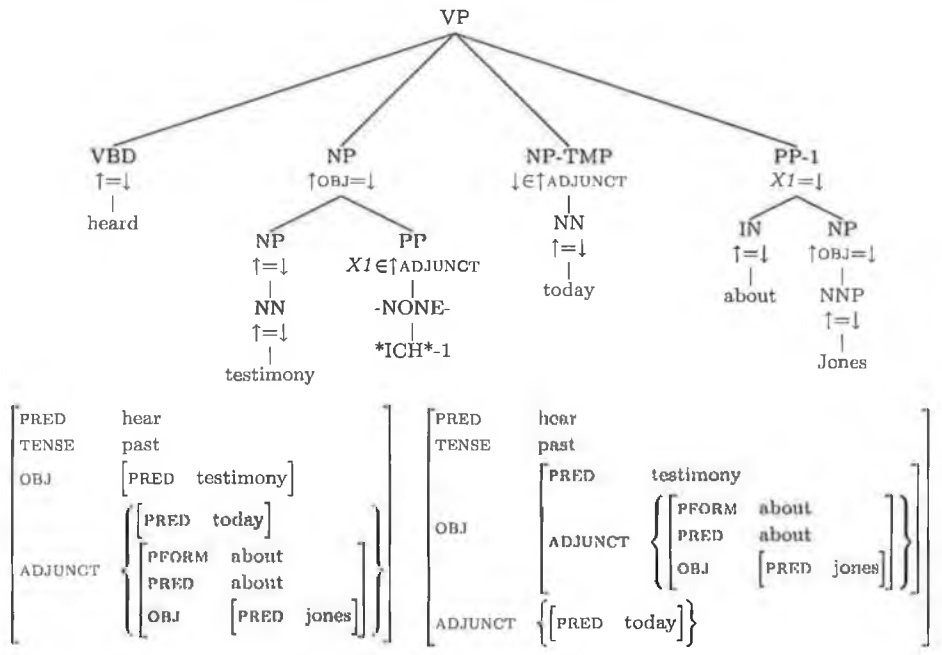


Figure 3.9: Automatically annotated Penn-II-style tree and f-structure for *heard testimony today about Jones.*

null node should be interpreted in more than one place. The annotation algorithm should locate the co-indexed constituent and ensure that the f-structure information associated with it is interpreted at the correct levels of attachment in the generated f-structure. The algorithm of McCarthy (2003) ignores *RNR* nodes and incorrectly interprets the co-indexed constituent locally as a result. I have extended the algorithm to handle *RNR* nodes by using the techniques described in Section 3.4.2 for *ICH* nodes.

Figure 3.10 provides the Penn-II-style tree for *She asked for and received refunds* annotated using the extended algorithm. Both *RNR* nodes are indexed 1. The co-indexed NP-1 is located and is then provided with a new annotation to identify it with the variable *X2*. The Left-Right Context Annotation module is invoked to provide annotations for the grandparent nodes of both *RNR* nodes. The ↓ meta-variables of both annotations are replaced by the variable *X2* resulting in the annotation ↑OBJ=*X2* for both nodes. The f-structure on the right of Figure 3.10 provides the resulting f-structure which correctly interprets *refunds* in the locations indicated by the *RNR* nodes. This f-structure features a further example of co-ordinated VPs with the subjects correctly percolated into

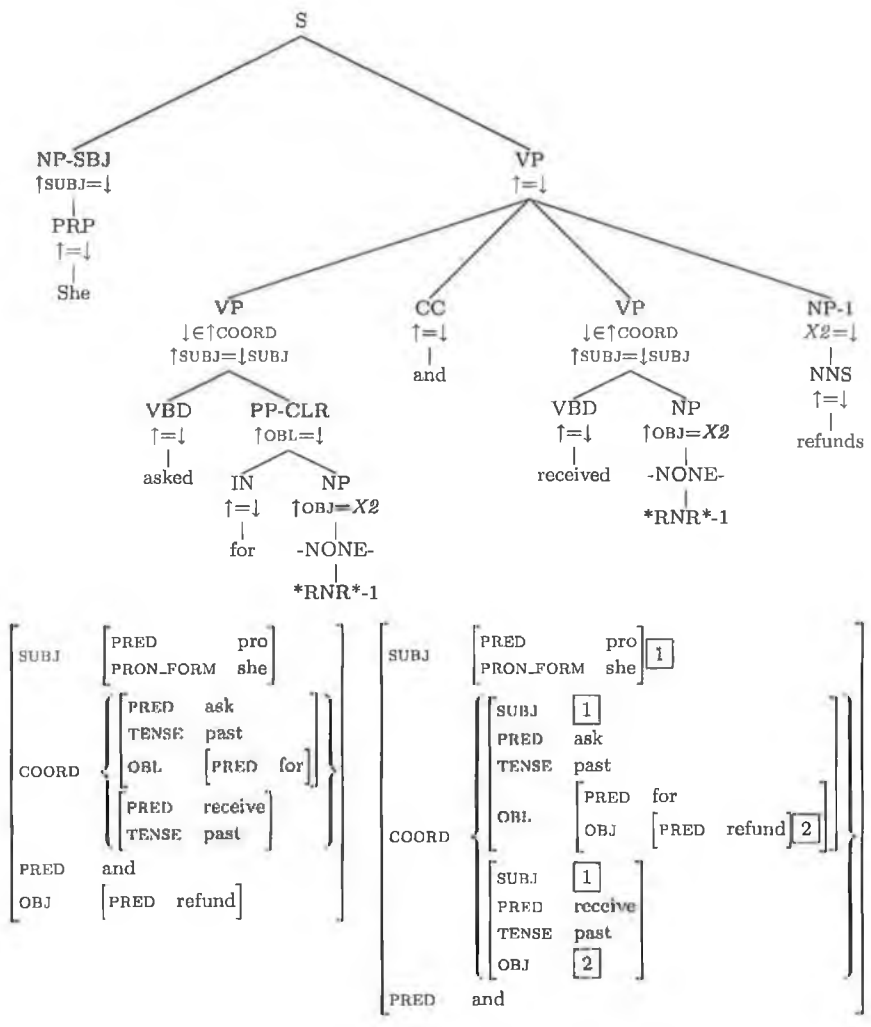


Figure 3.10: Automatically annotated Penn-II-style tree and resulting f-structure for *She asked for and received refunds.*

the co-ordinated elements (Section 3.3.1). The annotation algorithm of McCarthy (2003) produces the f-structure provided on the left of Figure 3.10. As the *RNR* null elements are ignored and the subject is not percolated into the co-ordinated elements, both the subject and object of the verb *receive* are not present in its local f-structure.

3.5 Review of DCU 105

An extensive manual review of the DCU 105 gold standard was performed to produce a more standardised, fine-grained analysis and to correct errors. The new analysis mirrors

Category	DCU 105 of McCarthy (2003)	Revised DCU 105
PRP (pronoun)	↑PRED=*headword*	↑PRED=*pro*, ↑PRON_FORM=*headword*
PRP$ (possessive pronoun)	↑PRED=*pro*, ↑CASE=*gen*, ↑WH=-	↑PRED=*pro*, ↑PRON_FORM=*headword*
WP (*wh* pronoun)	↑PRED=*pro*, ↑WH=+	↑PRED=*pro*, ↑PRON_FORM=*headword*
WP$ (*wh* possessive pronoun)	↑PRED=*pro*, ↑CASE=*gen*, ↑WH=+	↑PRED=*pro*, ↑PRON_FORM=*headword*

Table 3.1: Changes to pronoun analysis in DCU 105

several of the improvements to the annotation algorithm outlined in this chapter. The main changes are to the analysis of pronouns, set annotations and oblique agents with the features WH, LGS, TO, INF, CASE and CONJ no longer being used in the gold standard.

3.5.1 Pronouns

The feature PRON_FORM has been added to the gold standard for the analysis of pronouns (Table 3.1). All pronouns are now annotated ↑PRED=*pro* and ↑PRON_FORM=*headword*. In the DCU 105 of McCarthy (2003), possessive pronouns were annotated ↑PRED=*pro* and ↑CASE=*gen*, with the additional annotations ↑WH=+ or ↑WH=- used to indicate *wh* and non-*wh* pronouns, respectively. The PRON_FORM feature is used to ensure that all pronouns are annotated with the head word. Possessive and *wh* pronouns were not annotated with the head word in the DCU 105 of McCarthy (2003). All CASE and WH annotations have been removed.

3.5.2 Corrections to Set Annotations

Inconsistencies in the annotation of co-ordination and adjunct sets have been corrected to produce a more standardised analysis. In most cases, elements of co-ordination sets were analysed ↓∈↑CONJ in the DCU 105 of McCarthy (2003). However, occurrences of the equations ↑CONJ1=↓ and ↑CONJ2=↓ were also present. The feature COORD is now used instead of CONJ and all elements of co-ordination sets are now analysed as ↓∈↑COORD. The gold standard of McCarthy (2003) analysed the conjunct with the equation ↑PRED=*headword* in all cases. This analysis has been extended to add the equation ↑COORD_FORM=*headword*. The analysis of adjunct sets (↓∈↑ADJUNCT) has been maintained. However, several inconsistent analyses, e.g. ↑ADJUNCT=↓, have been removed.

3.5.3 Oblique Agents

Section 3.3.5 introduces changes to the annotation algorithm to standardise the annotation of oblique agents. Corresponding changes have been made to the DCU 105. The feature OBL_AG has been introduced for the analysis of oblique agents, replacing the original ADJUNCT analysis of the *by* prepositional phrase. The DCU 105 of McCarthy (2003) used the feature LGS (from the Penn-II -LGS functional tag) to represent the logical subject and this was the only indicator of the presence of an oblique agent. All occurrences of LGS have been removed from the DCU 105 as the new OBL_AG feature is sufficient.

3.5.4 Further Miscellaneous Changes

Further miscellaneous changes to the DCU 105 include:

- the conflation of the equations $\uparrow$TO=+ and $\uparrow$INF=+ for infinitives to $\uparrow$TO_INF=+.
- singular number now being analysed as $\uparrow$NUM=*sg* instead of $\uparrow$NUM=*sing*.
- possessives being analysed as $\uparrow$POSS=$\downarrow$ which replaces $\uparrow$POS=$\downarrow$.
- numerical noun modifiers being annotated $\uparrow$SPEC:QUANT=$\downarrow$ which replaces the inconsistent use of both $\uparrow$SPEC:ADJUNCT=$\downarrow$ and $\downarrow\in\uparrow$ADJUNCT.
- *wh*-less relative clauses now receiving the annotation $\downarrow$PRED=*pro* and $\downarrow$PRON_FORM=*null* (cf. Section 3.3.4).

3.6 Evaluation

3.6.1 Quantative Evaluation

All Penn-II trees (excluding trees with FRAG and X nodes) were annotated and the resulting f-structure equations were resolved. While the annotation algorithm of McCarthy (2003) already provided near complete coverage of Penn-II, a quantitative evaluation of the f-structures generated by the extended algorithm shows that coverage has been improved further. A single covering and connected f-structure is produced for 99.8% of all Penn-II trees (99.41% for McCarthy (2003)). No f-structures are produced for 45 trees (0.09%) due

to feature clashes which is an improvement (0.47% for McCarthy (2003)). Unannotated nodes resulted in two separate f-structure fragments being generated for 50 trees (0.103%).

# F-structure fragments	# Trees	% Treebank
0	45	0.093
1	48329	99.804
2	50	0.103

Table 3.2: Quantitative F-Structure Evaluation

3.6.2 Qualitative Evaluation

The Penn-II trees for the 105 gold standard sentences were automatically annotated and the quality of the resulting f-structures was evaluated against the DCU 105 gold standard f-structures using the methodology and software of Crouch et al. (2002) and Riezler et al. (2002). Table 3.3 provides the results of the evaluation for all grammatical functions and for preds-only in terms of precision, recall and f-score for each relation. The overall f-score is 96.93% for all grammatical functions and 94.28% for preds-only against the revised DCU 105 (Section 3.5) which are an improvement on the results of McCarthy (2003): f-scores of 94.11% and 90.86% for all grammatical functions and preds-only, respectively, against the original, coarse-grained, DCU 105.

Table 3.3 shows that the results are very high for the core grammatical functions, e.g. f-scores of 96% and 97% are achieved for SUBJ and OBJ, respectively. Correctly annotating SUBJ re-entrancies into XCOMPs and percolating SUBJ annotations into co-ordinated VPs has contributed to the high f-score for SUBJ. Although a relatively low f-score (82%) is achieved for apposition, this is a significant improvement as the algorithm of McCarthy (2003) did not correctly identify any of the 19 occurrences of apposition in the DCU 105. McCarthy (2003) achieved an f-score of 100% for two separate infinitival relations TO and INF. These relations have been conflated to TO_INF slightly reducing the overall f-score of the extended algorithm.

	Precision	Recall	F-Score
ADEGREE	11/12 = 92	11/12 = 92	92
ADJUNCT	669/716 = 93	669/714 = 94	94
APP	14/19 = 74	14/15 = 93	82
COMP	60/62 = 97	60/74 = 81	88
COORD	101/106 = 95	101/111 = 91	93
COORD_FORM	51/52 = 98	51/57 = 89	94
DET	196/196 = 100	196/197 = 99	100
FOCUS	1/1 = 100	1/1 = 100	100
IF	3/3 = 100	3/3 = 100	100
MODAL	22/22 = 100	22/22 = 100	100
NUM	836/836 = 100	836/836 = 100	100
OBJ	336/346 = 97	336/345 = 97	97
OBJ2	1/1 = 100	1/2 = 50	67
OBL	47/50 = 94	47/55 = 85	90
OBL2	2/2 = 100	2/2 = 100	100
OBL_AG	11/11 = 100	11/11 = 100	100
PART	7/7 = 100	7/9 = 78	88
PARTICIPLE	31/31 = 100	31/31 = 100	100
PASSIVE	66/66 = 100	66/71 = 93	96
PERS	836/836 = 100	836/836 = 100	100
POSS	48/50 = 96	48/52 = 92	94
PRON_FORM	94/95 = 99	94/94 = 100	99
QUANT	29/46 = 63	29/42 = 69	66
RELMOD	38/43 = 88	38/41 = 93	90
SUBJ	366/387 = 95	366/378 = 97	96
TENSE	241/241 = 100	241/241 = 100	100
THAT	17/17 = 100	17/18 = 94	97
TO_INF	32/32 = 100	32/32 = 100	100
TOPIC	12/12 = 100	12/13 = 92	96
TOPICREL	38/41 = 93	38/43 = 88	90
XCOMP	141/150 = 94	141/143 = 99	96
Overall	97.06	96.80	96.93
Preds-only	94.28	94.28	94.28

Table 3.3: Results by feature name of qualitative evaluation against the DCU 105

3.7 Summary

This chapter has presented an extensive overhaul, further development, extension and evaluation of the automatic f-structure annotation algorithm. Significant improvements have been made to the efficiency of the algorithm implementation enabling quicker development, testing and evaluation turnaround cycles and also improving the performance of the parsing technology which incorporates the annotation algorithm (Cahill et al., 2004b). Improvements to the existing annotation algorithm modules include the annotation of oblique agents and apposition. An extensive review of the DCU 105 gold standard was performed. The main changes to the annotation algorithm resulting from a review of the Penn-II annotation guidelines (Bies et al., 1995) have been presented. A quantitative and qualitative evaluation of the f-structures produced by the annotation algorithm has been performed. The algorithm achieves an f-score of 96.93% for all grammatical functions and

94.28% for preds-only against the DCU 105 gold standard.

Chapter 4

Evaluation of the Automatic F-Structure Annotation Algorithm against the PARC 700 Dependency Bank

4.1 Introduction

This thesis presents an automatic f-structure annotation algorithm which is a core component of a larger project (Burke et al., 2004b) for the automatic acquisition of high quality LFG lexicon and grammar resources. Chapter 2 describes the original, basic annotation algorithm of McCarthy (2003), while Chapter 3 corrects and extends this algorithm, providing an evaluation against the DCU 105, a gold standard consisting of f-structures for 105 randomly selected trees from WSJ Section 23 of Penn-II. There are a number of problems with evaluating against a gold standard of this size, most notably that of overfitting. There is a risk of assuming that the gold standard is a complete and balanced representation of the linguistic phenomena in a language and of basing design decisions on this. It is preferable to evaluate against a more extensive, independently constructed standard.

To overcome these difficulties with the DCU 105 evaluation, this chapter presents an evaluation of the automatic f-structure annotation algorithm against the PARC 700 De-

pendency Bank (King et al., 2003).[1] The PARC 700 is a larger, well-established, external gold standard which provides a more widely recognised benchmark against which annotation quality can be evaluated. This chapter presents conversion software to overcome systematic differences in linguistic analysis between the DCU 105 and PARC 700 representations. Importantly, this work can also be applied for the evaluation of the parsing technology of Cahill et al. (2004b) against the PARC 700. Furthermore, the work presented in this chapter allows the annotation algorithm to produce PARC 700-style dependencies, as well as DCU 105-style f-structures, for the entire Penn-II treebank.

Section 4.2 provides an overview of the PARC 700 and presents some of the systematic differences between the DCU 105 and PARC 700 representations. Section 4.3 describes each component of the conversion software which was designed to map the automatically acquired f-structures to overcome the systematic differences in representation and allow a fair evaluation against the PARC 700. Section 4.4 presents and analyses the results of the evaluation process. The automatically acquired and mapped f-structures achieve an f-score of 87.33% against the PARC 700 test set for the feature set of Kaplan et al. (2004). Section 4.5 summarises the chapter. An earlier version of this work has been published as Burke et al. (2004a).

4.2 The PARC 700 Dependency Bank

The PARC 700 Dependency Bank consists of dependency structures for 700 randomly selected sentences from Section 23 of the WSJ section of Penn-II. These sentences were automatically parsed by a hand-coded, deep LFG grammar of English using the XLE system (Maxwell and Kaplan, 1993). In cases where multiple parses were generated the best parse was manually chosen. The f-structures of the best parses were then automatically converted to dependency format (triples) and extended. The dependencies were manually examined and corrected by two independent reviewers.

The evaluation presented in this chapter replicates the experimental setup of Kaplan et al. (2004), with the PARC 700 divided into the same 140-sentence development set and 560-sentence test set. The set of features (Table 4.1) evaluated in the experiment form a

[1] Available from `http://www2.parc.com/istl/groups/nltt/fsbank/gold1-700-files.tar.Z`.

proper superset of preds-only and a proper subset of all grammatical functions (preds-only $\subset$ PARC $\subset$ all GFs). This feature set was selected in Kaplan et al. (2004) because the features carry important semantic information.

ADEGREE	degree of adjectives, adverbs, i.e. comparative, positive or superlative
ADJUNCT	adjuncts
AQUANT	adjectival quantifiers
COORD_FORM	form of a co-ordinating conjunct, e.g. *and*
COMP	complement clauses
CONJ	conjuncts in co-ordinate structures
DET_FORM	determiner forms, e.g. *the*
FOCUS_INT	fronted elements in interrogatives
MOD	noun-noun modifiers
NUM	number of nouns, e.g. singular (sg)
NUMBER	numbers modifying nouns
NUMBER_TYPE	type of a number phrase, i.e. cardinal or ordinal
OBJ	objects
OBJ_THETA	secondary objects
OBL	oblique
OBL_AG	demoted subject of a passive
OBL_COMPAR	comparative *than/as* clauses
PASSIVE	passive verb, e.g. *It was eaten*
PERF	perfective verb, e.g. *have eaten*
POSS	possessives, e.g. *John's book*
PRECOORD_FORM	*either, neither*
PROG	progressive verb, e.g. *were eating*
PRON_FORM	form of a pronoun, e.g. *she*
PRON_INT	interrogative pronouns
PRON_REL	relative pronouns
PROPER	type for proper nouns, e.g. name, location
PRT_FORM	particle in a particle verb, e.g. *They threw it out*
QUANT	quantifiers, e.g. *all*
STMT_TYPE	statement type, e.g. declarative
SUBORD_FORM	subordinating conjunction, e.g. *that*
TENSE	tense of a verb, e.g. past
TOPIC_REL	fronted element in relative clauses
XCOMP	non-finite complements, verbal and small clauses

Table 4.1: PARC 700 evaluation feature set of Kaplan et al. (2004)

Figure 4.1 displays as an AVM the PARC 700 dependency structure for the sentence *The principal-only securities will be repackaged by BT Securities into a Freddie Mac Remic, Series 103, that will have six classes.* The dependency structure was filtered using the PARC 700 evaluation feature set (Table 4.1). A comparison of this structure with the f-structure acquired by the annotation algorithm for the same sentence (Figure 4.2) highlights the five main classes of systematic differences between the DCU 105 and PARC 700 representations. The five classes are listed below and discussed in detail in Section 4.3.

- **Multi-Word Expressions** The f-structure annotation algorithm analyses the internal structure of all noun phrases fully, e.g. the noun phrase *Freddie Mac Remic* is represented as a PRED value *remic* modified by two ADJUNCTs, *freddie* and *mac.*

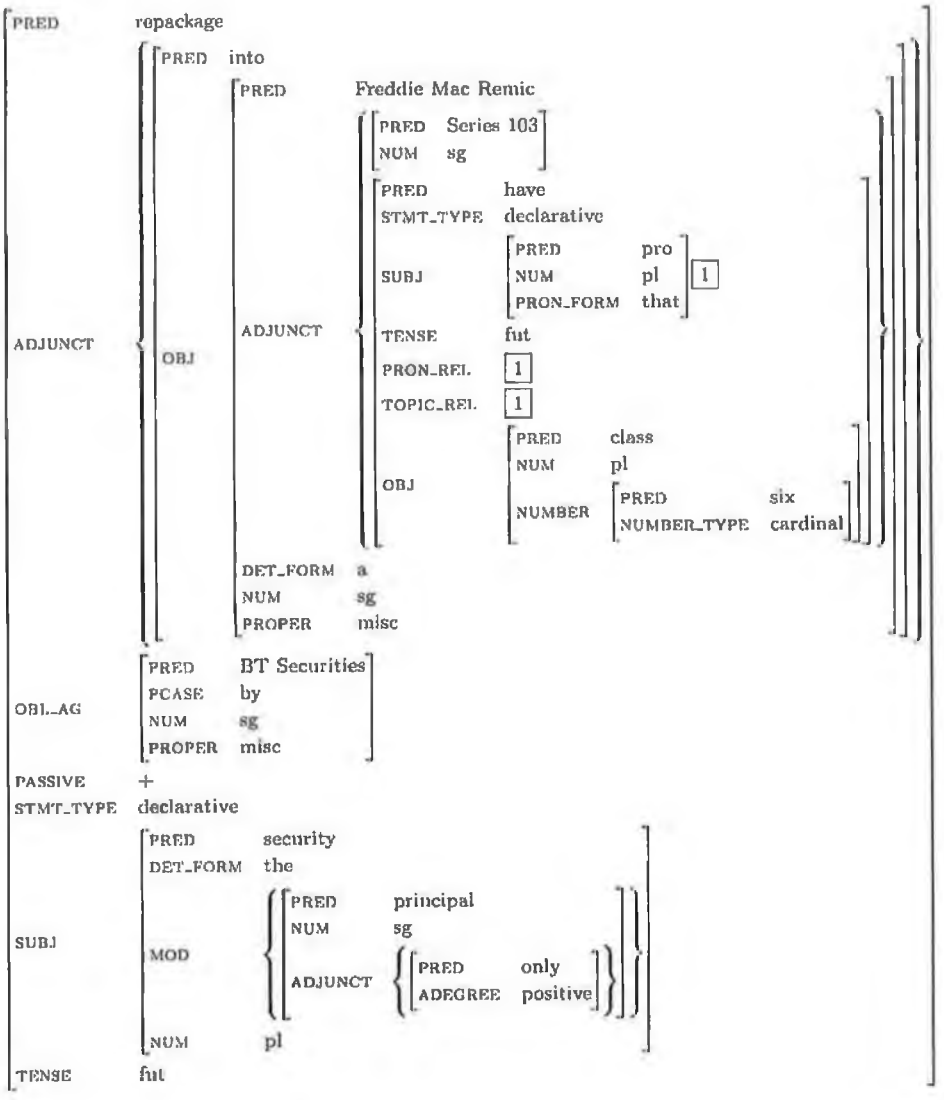


Figure 4.1: PARC 700 dependency structure, displayed as an AVM, for the sentence: *The principal-only securities will be repackaged by BT Securities into a Freddie Mac Remic, Series 103, that will have six classes.*

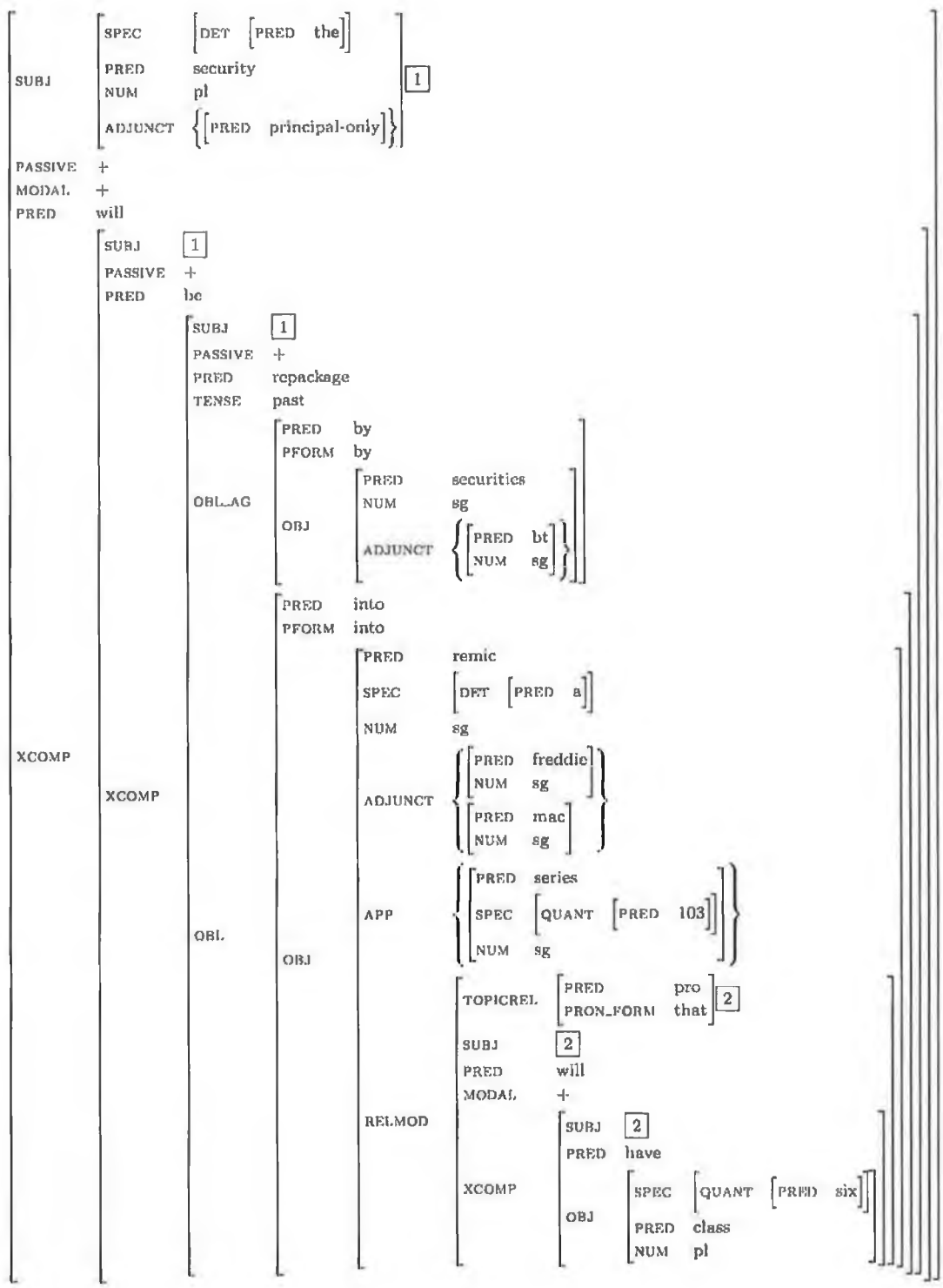


Figure 4.2: Automatically acquired f-structure for the sentence: *The principal-only securities will be repackaged by BT Securities into a Freddie Mac Remic, Series 103, that will have six classes.*

The PARC 700 analyses this and other named entities as multi-word expression predicates.

- **Feature Geometry** Although the phrase *by BT Securities* is analysed as an oblique agent (OBL_AG) in both representations, the internal feature geometry differs.

- **Feature Nomenclature** Determiners are annotated with the feature DET in the DCU 105 representation. The PARC 700 uses the DET_FORM feature.

- **Additional Features** The PARC 700 contains several features, e.g. STMT_TYPE and NUMBER_TYPE, which are not present in the DCU 105 analysis.

- **XCOMP Flattening** The representation of tense and aspect information differs greatly. The DCU 105 employs cascading XCOMPs to encode this information at f-structure level, while the same information is represented in the PARC 700 using a flat analysis with tense and aspect features. The automatically acquired f-structure of Figure 4.2 contains three XCOMPs, none of which are present in the PARC 700 dependency structure of Figure 4.1.

4.3 Conversion Software

4.3.1 Introduction

The task of evaluating the automatically acquired f-structures against the PARC 700 is non-trivial and time-consuming due to the systematic differences in linguistic analysis, feature geometry and nomenclature between the PARC 700 and the DCU 105 representations, as outlined in Section 4.2. This section presents the five modules of the conversion software (Figure 4.3) designed to overcome these systematic differences.

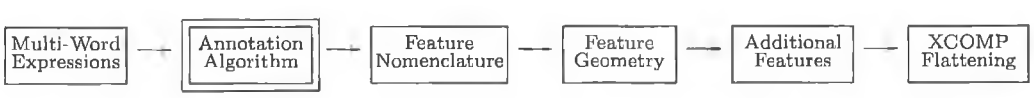


Figure 4.3: Conversion Software for mapping automatically annotated Penn-II trees from the DCU 105 analysis for evaluation against the PARC 700.

4.3.2 Multi-Word Expressions

Certain multi-word expressions, e.g. named entities, are treated as internally unanalysed units by the PARC 700, while the DCU 105 always fully analyses the internal structure of these strings. The *Multi-Word Expressions* pre-processing module identifies and tags multi-word expression predicates in Penn-II trees. Each tree is traversed in a top-down manner with the substrings represented at each subtree checked against a list of all multi-word expression predicates in the PARC 700. An NE (Named Entity) or MWE (other Multi-Word Expression) node is inserted as appropriate above the longest identified multi-word expression in each subtree. These nodes act as a cue for the annotation algorithm to produce the PARC 700 multi-word expression predicate analysis at f-structure level.

There are three cases for the insertion of an MWE or NE node:

- an entire subtree represents a multi-word expression predicate
- a partial subtree represents a multi-word expression predicate
- several subtrees represent a multi-word expression predicate

4.3.2.1 Entire subtree represents multi-word expression predicate

When an entire subtree is found to represent a multi-word expression predicate, a new node is inserted above all nodes in the subtree. Figure 4.4 illustrates the insertion of an NE node into the subtree representing *Freddie Mac Remic*. The f-structures automatically acquired by the annotation algorithm for both the original and the pre-processed trees are provided. The insertion of the NE node triggers the annotation algorithm to form the desired multi-word expression predicate *freddie mac remic*. No additional head rules or left/right context rules are required when an entire subtree represents a multi-word expression predicate as the inserted node will always be the head node and will have no nodes to its left or right.

4.3.2.2 Partial subtree represents multi-word expression predicate

When a partial subtree represents a multi-word expression predicate, a new node is inserted above the nodes representing the multi-word expression predicate only. For example, the

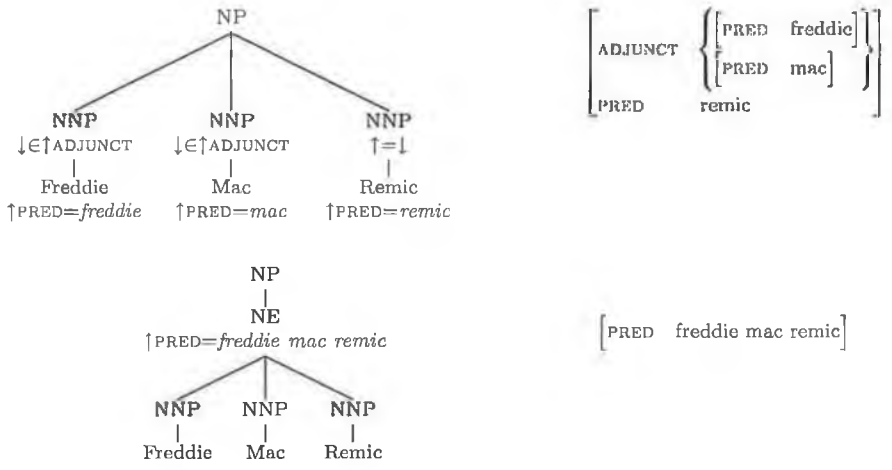


Figure 4.4: Entire subtree representing the multi-word expression predicate *Freddie Mac Remic*

named entity *White House* is contained within the Penn-II subtree representing the string *The official White House reaction.* The pre-processed subtree in Figure 4.5 shows the NE node inserted above the two nodes representing *White House.* A partial subtree representing a multi-word expression predicate can occur in the left or right context of the head node. As no entries exist in the left-right context annotation matrices for NE or MWE nodes, new entries had to be created. This task was trivial for NE nodes because the behaviour of NEs matches that of other nominal phrases for which left-right context entries already existed. The new entries were adapted from these existing nominal entries. These new entries allowed the inserted NE node in Figure 4.5 to be annotated (↓∈↑ADJUNCT).

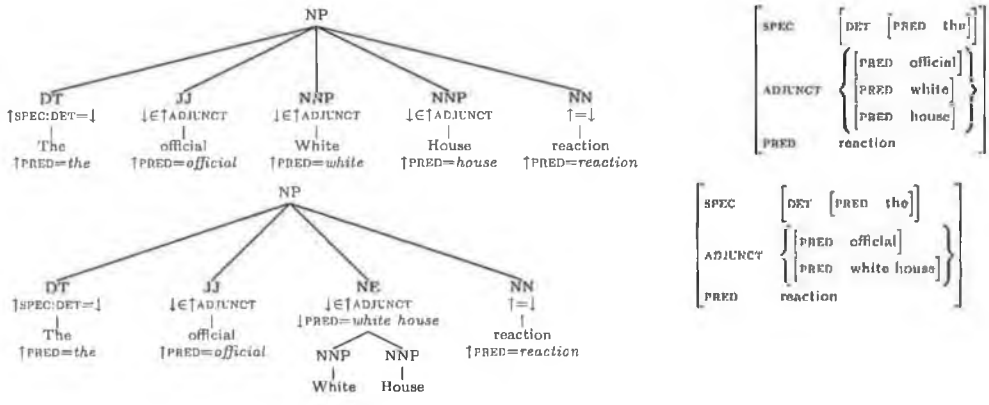


Figure 4.5: Partial subtree representing the multi-word expression predicate *White House*

The longest multi-word expression predicate contained within a subtree is always tagged, ignoring any shorter multi-word expression predicate substrings, e.g. in a subtree representing *New York Stock Exchange*, the entire subtree is tagged as a multi-word expression predicate ignoring the shorter multi-word expression predicate *New York*.

4.3.2.3 Several subtrees represent multi-word expression predicate

The final case of automatic node insertion occurs when a multi-word expression predicate is represented by several subtrees. In such subtrees, the deepest node governing the entire multi-word expression predicate is identified and all subordinate nodes excluding POS nodes are deleted, thus flattening the subtree. A new node is then inserted with the POS nodes representing the multi-word expression predicate as its daughters. Figure 4.6 provides the original tree for the noun phrase *The National Center for Education Statistics*, the pre-processed tree with the inserted NE node and the corresponding f-structures.

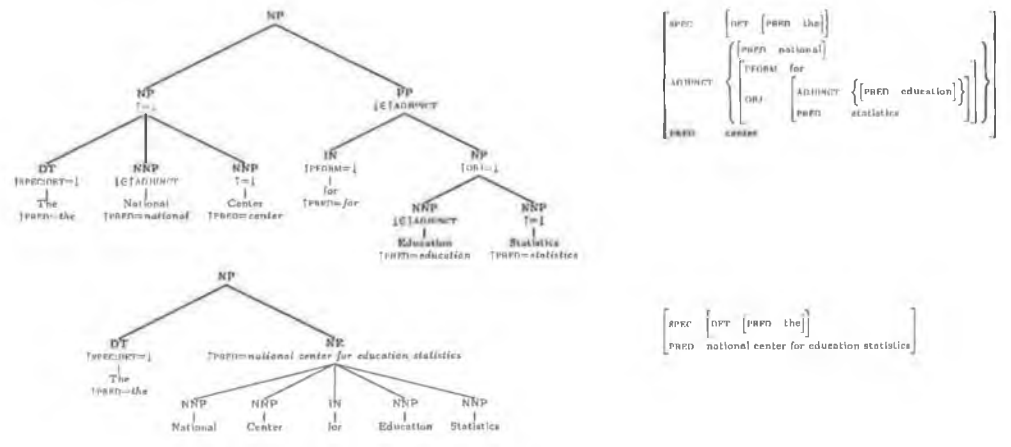


Figure 4.6: Several subtrees representing the multi-word expression predicate *National Center for Education Statistics*

4.3.3 Feature Nomenclature

The *Feature Nomenclature* module performs a straightforward mapping of the DCU 105 feature names for determiners, particles, co-ordinated elements and interrogatives to their PARC 700 equivalents (Table 4.2). This module originally mapped a larger number of features (Burke et al., 2004a), but the DCU 105 representation has since been adapted to

match the PARC 700 representation more closely, reducing the number of features which need to be mapped. Earlier versions of the conversion software (Burke et al., 2004a) also conflated several PARC 700 feature names to match the DCU 105 representation which did not make the same distinctions, e.g the PARC 700 features NUMBER, QUANT and AQUANT were conflated to QUANT. The PARC 700 evaluation feature set of Table 4.1 is now used fully for evaluation purposes without any conflation.

DCU 105	PARC 700
DET	DET_FORM
COORD	CONJ
FOCUS	FOCUS_INT
OBJ2	OBJ_THETA
PART	PRT_FORM

Table 4.2: Feature Nomenclature Mapping Table

4.3.4 Feature Geometry

The *Feature Geometry* module maps features which are common to both representations but with differing feature geometry. Oblique agents in the DCU 105 are analysed internally in the same manner as all other prepositional phrases, i.e. the preposition *by* receives PRED and PFORM annotations while the noun phrase is annotated ↑OBJ=↓. The PARC 700 uses a PCASE feature for *by* and omits the OBJ feature for the demoted subject of the passive clause. The *Feature Geometry* module maps the annotations ↑OBJ=↓ to ↑=↓ and replaces ↑PFORM=*by* and ↑PFORM=*by* with ↑PCASE=*by* in subtrees representing oblique agents. Figure 4.7 provides the original and mapped trees for the phrase *by BT Securities*. This example originates from the f-structures of Figures 4.1 and 4.2 and provides another instance of mapping by the *Multi-Word Expressions* module.

The PARC 700 provides an ADEGREE feature for adjectives and adverbs with three values: *comparative*, *positive* and *superlative*. While the DCU 105 feature set also includes ADEGREE, the value *positive*, which acts as the default ADEGREE value in the PARC 700, is not used. The constituent modified by the ADEGREE feature can also differ between both representations. Figure 4.8 shows the DCU 105 analysis of the phrase *less prolonged* and also the expected PARC 700 analysis. The DCU 105 annotates the adverb with the

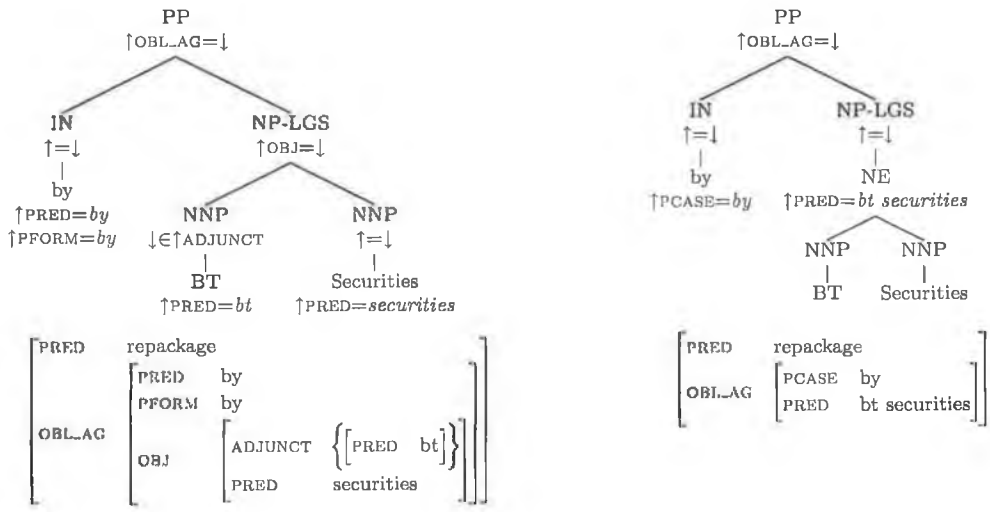


Figure 4.7: Feature geometry mapping for OBL_AG

ADEGREE feature, while the PARC 700 computes one ADEGREE feature which applies to the entire phrase.

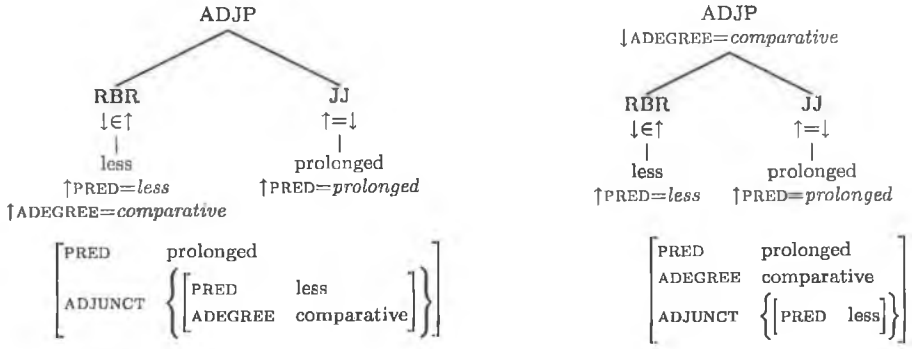


Figure 4.8: DCU 105 and PARC 700 ADEGREE analyses for the phrase *less prolonged*

This distinction complicates the task of calculating the *positive* ADEGREE value which was not previously present in the DCU 105 f-structures. Lexical macros in the annotation algorithm provide the annotation ↑ADEGREE=*comparative* for Penn-II POS tags JJR (comparative adjective) and RBR (comparative adverb), and ↑ADEGREE=*superlative* for JJS (superlative adjective) and RBS (superlative adverb). Simply creating lexical macros to annotate JJ (adjective) and RB (adverb) tags with ↑ADEGREE=*positive* will not pro-

duce the correct analysis, as shown by the incorrect f-structure in Figure 4.9 which would result for the phrase *less prolonged*.

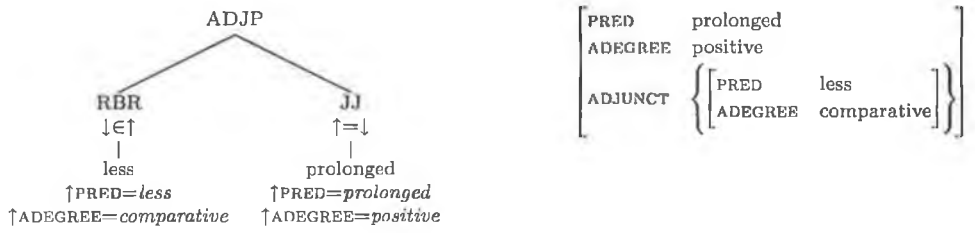


Figure 4.9: Incorrect mapping of ADEGREE *positive* for the phrase *less prolonged*

The first step towards achieving the desired ADEGREE analysis is to delete the annotations provided by the lexical macros for the tags JJR, JJS, RBR and RBS. Lexical macros were created for JJ and RB with the annotation ↑ADEGREE=*positive*. For phrases containing comparative or superlative adverbs or adjectives, all default ↑ADEGREE=*positive* annotations are deleted from within that phrase, and the parent node is annotated ↓ADEGREE=*comparative* or ↓ADEGREE=*superlative* as appropriate. This allows the desired PARC analysis of Figure 4.8 to be achieved for the phrase *less prolonged* instead of the incorrect f-structure of Figure 4.9.

4.3.5 Additional Features

The *Additional Features* module computes the following PARC 700 features which are not present in the DCU 105 analysis: AQUANT, MOD, NUMBER, NUMBER_TYPE, OBL_COMPAR, PRON_INT, PRON_REL, PROPER, STMT_TYPE and SUBORD_FORM.

4.3.5.1 AQUANT

The PARC 700 distinguishes adjectival quantifiers from adjectives and other quantifiers using the AQUANT feature. The DCU 105 analyses adjectival quantifiers as ADJUNCTs. The *Additional Features* module computes the AQUANT feature for the automatically annotated trees by mapping the annotation ↓∈↑ADJUNCT to ↑SPEC:AQUANT=↓ for JJ nodes occurring with the lemmas *many*, *more*, *most* and *several*. The ↑ADEGREE=*positive* annotation

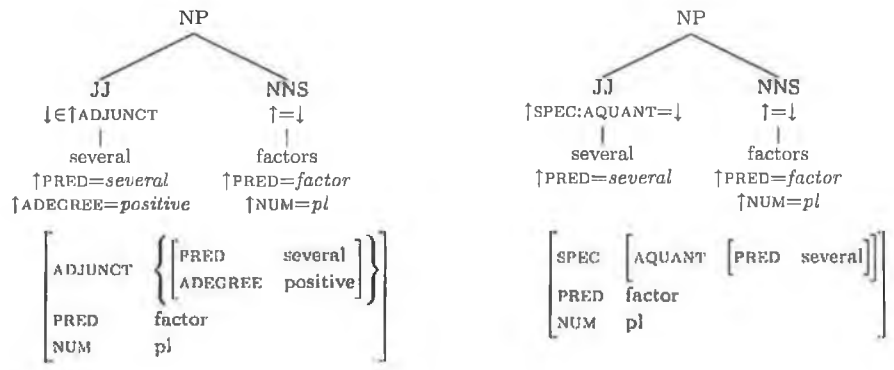


Figure 4.10: Computing AQUANT feature for the phrase *several factors*

which the *Feature Geometry* module adds for JJ nodes is deleted. An example of this mapping is provided in Figure 4.10 for the phrase *several factors*.

4.3.5.2 MOD

The MOD feature is used in the PARC 700 to analyse nominal modifiers within noun phrases. The DCU 105 treats nominal modifiers as ADJUNCTs. To achieve the PARC 700 analysis ↓∈↑ADJUNCT is mapped to ↓∈↑MOD for all nominal phrasal categories, nominal POS tags and named entities.

4.3.5.3 NUMBER

The PARC 700 analyses number modifiers within noun phrases using the NUMBER feature. The DCU 105 annotates cardinal numbers (CD) modifying NPs with ↑SPEC:QUANT=↓. This annotation is mapped to ↓∈↑NUMBER, for cardinal numbers modifying NPs only, to achieve the PARC 700 analysis. Ordinal numbers are tagged JJ in Penn-II and are annotated as ADJUNCTs in the DCU 105. A list of ordinal numbers is used to map the ADJUNCT annotations to the desired NUMBER analysis.

4.3.5.4 NUMBER_TYPE

The values of the PARC 700 NUMBER_TYPE feature are *cardinal* and *ordinal*. This feature is computed for the automatically annotated trees by adding the annotation ↑NUMBER_TYPE=*cardinal* to all CD nodes and by using the ordinal numbers list to provide

the annotation ↑NUMBER_TYPE=*ordinal*.

4.3.5.5 OBL_COMPAR

The PARC 700 feature OBL_COMPAR is used in comparative *than/as* clauses. The *Additional Features* module uses the ADEGREE feature to determine when the annotation ↑OBL_COMPAR=↓ should be added to the automatically annotated trees. The annotation is added to nodes headed by *than* or *as* and preceded in the local subtree by a node with a *comparative* ADEGREE value. Figure 4.11 provides the annotated tree and f-structure produced by the conversion software for the phrase *only modestly higher than normal*. The prepositional phrase *than normal* is annotated ↑OBL_COMPAR=↓ because it is headed by *than* and is preceded in the local subtree by a node (ADJP) which has a *comparative* ADEGREE value.

4.3.5.6 PRON_REL

The PARC 700 PRON_REL feature annotates the relative pronoun in relative clauses. In the automatically annotated trees, nodes annotated with the TOPIC_REL feature were also given a PRON_REL feature with the value *pro*. The relative pronoun is provided by the PRON_FORM feature which modifies this *pro* value.

4.3.5.7 PROPER

The PARC 700 annotates proper noun types using the PROPER feature with values *date*, *location*, *misc*, *name* and *title*. Lists of proper nouns are used to add this feature to the automatically annotated trees. A list of the days of the week and months are used to annotate *date*. The annotation ↑PROPER=*location* is provided using a list of countries, cities and US states. A list of common names and titles such as *Mr* trigger the annotation ↑PROPER=*name*. Noun phrases beginning with *Ambassador*, *Attorney*, *Director* or *Justice* are annotated as *titles*. The default value *misc* is applied to all *NE* nodes that have not received a PROPER annotation.

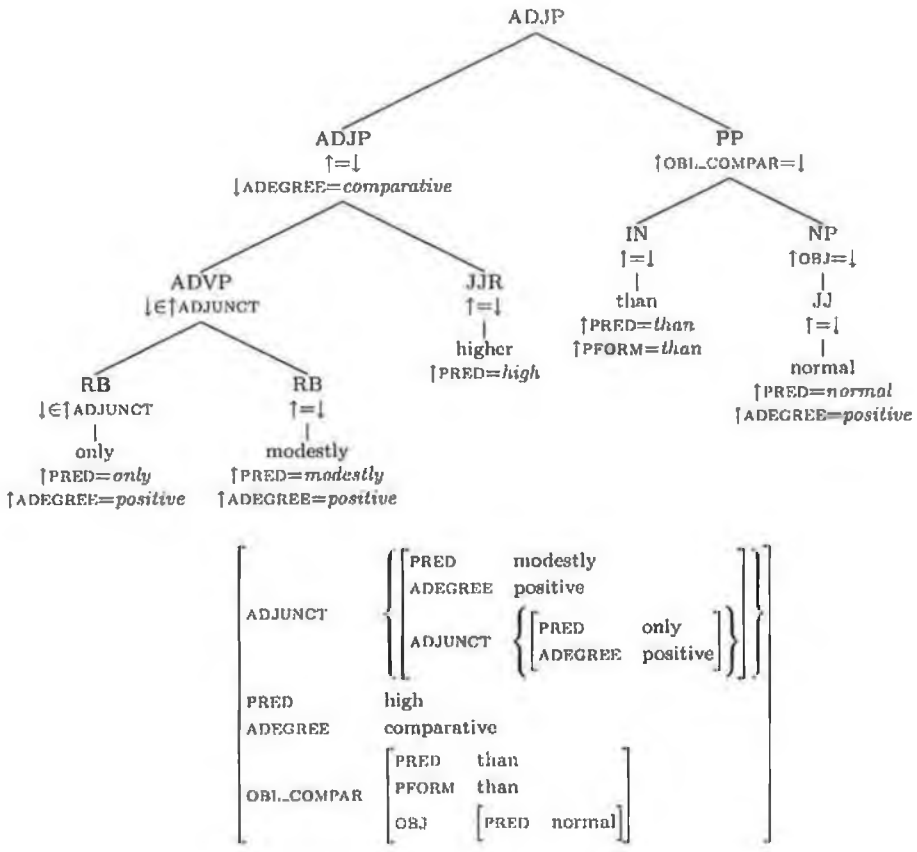


Figure 4.11: Computing OBL_COMPAR feature for the phrase *only modestly higher than normal*

4.3.5.8 STMT_TYPE

Two values of the PARC 700 statement type feature (STMT_TYPE) are computed by the *Additional Features* module: *declarative* and *header*. All finite sentential clauses (S), excluding those with the nominal (-NOM) and purpose (-PRP) functional tags, are annotated with the *declarative* statement type. Trees with NP as the root node are annotated ↓STMT_TYPE=*header*.

4.3.5.9 SUBORD_FORM

The PARC 700 SUBORD_FORM feature represents subordinating conjunctions. The DCU 105 analyses these constituents with the features IF, THAT and WHETHER which always have the value +. The annotations ↑THAT=+ and ↑WHETHER=+ are mapped to ↑SUBORD_FORM=*that* and ↑SUBORD_FORM=*whether* respectively. Subordinate clauses with no overt subordinating conjunction are annotated ↑SUBORD_FORM=*null*.

4.3.6 XCOMP Flattening

The most noticeable difference between the DCU 105 and PARC 700 analyses is the representation of tense and aspect information. Cascading XCOMPs encode this information in the DCU 105, while the PARC 700 uses a flat analysis with tense and aspect features. Figure 4.12 provides the automatically acquired f-structure and the desired PARC 700 analysis for the sentence *Unlike 1987, interest rates have been falling this year.*

The DCU 105 analysis introduces a new XCOMP AVM for both the auxiliaries *have* and *been*. The subject of the sentence (*interest rates*) is re-entrant as the subject of both XCOMPs. Each verb lemma receives a local PARTICIPLE or TENSE annotation provided by the annotation algorithm's lexical macros. In contrast, the PARC 700 analysis provides no XCOMP annotation and so there is no need for subject re-entrancies. The absence of XCOMPs allows the ADJUNCTs *Unlike 1987* and *this year* to be merged in one ADJUNCT set at sentence level. There is only one verbal PRED value in the f-structure, *fall*, and one corresponding TENSE feature. The information provided by the auxiliaries is encoded by the PROG(ressive) and PERF(ective) features.

The final post-processing module, *XCOMP Flattening*, implements a systematic map-

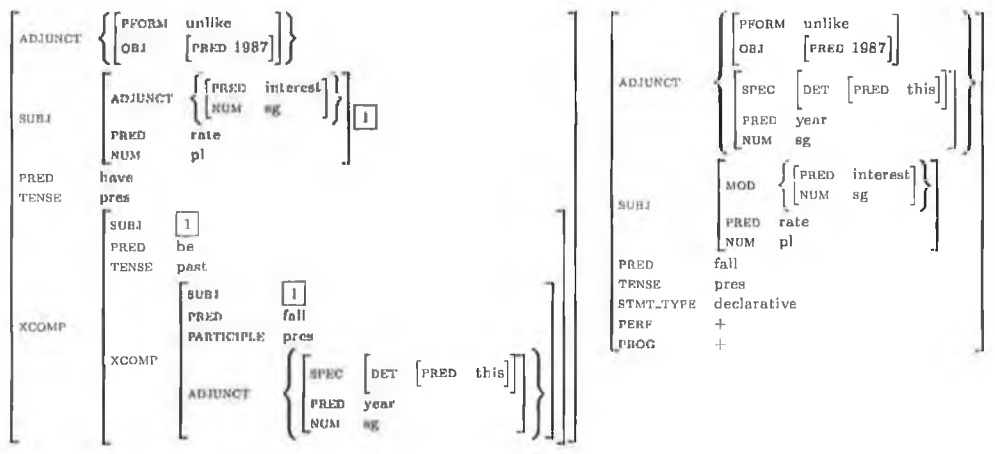


Figure 4.12: DCU 105 and PARC 700 analyses for the sentence *Unlike 1987, interest rates have been falling this year*

ping to overcome these differences in analysis. Figure 4.13 provides the unmapped automatically annotated Penn-II tree for the example sentence. These annotations produce the cascading XCOMP analysis at f-structure level (Figure 4.12).

The first step carried out by the *XCOMP Flattening* module is to delete auxiliary PRED annotations, while maintaining the PRED value of the main verb which is found at the deepest level of XCOMP embedding. The annotations ↑PRED=*have* and ↑PRED=*be* are removed from the example annotated tree in Figure 4.13.

Secondly, the TENSE annotations on all nodes except the first auxiliary are deleted. The annotation ↑TENSE=*past* on the lemma *be* in Figure 4.13 is removed.

Thirdly, the PARC 700 aspect features PROG and PERF are computed. Progressive aspect is represented in the automatically generated f-structures by the PARTICIPLE feature occurring with the value *pres*. The annotation ↑PARTICIPLE=*pres* is replaced by ↑PROG=+. Nodes representing the auxiliary lemma *have* are annotated ↑PERF=+. Both aspect features are added to the annotated tree in Figure 4.13.

The final step flattens the XCOMP cascade, while grouping and maintaining the adjuncts from each level. The XCOMP and subject re-entrancy annotations are removed from all nodes, except VP complements of modal verbs, and are replaced by ↑=↓. Removing all XCOMP annotations in this manner flattens the f-structure and groups the ADJUNCTs in larger sets.

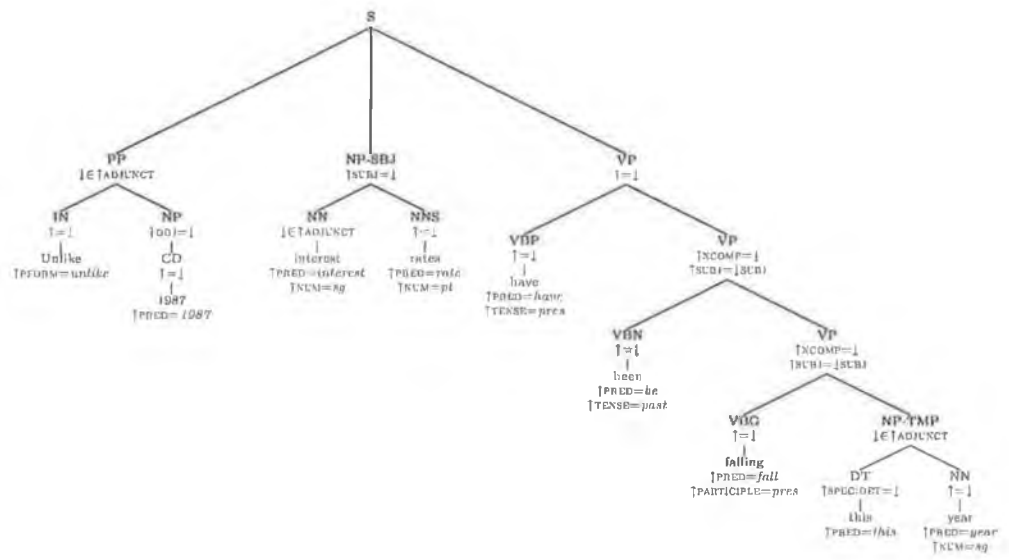


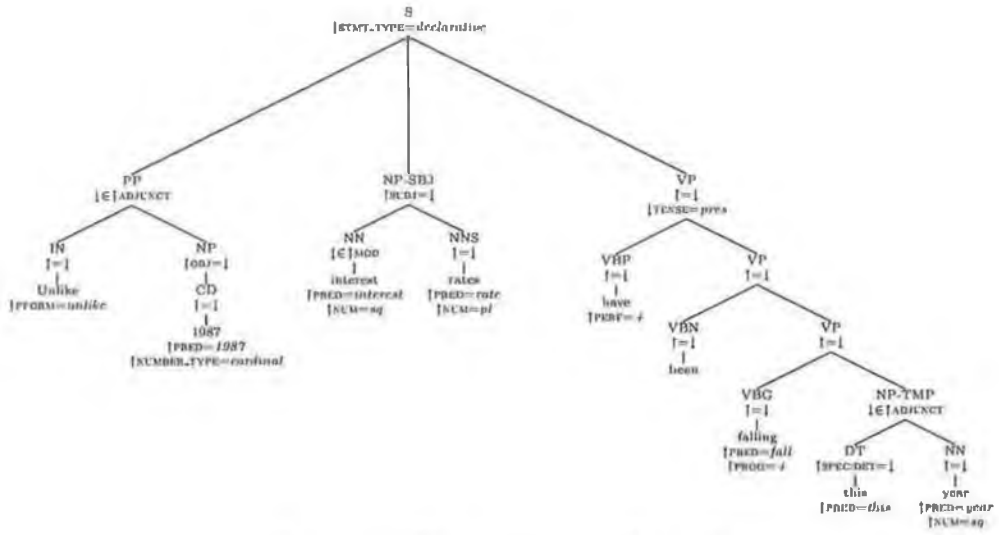
Figure 4.13: Automatically annotated tree for the sentence *Unlike 1987, interest rates have been falling this year*

Figure 4.14 provides the annotated tree and corresponding f-structure resulting from the application of the conversion software to the automatically annotated tree of Figure 4.13. The f-structure has the desired flat analysis with the correct PARC 700 tense and aspect features and the adjuncts *unlike 1987* and *this year* grouped in a single adjunct set.

4.4 Evaluation

4.4.1 Results

The 700 sentences comprising the PARC 700 were split into a development set of 140 sentences and a test set of 560 for the experiments described in Kaplan et al. (2004). The same sets were used for the processes of developing and testing the conversion software. The 560 sentences of the test set were annotated by the automatic annotation algorithm and mapped using the software outlined in the previous section. The resulting f-structures were evaluated against the PARC 700 using the evaluation methodology and software presented in Crouch et al. (2002) and Riezler et al. (2002). The mapped f-structures for the 560-sentence test set achieved an f-score of 87.33% against the PARC 700 dependencies.



$$
\begin{bmatrix}
\text{ADJUNCT} & \left\{ \begin{bmatrix} \text{PRED} & \text{unlike} \\ \text{PFORM} & \text{unlike} \\ \text{OBJ} & [\text{PRED } 1987] \end{bmatrix}, \begin{bmatrix} \text{SPEC} & [\text{DET} & [\text{PRED} & \text{this}]] \\ \text{PRED} & \text{year} \end{bmatrix} \right\} \\
\text{SUBJ} & \begin{bmatrix} \text{ADJUNCT} & \{[\text{PRED} & \text{interest}]\} \\ \text{PRED} & \text{rate} \end{bmatrix} \\
\text{PRED} & \text{fall} \\
\text{TENSE} & \text{pres} \\
\text{STMT\_TYPE} & \text{declarative} \\
\text{PERF} & + \\
\text{PROG} & +
\end{bmatrix}
$$

Figure 4.14: Annotated tree and corresponding f-structure resulting from the application of the conversion software to the annotated tree of Figure 4.13

	Precision	Recall	F-Score
ADEGREE	1037/1216 = 85	1037/1290 = 80	83
ADJUNCT	2295/3040 = 75	2295/2995 = 77	76
AQUANT	9/10 = 90	9/13 = 69	78
COMP	217/241 = 90	217/257 = 84	87
CONJ	468/534 = 88	468/552 = 85	86
COORD_FORM	230/307 = 75	230/252 = 91	82
DET_FORM	949/962 = 99	949/964 = 98	99
FOCUS_INT	0/0 = 0	0/5 = 0	0
MOD	411/483 = 85	411/573 = 72	78
NUM	3721/3983 = 93	3721/4145 = 90	92
NUMBER	262/295 = 89	262/297 = 88	89
NUMBER_TYPE	409/424 = 96	409/440 = 93	95
OBJ	1678/1801 = 93	1678/1866 = 90	92
OBJ_THETA	5/12 = 42	5/11 = 45	43
OBL	127/236 = 54	127/173 = 73	62
OBL_AG	38/43 = 88	38/45 = 84	86
OBL_COMPAR	5/8 = 62	5/15 = 33	43
PASSIVE	186/197 = 94	186/238 = 78	86
PCASE	40/43 = 93	40/52 = 77	84
PERF	79/86 = 92	79/86 = 92	92
POSS	186/200 = 93	186/205 = 91	92
PRECOORD_FORM	0/0 = 0	0/6 = 0	0
PROG	169/174 = 97	169/203 = 83	90
PRON_FORM	507/547 = 93	507/531 = 95	94
PRON_INT	0/0 = 0	0/6 = 0	0
PRON_REL	103/145 = 71	103/119 = 87	78
PROPER	625/761 = 82	625/744 = 84	83
PRT_FORM	32/39 = 82	32/46 = 70	75
QUANT	55/69 = 80	55/71 = 77	79
STMT_TYPE	962/1066 = 90	962/1094 = 88	89
SUBJ	1580/1716 = 92	1580/1779 = 89	90
SUBORD_FORM	159/193 = 82	159/195 = 82	82
TENSE	1002/1022 = 98	1002/1051 = 95	97
TOPIC_REL	105/145 = 72	105/119 = 88	80
XCOMP	416/461 = 90	416/478 = 87	89
Overall	88.31	86.38	87.33
Preds-only	84.71	84.21	84.45

Table 4.3: Results by feature name of evaluation against the PARC 700

Table 4.3 provides the results for each feature in terms of precision, recall and f-score.

4.4.2 Analysis

There is a wide gap between the results achieved by the annotation algorithm when evaluated against the DCU 105 (96.93% and 94.28% f-score for all grammatical functions and preds-only) and, using the conversion software, against the PARC 700 (87.33% and 84.45% f-score for the PARC 700 evaluation feature set of Kaplan et al. (2004) and for preds-only). A number of reasons for the poorer results against the PARC 700 are analysed in this section.

4.4.2.1 Penn-II POS Tagging

The analysis of modifiers as nominal or adjectival in the PARC 700 cannot accurately be predicted from the Penn-II POS tags. The annotation algorithm annotates nominal tags with ↑NUM=*sg* or ↑NUM=*pl*, while the conversion software is used to annotate adjectives ↑ADEGREE=*positive*. Nominal modifiers are annotated with a MOD feature while adjectival modifiers receive an ADJUNCT annotation. Penn-II POS tags are also used to make this distinction in the conversion software.

In most cases the Penn-II POS tags and the PARC 700 analysis match, allowing the annotation algorithm to produce the desired feature annotations. However, there is a significant amount of divergence and in every such case the annotation algorithm is penalised when evaluated against the PARC 700. The same penalty is not incurred when evaluating the automatically acquired f-structures against the DCU 105, because Penn-II POS tags were used to determine whether a NUM feature is used in the DCU 105 gold standard. The MOD feature and ↑ADEGREE=*positive* annotation do not occur in the DCU 105.

Table 4.3 shows that the annotation algorithm achieves an f-score of 92% for NUM which is by far the most frequently occurring feature in the PARC 700. Against the DCU 105, the acquired f-structures achieve 100% accuracy for this feature.

4.4.2.2 Hyphenation

In the PARC 700 dependencies, approximately one-third of all occurrences of hyphenated words are split into separate lemmas, each with their own feature-value pairs. The annotation algorithm, conversion software and DCU 105 leave hyphenation intact. The automatically acquired f-structures are penalised when mapped and evaluated against the PARC 700 for every split hyphenated word. This penalty is not incurred when evaluating against the DCU 105.

Figure 4.15 shows the f-structure acquired by the annotation algorithm for the phrase *investment-grade quality properties*, together with its PARC 700 analysis. The lemma *investment-grade* is tagged in Penn-II as an adjective. The PARC 700 removes the hyphen producing two lemmas *investment* and *grade*, both of which it analyses as nominal

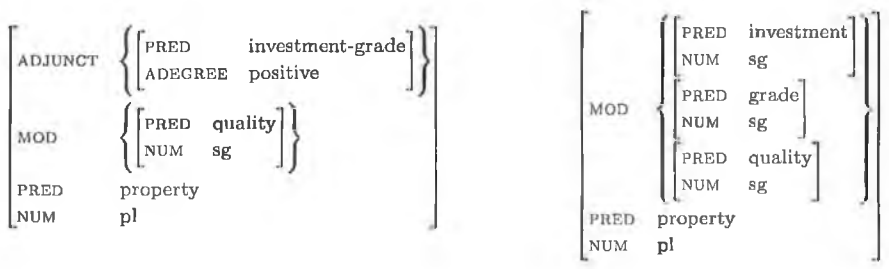


Figure 4.15: DCU 105 and PARC 700 treatment of hyphenation in the phrase *investment-grade quality properties*

modifiers. The annotation algorithm would achieve an f-score of 100% for this NP (and most simple NPs) in the DCU 105. Against the PARC 700, the f-score achieved for this phrase is 50%.

4.4.2.3 Computational Error Margins

As outlined in Section 4.3, the conversion software consists of five modules. It is inevitable that each additional computation module adds its own margin of error: these are cases where a conversion mapping is carried out inappropriately or a required mapping is missed. The five additional modules required for evaluation against the PARC 700 gold standard are lossy and produce a higher computational error margin than the simpler process of evaluating against the DCU 105.

4.4.2.4 Characteristics of both Gold Standards

The origin of both gold standards must also impact on the results achieved by the automatically acquired f-structures. The DCU 105 was designed for the purpose of evaluating f-structures produced by the automatic f-structure annotation algorithm and the derived parsing technology. The PARC 700 is based on the f-structures for the 700 sentences provided by the hand-crafted broad-coverage LFG grammar of English using the XLE system (Maxwell and Kaplan, 1993). As a result, in each case there is some systematic bias towards a particular style of analysis. The most obvious example of this bias is the lemmas used. As the lemmas which are used in both the DCU 105 and the automatically generated f-structures are derived from a common source, there is a 100% match. While

efforts were made to align the lemmas of the automatically generated f-structures with those used in the PARC 700, there are some inconsistencies which could not be systematically resolved. This inconsistency results in an additional margin of error when evaluating against the PARC 700.

The DCU 105 is a relatively small gold standard. There are a number of problems with evaluating against a gold standard of this size, most notably that of overfitting. There is a risk of assuming that the gold standard is a complete and balanced representation of the linguistic phenomena in a language and then basing design decisions on this assumption. The possibility that the annotation algorithm overfits the DCU 105 may be a contributory factor to the gap between the evaluation results.

4.5 Summary

This chapter presents an evaluation of the automatic f-structure annotation algorithm against the PARC 700 Dependency Bank. Some of the systematic differences between the DCU 105 and PARC 700 representations were outlined, motivating the development of a suite of conversion software to map the automatically annotated trees to overcome these differences. The five modules of the conversion software were described in detail and applied to the automatically f-structure-annotated Penn-II trees for the PARC 700 strings. The results of the evaluation process were provided and analysed. The automatically acquired and mapped f-structures achieve an f-score of 87.33% against the PARC 700 test set for the feature set of Kaplan et al. (2004). Differences in linguistic analysis, which could not be resolved by the systematic mappings of the conversion software, were illustrated as these problems contribute to the difference in results achieved by the annotation algorithm against the PARC 700 and DCU 105.

While the conversion software was established for evaluation purposes, it can also be used to produce a version of the Penn-II treebank annotated with f-structure information in the style of those generated by the hand-crafted grammars developed in the ParGram project (Butt et al., 2002) underlying the PARC 700 dependencies. Furthermore, the conversion software is designed to allow the parsing technology presented in Cahill et al. (2004b) to also be evaluated against the PARC 700. The results presented in this chapter

provide an upper bound for the results which can be achieved by the parsing technology of Cahill et al. (2004b).

Chapter 5

Evaluation of the Automatic F-Structure Annotation Algorithm against PropBank

5.1 Introduction

This thesis presents an automatic method for the annotation of treebank trees with LFG f-structure information. This work is a core component of a larger project (Burke et al., 2004b) for the automatic acquisition of high quality LFG lexicon and grammar resources. Chapters 2 and 3 present the process of automatically annotating Penn-II with LFG f-structure information. Chapter 4 extensively evaluates the automatically acquired LFG f-structures for English against the PARC 700. Evaluation of the automatic f-structure annotation algorithm is motivated by the importance of the algorithm for the automatic acquisition of LFG resources presented by Cahill et al. (2004b) and O'Donovan et al. (2004, 2005a). This chapter evaluates the annotation algorithm against PropBank (Kingsbury and Palmer, 2002).

In contrast to the DCU 105 and PARC 700, PropBank provides a layer of semantic annotation for the syntax trees of the Penn-II treebank. Evaluating against PropBank provides a semantic evaluation of the automatically acquired f-structures. This poses new challenges as annotation quality has so far only been measured against the syntax-

based DCU 105 and PARC 700 gold standards. PropBank also allows a much larger scale evaluation than was previously possible, in principle allowing f-structure quality to eventually be evaluated against PropBank data for the *entire* Penn-II treebank.

As PropBank was developed independently of any grammar formalism, it provides a platform which allows more meaningful comparisons to be made between parsing technologies than was previously possible. PropBank has been used for the evaluation of CCG (Gildea and Hockenmaier, 2003) and HPSG (Miyao and Tsujii, 2004) parsers. The methodology presented in this chapter will allow the parsing technology of Cahill et al. (2004b) to be evaluated against PropBank in future and for comparisons with CCG, HPSG and other parsers to be made.

Evaluation proceeds as follows: first, semantic role-based PropBank annotations (ARG0, ..., ARGM) are converted into a dependency format (triples); second, automatically generated f-structures are converted into LFG grammatical function-based triples (SUBJ, OBJ, ...); third, conversion software systematically maps the LFG grammatical function-based triples encoding to the PropBank semantic role-based triples encoding; fourth, the evaluation software of Crouch et al. (2002) and Riezler et al. (2002) is used to compute precision, recall and f-score.

Section 5.2 provides an overview of PropBank and the process of converting the PropBank semantic annotations into dependency format for evaluation purposes. Section 5.3 describes the conversion software which systematically converts the triples extracted from the automatically generated f-structures for evaluation against PropBank. Section 5.4 presents and analyses the results of the evaluation process. Using the Penn-II WSJ section 24 as the development set, we currently achieve an f-score of 76.58% against PropBank for the WSJ section 23 test set. Section 5.5 outlines possibilities for future work. Section 5.6 summarises the chapter. Most of the work presented here is published by Burke et al. (2005).

5.2 PropBank

5.2.1 Overview

PropBank (Kingsbury and Palmer, 2002) adds a layer of semantic annotation to the syntax trees of Penn-II. The process of semantic role annotation was semi-automatic. A rule-based automatic argument tagger encodes class-based mappings between grammatical and semantic roles with 83% accuracy. The annotations were manually corrected and extended. PropBank contains a set of semantic frames for each Penn-II verb. The semantic frames define particular meanings for each verb and the roles played by their semantic arguments in each case. PropBank annotates Penn-II by identifying token verb occurrences, assigning a semantic frame to those verbs and marking the semantic arguments of the verbs. PropBank does not annotate or provide semantic roles for *be*.

5.2.2 Semantic Frames

PropBank assigns a set of semantic frames for every verb in Penn-II. Each semantic frame provides a definition for the semantic role labels relevant to that particular instance of the verb. Table 5.1 provides the three semantic frames for the predicate *yield*. The first semantic frame for *yield* defines the semantic role labels for the meaning *to result in*: ARG0 is the "thing yielding" and ARG1 is the "thing yielded".

	(yield.01) To result in	(yield.02) To give way	(yield.03) To give a dividend
ARG0	thing yielding	thing giving way	thing providing a dividend
ARG1	thing yielded	what's lost	dividend, earnings
ARG2	n/a	what's preferred	recipient

Table 5.1: PropBank semantic frame set for the predicate *yield*

Annotated example sentences for the three semantic frames for *yield* are:

(1) Frame 1: "To result in"

[$_{ARG0}$ *A single acre of grapes*] *yielded* [$_{ARG1}$ *a mere 75 cases*] [$_{ARGM-TMP}$ *in 1987*].

Frame 2: "To give way"

[$_{ARG0}$ *John*] *yielded* [$_{ARG1}$ *the right-of-way*] *to* [$_{ARG2}$ *the Mack truck*].

Frame 3: "To give a dividend"

The Canadian government announced [$_{ARG0}$ *a new, 12-year Canada Savings Bond issue*] *that will yield* [$_{ARG2}$ *investors*] [$_{ARG1}$ *10.5%*] [$_{ARGM-TMP}$ *in the first year*].

The semantic role label annotations indicate that in the first example sentence *a single acre of grapes* is the "thing yielding" while *a mere 75 cases* is the "thing yielded". The phrase *in 1987* is annotated as an optional modifier ARGM-TMP.

5.2.3 Semantic Argument Annotation

PropBank provides a file of semantic annotations for Penn-II in the following format. The annotations first identify the relevant Penn-II tree by providing the Penn-II file name and line number, e.g. line 12 in `wsj/00/wsj_0004.mrg` identifies the tree shown in Figure 5.1 for the sentence *The top money funds are currently yielding well over 9%*. The annotation then identifies the verb being annotated and the relevant semantic frame for this occurrence of the verb, which in this case is "yield.01", the frame "to result in" as outlined in Table 5.1. The semantic arguments are then listed in the form `terminal number:node height-semantic role`. Terminals are numbered from left to right starting with zero.

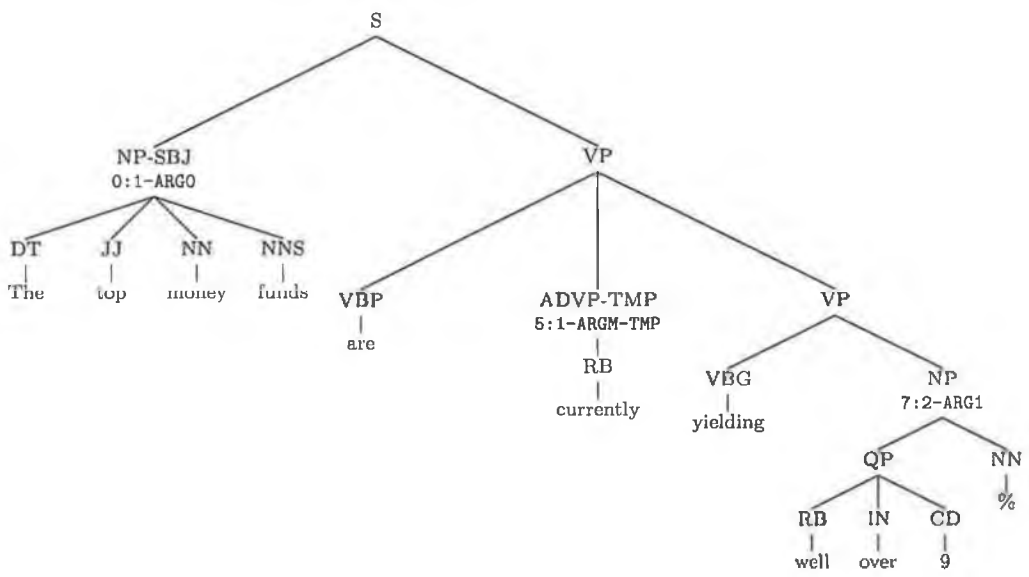


Figure 5.1: Penn-II tree for the sentence *The top money funds are currently yielding well over 9%*

The semantic arguments for the example sentence are annotated by PropBank as: `0:1-ARG0`, `5:1-ARGM-TMP` and `7:2-ARG1`. The annotation `0:1-ARG0` indicates that the node NP-SBJ which represents the noun phrase *The top money funds* is a semantic argument of the verb *yield* with the semantic role ARG0. This node is found in the tree of Figure 5.1 by starting with the POS tag of terminal 0 in the tree, i.e. DT, and traversing one node, i.e. `0:1`, upwards from that node. Similarly, the argument paths `5:1-ARGM-TMP` and `7:2-ARG1` indicate that the semantic roles ARGM-TMP and ARG1 are played by the nodes ADVP-TMP and NP representing *currently* and *well over 9%*, respectively.

5.2.4 Creating Gold Standard PropBank Dependencies

In order to evaluate the automatic f-structure annotation algorithm, the PropBank semantic annotations were converted into a dependency format (triples). By also mapping the automatically generated f-structures into a set of semantic role triples, the methodology and software of Crouch et al. (2002) and Riezler et al. (2002) could be used to evaluate the annotation algorithm in terms of precision, recall and f-score.

The PropBank semantic annotations were automatically converted into triples of the form: SEMANTIC ROLE(verb, argument). The Penn-II nodes representing the semantic roles were identified by automatically traversing the argument paths as outlined in Section 5.2.3. For each node, the head word of the subtree represented by that node was identified using the head-lexicalisation rules of the f-structure annotation algorithm, which are a modified version of the rule set of Magerman (1994). The verbs and head words were lemmatised with the XLE lemmatiser also used by the annotation algorithm. The PropBank semantic roles were conflated, removing the different subtypes of ARGM modifiers (Table 5.2), to the subset: ARG0, ARG1, ARG2, ARG3, ARG4, ARG5 and ARGM.

ADV	adverbial	MOD	modal verb
CAU	cause	NEG	negation
DIR	direction	PNC	purpose not cause
DIS	discourse connectives	PRD	predication
EXT	extent	REC	reciprocal
LOC	location	TMP	temporal
MNR	manner		

Table 5.2: PropBank ARGM subtypes

To create PropBank triples for the sentence *The top money funds are currently yielding*

well over 9%, the head words of the nodes NP-SBJ, ADVP-TMP and NP (Figure 5.1) were automatically identified as *funds*, *currently* and *%*, respectively. After lemmatising all words and conflating the semantic roles, the triples ARG0(yield, fund), ARG1(yield, percent) and ARGM(yield, currently) were created. This process was applied to all trees in the treebank.

5.3 Converting F-Structures into Semantic Roles

I developed conversion software to produce PropBank-style semantic role annotations in the dependency format introduced in Section 5.2.4. F-structures are automatically acquired by the annotation algorithm from Penn-II trees. Triples are extracted from these f-structures and then post-processed by the conversion software to produce semantic role annotations. The conversion procedure employs default mappings from LFG feature names to PropBank semantic roles before handling the following phenomena which require more complex mappings:

- Particles,
- Modal verbs,
- Mapping to ARG3, ARG4 and ARG5,
- Verbs deviating from default mapping patterns,
- Filtering remaining unwanted triples.

5.3.1 Default Mappings

Default mappings are used to map LFG feature names to PropBank semantic role annotations. Table 5.3 lists these mappings for active verbs. Passive voice is identified by the annotation algorithm which results in PASSIVE triples being extracted from the automatically generated f-structures. These triples are used by the conversion software to map the SUBJ triple of passive verbs to ARG1 (subjects of active verbs are mapped by default to ARG0), while oblique agents are mapped to ARG0.

LFG feature name	PropBank semantic role
SUBJ	ARG0
OBJ	ARG1
COMP	ARG1
XCOMP	ARG1
OBJ_THETA	ARG2
OBL	ARG2
OBL2	ARG2
ADJUNCT	ARGM

Table 5.3: Default mappings from LFG feature names to PropBank semantic roles for verbs with active voice

The default mappings of Table 5.3 were applied to the automatically generated LFG triples for the active verb *yield* in the sentence *The top money funds are currently yielding well over 9%.* The resulting mapped PropBank-style triples and the original LFG triples are provided in Table 5.4. The default mappings are successful for this sentence, producing the desired PropBank triples.

Automatically generated LFG triples	Mapped PropBank-style triples	Gold standard PropBank triples
SUBJ(yield, fund)	ARG0(yield, fund)	ARG0(yield, fund)
OBJ(yield, percent)	ARG1(yield, percent)	ARG1(yield, percent)
ADJUNCT(yield, currently)	ARGM(yield, currently)	ARGM(yield, currently)

Table 5.4: Default mappings applied to automatically generated triples for *The top money funds are currently yielding well over 9%*

5.3.2 Particles

PropBank annotates phrasal verbs by grouping all nodes representing the phrasal verb and providing their semantic arguments as normal. When creating the gold standard PropBank triples, we combined the grouped nodes to form a multi-word expression for the phrasal verb. Phrasal verbs have a single triple for each semantic argument as with all other verbs. The third column of Table 5.5 provides the gold standard triples we extracted from PropBank for the phrasal verb *snap up* in the sentence *Earlier this year, Japanese investors snapped up a similar fund.* The first column provides a subset of the triples produced by the f-structure annotation algorithm for the same sentence, while the second column shows the PropBank-style triples produced by the application of the default mappings to the triples of column one.

An f-score of zero will be achieved for this sentence unless the multi-analysis is adopted

for the phrasal verb. The Penn-II PRT (particle) tag is automatically annotated ↑PART=↓, which results in the triple PART(snap, up) in this example. The conversion software uses the PART triple to create the multi-word expression predicate which replaces all occurrences of the bare verb in the mapped triples. This allows the desired gold standard triples to be produced by the mapping module.

Automatically generated LFG triples	Triples created by default mappings	Gold standard PropBank triples
SUBJ(snap, investor)	ARG0(snap, investor)	ARG0(snap_up, investor)
OBJ(snap, fund)	ARG1(snap, fund)	ARG1(snap_up, fund)
ADJUNCT(snap, year)	ARGM(snap, year)	ARGM(snap_up, year)
PART(snap, up)		

Table 5.5: Automatically generated LFG triples, triples created by the default mappings and gold standard PropBank triples for *Earlier this year, Japanese investors snapped up a similar fund*

5.3.3 Modal Verbs

Modal verbs are represented in PropBank as optional arguments of the main verb. This treatment differs markedly from the cascading XCOMP analysis of the automatically generated f-structures and triples. Table 5.6 provides a subset of the automatically generated LFG triples and gold standard PropBank triples for the sentence *France can boast the lion's share of high-priced bottles.*

Automatically generated LFG triples	Triples created by default mappings	Gold standard PropBank triples
SUBJ(can, france)	ARG0(can, france)	
MODAL(can, +)		ARGM(boast, can)
XCOMP(can, boast)	ARG1(can, boast)	
SUBJ(boast, france)	ARG0(boast, france)	ARG0(boast, france)
OBJ(boast, share)	ARG1(boast, share)	ARG1(boast, share)

Table 5.6: Automatically generated LFG triples, triples created by the default mappings and gold standard PropBank triples for *France can boast the lion's share of high-priced bottles.*

The annotation algorithm uses the Penn-II MD tag to annotate modal verbs. The MODAL triple triggers the creation of an ARGM triple in the mapping module. The cascading XCOMP triples are traversed from the modal verb to identify the main verb which is then modified by the new ARGM triple. Having created this new triple, all other triples associated with the modal verb are removed. This procedure, together with the default mappings, allows the gold standard PropBank analysis to be achieved.

Automatically generated LFG triples	LFG triples without relative pronouns	Gold standard PropBank triples
RELMOD(right, expire)	RELMOD(right, expire)	
PRON_FORM(pro, which)	PRON_FORM(right, which)	
TOPICREL(expire, pro)	TOPICREL(expire, right)	
SUBJ(expire, pro)	SUBJ(expire, right)	ARG0(expire, right)
ADJUNCT(expire, november)	ADJUNCT(expire, november)	ARGM(expire, november)

Table 5.7: Automatically generated LFG triples and mapped PropBank triples for the fragment *The rights, which expire Nov. 21*

5.3.4 Relative Clauses

The gold standard triples extracted from PropBank do not contain relative pronouns. Instead, the head noun being modified by the relative clause takes the place of relative pronouns in the gold standard triples. As the default mappings are not sufficient to compute the desired PropBank-style triples from the automatically generated LFG triples for verbs embedded within relative clauses, a further mapping step handles relative pronouns.

The automatically generated LFG triples indicate the presence of a relative clause through RELMOD and TOPICREL triples. The first column of Table 5.7 provides a subset of the automatically generated LFG triples for the fragment *The rights, which expire Nov. 21*. The RELMOD triple indicates that the noun (lemmatised as) *right* is modified by a relative clause which has *expire* as its main verb. The value *pro* represents the relative pronoun, whose surface form *which* is provided by the PRON_FORM triple. The TOPICREL triple links the *pro* value to the verb, indicating which pronoun is the fronted element of the relative clause. The SUBJ triple indicates that the relative pronoun is the subject of the relative clause.

Applying the default mappings to SUBJ(expire, pro) would produce the incorrect PropBank triple ARG0(expire, pro). To overcome this problem, the conversion software first locates RELMOD triples. A RELMOD triple indicates that a noun is modified by a relative clause and provides the main verb of that clause. The TOPICREL triple associated with that main verb is then found. This triple provides the relative pronoun. Every occurrence of that relative pronoun in all triples for that sentence is replaced with the noun from the RELMOD triple (Table 5.7, second column). With this step in place, the default mappings (in this case from SUBJ to ARG0) are used to achieve the correct analysis.

5.3.5 Mapping to ARG3, ARG4 and ARG5

The mappings outlined so far will not generate any triples for the semantic roles ARG3, ARG4 and ARG5. While using the WSJ section 24 of Penn-II as a development set, it became clear that a significant number of ARG3 and ARG4 annotations occur in pairs with verbs taking two oblique prepositional phrases, headed by *from* and *to*. The PP headed by *from* was usually annotated ARG3, while the PP headed by *to* was annotated ARG4. This information was encoded in the conversion software to produce the desired ARG3 and ARG4 triples instead of mapping by default to ARG2. ARG5 occurs very infrequently (only 5 times in WSJ section 23). No mapping was developed for this semantic role.

5.3.6 Mappings for Specific Verbs

In many cases, even when the annotation algorithm generates a correct f-structure, there are no syntactic cues which can be used to produce the expected PropBank triples. The syntactic information available through the automatically generated f-structures and triples is insufficient for mapping the semantic roles of, for example, *climb*. Table 5.8 provides three sets of triples for the sentence *Net profit climbed to 30%*; (i) the triples produced by the f-structure annotation algorithm, (ii) the mapped triples produced using the conversion software described so far and (iii) the expected PropBank triples.

Automatically generated LFG triples	Mapped triples	Gold standard PropBank triples
SUBJ(climb, profit)	ARG0(climb, profit)	ARG1(climb, profit)
ADJUNCT(profit, net)		
OBL(climb, to)	ARG2(climb, to)	ARG4(climb, to)
OBJ(to, percent)		
QUANT(percent, 30)		

Table 5.8: Automatically generated LFG triples, mapped triples and PropBank triples for *Net profit climbed to 30%*

Applying the default mappings to the automatically generated triples produces ARG0 and ARG2 triples which should actually be ARG1 and ARG4, respectively. Having reviewed the development set, this is the normal expected behaviour for the verb *climb*. There is no further syntactic information available which could be used in a general mapping rule to produce the correct triples in this case, without degrading the overall performance of the conversion software for most verbs. Instead of introducing a general rule to deal with

this case, a specific rule was introduced for the verb *climb* mapping SUBJ to ARG1, OBJ to ARG2, OBL to ARG3 for PPs headed by *from* and to ARG4 for PPs headed by *to*.

Other verbs in the development set displayed the same behaviour as *climb*. On examination of the VerbNet (Kipper et al., 2000) classes containing *climb*, class 45.6 provided many verbs which required the mapping outlined above:

(2) *appreciate, balloon, climb, decline, decrease, depreciate, differ, diminish, drop, fall, fluctuate, gain, grow, increase, jump, lessen, mushroom, plummet, plunge, rise, rocket, skyrocket, soar, surge, tumble, vary*

This list was amended on further analysis of the development set, with *lessen* removed and *return* added to the list of verbs mapped in the same manner as *climb*.

A number of other specific mappings were created for groups of verbs, e.g. VerbNet class 48.1.1:

(3) *appear, arise, awake, awaken, break, burst, come, dawn, derive, develop, emanate, emerge, erupt, evolve, exude, flow, form, grow, gush, issue, materialize, open, plop, result, rise, spill, spread, steal, stem, stream, supervene, surge, wax*

For active occurrences of a subset of these verbs, SUBJ is mapped to ARG1. The defaults and other general mappings are used for all other triples with these verbs.

5.3.7 Filtering

Penn-II verbal POS tags and phrasal bracketing cannot always be used to accurately predict which words are annotated by PropBank. Errors in Penn-II POS tagging would result in the annotation algorithm producing PropBank triples for words which are not annotated by PropBank. In some cases, words which are correctly tagged in Penn-II as verbs and bracketed as the head of a VP are not annotated by PropBank. The annotation algorithm would be punished in these cases for correctly producing PropBank-style triples.

The original version of the conversion software Burke et al. (2005) used the PropBank gold standard triples to overcome this problem. The gold standard triples were consulted to indicate which words were annotated as verbs in PropBank. The conversion software only produced PropBank-style triples for those lemmas. This procedure has since been

removed and the conversion software no longer refers to the gold standard triples, relying instead on Penn-II POS tagging and bracketing only.

For the purpose of evaluation, a CAT(egory) feature with the value v is added to the f-structures produced by the annotation algorithm for all words POS-tagged in Penn-II as verbs and bracketed as the head of a VP, ADJP, PP or any category annotated with the Penn-II -PRD (predicative) functional tag. CAT triples are extracted from the automatically generated f-structures and are used to filter the PropBank-style triples produced by the conversion software. PropBank-style triples are only produced for lemmas occurring with a CAT triple.

The new procedure is preferred to the original consultation of the gold standard PropBank triples to identify the annotated verbs as it is more methodologically sound and the results presented in Table 5.9 are derived with the new procedure. The new procedure achieves an f-score which is only 0.32% lower than the original procedure.

5.4 Evaluation

5.4.1 Results

The 2,416 trees in WSJ section 23 of Penn-II were annotated by the automatic f-structure annotation algorithm. Triples were extracted from the resulting f-structures and passed through the conversion software outlined in Section 5.3. These triples were evaluated against the gold standard triples extracted from the PropBank annotations for the same sentences using the methodology and software presented in Crouch et al. (2002) and Riezler et al. (2002). Without specific verb mappings an f-score of 73.42% is achieved, with precision and recall at 75.14% and 71.77%, respectively. Including specific verb mappings sees the overall f-score increase to 76.58% as a result of improved precision and recall scores of 78.44% and 74.81%. Table 5.9 provides the results in terms of precision, recall and f-score for each semantic role with and without specific verb mappings.

	Without Specific Verb Mappings			With Specific Verb Mappings		
	Precision	Recall	F-score	Precision	Recall	F-score
ARG0	3176/4289=74	3176/3708=86	79	3127/3887=80	3127/3708=84	82
ARG1	3408/4297=79	3408/5009=68	73	3685/4506=82	3685/5009=74	77
ARG2	349/775=45	349/1115=31	37	460/863=53	460/1115=41	47
ARG3	25/28=89	25/173=14	25	54/60=90	54/173=31	46
ARG4	24/28=86	24/102=24	37	50/54=93	50/102=49	64
ARG5	0/0=0	0/5=0	0	0/0=0	0/5=0	0
ARGM	2978/3837=78	2978/3765=79	78	3006/3865 = 78	3006/3765 = 80	79
Overall	75.14	71.77	73.42	78.44	74.81	76.58

Table 5.9: Annotation quality measured against PropBank for WSJ section 23 of Penn-II, with and without mappings for specific verbs

5.4.2 Analysis

The overall f-score of 76.58% achieved by the annotation algorithm against PropBank for WSJ section 23 of Penn-II is lower than the results in previous evaluation experiments. Against the DCU 105 an f-score of 96.93% was achieved for complete f-structures and 94.28% for preds-only f-structures, while against the PARC 700 Dependency Bank using the feature set of Kaplan et al. (2004), the f-score was 87.33%. When evaluating the automatically generated f-structures — a syntax-based resource — against a gold standard of semantic relations such as PropBank, lower results should be expected than in experiments evaluating the f-structures against syntax-based gold standards, such as the DCU 105 and PARC 700.

Overall, precision is higher than recall, indicating that our algorithm is more likely to produce a partial analysis than an incorrect one. The only semantic role with precision lower than recall is ARG0. The conversion software attempts to map the semantic arguments of specific verbs which deviate from the behaviour captured in the default mappings. Most mappings for specific verbs map the SUBJ triple to ARG1 instead of the default mapping to ARG0. These mappings result in an improvement in f-scores for ARG0 and ARG1 of 3% and 4%, respectively. However, the conversion software does not provide specific mappings for enough verbs which results in too many SUBJ triples still being incorrectly mapped to ARG0.

Creating different SUBJ mappings for animate and inanimate arguments may be advantageous. The default mapping of SUBJ to ARG0 for active verbs is often incorrect for inanimate subjects, e.g. *The share price climbed.* Mapping SUBJ to ARG1 for inanimate

objects may help improve the identification of both ARG0 and ARG1.

A further, albeit less significant, explanation for the lower precision score for ARG0 is the failure of the annotation algorithm in some cases to identify a verb occurrence as having passive voice. In a syntax-based evaluation, this results in a missing PASSIVE triple which lowers recall and leaves precision unchanged. Unlike in the semantic role evaluation, the net effect is less significant as there is a larger number of triples per sentence in the syntactic evaluation. By contrast, a missing passive marker in the semantic evaluation means that the SUBJ triple is mapped by default to ARG0 instead of ARG1. This results in lower precision for ARG0 and lower recall for ARG1. This is reflected in the scores for ARG1; precision 82%, recall 74%.

The best results are achieved for the semantic roles ARG0, ARG1 and ARGM with f-scores of 82%, 77% and 79%, respectively. As these semantic roles are the most frequently occurring, accounting for 90% of all gold standard triples, the development of mappings for these triples was the main focus of this research. In addition, however, when the conversion software does produce the less frequently occurring ARG3 and ARG4 triples, they are usually correct, as shown by the high precision scores of 90% and 93%, respectively. The low recall scores of 31% and 49% indicate that far too few ARG3 and ARG4 triples are being mapped.

These infrequently occurring semantic roles do not have obvious default equivalent LFG feature names which makes them particularly difficult to map. The specific verb mappings allow significant improvements to be made: f-scores increase for ARG3 and ARG4 by 21% and 27%, respectively. A relatively conservative approach was taken when mapping these semantic roles which accounts for some of the shortfall. Another reason for the scarcity of these triples is that they are only produced through the mapping of OBL triples generated by the annotation algorithm. Distinguishing between obliques and adjuncts is an area fraught with difficulty. The annotation algorithm relies on the Penn-II -CLR and -DTV functional tags for the annotation of obliques. In the original Penn-II annotation, these functional tags were employed relatively inconsistently and infrequently which may contribute to the shortage of ARG3 and ARG4 triples. This fact also partially explains the poor results for ARG2, which has higher precision than recall, caused by ARG2

triples not being produced in sufficient volume. Obliques are one source of ARG2 triples.

No mappings have been developed to produce ARG5 triples as they occur too infrequently for any general pattern to be established.

5.5 Future Work

5.5.1 Harnessing Results to Improve Annotation Quality

The evaluation of the annotation algorithm against PropBank provides a benchmark for annotation quality. As this is our largest evaluation of f-structure annotation quality, it provides more valuable feedback than was previously available to us. Focus should now be placed on analysing the evaluation results for the purpose of improving the annotation algorithm itself and not just the mapping software. The analysis of the results to date has shown that the identification of passive voice is one area which needs to be improved. Further analysis should highlight other problem areas and allow improvements to be made to the annotation algorithm and the extraction of lexical resources (O'Donovan et al., 2004, 2005a) and parsing technology Cahill et al. (2004b) based on the algorithm.

5.5.2 Alternative Mapping Procedure

An alternative approach to the mapping process may be required, as there are clear limitations to the improvements which can be made to the current mapping software. I have examined one alternative mapping procedure, similar to the methodology of Miyao and Tsujii (2004), which may provide a better solution in the long term than the conversion software described in this chapter. A mapping from f-structure annotations to PropBank annotations could be learned from a training set of Penn-II trees, e.g. WSJ sections 02 to 21.

The annotation algorithm would be used to produce f-structures for the training set, from which triples would then be extracted. By aligning these automatically generated triples with their gold standard PropBank equivalents, the LFG features for each verb occurrence in the training set could be listed with their equivalent PropBank semantic roles. The passive markers of the annotation algorithm could be used to indicate whether

a verb occurs with passive voice. A ranked list could be compiled for each verb of their most frequent active and passive mappings from LFG features to PropBank semantic roles.

For the test set (WSJ section 23), the Penn-II trees would be automatically annotated and triples would be extracted from the resulting f-structures. The LFG features and passive markers would be retrieved from the triples for each verb occurrence. The highest-ranked mapping for that verb occurrence with the given LFG features would be retrieved and used to map those triples to the corresponding PropBank semantic roles.

Preliminary examination of this approach has shown that it is potentially a better long-term solution than our current approach. To some extent the specific verb mappings of the current conversion software could be seen as a manual attempt at developing a similar mapping to that outlined by this alternative approach.

5.5.3 Universal Gold Standard Triples

As PropBank was developed independently of any grammar formalism, it provides a platform for making more meaningful comparisons between parsing technologies than was previously possible. However, given the format of the PropBank annotations and the need to convert these annotations to allow evaluation to take place, currently it is not straightforward to draw clear conclusions from such comparisons. There is a need for greater transparency in the evaluation process used to produce published results. This could be achieved through collaboration on the development and publication of a universal set of gold standard PropBank triples across a number of research groups.

5.5.4 Evaluation of Parsing Technology

The ultimate goal of this work is the evaluation of the parsing technology of Cahill et al. (2004b). The conversion software presented in Chapter 4 for the evaluation of the annotation algorithm against the PARC 700 is more refined than the PropBank conversion software of this chapter. The PropBank conversion software needs to be improved or replaced by an alternative approach to allow a true evaluation of the parsers and the annotation algorithm to be performed, as was possible against the PARC 700. Furthermore, a universally agreed set of gold standard triples derived from the PropBank resources is

required to support meaningful comparisons between parsers.

5.6 Summary

This chapter has presented an evaluation of the automatic f-structure annotation algorithm against PropBank for the test set, WSJ section 23 of Penn-II. A dependency-format gold standard was extracted from PropBank to facilitate the evaluation process. The Penn-II trees were automatically annotated to produce LFG f-structures, from which triples were extracted. Conversion software was developed to map these triples to produce PropBank-style semantic annotations in dependency format. WSJ section 24 of Penn-II and PropBank was used as the development set for the conversion software. An f-score of 76.58% was achieved against PropBank for the test set. A detailed analysis of the results was provided followed by several avenues for extending this research.

Chapter 6

Automatic Acquisition of Chinese LFG Resources

6.1 Introduction

Deep unification- or constraint-based grammars are usually hand-crafted. Scaling such grammars to handle unrestricted text is an expensive process, particularly in the case of multilingual grammar development. This thesis presents a methodology for the automatic acquisition of high-quality, wide-coverage LFG resources from treebanks. Chapters 2 and 3 present the process of automatically annotating the Penn-II treebank (Marcus et al., 1994) with LFG f-structure information, while Chapters 4 and 5 provide an extensive evaluation of the automatically acquired LFG f-structures for English against independent linguistic resources. This chapter demonstrates that our technology can be deployed for rapid grammar acquisition for Mandarin Chinese, reporting on a joint research project with colleagues at the University of Hong Kong (Adams Bodomo, Olivia Lam and Rowena Chan) to explore the application of our technology to the Penn Chinese Treebank (CTB) (Xue et al., 2002), published as Burke et al. (2004c). We have also successfully applied this rapid multilingual grammar acquisition strategy to German and Spanish treebanks (Cahill et al., 2005; O'Donovan et al., 2005b). This is possible because all components of the parsing architectures, except for the automatic f-structure annotation algorithm, are language-independent. The annotation algorithm is the only component that needs to be

adapted in order to acquire a grammar for a new language.

Seeding the annotation algorithm with linguistic generalisations is a semi-automatic process. For each parent category in the treebank, the most frequent CFG rule types providing >= 85% coverage of rule tokens are automatically extracted. The extracted seed rules are then automatically partially annotated with default and head annotations. The annotation of the seed rules is then manually corrected and completed. Generalisations in the form of annotation matrices are then extracted from the annotated seed rules. The completed annotation matrices are incorporated into the generic annotation algorithm. The co-ordination module must also be implemented. Rules for identifying the conjunct and the co-ordinated elements are extracted from the annotated seed rules.

Once the seeding of the annotation algorithm is complete, the annotation algorithm is ready to be applied for grammar induction and parsing. As the subcategorisation frame extraction algorithm operates at f-structure level, it is also language-independent. A lexicon can be automatically acquired once the seeding process is complete and the automatic f-structure annotation algorithm has been applied to the treebank.

The modularised nature of the technology allows components to be developed independently. To date, the main benefit of this has been the possibility to experiment with different (externally developed) parsers. A further benefit is the potential to separate the procedural, technical aspects of the technology from the linguistic seeding of the annotation algorithm. This provides a clean mechanism for collaborating with external linguists and incorporating their expert knowledge of the linguistics of a particular language. The development of the Mandarin Chinese resources took advantage of this possibility and this collaboration (Burke et al., 2004c) provides a model for future work.

Section 6.2 provides an overview of the CTB. Section 6.3 describes the process of extracting linguistic generalisations from the CTB to seed the annotation matrices of the automatic f-structure annotation algorithm. Section 6.4 provides an overview of the annotation algorithm modules and a quantitative and qualitative evaluation of the acquired proto-f-structures. Section 6.5 reviews the application of the subcategorisation frame extraction algorithm to the automatically acquired proto-f-structures. Three parsing experiments incorporating the Mandarin Chinese f-structure annotation algorithm are presented

in Section 6.6. The chapter concludes with a summary and several avenues for further related research.

The main results are summarised as follows: For 95.123% of the CTB training set trees, the annotation algorithm generates a single covering and connected f-structure, 4.805% are associated with more than one f-structure fragment, while the algorithm fails to generate any f-structure fragments for 0.072% of the training set trees due to feature clashes. A total of 10,479 semantic form tokens with 26 distinct frame types are extracted from these proto-f-structures. Verbal semantic forms account for 2,510 tokens instantiating all 26 frame types. The methodology for extracting four grammars incorporating the Mandarin Chinese annotation algorithm is described. Three experiments are performed to evaluate the performance of the grammars. The best-performing grammar on the tree-based evaluation is PCFG-P-A which achieves a labelled f-score of 81.77%, outperforming the previous best reported labelled f-scores of 78.8% by Levy and Manning (2003) and 79.9% by Chiang and Bikel (2002). PCFG-P-F performs best in both dependency-based evaluations achieving an f-score of 83.89% for all grammatical functions against the 50-sentence manually corrected gold standard f-structures and an f-score of 85.86% for all grammatical functions against the automatically annotated full CTB test set.

6.2 Penn Chinese Treebank

The CTB consists of syntax trees for Mandarin Chinese sentences. The automatic f-structure annotation algorithm was developed and evaluated using CTB version 2.0. This version consists of 325 articles of Xinhua newswire text from 1994 to 1998. The 4,183 sentences of CTB version 2.0 are segmented into 99,529 words, which corresponds to approximately one tenth of the English Penn-II treebank.

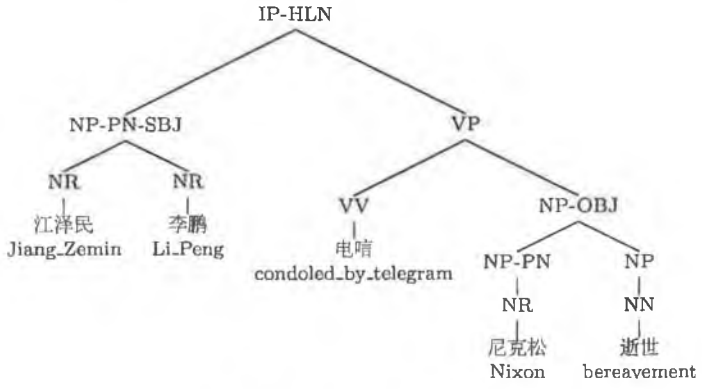
The CTB functional tag set (Table 6.1) provides more information than the corresponding Penn-II functional tag set. In addition to Penn-II's -SBJ (subject) tag, the CTB identifies objects and distinguishes between direct and indirect objects with the -OBJ and -IO tags respectively. Functional tags also provide tense, aspect and mood information, e.g. -CND (conditional), and indicate statement types, e.g. -IMP (imperative).

As with Penn-II, multiple CTB functional tags can be attached to CTB phrasal tags

Tag	Description	Tag	Description
ADV	adverbial	MNR	manner
APP	appositive	OBJ	direct object
BNF	benefactive	PN	proper names
CND	conditional	PRD	predicate
DIR	direction	PRP	purpose/reason
EXT	extent	Q	question
FOC	focus	SBJ	subject
HLN	headline	SHORT	short form
IJ	interjection	TMP	temporal
IMP	imperative	TPC	topic
IO	indirect object	TTL	title
LGS	logical subject	WH	wh-phrase
LOC	locative	VOC	vocative

Table 6.1: CTB functional tag set

to form complex categories. The CTB annotation scheme consists of 17 such phrasal tags, plus 33 POS tags, 6 verb compound tags and 7 empty categories. Figure 6.1 provides an example CTB tree for the sentence 江泽民李鹏电唁尼克松逝世 (*Jiang Zemin and Li Peng condoled the bereavement of Nixon by telegram*). The sentence is a headline as indicated by the statement type functional tag -HLN. The functional tag -PN distinguishes proper nouns from common nouns.



江泽民李鹏电唁尼克松逝世
Jiang Zemin and Li Peng condoled the bereavement of Nixon by telegram.

Figure 6.1: Example CTB tree

One noticeable difference between Penn-II and the CTB is the CTB's far more frequent usage of the FRAG(ment) tag which occurs in approximately 15% of all trees, always as the root node. Xinhua newswire articles begin with a headline followed by the reporter's name and location. The CTB retains this information as two trees: one tree for the headline, usually an IP with the -HLN functional tag, and another tree for the reporter's

details with FRAG as the root node. This consistency allows some generalisations to be made within our CTB annotation algorithm for the annotation of FRAGs. FRAG nodes were ignored when designing our annotation algorithm for Penn-II.

6.3 Seeding the Annotation Algorithm with Mandarin Chinese Linguistic Generalisations

The generic automatic f-structure annotation algorithm was seeded with Mandarin Chinese linguistic generalisations as outlined in Figure 6.2. As in our approach for English, for each CTB parent category the most frequent CFG rule types were extracted which expand that category to provide joint coverage of >= 85% of total CFG rule token occurrences for that parent category. The 645 most frequent rule tokens which were extracted for all CTB parent categories were then automatically partially annotated using head finding rules and default annotations triggered by CTB functional tags. The head RHS node of each CFG rule was automatically identified using the head rules of Levy and Manning (2003) and annotated $\uparrow=\downarrow$. Six default annotations were used to annotate RHS nodes marked with CTB functional tags (Table 6.2). The annotation of the 645 seed CFG rules was manually corrected and completed by our colleagues at the University of Hong Kong (Burke et al., 2004c). I then used the seed CFG rules with completed annotations to produce the Left-Right Context Annotation matrices for the Mandarin Chinese annotation algorithm.

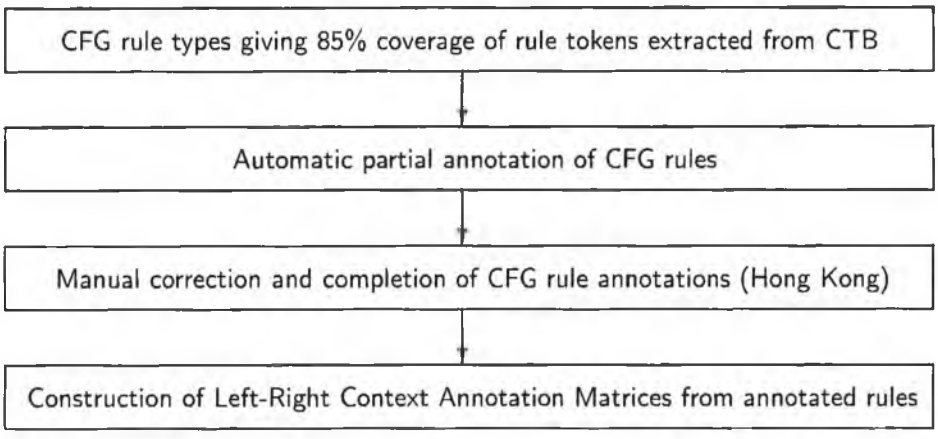


Figure 6.2: Procedure for seeding annotation algorithm with Mandarin Chinese linguistic generalisations

CTB Functional Tag	Default Annotation
-SBJ	↑SUBJ=↓
-OBJ	↑OBJ=↓
-IO	↑OBJ2=↓
-ADV	↓∈↑ADJN
-LOC	↓∈↑ADJN
-MNR	↓∈↑ADJN
-TMP	↓∈↑ADJN

Table 6.2: Default annotations for automatic partial annotation of extracted CTB seed rules

6.4 Automatic F-Structure Annotation Algorithm

6.4.1 Introduction

The Mandarin Chinese automatic f-structure annotation algorithm acquires proto-f-structures from the CTB using three modules of the generic annotation algorithm: the Left-Right Context Annotation module, the Co-ordination module and the Catch-All and Clean-Up module (Figure 6.3). The implementation of the Traces module of the annotation algorithm has been left for future work. The Traces module would use the CTB's null elements and co-indexation to capture long distance dependencies to allow the annotation algorithm to acquire proper f-structures.

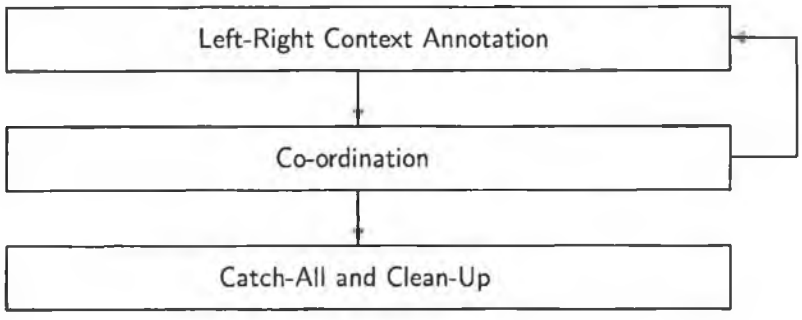


Figure 6.3: Mandarin Chinese Annotation Algorithm Modules

6.4.1.1 Left-Right Context Annotation

The first module of the Mandarin Chinese automatic f-structure annotation algorithm — Left-Right Context Annotation — head-lexicalises the CTB using the head-lexicalisation

rules of Levy and Manning (2003). This process creates a bi-partition of each local subtree, with nodes lying in either the left or right context of the head. The annotation algorithm annotates nodes in the left and right contexts by consulting the Left-Right Context Annotation matrices.

To gain maximum value from the manually completed 645 seed CFG rule annotations, the annotation algorithm first consults these rules when annotating a subtree to check if a manually annotated seed CFG rule matches exactly the CFG rule representing that local subtree. The manual annotations are applied to the subtree if a match is found, otherwise the Left-Right Context annotation matrices are consulted.

6.4.1.2 Co-ordination module

As with the annotation algorithm for Penn-II, the annotation of co-ordinate structures in CTB trees is handled by a separate co-ordination module in the Mandarin Chinese annotation algorithm. The relatively flat analysis of co-ordination in the both treebanks would complicate the Left-Right Context Annotation matrices, making them harder to maintain and extend. The co-ordination module annotates the co-ordinating conjunct (if present) as the head of the co-ordinate structure and identifies the elements of the co-ordination set. The Left-Right Context module is then reused to annotate any remaining unannotated nodes.

To give a simple example, the co-ordination module correctly annotates the co-ordination of proper nouns in structures which do not contain a co-ordinating conjunct (Figure 6.4). Noun phrases marked with the -PN (proper noun) functional tag may contain only nodes with the NR (proper noun) POS tag. Each NR node is annotated as an element of the co-ordination set ($\downarrow \in \uparrow$COORD). As no co-ordinating conjunct is present, a COORD_FORM feature is added with the value NULL.

6.4.1.3 Catch-All and Clean-Up module

The Catch-All and Clean-Up module provides default annotations for remaining unannotated nodes that are labelled with CTB functional tags. The functional tag -SBJ, for example, is annotated $\uparrow$SUBJ=$\downarrow$, while phrasal categories bearing -LOC or -TMP tags are

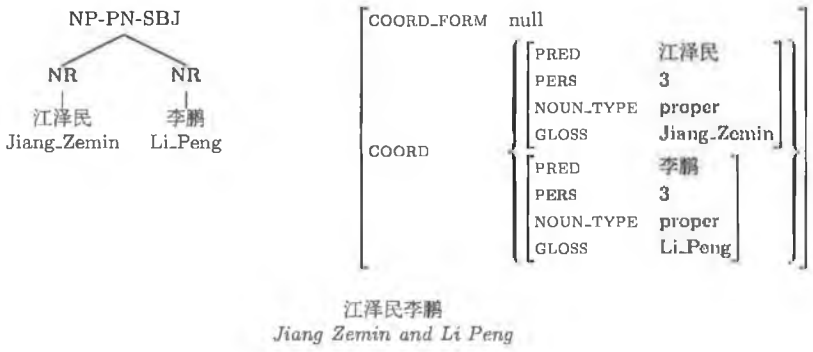


Figure 6.4: Annotation of Mandarin Chinese co-ordinate structure with no co-ordinating conjunct

annotated as adjunct set elements, $\downarrow \in \uparrow$ADJN. A small amount of over-generalisation is accepted within the first two annotation algorithm modules to allow a concise statement of linguistic generalisations. In the final Catch-All and Clean-Up module some annotations are overwritten to counter this problem and to systematically correct other potential feature clashes.

6.4.2 Annotation Algorithm Evaluation

The annotation algorithm is applied to each CTB tree and assigns functional annotations to nodes in CTB trees. The resulting annotations are collected, passed to a constraint solver and LFG f-structures are generated. The f-structures are evaluated quantitatively and qualitatively.

6.4.2.1 Quantitative Evaluation: Fragmentation

Following Levy and Manning (2003), in our experiments we split the 4,183 trees of CTB version 2.0 into the development set of 352 trees, the test set of 348 trees and the training set of 3,483 trees. The annotation algorithm achieves good coverage for the CTB with 95.123% of CTB trees receiving a single connected and covering f-structure. Table 6.3 provides a quantitative evaluation of the f-structures automatically acquired by the annotation algorithm. Feature clashes in the annotation of 3 trees (0.072% of the CTB) result in no f-structure being produced for those sentences. Multiple f-structure fragments, caused by nodes which are left unannotated by the annotation algorithm, are generated

for 201 trees (4.805%).

FRAGMENTS (#)	SENTENCES (#)	SENTENCES (%)
0	3	0.072
1	3979	95.123
2	184	4.398
3	13	0.311
4	2	0.048
7	1	0.024
9	1	0.024

Table 6.3: Quantitative Evaluation of Mandarin Chinese Annotation Algorithm

6.4.2.2 Qualitative Evaluation: Dependency Evaluation against a Gold Standard

The annotation algorithm and the acquired f-structures play an important role in the extraction of wide-coverage, probabilistic lexical resources and LFG parsing technology and need therefore to be of a high standard. To measure annotation quality a gold standard set of 50 trees were randomly selected from the 348-sentence CTB test set. Following the methodology of Cahill et al. (2002c) and King et al. (2003), the 50 gold standard trees were automatically annotated by the f-structure annotation algorithm. The f-structure annotations were then manually corrected, extended and checked over a number of iterations to create the gold standard set of f-structures.

Annotation quality is measured in terms of precision and recall against dependencies derived from the gold standard f-structures. Using the evaluation methodology and software presented by Crouch et al. (2002) and Riezler et al. (2002), the gold standard f-structures and the f-structures generated by the annotation algorithm were translated into dependency triples and evaluated. The automatic f-structure annotation algorithm achieves an f-score of 90.91% for all grammatical functions and 83.79% for preds-only f-structures (Table 6.4).

	PRECISION (%)	RECALL (%)	F-SCORE (%)
All Grammatical Functions	90.13	91.70	90.91
Preds-Only	81.44	86.29	83.79

Table 6.4: Qualitative Evaluation of Mandarin Chinese Annotation Algorithm

Table 6.5 provides a breakdown of annotation results by feature name for all grammatical functions. Note that a number of features (CLASSIFIER and OBL) have been added manually to the gold standard but are currently not supported by the annotation algorithm, while OBJ2 is produced by the annotation algorithm but does not occur in the gold standard.

DEPENDENCY	PRECISION (%)	RECALL (%)	F-SCORE (%)
ADJUNCT	91	85	88
APP	75	100	86
CLASSIFIER	0	0	0
COMP	22	43	29
COORD	85	99	91
DET	50	100	67
NOUN_TYPE	100	100	100
NUMBER_TYPE	100	100	100
OBL	0	0	0
OBJ	74	91	81
OBJ2	0	0	0
OBL	0	0	0
PERS	100	100	100
POSS	98	90	94
QUANT	58	58	58
SUBJ	84	83	84
TOPIC	100	100	100
XCOMP	70	82	76

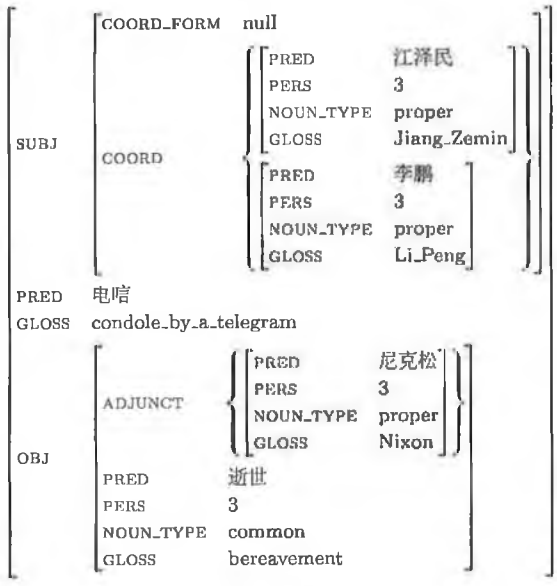
Table 6.5: Quantitative Evaluation of Mandarin Chinese Annotation Algorithm by feature name for all grammatical functions

6.5 Extraction of Semantic Forms

The semantic form extraction methodology presented by O'Donovan et al. (2004) can be applied to the f-structures automatically acquired from the CTB. LFG subcategorisation requirements are expressed at f-structure level in terms of semantic forms. A semantic form of type PRED<SUBJ, OBJ> states that the predicate PRED locally requires a SUBJ(ect) and an OBJ(ect) grammatical function. In this example, <SUBJ, OBJ> is the semantic frame type. LFG distinguishes between subcategorisable (arguments: SUBJ, OBJ, OBJ2, COMP, XCOMP, etc.) and non-subcategorisable grammatical functions (adjuncts: ADJN, APP, etc.). If the f-structures generated by the automatic f-structure annotation algorithm on the treebank trees are of good quality, then reliable semantic forms can be extracted following the methodology presented by O'Donovan et al. (2004): "for each f-structure, for each level of embedding, determine the local PRED and collect all subcategorisable

grammatical functions present at that level".

The semantic form extraction algorithm was applied to the automatically f-structure-annotated CTB. Figure 6.5 provides the f-structure produced by the annotation algorithm for the CTB tree in Figure 6.1 for the sentence 江泽民李鹏电唁尼克松逝世 (*Jiang Zemin and Li Peng condoled the bereavement of Nixon by a telegram*). Figure 6.5 also shows the semantic form extracted from the f-structure for this sentence: 电唁<SUBJ, OBJ>. A total of 10,479 semantic form tokens with 26 distinct frame types were extracted from the f-structure-annotated CTB. There are 2,510 verbal semantic forms which occur with all 26 distinct frame types (Table 6.6).



Semantic form: 电唁<SUBJ, OBJ>

江泽民李鹏电唁尼克松逝世

Jiang Zemin and Li Peng condoled the bereavement of Nixon by a telegram

Figure 6.5: Example f-structure and semantic form acquired from CTB

	TOKENS	TYPES
All forms	10469	26
Verbal	2510	26
Nominal	6227	4
Adjectival	715	1
Adverbial	579	1

Table 6.6: Semantic forms extracted from CTB

The quality of the extracted Chinese semantic forms has not been evaluated. An extensive evaluation of semantic form quality against an independent linguistic resource, following the evaluation of English semantic forms of O'Donovan et al. (2004), will be an important future extension of this research.

6.6 Parsing Experiments

6.6.1 Methodology

The automatic f-structure annotation algorithm and semantic forms extracted from acquired f-structures, following the methodology of O'Donovan et al. (2004, 2005a), are core components of the parsing technology presented by Cahill et al. (2004b). We use the Mandarin Chinese annotation algorithm in both the pipeline and integrated parsing models to parse raw text from the CTB test set into proto-f-structures. The BitPar parsing software of Schmid (2004) was used with both models.

In the pipeline parsing model, a PCFG is extracted from the CTB to parse unseen text. The parse trees are annotated by the automatic f-structure annotation algorithm and the f-structure annotations are passed to a constraint solver to produce f-structures. In the integrated model, the automatic f-structure annotation algorithm annotates the CTB. An annotated PCFG, which combines CTB categories and the f-structure annotations provided by the annotation algorithm into monadic categories for grammar extraction and parsing, is then extracted from the f-structure-annotated CTB. Raw text is parsed with the annotated PCFG to produce f-structure-annotated parse trees. The f-structure annotations are collected from the parse trees and passed to the constraint solver to produce f-structures.

Two PCFGs were extracted for the pipeline model. PCFG-F consists of nodes with CTB categories and CTB functional tags. PCFG-P-F expands PCFG-F with the parent transformation (Johnson, 1999). The parent transformation annotates each phrasal node with its parent category, encoding useful contextual information. Two annotated PCFGs were extracted for the integrated model, one with the parent transformation (PCFG-P-A). The annotated PCFGs both contain CTB categories, but not CTB functional tags

as this (and further) information is already encoded in the annotations provided by the annotation algorithm. In each case the usual pre-processing steps are performed prior to grammar extraction: deletion of empty nodes and cyclic unary productions (Levy and Manning, 2003). Table 6.7 compares the four grammars.

Parsing Model	Pipeline		Integrated	
Grammar	PCFG-F	PCFG-P-F	PCFG-A	PCFG-P-A
CTB Categories	✓	✓	✓	✓
CTB Functional Tags	✓	✓	✗	✗
Parent Transformation	✗	✓	✗	✓
F-Structure Annotations	✗	✗	✓	✓

Table 6.7: Grammars Extracted for Mandarin Chinese Parsing

6.6.2 Evaluation

6.6.2.1 Experiment 1 (Tree-Based Evaluation)

Following the experimental setup of Chiang and Bikel (2002) and Levy and Manning (2003), experiment 1 evaluates the CFG parse trees produced by each grammar against the original trees for the 300 sentences of length <= 40 in the 348 sentence CTB test set. Table 6.8 presents f-scores for labelled and unlabelled bracketings using evalb (Sekine and Collins, 1997) as well as the number of rules and coverage statistics for each grammar. All four grammars produce a parse for each of the 300 sentences in this experiment. In the pipeline model, PCFG-P-F outperforms PCFG-F as expected. The parent transformation increases grammar size and improves parse quality by approximately 2.5%. The pipeline model grammars are outperformed by both integrated model grammars. The addition of the parent transformation to the f-structure-annotated PCFG-A again results in an increase of approximately 2.5% in parse quality. PCFG-P-A is the best-performing grammar, with a labelled f-score of 81.77%, which outperforms the previous best reported labelled f-scores of 78.8% by Levy and Manning (2003) and 79.9% by Chiang and Bikel (2002).

6.6.2.2 Experiment 2 (Dependency Evaluation against Manually Corrected Gold Standard)

Experiment 2 evaluates the f-structures generated by our grammars against the manually corrected 50 gold standard f-structures for trees randomly selected from the CTB test

Parsing Model	Pipeline		Integrated	
Grammar	PCFG-F	PCFG-P-F	PCFG-A	PCFG-P-A
#Rules	3508	6479	3406	7234
#Parses	300	300	300	300
Labelled F-Score	76.03	78.78	79.07	81.77
Unlabelled F-Score	77.11	79.55	79.73	82.29

Table 6.8: Tree-based Results for CTB Parsing (Experiment 1)

set. Results are calculated as f-scores using the triple-based dependency encoding and evaluation software of Crouch et al. (2002) and Riezler et al. (2002). Table 6.9 provides f-score results for both preds-only and all grammatical functions and also indicates the percentage of sentences receiving at least one f-structure fragment. Preds-only is a stricter measure than the evaluation of all grammatical functions as it removes 'minor' feature-value pairs, e.g. person information, which tend to be associated with the correct local PRED even if the PRED itself is misattached in the f-structure. The best-performing grammar in this experiment is PCFG-P-F from the pipeline model with an f-score of 83.89% for all grammatical functions, reversing the trend of the tree-based evaluation reported in Experiment 1. This unexpected result occurs because the annotations on the parse-tree produced by PCFG-P-A for one of the gold standard sentences could not be resolved to form an f-structure. PCFG-P-A achieves an f-score of zero for this sentence which reduces its overall f-score. As in Experiment 1, the addition of the parent transformation improves the grammars in both models.

Parsing Model	Pipeline		Integrated	
Grammar	PCFG-F	PCFG-P-F	PCFG-A	PCFG-P-A
All Grammatical Functions	80.11	83.89	81.10	82.12
Preds-only	63.06	70.52	65.63	68.74
Fragmentation	100	100	100	98

Table 6.9: Dependency-based Results for CTB Parsing (Experiment 2)

6.6.2.3 Experiment 3 (Dependency Evaluation against Automatically Annotated Treebank Trees)

Experiment 3 evaluates the f-structures generated by our grammars for the full 348 sentence CTB test set against the f-structures acquired by the automatic f-structure annotation algorithm from the original CTB trees for the same sentences using the triple-based

dependency encoding and evaluation software from Crouch et al. (2002) and Riezler et al. (2002). Table 6.10 provides the f-score results for this experiment.

The overall results in experiment 3 are higher than in experiment 2 which is to be expected as evaluation against a manually corrected and extended gold standard is more taxing than evaluation against the automatically f-structure-annotated original treebank trees. PCFG-P-A in the integrated model, the best-performing grammar in the tree-based evaluation, generates f-structures for 95.96% of the sentences in the test set, while the other three grammars have full coverage. In spite of this, PCFG-P-A achieves similar results to the other grammars. The relatively small size of the training set hampers the performance of PCFG-P-A in the dependency-based experiments. Of all four grammars, PCFG-P-A contains the most detailed CFG rules with both f-structure annotations and parent information. The training data does not contain occurrences of all the CFG rules required for PCFG-P-A to attain 100% coverage of the test data. Scaling the annotation algorithm to the larger CTB version 5.0 should see the trends of experiment 1 repeated in experiments 2 and 3 as this sparse data problem is alleviated.

Parsing Model	PIPELINE		INTEGRATED	
Grammar	PCFG-F	PCFG-P-F	PCFG-A	PCFG-P-A
All Grammatical Functions	84.44	85.86	85.79	84.18
Preds-only	70.05	72.70	73.05	72.80
Fragmentation	100	100	100	95.96

Table 6.10: Dependency-based Results for CTB Parsing (Experiment 3)

6.7 Summary and Future Work

This chapter has presented the application of the technology introduced in the previous chapters of this thesis to the CTB to induce wide-coverage LFG resources for Mandarin Chinese. Linguistic generalisations for Mandarin Chinese were formed to seed the automatic f-structure annotation algorithm. The proto-f-structures acquired from the CTB by the annotation algorithm were quantitatively and qualitatively evaluated. For 95.123% of the CTB training set trees, the annotation algorithm generates a single covering and connected f-structure, while 4.805% are associated with more than one f-structure fragment. The algorithm fails to produce an f-structure fragment for 0.072% of all training set

trees. A total of 10,479 semantic form tokens with 26 distinct frame types were extracted from these proto-f-structures. Verbal semantic forms account for 2,510 tokens instantiating all 26 frame types. The methodology for extracting four grammars incorporating the Mandarin Chinese annotation algorithm was described. Three experiments were performed to evaluate the performance of the grammars. The best-performing grammar on the tree-based evaluation was PCFG-P-A which achieves a labelled f-score of 81.77%, outperforming the previous best reported labelled f-scores of 78.8% by Levy and Manning (2003) and 79.9% by Chiang and Bikel (2002). PCFG-P-F performed best in both dependency-based evaluations achieving an f-score of 83.89% for all grammatical functions against the 50-sentence manually corrected gold standard f-structures and an f-score of 85.86% for all grammatical functions against the automatically annotated full CTB test set.

In comparison with the acquisition of wide-coverage LFG resources for English described in Chapters 2 to 6, a relatively short amount of time has been spent on acquiring Mandarin Chinese resources from the CTB. The results, while encouraging, can be improved significantly given further concerted research effort. Outlined below are several tasks which would extend and improve the quality of the automatically acquired LFG resources for Mandarin Chinese described in this chapter:

- Extend the feature set used by the annotation algorithm to produce a more detailed f-structure analysis.
- Implement the Traces module of the annotation algorithm to capture long distance dependencies at f-structure level and produce proper f-structures.
- Extend the size of the gold standard to allow a more extensive evaluation.
- Update the gold standard f-structures to reflect the more detailed proper f-structures produced by the annotation algorithm.
- Scale the annotation algorithm to CTB version 5.0 which is now available containing 507,222 words in 18,782 sentences.

- Evaluate the acquired f-structures and parser output against an independent resource, e.g. Chinese Propbank (Xue and Palmer, 2003).
- Evaluate the extracted semantic forms.
- Investigate the porting of the technology to other Chinese treebanks, e.g. the Academia Sinica treebank (Chen et al., 2003).

The results reported in this chapter and previous experience with inducing wide-coverage LFG resources for English suggest that the treebank-based, grammar acquisition methodology is attractive as it succeeds in generating multi-lingual wide-coverage resources at a much faster rate than traditional hand-coding of similar resources.

Chapter 7

Conclusions

7.1 Thesis Contributions

This thesis presents the development and evaluation of an automatic LFG f-structure annotation algorithm which is a core component in a larger project (Burke et al., 2004b) for large-scale lexicon and grammar development, addressing the knowledge acquisition bottleneck familiar from traditional rule-based approaches to NLP and AI.

The work presented in this thesis has:

- reviewed the basic automatic f-structure annotation algorithm of McCarthy (2003) and provided an extensive overhaul, further development and extension of the annotation algorithm.
- corrected and standardised the DCU 105 gold standard and evaluated the f-structures produced by the annotation algorithm against this gold standard.
- evaluated the annotation algorithm against the larger PARC 700 Dependency Bank (King et al., 2003) by developing conversion software to overcome the systematic differences in analysis between the automatically acquired f-structures and PARC 700 dependencies.
- evaluated the annotation algorithm against PropBank (Kingsbury and Palmer, 2002) by automatically converting the PropBank semantic annotations for Section 23 of

Penn-II into triples format and developing conversion software to map the automatically acquired f-structures into the same format.

- acquired grammars and lexical resources for Mandarin Chinese from the Penn Chinese Treebank (CTB) using a generic version of the annotation algorithm, seeded with linguistic generalisations for Mandarin Chinese.

As a result of the extensive overhaul, further development and extension of the basic annotation algorithm of McCarthy (2003), the revised annotation algorithm provides increased speed, coverage, granularity and quality. The algorithm has become more complex to provide a more fine-grained f-structure analysis. Despite this increased complexity, a 4-fold reduction in processing time has been achieved, thereby enabling faster development than heretofore. Maintaining wide coverage while providing a more fine-grained analysis is difficult task, yet the algorithm produces a single covering and connected f-structure for 99.8% of all Penn-II trees, an increase of 0.39% on the coverage provided by the more coarse-grained annotation algorithm of McCarthy (2003). An intensive manual review of the DCU 105 produced a corrected, more standardised and fine-grained gold standard set of f-structures. Using the evaluation methodology and software of Crouch et al. (2002) and Riezler et al. (2002), the revised annotation algorithm achieved f-scores of 96.93% for all grammatical functions and 94.28% for preds-only against the reviewed DCU 105. These results show that f-structure quality has improved significantly when compared with the annotation algorithm of McCarthy (2003) which achieved f-scores of 94.11% and 90.86% for all grammatical functions and preds-only, respectively, against the original, more coarse-grained DCU 105.

Conversion software was developed to overcome systematic differences in linguistic analysis, feature geometry and nomenclature between the PARC 700 dependencies and the automatically acquired f-structure representations. The main purpose of this software was to allow a fair evaluation of the annotation algorithm against the PARC 700 Dependency Bank: the automatically acquired and mapped f-structures achieve an f-score of 87.33% against the PARC 700 test set for the feature set of Kaplan et al. (2004). However, the conversion software can also be applied to produce f-structures for the entire Penn-II treebank in the style of those generated by the hand-crafted grammars developed in the

ParGram project (Butt et al., 2002) underlying the PARC 700 dependencies.

An automated process for the extraction of a dependency-format gold standard from PropBank semantic annotations was developed to facilitate the evaluation of the annotation algorithm against PropBank. This process was applied to the test set (WSJ Section 23 of Penn-II). The Penn-II trees for the test set were automatically annotated to produce LFG f-structures, from which triples were extracted. Conversion software was developed to map these triples to produce PropBank-style semantic annotations in dependency format. WSJ section 24 of Penn-II and PropBank was used as the development set for the conversion software. An f-score of 76.58% was achieved against PropBank for the test set.

A relatively short amount of time was spent on the acquisition of wide-coverage Mandarin Chinese grammatical and lexical resources from the CTB. The generic annotation algorithm was seeded with linguistic generalisations for Mandarin Chinese to acquire proto-f-structures from the CTB (Xue et al., 2002). For 95.123% of the CTB training set trees, the annotation algorithm generates a single covering and connected f-structure. A total of 2,510 verbal semantic form tokens with 26 distinct frame types were extracted from these proto-f-structures. Of the extracted and evaluated grammars, the best-performing grammar on the tree-based evaluation was PCFG-P-A which achieves a labelled f-score of 81.77%, outperforming the previous best reported labelled f-scores of 79.9% by Chiang and Bikel (2002). These results suggest that the treebank-based, grammar acquisition methodology is attractive as it succeeds in generating multi-lingual wide-coverage resources at a much faster rate than traditional hand-coding of similar resources.

7.2 Applications

7.2.1 Current Applications

This section presents current applications of the automatic f-structure annotation algorithm. I have contributed to the development of these applications by adapting and improving the annotation algorithm to incorporate feedback from the respective evaluation processes. I wrote the PARC 700 conversion software (Chapter 4) which allows the parsing technology to be evaluated against the PARC 700 and comparisons with the resources

created by other research groups to be made.

7.2.1.1 Lexicon Acquisition

Modern unification-based NLP systems depend upon wide-coverage lexical resources. Automatic lexicon acquisition is an attractive option as manually constructing lexical resources is an error-prone and expensive process and it is very difficult to achieve full coverage for unrestricted text. Van Genabith et al. (1999) presents a methodology for acquiring subcategorisation frames from f-structures (automatically acquired from treebanks):

> Given a set of subcategorisable grammatical functions, for each f-structure and for each level of embedding in those f-structures, determine the PRED value at that level and collect the subcategorisable grammatical functions present at that level.

This subcategorisation frame extraction algorithm has been applied to the f-structures acquired by the annotation algorithm from Penn-II (O'Donovan et al., 2004) and Penn-III (O'Donovan et al., 2005a). Subcategorisation frames are acquired automatically from the f-structures without any pre-definition of frame types and are evaluated extensively against COMLEX.

The subcategorisation frame extraction algorithm was applied to the automatically f-structure-annotated Penn-III treebank producing frames for 4,362 distinct verb lemma types. 15,166 distinct subcategorisation frame types (lemma-frame pairs) were extracted for those verbs, of which 4,128 were marked as passive. Including specific prepositions for OBL arguments and particles in the subcategorisation frame extraction procedure produces 21,005 distinct subcategorisation frames, 5,005 of which are passive. Experiments were performed to evaluate the quality of the extracted lexical resource against COMLEX (Macleod et al., 1994), a hand-crafted lexicon containing 138 distinct frame types for verbs. The experiments evaluated the 3,529 active verbs that are common to both resources in what is, to our knowledge, the largest and most complete evaluation of automatically acquired English subcategorisation frames. Our experiments follow the evaluation procedure for

(a parsing-based acquisition of) subcategorisation frames for 3,000 active German verbs by Schulte im Walde (2002), the only evaluation for acquired subcategorisation frames comparable in size to ours, as far as we are aware.

O'Donovan et al. (2005a) and O'Donovan (2006) describe the mappings required to perform the evaluation against COMLEX and provides a detailed analysis of the results. Relative thresholds were placed on the conditional probabilities of the automatically extracted frames to create two sets of experiments. Frames with probabilities of ≤ 0.01 and 0.05 were filtered out. The experiments were further parameterised to filter/maintain (i) OBL arguments, (ii) specific prepositions associated with OBLs and (iii) specific particles in the automatically extracted frames. The extracted frames with OBL arguments, but without specific prepositions and particles, achieve f-scores of 63.6% and 62.2% against COMLEX with thresholds of 1% and 5% respectively.

7.2.1.2 Parsing Technology

The annotation algorithm and the probabilistic subcategorisation frames extracted from the f-structure-annotated Penn-II are core components in two parsing architectures (Cahill et al., 2004b) which generate probabilistic approximations of LFG grammars (Figure 7.1). The pipeline architecture extracts PCFGs from Penn-II or uses history-based c-structure parsers trained on Penn-II (Charniak, 2000; Bikel, 2002). Raw text is parsed with this grammar and the resulting parse trees are automatically annotated with LFG f-structure equations. The integrated model first automatically annotates Penn-II with LFG f-structure equations. An annotated PCFG (A-PCFG) is extracted from the annotated treebank. Non-terminal symbols in the A-PCFG combine Penn-II syntactic categories with LFG f-structure annotations, e.g. NP[$\uparrow$OBJ=$\downarrow$] $\rightarrow$ JJ[$\downarrow\in\uparrow$ADJUNCT] NNS[$\uparrow$=$\downarrow$]. The A-PCFG parses raw text producing f-structure-annotated parse trees.

The f-structure equations on the annotated parse trees produced by both architectures form proto-f-structures, with long distance dependencies (LDDs) unresolved. Cahill et al. (2004b) presents a methodology for resolving LDDs to produce proper f-structures. The conversion software presented in Chapter 4 is used with both parsing architectures to produce PARC 700-style dependencies.

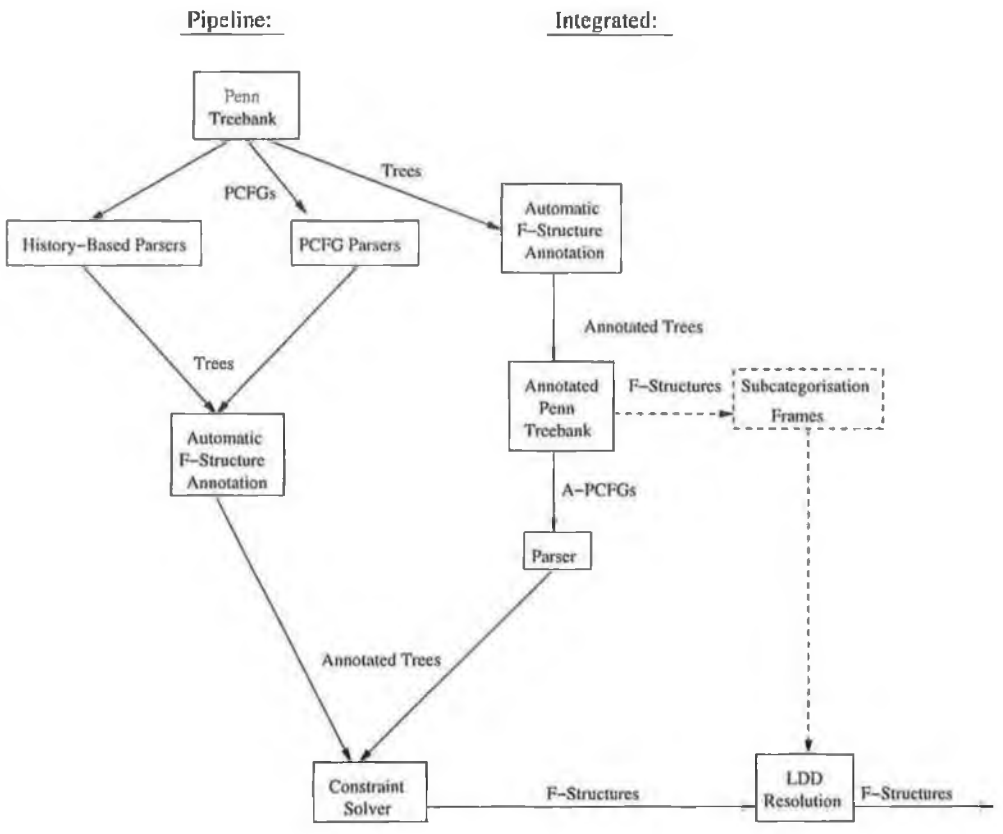


Figure 7.1: Parsing Architectures

Following the experimental set-up of Kaplan et al. (2004), with the same PARC 700 development and test sets of 140 and 560 sentences, the test set sentences were parsed using Bikel's (2002) retrained parser in the pipeline model, using the PARC 700 conversion software for evaluation. An f-score of 83.08% was achieved against the PARC 700 using the feature set of Kaplan et al. (2004). The parsing technology outperforms the hand-crafted grammar and XLE-based system of Kaplan et al. (2004), which reported an f-score of 79.6% for this experiment. The difference in results was shown to be statistically significant using the Approximate Randomisation Test (Noreen, 1989).

7.2.2 Possible Future Applications

7.2.2.1 Question Answering

Question Answering (QA) is a major natural language processing research field. A typical QA system takes a question expressed in natural language and seeks an answer from a collection of available documents. The correct interpretation of the question is a key issue for all QA systems. Judge et al. (2005) investigate whether the annotation algorithm presented in this thesis and the parsing technology of Cahill et al. (2004b) can be ported to the ATIS corpus (Hemphill et al., 1990). The ATIS corpus is a transcription of spoken dialogue with an automated air travel information system which presents a different style of language from the WSJ newswire texts of Penn-II. Judge et al. (2005) conclude from their experiments that the annotation algorithm is robust with respect to domain variance indicating that the parsing technology of Cahill et al. (2004b) is of potential value to a question answering system.

7.2.2.2 Text Condensation

The parsing technology of Cahill et al. (2004b) could be applied to the task of text condensation. A post-processing module would condense the f-structures produced by the parsing technology by removing "non-governable" grammatical functions including adjunct sets, apposition sets and relative clauses. These functions would have to be analysed to avoid deleting f-structure information indicating negation. A generation module would be required to produce the condensed raw text output from the resulting f-structure.

7.2.2.3 Multi-Document Summarisation

Multi-Document summarisation is a further possible application of the parsing technology of Cahill et al. (2004b). F-structures produced by the parsing technology for the sentences of multiple documents could be analysed to extract repeated or shared f-structures. A generation module could be applied to these core f-structures to produce sentences which summarise the contents of the documents. This application would be particularly useful for summarising the content of documents returned by a search engine.

7.3 Future Work

7.3.1 Overview

The main future goal for this project is the ongoing improvement and extension of the annotation algorithm as this results in better quality lexical and grammatical resources. Feedback from the syntax-based DCU 105 and PARC 700 evaluation processes and linguistic information derived from the Penn-II annotation guidelines (Bies et al., 1995) have driven much of the significant improvements to f-structure quality which have been made in this thesis. PropBank provides a much larger development and evaluation resource. This section outlines some possibilities for making greater use of this resource. The future development of the automatically acquired Mandarin Chinese resources is also described. The development of question answering, text condensation and multi-word document summarisation applications are further possible extensions of the work presented in this thesis.

7.3.2 Alternative PropBank Mapping Procedure

There are clear limitations to the improvements which can be made to the current PropBank mapping software. An alternative procedure, similar to the methodology of Miyao and Tsujii (2004), may provide a better long-term solution. A mapping from f-structure annotations to PropBank annotations could be learned from a training set of Penn-II trees. The annotation algorithm would be used to produce f-structures for the training set, from which triples would then be extracted. By aligning these automatically generated triples with their gold standard PropBank equivalents, the LFG features for each verb occurrence in the training set could be listed with their equivalent PropBank semantic roles. The passive markers of the annotation algorithm would be used to indicate whether a verb occurs with passive voice. A ranked list could be compiled for each verb of their most frequent active and passive mappings from LFG features to PropBank semantic roles.

For the test set, Penn-II trees would be automatically annotated and triples would be extracted from the resulting f-structures. The LFG features and passive markers would be retrieved from the triples for each verb occurrence. The highest-ranked LFG–PropBank mapping for that verb occurrence with the given LFG features would be retrieved and

used to map those triples to the corresponding PropBank semantic roles. Preliminary examination of this approach has shown that it is potentially a better long-term solution than our current approach.

7.3.3 Universal PropBank Gold Standard Triples

As PropBank was developed independently of any grammar formalism, it provides a platform for making more meaningful comparisons between parsing technologies than was previously possible. However, given the format of the PropBank annotations and the need to convert these annotations into a format compatible with evaluation software, currently it is not straightforward to draw clear conclusions from such comparisons. There is a need for greater standardisation and transparency in the evaluation process used to produce published results. This could be achieved through collaboration on the development and publication of a universal set of gold standard PropBank triples across a number of research groups.

7.3.4 Evaluation of Parsing Technology

This thesis has presented techniques for the evaluation of the annotation algorithm against PropBank. The ultimate goal of this work is the evaluation of the parsing technology of Cahill et al. (2004b). The conversion software for the evaluation of the annotation algorithm against the PARC 700 (Chapter 4) is more refined than the PropBank conversion software (Chapter 5). The PropBank conversion software needs to be improved or replaced by an alternative approach to allow a proper evaluation of the parsers and the annotation algorithm to be performed, as was possible against the PARC 700.

7.3.5 Mandarin Chinese Resources

The results achieved by the current Mandarin Chinese annotation algorithm have been encouraging, but can be improved significantly given further concerted research effort. The current coarse-grained f-structure analysis should be extended to produce a more detailed feature set. The Traces module of the annotation algorithm should be implemented to capture LDDs as re-entrancies at f-structure level for Mandarin Chinese. The number of

sentences in the gold standard should be increased to allow a more extensive evaluation. The gold standard f-structures should be reviewed to reflect the more detailed proper f-structures produced by an updated annotation algorithm. The algorithm should be scaled to provide coverage of CTB version 5.0 which contains 18,782 sentences. The acquired f-structures and parser output should be evaluated against an independent resource, e.g. Chinese PropBank (Xue and Palmer, 2003). An evaluation of the extracted lexical resources should also be performed. Porting the technology to other Chinese treebanks, e.g. the Academia Sinica treebank (Chen et al., 2003), would provide further interesting research possibilities.

Appendix A

Lexical Macros for Penn-II

Penn-II POS Tag	Lexical Macro
CC	↑PRED=*headword*, ↑COORD_FORM=*headword*
CD	↑PRED=*headword*
DT	↑PRED=*headword*
EX	↑PRED=*headword*, ↑FORM=*headword*
FW	↑PRED=*headword*
IN	↑PRED=*headword*, ↑PFORM=*headword*
JJ	↑PRED=*headword*
JJR	↑PRED=*headword*, ↑ADEGREE=*comparative*
JJS	↑PRED=*headword*, ↑ADEGREE=*superlative*
LS	↑PRED=*headword*
MD	↑PRED=*headword*, ↑MODAL=+
NN	↑PRED=*headword*, ↑NUM=*sg*, ↑PERS=*3*
NNP	↑PRED=*headword*, ↑NUM=*sg*, ↑PERS=*3*
NNPS	↑PRED=*headword*, ↑NUM=*pl*, ↑PERS=*3*
NNS	↑PRED=*headword*, ↑NUM=*pl*, ↑PERS=*3*
PDT	↑PRED=*headword*
POS	
PRP	↑PRED=*pro*, ↑PRON_FORM=*headword*
PRP$	↑PRED=*pro*, ↑PRON_FORM=*headword*
RB	↑PRED=*headword*
RBR	↑PRED=*headword*, ↑ADEGREE=*comparative*
RBS	↑PRED=*headword*, ↑ADEGREE=*superlative*
RP	↑PART=*headword*
SYM	↑PRED=*headword*
TO	↑TO_INF=+
UH	↑PRED=*headword*
VB	↑PRED=*headword*
VBD	↑PRED=*headword*, ↑TENSE=*past*

Penn-II POS Tag	Lexical Macro
VBG	↑PRED=*headword*, ↑PARTICIPLE=*pres*
VBN	↑PRED=*headword*, ↑TENSE=*past*
VBP	↑PRED=*headword*, ↑TENSE=*pres*
VBZ	↑PRED=*headword*, ↑TENSE=*pres*, ↑NUM=*sg*, ↑PERS=*3*
WDT	↑PRED=*pro*, ↑PRON_FORM=*headword*
WP	↑PRED=*pro*, ↑PRON_FORM=*headword*
WP$	↑PRED=*pro*, ↑PRON_FORM=*headword*
WRB	↑PRED=*pro*, ↑PRON_FORM=*headword*

Appendix B

Head-lexicalisation rules for Penn-II

Category	Direction	Ranked head candidates
ADVP	right	RB RBR RBS FW ADVP CD JJR JJS JJ NP
ADJP	right	% QP JJ VBN VBG ADJP $ JJR JJS DT FW IN RBR RBS RB
CONJP	left	CC RB IN
LST	left	LS :
PP	left	IN TO FW
PRT	left	RP
RRC	left	VP NP ADVP ADJP PP
UCP	left	CC S ADVP RB PRN
VP	left	MD VBD VBN VBZ VB VBG VBP POS VP TO ADJP JJ NP
WHADVP	left	WRB
WHPP	left	IN TO FW
NAC	right	NN NNS NNP NNPS NP NAC EX $ CD QP PRP VBG JJ JJS JJR ADJP FW
NP	right	(Any nominal phrasal or POS tag) EX $ CD PRP VBG JJ QP JJS JJR ADJP DT FW RB SYM PRP$ PRN POS
QP	right	NN $ % CD QP JJ JJR JJS DT
S	right	TO MD VBD VBN VBZ VB VBG VBP POS VP SBAR ADJP UCP NP PP-PRD ADJP-PRD NP-PRD
SBAR	right	IN S SQ SINV SBAR
SBARQ	right	SQ S SINV SBARQ
SINV	right	MD IN VBZ VBD VBP VB VP S SINV ADJP NP
SQ	right	MD VBZ VBD VBP VB VP SQ
WHADJP	right	JJ ADJP
WHNP	right	NN NNS NNP NNPS NP WDT WHADJP WHNP WP WP$ JJ JJR JJS DT CD QP WHPP

Bibliography

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

Bikel, D. (2002). Design of a Multi-lingual Parallel-processing Statistical Parsing Engine. In *Proceedings of the Human Language Technology Conference 2002*, pages 24–27, San Diego, CA.

Bresnan, J. (2001). *Lexical-Functional Syntax.* Blackwell, Oxford.

Burke, M., Cahill, A., O'Donovan, R., van Genabith, J., and Way, A. (2004a). The Evaluation of an Automatic Annotation Algorithm against the PARC 700 Dependency Bank. In *Proceedings of the Ninth International Conference on LFG*, pages 101–121, Christchurch, New Zealand.

Burke, M., Cahill, A., O'Donovan, R., van Genabith, J., and Way, A. (2004b). Treebank-Based Acquisition of Wide-Coverage, Probabilistic LFG Resources: Project Overview, Results and Evaluation. In *The First International Joint Conference on Natural Language Processing (IJCNLP-04), Workshop "Beyond Shallow Analyses - Formalisms and Statistical Modeling for Deep Analyses"*, Hainan Island, China [no page numbers].

Burke, M., Cahill, A., van Genabith, J., and Way, A. (2005). Evaluating Automatically Acquired F-Structures against PropBank. In *Proceedings of the Tenth International Conference on LFG*, Bergen, Norway, [to appear].

Burke, M., Lam, O., Cahill, A., Chan, R., O'Donovan, R., Bodomo, A., van Genabith, J., and Way, A. (2004c). Treebank-Based Acquisition of a Chinese Lexical-Functional

Grammar. In *Proceedings of the 18th Pacific Asia Conference on Language, Information and Computation (PACLIC-18)*, pages 161–172, Tokyo, Japan.

Butt, M., Dyvik, H., King, T. H., Masuichi, H., and Rohrer, C. (2002). The Parallel Grammar Project. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING 2002), Workshop on Grammar Engineering and Evaluation*, pages 1–7, Taipei, Taiwan.

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

Cahill, A., Burke, M., McCarthy, M., O'Donovan, R., van Genabith, J., and Way, A. (2004a). Evaluating Automatic F-Structure Annotation for the Penn-II Treebank. In Erhard Hinrichs and Kiril Simov, editor, *Journal of Language and Computation; Special Issue on "Treebanks and Linguistic Theories"*, pages 523–547, Dordrecht/Boston/London, The Netherlands. Kluwer Academic Press.

Cahill, A., Burke, M., O'Donovan, R., van Genabith, J., and Way, A. (2004b). Long-Distance Dependency Resolution in Automatically Acquired Wide-Coverage PCFG-Based LFG Approximations. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 320–327, Barcelona, Spain.

Cahill, A., Forst, M., Burke, M., McCarthy, M., O'Donovan, R., Rohrer, C., van Genabith, J., and Way, A. (2005). Treebank-Based Acquisition of Multilingual Unification Grammar Resources. In Emily Bender and Dan Flickinger and Frederik Fouvry and Melanie Siegel, editor, *Journal of Research on Language and Computation; Special Issue on "Shared Representations in Multilingual Grammar Engineering"*, Dordrecht/Boston/London, The Netherlands. Kluwer Academic Press (to appear).

Cahill, A., McCarthy, M., van Genabith, J., and Way, A. (2002a). Automatic Annotation of the Penn Treebank with LFG F-Structure Information. In *Proceedings of the The Third International Conference on Language Resources and Evaluation (LREC 2002):*

Workshop on Linguistic Knowledge Acquisition and Representation: Bootstrapping Annotated Language Data, pages 8–15, Las Palmas, Canary Islands, Spain.

Cahill, A., McCarthy, M., van Genabith, J., and Way, A. (2002b). Evaluating Automatic F-Structure Annotation for the Penn-II Treebank. In Erhard Hinrichs and Kiril Simov, editor, *Proceedings of the Treebanks and Linguistic Theories Workshop (TLT'02)*, pages 42–60, Sozopol, Bulgaria.

Cahill, A., McCarthy, M., van Genabith, J., and Way, A. (2002c). Parsing with PCFGs and Automatic F-Structure Annotation. In Miriam Butt and Tracy Holloway King, editor, *Proceedings of the Seventh International Conference on LFG*, pages 76–95, Stanford, CA. CSLI Publications.

Charniak, E. (1996). Tree-Bank Grammars. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 1031–1036, Menlo Park, CA.

Charniak, E. (2000). A Maximum Entropy Inspired Parser. In *Proceedings of the First Annual Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL 2000)*, pages 132–139, Seattle, WA.

Chen, K.-J., Huang, C.-R., Chen, F.-Y., Luo, C.-C., Chang, M.-C., Chen, C.-J., and Gao, Z.-M. (2003). Sinica Treebank: Design Criteria, Representational Issues and Implementation. In Abeillé, A., editor, *Building and Using Parsed Corpora*, volume 20 of *Text, Speech and Language Technology*, pages 231–248. Kluwer Academic Publishers, Dordrecht/Boston/London, The Netherlands.

Chiang, D. and Bikel, D. (2002). Recovering Latent Information in Treebanks. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING 2002)*, pages 183–198, Taipei, Taiwan.

Collins, M. (1996). A New Statistical Parser Based on Bigram Lexical Dependencies. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 184–192, Santa Cruz, CA.

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

Crouch, R., Kaplan, R., King, T. H., and Riezler, S. (2002). A Comparison of Evaluation Metrics for a Broad Coverage Parser. In *Proceedings of the The Third International Conference on Language Resources and Evaluation (LREC 2002): Workshop: Beyond PARSEVAL – Towards Improved Evaluation Measures for Parsing Systems*, pages 67–74, Las Palmas, Canary Islands, Spain.

Dalrymple, M. (2001). *Lexical-Functional Grammar.* San Diego, CA and Academic Press, London.

Frank, A. (2000). Automatic F-Structure Annotation of Treebank Trees. In Miriam Butt and Tracy Holloway King, editor, *Proceedings of the Fifth International Conference on LFG*, pages 140–160, Stanford, CA. CSLI Publications.

Gildea, D. and Hockenmaier, J. (2003). Identifying Semantic Roles Using Combinatory Categorial Grammar. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 57–64, Sapporo, Japan.

Hemphill, C., Godfrey, J., and Doddington, G. (1990). The ATIS Spoken Language Systems Pilot Corpus. In *Proceedings of DARPA Speech and Natural Language Workshop*, pages 96–101, Hidden Valley, PA.

Hockenmaier, J. (2003). Parsing with Generative Models of Predicate-Argument Structure. In *Proceedings of the 41st Annual Conference of the Association for Computational Linguistics*, pages 359–366, Sapporo, Japan.

Johnson, M. (1999). PCFG Models of Linguistic Tree Representations. *Computational Linguistics*, 24(4):613–632.

Johnson, M. (2002). A Simple Pattern-Matching Algorithm for Recovering Empty Nodes and Their Antecedents. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL 02)*, pages 136–143, Philadelphia, PA.

Judge, J., Burke, M., Cahill, A., O'Donovan, R., van Genabith, J., and Way, A. (2005). Strong Domain Variation and Treebank-Induced LFG Resources. In *Proceedings of the Tenth International Conference on LFG*, Bergen, Norway, [to appear].

Kaplan, R. and Bresnan, J. (1982). Lexical Functional Grammar, a Formal System for Grammatical Representation. In Joan Bresnan, editor, *The Mental Representation of Grammatical Relations*, pages 173–281. MIT Press, Cambridge, MA.

Kaplan, R., Riezler, S., King, T. H., Maxwell, J. T., Vasserman, A., and Crouch, R. (2004). Speed and Accuracy in Shallow and Deep Stochastic Parsing. In *Proceedings of the Human Language Technology Conference and the Fourth Annual Meeting of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL'04)*, pages 97–104, Boston, MA.

Kay, M., Gawron, J. M., and Norvig, P. (1994). *Verbmobil. A Translation System for Face-to-Face Dialog*, volume 33. CSLI Lecture Notes. Chicago University Press.

King, T. H., Crouch, R., Riezler, S., Dalrymple, M., and Kaplan, R. (2003). The PARC700 Dependency Bank. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2003): Workshop on Linguistically Interpreted Corpora (LINC 2003)*, pages 1–8, Budapest, Hungary.

Kingsbury, P. and Palmer, M. (2002). From Treebank to PropBank. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC 2002)*, pages 1989–1993, Las Palmas, Canary Islands, Spain.

Kipper, K., Dang, H. T., and Palmer, M. (2000). Class-Based Construction of a Verb Lexicon. In *Proceedings of the 17th National Conference on Artificial Intelligence and 12th Conference on Innovative Applications of Artificial Intelligence*, pages 691–696, Austin, TX.

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

Leech, G. and Garside, R. (1991). Running a Grammar Factory: on the complilation of parsed corpora, or 'treebanks'. In S. Johansson and A.-B. Stenstr om, editor, *English Computer Corpora: Selected Papers and Research Guide*, pages 15–32. Mouton de Gruyter, Berlin, Germany.

Levy, R. and Manning, C. (2003). Is it harder to parse Chinese, or the Chinese Treebank? In *Proceedings of the 41st Conference of the Association for Computational Linguistics*, pages 439–446, Sapporo, Japan.

Macleod, C., Meyers, A., and Grishman, R. (1994). The COMLEX Syntax Project: The First Year. In *Proceedings of the ARPA Workshop on Human Language Technology*, pages 669–703, Princeton, NJ.

Magerman, D. (1994). *Natural Language Parsing as Statistical Pattern Recognition.* PhD thesis, Department of Computer Science, Stanford University, CA.

Marcus, M., Kim, G., Marcinkiewicz, M. A., MacIntyre, R., Bies, A., Ferguson, M., Katz, K., and Schasberger, B. (1994). The Penn Treebank: Annotating Predicate Argument Structure. In *Proceedings of the ARPA Workshop on Human Language Technology*, pages 110–115, Princton, NJ.

Maxwell, J. and Kaplan, R. (1993). The interface between phrasal and structural constraints. *Computational Linguistics*, 19(4):571–589.

McCarthy, M. (2003). Design and Evaluation of the Linguistic Basis of an Automatic F-Structure Annotation Algorithm for the Penn-II Treebank. Master's thesis, School of Computing, Dublin City University, Dublin, Ireland.

Miyao, Y. and Tsujii, J. (2004). Deep Linguistic Analysis for the Accurate Identification of Predicate-Argument Relations. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING 2004)*, pages 1392–1397, Geneva, Switzerland.

Noreen, E. W. (1989). *Computer Intensive Methods for Testing Hypotheses: An Introduction.* Wiley, New York.

O'Donovan, R. (2006). *Large-Scale Multilingual Lexical Extraction.* PhD thesis, School of Computing, Dublin City University, Dublin, Ireland.

O'Donovan, R., Burke, M., Cahill, A., van Genabith, J., and Way, A. (2004). Large-Scale Induction and Evaluation of Lexical Resources from the Penn-II Treebank. In *Pro-*

ceedings of the 42nd Annual Meeting of the Association for Computational Linguistics, pages 368–375, Barcelona, Spain.

O'Donovan, R., Burke, M., Cahill, A., van Genabith, J., and Way, A. (2005a). Large-Scale Induction and Evaluation of Lexical Resources from the Penn-II and Penn-III Treebanks. *Computational Linguistics*, 31(3):329–365.

O'Donovan, R., Cahill, A., van Genabith, J., and Way, A. (2005b). Automatic Acquisition of Spanish LFG Resources from the CAST3LB Treebank. In *Proceedings of the Tenth International Conference on LFG*, Bergen, Norway, (to appear).

Pollard, C. and Sag, I. (1994). *Head-driven Phrase Structure Grammar.* CSLI Publications, Stanford, CA.

Riezler, S., King, T., Kaplan, R., Crouch, R., Maxwell, J. T., and Johnson, M. (2002). Parsing the Wall Street Journal using a Lexical-Functional Grammar and Discriminative Estimation Techniques. In *Proceedings of the 40th Annual Conference of the Association for Computational Linguistics (ACL-02)*, pages 271–278, Philadelphia, PA.

Sadler, L., van Genabith, J., and Way, A. (2000). Automatic F-Structure Annotation from the AP Treebank. In Miriam Butt and Tracy Holloway King, editor, *Proceedings of the Fifth International Conference on LFG*, pages 226–243, Stanford, CA. CSLI Publications.

Sampson, G. (1995). *English for the Computer.* Oxford University Press.

Schmid, H. (2004). Efficient Parsing of Highly Ambiguous Context-Free Grammars with Bit Vectors. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING 2004)*, pages 162–168, Geneva, Switzerland.

Schulte im Walde, S. (2002). Evaluating Verb Subcategorisation Frames learned by a German Statistical Grammar against Manual Defintions in the Duden Dictionary. In *Proceedings of the 10th EURALEX International Congress*, pages 187–197, Copenhagen, Denmark.

Sekine, S. and Collins, M. J. (1997). The evalb Software. URL. `http://nlp.cs.nyu.edu/evalb/`.

Xue, N., Chiou, F.-D., and Palmer, M. (2002). Building a Large-Scale Annotated Chinese Corpus. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING 2002)*, pages 1–8, Taipei, Taiwan.

Xue, N. and Palmer, M. (2003). Annotating the Propositions in the Penn Chinese Treebank. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics: 2nd SIGHAN Workshop on Chinese Language Processing*, pages 47–54, Sapporo, Japan.