EVALUATION OF AN AUTOMATIC F-STRUCTURE ANNOTATION ALGORITHM AGAINST THE PARC 700 DEPENDENCY BANK

Michael Burke, Aoife Cahill, Ruth O'Donovan, Josef van Genabith and Andy Way
National Centre for Language Technology and School of Computing,
Dublin City University, Dublin, Ireland

Proceedings of the LFG04 Conference

University of Canterbury

Miriam Butt and Tracy Holloway King (Editors)

2004

CSLI Publications

http://csli-publications.stanford.edu/

— Abstract —

An automatic method for annotating the Penn-II Treebank (Marcus et al., 1994) with high-level Lexical Functional Grammar (Kaplan and Bresnan, 1982; Bresnan, 2001; Dalrymple, 2001) f-structure representations is described in (Cahill et al., 2002; Cahill et al., 2004a; Cahill et al., 2004b; O'Donovan et al., 2004). The annotation algorithm and the automatically-generated f-structures are the basis for the automatic acquisition of wide-coverage and robust probabilistic approximations of LFG grammars (Cahill et al., 2002; Cahill et al., 2004a) and for the induction of LFG semantic forms (O'Donovan et al., 2004). The quality of the annotation algorithm and the f-structures it generates is, therefore, extremely important. To date, annotation quality has been measured in terms of precision and recall against the DCU 105. The annotation algorithm currently achieves an f-score of 96.57% for complete f-structures and 94.3% for preds-only f-structures. 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 basing design decisions on this. It is, therefore, preferable to evaluate against a more extensive, external standard. Although the DCU 105 is publicly available,[1] a larger well-established external standard can provide a more widely-recognised benchmark against which the quality of the f-structure annotation algorithm can be evaluated. For these reasons, we present an evaluation of the f-structure annotation algorithm of (Cahill et al., 2002; Cahill et al., 2004a; Cahill et al., 2004b; O'Donovan et al., 2004) against the PARC 700 Dependency Bank (King et al., 2003). Evaluation against an external gold standard is a non-trivial task as linguistic analyses may differ systematically between the gold standard and the output to be evaluated as regards feature geometry and nomenclature. We present conversion software to automatically account for many (but not all) of the systematic differences. Currently, we achieve an f-score of 87.31% for the f-structures generated from the original Penn-II trees and an f-score of 81.79% for f-structures from parse trees produced by Charniak's (2000) parser in our pipeline parsing architecture against the PARC 700.

1 Introduction

An automatic method for annotating the Penn-II Treebank (Marcus et al., 1994) with high-level Lexical Functional Grammar (Kaplan and Bresnan, 1982; Bresnan, 2001; Dalrymple, 2001) f-structure representations is described in (Cahill et al., 2002; Cahill et al., 2004a; Cahill et al., 2004b; O'Donovan et al., 2004). Annotation coverage is near complete with 99.83% of the 48K Penn-II sentences receiving a single, connected and covering f-structure. The annotation algorithm and the automatically-generated f-structures

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

are the basis for the automatic acquisition of wide-coverage and robust probabilistic approximations of LFG grammars (Cahill et al., 2002; Cahill et al., 2004a) and for the induction of LFG lexical resources (O'Donovan et al., 2004). The quality of the annotation algorithm and the f-structures it generates is, therefore, extremely important. To date, annotation quality has been measured in terms of precision and recall against the DCU 105, a set of manually constructed, gold-standard f-structures for 105 randomly selected sentences from Section 23 of the WSJ section of Penn-II. The annotation algorithm currently achieves an f-score of 96.57% for complete f-structures and 94.3% for preds-only f-structures using the evaluation methodology and software presented in (Crouch et al., 2002) and (Riezler et al., 2002).

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 basing design decisions on this. It is, therefore, preferable to evaluate against an independently constructed, more extensive, external standard. A larger well-established external standard can provide a more widely-recognised benchmark against which the quality of the f-structure annotation algorithm can be evaluated. For these reasons, we present an evaluation of the f-structure annotation algorithm of (Cahill et al., 2002; Cahill et al., 2004a; Cahill et al., 2004b; O'Donovan et al., 2004) against the PARC 700 Dependency Bank (King et al., 2003). The PARC 700 comprises 700 randomly selected sentences from Section 23 of the WSJ section of Penn-II which were parsed by a hand-coded, deep LFG, converted to dependency format (triples) and manually corrected and extended. We use the annotation algorithm of (Cahill et al., 2002; Cahill et al., 2004a; Cahill et al., 2004b; O'Donovan et al., 2004) to generate f-structures for those 700 Penn-II trees and also a subset of 560 following the experimental setup of (Kaplan et al., 2004).

Evaluation against an external standard is a non-trivial and time-consuming task, in this case due primarily to systematic differences in linguistic analysis, feature geometry and nomenclature. In order to carry out the evaluation we developed conversion software to automatically handle some, but not all, of the systematic differences (Figure 1). Before annotating Penn-II trees we deal with named entity recognition. The PARC 700 analyses certain names (e.g. 'Merrill Lynch') as complex predicates while the annotation algorithm analyses the same string fully parsed as a head ('Lynch') modified by an adjunct ('Merrill'). Our pre-processing module identifies and tags named entities in the Penn-II trees. The trees are then annotated by the f-structure annotation algorithm (Cahill et al., 2002; Cahill et al., 2004a; Cahill et al., 2004b; O'Donovan et al., 2004) and passed through three post-processing modules.

A significant number of feature names differ between the PARC 700 dependencies and the automatically-generated f-structures. The first post-processing module (Feature Geometry and Renaming) implements a mapping to establish common feature names, while also resolving some systematic structural differences between the gold standard analyses, including the analysis of oblique agents and quoted speech. A number of features in the PARC 700 are not computed by the automatic f-structure annotation algorithm, while some

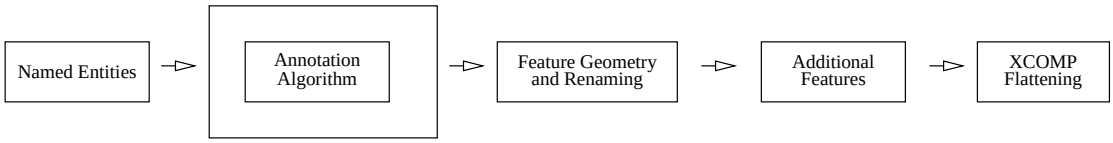

Figure 1: Conversion Software

common features have differing value ranges. The second post-processing module (Additional Features) systematically annotates the trees with as many of the missing features and values as possible.

One key difference in the f-structures automatically-generated by the annotation algorithm and those in the PARC 700 is the representation of tense and aspect information. While our annotation algorithm uses a system of cascading XCOMPs to encode this information at f-structure level, the same details are represented in the PARC 700 dependencies using a flat analysis with tense and aspect features. To cope with this, we automatically flatten the f-structures generated by the annotation algorithm (XCOMP Flattening).

Section 2 of this paper provides a brief overview of the automatic f-structure annotation algorithm. The components of the conversion software used to systematically convert the automatically-generated f-structures for evaluation against the PARC 700 are described in detail in Section 3. Section 4 outlines and analyses the results of the evaluation process. Conclusions and possibilities for future work follow in Section 5.

2 Automatic F-Structure Annotation Algorithm

This section provides a brief overview of the automatic f-structure annotation algorithm of (Cahill et al., 2002; Cahill et al., 2004a; Cahill et al., 2004b; O'Donovan et al., 2004). The generic algorithm is modular, as outlined in Figure 2, and is language and treebank-independent. The modules of the annotation algorithm must be manually seeded with linguistic information for the specific language/treebank pair, in this case the Penn-II treebank for English.

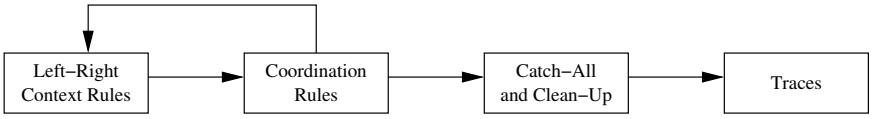

Figure 2: Annotation Algorithm modules

The first module, Left-Right Context Rules, head-lexicalises the treebank using a modified version of Magerman's scheme (Magerman, 1994). This process creates a bi-partition of each local subtree, with nodes lying in either the left or right context of the head. An annotation matrix is manually constructed for

each parent category in the treebank. For each parent category the task of matrix construction is greatly minimised by manually analysing only the most frequent CFG rule types that give at least 85% coverage of rule tokens for that parent category in the treebank. For example, only the most frequent 102 NP rule types were analysed to produce the NP annotation matrix which generalises to provide default annotations for all 6595 NP rule types in the treebank. Default annotations are read from these matrices by the annotation algorithm to annotate nodes in the left and right context of each subtree.

The annotation of co-ordinate structures is handled by a separate module in the annotation algorithm, because the relatively flat analysis of co-ordination in Penn-II would complicate the Left-Right Context Rules, making them harder to maintain and extend. Once the elements of the co-ordination set have been identified, the Left-Right Context Rules module may be re-used to provide default annotations for any remaining unannotated nodes in a co-ordinate construction.

The Catch-All and Clean-Up module provides default annotations for remaining unannotated nodes that are labelled with Penn functional tags, e.g. -SBJ. A small amount of over generation is accepted within the first two annotation algorithm modules to allow a concise statement of linguistic generalisations. Some annotations are overwritten to counter this problem and to systematically correct other potential feature clashes.

The first three modules of the annotation algorithm produce proto-f-structures which do not account for non-local dependencies. To create "proper" f-structures, the Traces module uses the wide range of trace information encoded in Penn-II to capture dependencies introduced by topicalisation, passivisation, relative clauses and questions. Figure 3 illustrates a Penn-II style tree and corresponding proto- and proper f-structures for the sentence "U.N. signs treaty, the headline said." The Trace module translates the Penn trace and co-indexation information to capture the long-distance dependency in terms of a corresponding re-entrancy in the proper f-structure which is absent from the proto-f-structure.

The annotation algorithm achieves excellent coverage for the WSJ section of Penn-II with 99.83% of the 48K sentences receiving a single connected and covering f-structure. Figure 4 provides a quantitative evaluation of the f-structures produced by the annotation algorithm. Feature clashes in the annotation of 85 trees result in no f-structure being produced for those sentences. Nodes left unannotated by the annotation algorithm in two trees caused two separate f-structure fragments for both sentences.

While achieving such wide coverage is important, the annotation quality must be of a high standard, particularly as the annotation algorithm plays a vital role in the generation of wide-coverage, probabilistic LFG parsing technology (Cahill et al., 2002; Cahill et al., 2004a) and lexical resources (O'Donovan et al., 2004). To date, annotation quality has been measured in terms of precision, recall and f-score against the DCU 105, a set of manually constructed, gold-standard f-structures for 105 randomly selected sentences from Section 23 of the WSJ part of Penn-II.

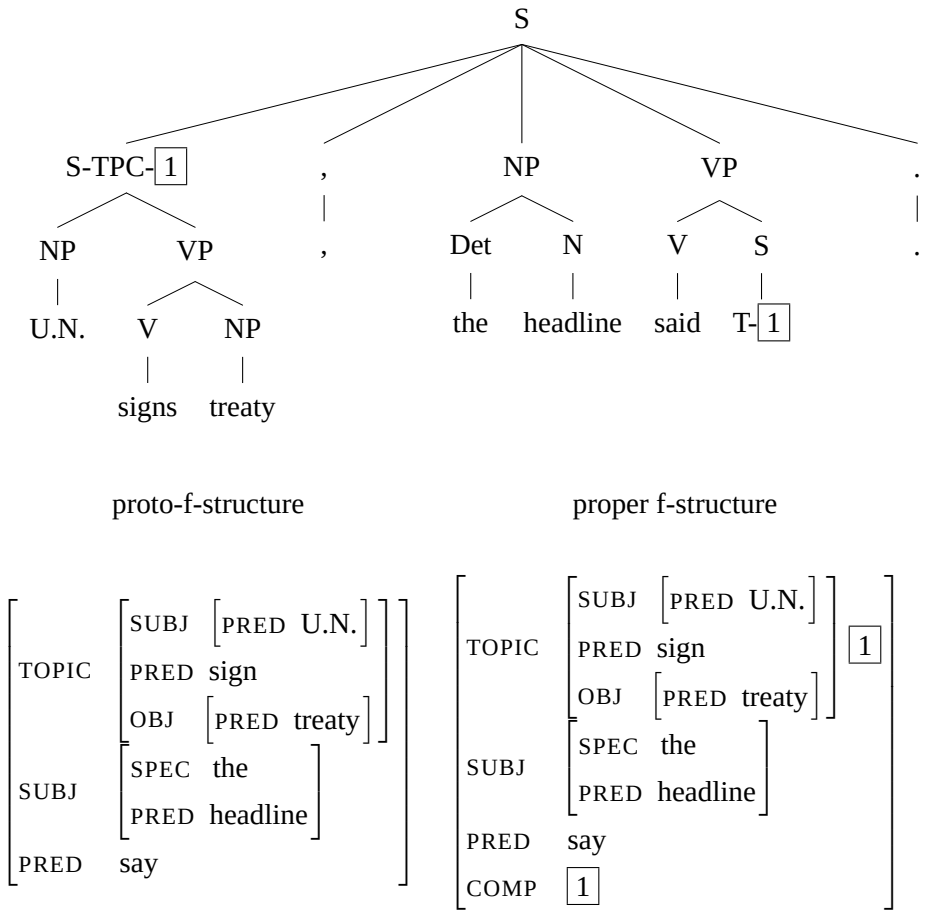

Figure 3: Penn-II style tree with LDD trace and corresponding re-entrancy in proper f-structure

Using the evaluation methodology and software presented in (Crouch et al., 2002) and (Riezler et al., 2002) the annotation algorithm currently achieves an f-score of 96.57% for complete f-structures and 94.3% for preds-only f-structures (Figure 5).[2]

As previously indicated, there are a number of problems with evaluating against a gold standard of this size. There is a risk of assuming that the gold standard is a complete and balanced representation of the linguistic phenomena in a language and basing design decisions on this. It is, therefore, preferable to evaluate against an independently constructed, more extensive, external standard which can provide a more widely-recognised benchmark for the evaluation of annotation quality.

The PARC 700 Dependency Bank (King et al., 2003) was chosen for this purpose. In an initial experiment the 700 Penn-II trees represented in the PARC 700 were annotated by the automatic f-structure

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

# f-structures	# sentences	Treebank Percentage
0	85	0.176
1	48337	99.820
2	2	0.004

Figure 4: Quantitative Evaluation

annotation algorithm. As expected, the results were poor because the DCU 105 and the automatically-generated f-structures differ substantially in linguistic analysis, feature geometry and nomenclature from the PARC 700 dependencies. An f-score of 49% was achieved which compares very poorly with the results achieved against the DCU 105, as illustrated in Figure 5.

	DCU 105		**PARC 700**
	All grammatical functions	*Preds only*	*Feature set of (Kaplan et al., 2004)*
Precision	96.58	94.53	46.32
Recall	96.55	94.07	51.99
F-Score	96.56	94.30	49.00

Figure 5: Initial qualitative evaluation against PARC 700

In order to achieve a fair evaluation, conversion software was developed to overcome some, but not all, of the systematic differences between the DCU 105 and PARC 700 representations. This software is presented in detail in Section 3.

3 Conversion Software

3.1 Introduction

The previous section provided an overview of the annotation algorithm used to generate f-structures for the WSJ section of Penn-II. An evaluation of the annotation quality against the DCU 105 was presented and the need for evaluation against a more extensive, external standard was motivated.

The chosen external standard, the PARC 700 Dependency Bank (King et al., 2003) comprises 700 randomly selected sentences from Section 23 of the WSJ section of Penn-II. The sentences were parsed by a hand-coded, deep LFG, converted to dependency format (triples) and manually corrected and extended.

This section presents the conversion software developed to overcome some of the systematic differences in linguistic analysis, feature geometry and nomenclature between the automatically-generated f-structures

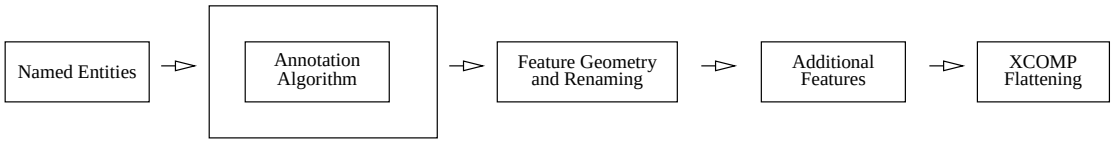

Figure 6: Conversion Software

and the PARC 700 dependency structures. An overview of the conversion software is provided in Figure 6. Penn-II trees are passed through a pre-processing module, Named Entities, before being automatically annotated by the f-structure annotation algorithm, as outlined in Section 2. The annotated trees are then modified by three post-processing modules, Feature Geometry and Renaming, Additional Features and XCOMP Flattening. All four modules of the conversion software are described in the following sections.

3.2 Named Entities

The first module of the conversion software handles differences in the analysis of named entities. The PARC 700 treats certain names (e.g. 'Merrill Lynch') as complex predicates while the annotation algorithm analyses the same string fully parsed as a head ('Lynch') modified by an adjunct ('Merrill'). Our Named Entities pre-processing module identifies and tags named entities in the Penn-II trees, allowing the annotation algorithm to produce the complex predicate analysis expected by the PARC 700 gold standard dependencies.

The identification of named entities is carried out using a list of all the named entities in the PARC 700. The string represented by each subtree is checked against this list. Identified named entities are tagged using a new category NE. A new node, labelled NE, is inserted in the tree above the part of speech tags representing the named entity. This node indicates that the words represented by its daughter nodes should be combined to form a complex predicate at f-structure level.

There are three cases for the automatic insertion of an NE node, the simplest of which occurs when an entire subtree represents a named entity. An NE node is inserted above all part of speech tags in the subtree. Figure 7 illustrates the insertion of an NE node into the subtree representing the named entity "Merrill Lynch".[3] The f-structures automatically-generated by the annotation algorithm for both the original and the pre-processed trees are provided. The NE node allows the correct complex PRED value ("merrill lynch") to be created to match with the version in the corresponding PARC 700 dependency.

A more complex case for the automatic insertion of an NE node occurs when a partial subtree represents a named entity. In such cases, a node labelled NE is inserted above the part of speech tags representing the named entity only. For example, the named entity "White House" is contained in the Penn-II subtree

[3]The conversion software converts the lemmas of the automatically-generated f-structures and the PARC 700 dependencies into lowercase for evaluation purposes.

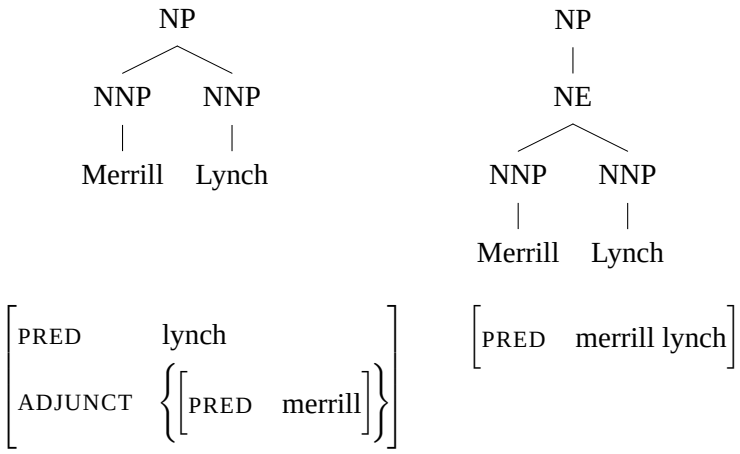

Figure 7: Simple case of named entity node insertion for the phrase

representing the string “The official White House reaction”. The modified subtree in Figure 8 shows the NE node inserted above the four part of speech nodes representing the named entity. The inserted node and the remaining part of speech nodes are now siblings.

The original annotation algorithm did not provide an annotation for the inserted NE node as no left-right context matrix contained entries for NE tags. Therefore, new left-right context entries were created for the new tag. This task was trivial, because the behaviour of named entities is similar to other nominal phrases for which left-right context entries already existed. The new entries were adapted from these existing nominal entries and allowed the inserted NE node in Figure 8 to be annotated as an adjunct.

The final and most complex case of automatic node insertion occurs when a named entity is represented by multiple subtrees. In such subtrees, the parent node is identified and all subordinate nodes, excluding part of speech nodes, are deleted, thus flattening the subtree. An NE node is then inserted as a daughter of the parent node, with the part of speech nodes representing the named entity as its daughters. The original and modified tree is illustrated in Figure 9.

3.3 Feature Geometry and Renaming

Feature Geometry and Renaming is the first post-processing module of the conversion software and is applied to trees that have first been treated for named entities and then automatically annotated by the f-structure annotation algorithm.

A significant number of feature names differ between the PARC 700 dependencies and the automatically-generated f-structures. The Feature Geometry and Renaming module implements a mapping to establish common feature names. Table 1 provides the details of this mapping. The feature names used in the

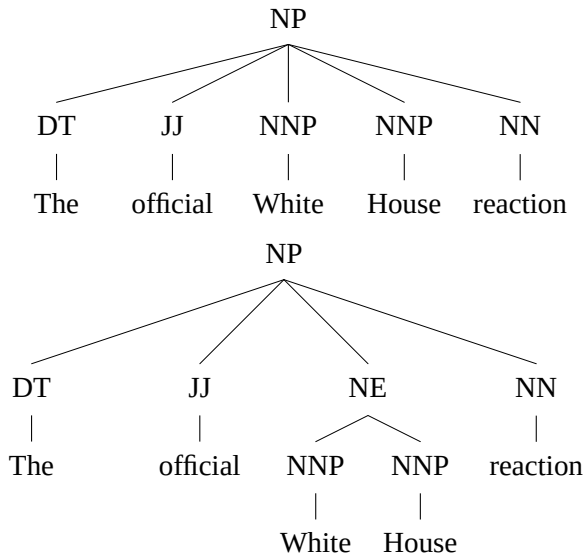

Figure 8: Named entity forming part of a local subtree

automatically-generated f-structures for determiners, particles, co-ordinated elements and interrogatives are mapped to match those present in the PARC 700. The automatic f-structure annotation algorithm does not distinguish between modifiers and other adjuncts, so the PARC 700 MOD feature is mapped to ADJUNCT. Similarly, the PARC 700 features AQUANT and NUMBER are mapped to QUANT, while OBL_COMPAR becomes OBL.

The conversion software encodes feature geometry mappings to resolve some systematic structural differences between the analyses of the DCU 105 and PARC 700 gold standards. The treatment of oblique agents in the DCU 105 and the f-structures automatically-generated by the annotation algorithm is illustrated in Figure 10(a). In order to match the PARC 700 structure Figure 10(b), the PRED value of the oblique agent must be moved and the OBJ feature removed. This is achieved by mapping the annotation of the noun phrase from “up-obj=down” to “up=down”. The feature PFORM is mapped to PCASE; a mapping which only occurs in the context of oblique agents.

The feature geometry mapping for oblique agents systematically overcomes the structural difference in analysis between the gold standards. Structural differences in the analysis of quoted speech and the distribution of shared subjects and objects into co-ordinate structures are also resolved using feature geometry mappings.

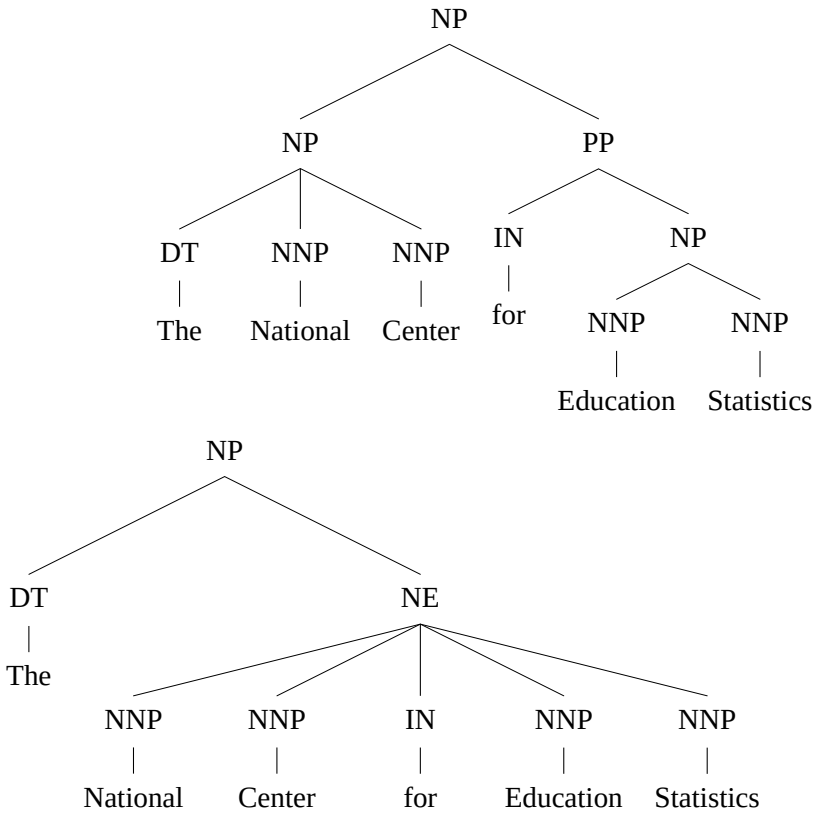

Figure 9: Named entity represented by several subtrees

3.4 Additional Features

The second post-processing module, Additional Features, annotates the trees with PARC 700 features which are not computed by the f-structure annotation algorithm. Furthermore, some features are common to both representations but with differing value ranges. This module also annotates the trees with the missing values.

The first feature to be added is NUMBER_TYPE which has the values "cardinal" and "ordinal". This module annotates all nodes labelled CD (cardinal number) with the PARC 700 NUMBER_TYPE feature. String analysis is used to identify ordinal numbers which are given the value "ordinal" for this new feature. All other CD nodes are given the value "cardinal" by default. The PARC 700 feature PRECOORD_FORM, with values such as "both" and "either", is also computed by the Additional Features module. Subtrees containing multiple CC nodes are identified, and the feature is added if the leftmost node is a CC.

The automatically-generated f-structures do not contain the feature STMT_TYPE (statement type), one of the most frequent features in the PARC 700. The value "header" is computed for this feature if the root node of the tree is a noun phrase. STMT_TYPE is added to all S nodes with the value "declarative".

The feature ADEGREE has two possible values in the automatically-generated f-structures, "compara-

DCU 105	PARC 700	Common feature name
DET	DET_FORM	DET_FORM
PART	PRT_FORM	PRT_FORM
COORD	CONJ	CONJ
FOCUS	FOCUS_INT	FOCUS_INT
ADJUNCT	MOD	ADJUNCT
OBL	OBL_COMPAR	OBL
QUANT	AQUANT	QUANT
QUANT	NUMBER	QUANT

Table 1: Feature Mapping table

$$\begin{bmatrix} \text{PRED} & \text{purchase} \\ \text{OBL\_AG} & \begin{bmatrix} \text{PFORM} & \text{by} \\ \text{OBJ} & \begin{bmatrix} \text{PRED} & \text{government} \end{bmatrix} \end{bmatrix} \end{bmatrix} \qquad \begin{bmatrix} \text{PRED} & \text{purchase} \\ \text{OBL\_AG} & \begin{bmatrix} \text{PCASE} & \text{by} \\ \text{PRED} & \text{government} \end{bmatrix} \end{bmatrix}$$

(a) Automatically-generated f-structure (b) PARC 700 representation

Figure 10: Feature geometry mapping for oblique agents

tive" and "superlative", which are provided by the lexical macros for comparative adverbs (RBR), comparative adjectives (JJR), superlative adverbs (RBS) and superlative adjectives (JJS). Every adjective in the PARC 700 has an ADEGREE feature, which may be "comparative", "superlative" or "positive", which acts as a default value. Simply annotating all adjectives (JJ) with the ADEGREE value "positive" will provide many of the missing annotations. However, in the cases where a comparative or superlative adverb modifies an adjective, the adverb's ADEGREE feature must be moved to overwrite the adjective's new "positive" ADEGREE annotation.

Figure 11 provides four f-structure fragments for the phrase "most troublesome" to illustrate this mapping. An f-structure representation of the PARC 700 dependency and the automatically-generated f-structure for this phrase are provided in the first row. The ADEGREE features are attached at different levels. The "first pass" f-structure is created when the module annotates all adjectives with the value "positive" for the ADEGREE feature. For this particular phrase, adding this feature-value pair actually increases the divergence between the automatically-generated f-structure and the PARC 700 version. Overwriting the adjective's ADEGREE feature with that of the adverb corrects this problem. The PRED value for the adverb "most" does not occur in the PARC 700 dependency, but is still present in the automatically-generated f-structure.

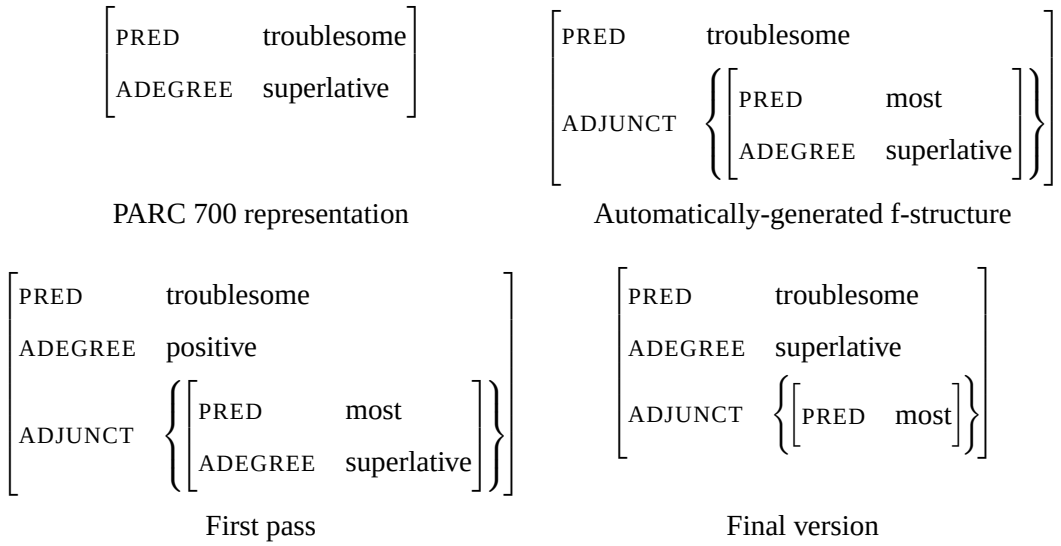

Figure 11: Adegree complications

3.5 XCOMP Flattening

The most noticeable difference between automatically-generated f-structures and PARC 700 dependencies is the representation of tense and aspect information. While our annotation algorithm uses a system of cascading XCOMPs to encode this information, as shown in Figure 12 for the sentence "Unlike 1987, interest rates have been falling this year", the PARC 700 uses a flat analysis with tense and aspect features. This final post-processing module, XCOMP Flattening, implements a systematic mapping to overcome this difference.

The first step carried out by the "XCOMP flattening" module is the identification of the correct PRED and TENSE values to maintain. The correct PRED value is that of the main verb found at the "deepest" level of the cascade of XCOMPs. The TENSE value at the "outer" level is also maintained. All other PRED and TENSE values are deleted.

Secondly, the PARC 700 aspect features, PROGressive and PERFective, are computed. Progressive aspect is represented in the automatically-generated f-structures by the PARTICIPLE feature occurring with the value "pres". If this feature-value pair is found at any XCOMP level, it is replaced by the PARC 700 feature PROG with value "+". The PERF feature is added with value "+" if the PRED value "have" is found at any XCOMP level before the "deepest" level.

The final step in this module achieves the task of flattening the XCOMP cascade, while grouping and maintaining the adjuncts from each level. The XCOMP annotation is removed from all nodes, except modals, and is replaced with the "up=down" annotation. Removing all XCOMP annotations in this manner "flattens" the f-structure. The process of unification invoked by the constraint solver groups together all the adjuncts

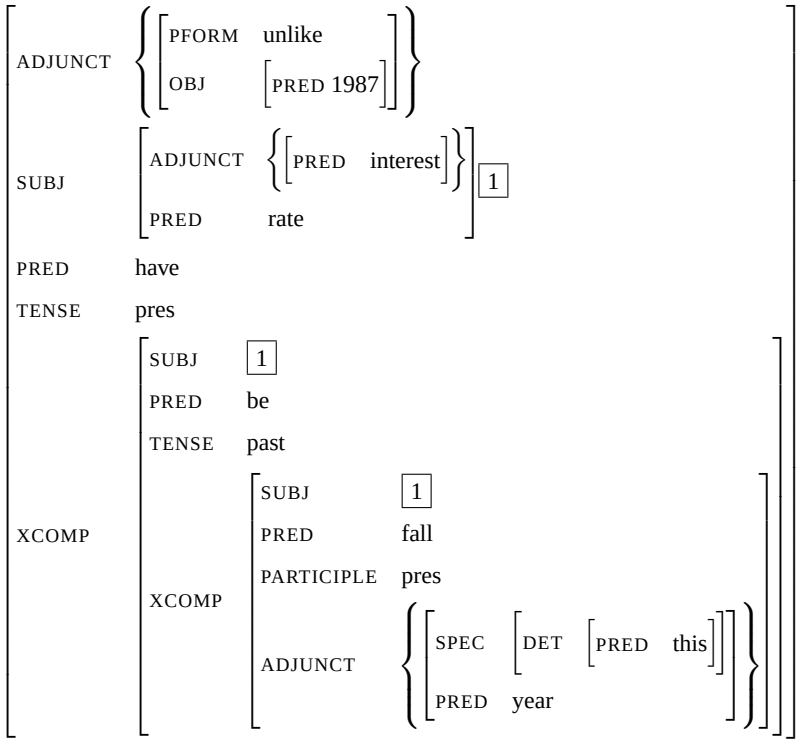

Figure 12: Cascading XCOMP example

in one set. Similarly, the information that has been maintained and added at various levels in the XCOMP cascade, e.g. PRED values and aspect features, are retained through unification. The entire process of XCOMP flattening is achieved through a process of systematically rewriting annotations on the trees and without moving any annotations.

Figure 13 provides the "flattened" f-structure produced by the conversion software for the sentence of Figure 12: "Unlike 1987, interest rates have been falling this year". The adjuncts "unlike 1987" and "this year" are both contained in a single adjunct set at sentence level in the flattened version. The "deepest" PRED value in the cascade of XCOMPs has been maintained. The "outer" TENSE is maintained, while PROG and PERF features have been added. All other feature-value pairs have been removed.

3.6 Conclusions

Section 2 provided a brief overview of the automatic f-structure annotation algorithm of (Cahill et al., 2002; Cahill et al., 2004a; Cahill et al., 2004b; O'Donovan et al., 2004). This section has described the four modules of the conversion software developed to systematically map the automatically-generated f-structures for evaluation against the PARC 700. This software was applied to the 700 sentences that comprise the PARC 700. Using the evaluation methodology and software presented in (Crouch et al., 2002) and (Riezler et al.,

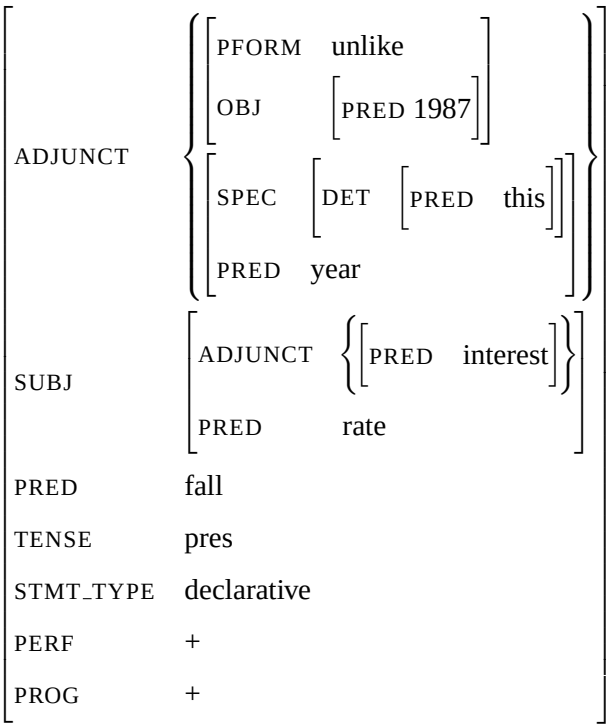

Figure 13: Flattened version of Figure 12

2002), the converted f-structures were evaluated against the PARC 700 dependencies. Section 4 provides an analysis of the results of this evaluation.

4 Evaluation

4.1 Introduction

Section 2 provided an overview of the automatic f-structure annotation algorithm of (Cahill et al., 2002; Cahill et al., 2004a; Cahill et al., 2004b; O'Donovan et al., 2004). Section 3 described conversion software which allows the annotation algorithm to be evaluated against the PARC 700. This section provides the results of the evaluation process. The results are analysed in comparison with the results achieved against the DCU 105 gold standard. The conversion software was also used to evaluate the output generated by our CFG parsing technology based on the f-structure annotation algorithm against the PARC 700. The results of this evaluation are also provided.

4.2 Qualitative Evaluation

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 converted 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 converted f-structures for the 560 sentence test set achieved an f-score of 87.31% against the PARC 700 dependencies. The f-score for all 700 sentences of the PARC 700 was 87.36%.

	PARC 700: 560 sentence test set	**DCU 105**	
	Feature set of (Kaplan et al., 2004)	*All grammatical functions*	*Preds only*
Precision	88.57	96.52	94.45
Recall	86.10	96.63	94.16
F-Score	87.31	96.57	94.30

Table 2: Evaluation Results against PARC 700 and DCU 105

Table 2 illustrates the results in terms of precision, recall and f-score[4]. The results achieved by the annotation algorithm against the DCU 105 for all grammatical functions and for preds-only are also provided. An analysis of the different results achieved against both gold standards follows.

4.3 Analysis of Results

There is a wide gap between the results achieved by the annotation algorithm when evaluated against the DCU 105 and, using the conversion software, against the PARC 700. There are a number of reasons for the poorer results against the PARC 700, most of which are related to differences between the representations used in the automatically-generated f-structures and the PARC 700 which could not be captured using the systematic mappings of the conversion software. Some of these differences will now be analysed, the first of which is the treatment of hyphenated words.

[4]Precision, Recall and F-Score were calculated according to the following equations:

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

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

$F - Score = \frac{2\ X\ Precision\ X\ Recall}{Precision + Recall}$

4.3.1 Hyphenated Words

In the PARC 700 dependencies, hyphenated words are often, but not always, split into separate lemmas, each with their own feature-value pairs. Hyphenated words remain as a single unit in the f-structures automatically-generated by the annotation algorithm. The conversion software does not attempt to split these words and crucially, hyphenated words also remain intact in the DCU 105. Therefore, the converted f-structures are penalised in the evaluation against the PARC 700 for every hyphenated word that is split.

adjunct(property, investment-grade)	adjunct(property, investment)
	adjunct(property, grade)
Automatically-generated triple	PARC 700 triples

Figure 14: Hyphenation problems

Figure 14 illustrates the PARC 700 triples and those produced by the annotation algorithm for the phrase "investment-grade property". The annotation algorithm produces one triple which does not occur in the gold standard, which in turn contains two triples that are not produced by the annotation algorithm. Thus, the annotation algorithm is penalised despite correctly producing the adjunct relationship.

Not all hyphenated words are split in this manner in the PARC 700 dependencies. Given the resources available to us, i.e. Penn tags, there is no systematic pattern which can be used to predict the PARC 700 treatment of hyphenation, so no attempt is made within the conversion software to solve this problem.

4.3.2 Penn-II POS Tagging

The annotation algorithm annotates Penn-II part of speech tags using a set of lexical macros. Singular nouns (NN) are annotated with a NUM feature with value "sg", while nodes labelled NNS receive NUM value "pl". Section 3 explained how all adjectives (JJ) receive an ADEGREE feature in the Additional Features module of the conversion software. Adjuncts in the converted f-structures may have an ADEGREE or NUM feature, the choice of which is determined entirely by the Penn-II POS tags.

The analysis of adjuncts as nominal or adjectival in the PARC 700 dependencies cannot be accurately predicted from the Penn-II POS tags. In the majority of cases the Penn-II tagging and PARC 700 analyses match, but there is a significant amount of divergence and in every such case, the converted f-structures are penalised. As with the treatment of hyphenated words, no attempt is made to solve this problem in the conversion software, as there is no systematic way of doing so using the available resources, i.e. Penn-II POS tags.

As outlined above, the annotation algorithm provides NUM annotations through the lexical macros for each Penn-II POS tag. However, named entities in the PARC 700 dependencies also receive a NUM value

which cannot be predicted from POS tags or other indicators. The conversion software provides a default "sg" NUM value for all named entities, although this is not always correct.

4.3.3 Computational Error Margins

As outlined in Section 3, the conversion software consists of five modules, one of which is the annotation algorithm itself. 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. The four additional modules required for evaluation against the PARC 700 gold standard must produce a higher computational error margin than the simpler process of evaluating against the DCU 105.

4.3.4 Characteristics of both Gold Standards

The origin of both gold standards must also impact on the results achieved by the automatically-generated f-structures. The DCU 105 was designed for the purpose of evaluating f-structures produced by the annotation algorithm and the derived parsing technology. The PARC 700, in turn, 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 III 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.

4.4 Evaluation of Parsing Technology

The conversion software described in Section 3 can also be used to evaluate the performance of the parsing technology of (Cahill et al., 2002; Cahill et al., 2004a). Two parsing architectures are presented in detail: an integrated model and a pipeline model. The best PCFG induced using the integrated model achieved an f-score of 80.33% against the 560 sentence test subset of the PARC 700. The pipeline model, using the output of Charniak's parser (Charkiak, 2000), achieved an f-score of 81.79% against the same test set. This

result is an improvement of 2.19% on the previous best published results against this test set in (Kaplan et al., 2004).

5 Conclusions

This paper has presented an evaluation of the automatic f-structure annotation algorithm of (Cahill et al., 2002; Cahill et al., 2004a; Cahill et al., 2004b; O'Donovan et al., 2004) against the PARC 700 Dependency Bank (King et al., 2003). A brief outline of the annotation algorithm was provided in Section 2 and the need for an evaluation against a larger well-established external standard was motivated. The differences in linguistic analysis, feature geometry and nomenclature between the automatically-generated f-structures and the dependency structures of the chosen gold standard, the PARC 700, necessitated the development of an automatic conversion process. The conversion software developed for the purpose of overcoming systematic differences between the representations was presented in Section 3. The results of the evaluation process were provided and analysed in Section 4. 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. Currently we achieve an f-score of 96.57% for full f-structures and 94.3% for preds-only f-structures against the DCU 105. Using our conversion software we achieve an f-score of 87.31% against the PARC 700 for the feature set of (Kaplan et al., 2004). Evaluation against an external gold standard is non-trivial and we expect improvements in the conversion software to yield corresponding improvements in the results.

The conversion software presented in this paper also allows the parsing technology of (Cahill et al., 2002; Cahill et al., 2004a) to be evaluated against the PARC 700. Currently we achieve an f-score of 81.79% using the output of Charniak's (2000) parser in our pipeline architecture, an improvement of 2.19% over the previous best result of (Kaplan et al., 2004). This is a significant development as it provides a more widely-recognised benchmark for the parser quality and allows more direct comparisons to be made with the published results of others.

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. Scaling up the Named Entities module for the identification of all named entities in the treebank is a task for further work. The evaluation of the automatically-generated f-structures against the larger PARC 700 provides many opportunities for the future improvement of the automatically-generated grammatical and lexical resources presented in (Cahill et al., 2002; Cahill et al., 2004a; Cahill et al., 2004b; O'Donovan et al., 2004).

References

Bresnan, J. 2001. *Lexical Functional Syntax*. Blackwell, Oxford.

Butt, M., H. Dyvik, T. King, H. Masuichi, and C. Rohrer. 2002. The Parallel Grammar Project. In *Proceedings of COLING-2002 Workshop on Grammar Engineering and Evaluation*, pages 1–7, Taipei, Taiwan. printed.

Cahill, A., M. Burke, R. O'Donovan, J. van Genabith, and A. Way. 2004a. Long-Distance Dependency Resolution in Automatically Acquired Wide-Coverage PCFG-Based LFG Approximations. In *Proceedings of 42nd Conference of the Association for Computational Linguistics*, pages 319–326, Barcelona, Spain. printed.

Cahill, A., M. McCarthy, M. Burke, R. O'Donovan, J. van Genabith, and A. Way. 2004b. Evaluating Automatic F-Structure Annotation for the Penn-II Treebank. In *Journal of Research on Language and Computation*. Kluwer Academic Publishers.

Cahill, A., M. McCarthy, J. van Genabith, and A. Way. 2002. Parsing with PCFGs and Automatic F-Structure Annotation. In M. Butt and T. Holloway-King, editors, *Proceedings of the Seventh International Conference on Lexical-Functional Grammar*, pages 76–95, Athens, Greece. CSLI Publications, Stanford, CA. [in press]. printed.

Charkiak, E. 2000. A Maximum-Entropy-Inspired Parser. In *Proceedings of the First Annual Meeting of the North American Chapter of the Association of Computational Linguistics*, pages 132–139, Seattle, WA. printed.

Crouch, R., R. Kaplan, T. King, and S. Riezler. 2002. A Comparison of Evaluation Metrics for a Broad Coverage Parser. In *Beyond PARSEVAL Workshop at 3rd Int. Conference on Language Resources and Evaluation*, pages 67–74, Las Palmas, Canary Islands, Spain.

Dalrymple, M. 2001. *Lexical-Functional Grammar*. San Diego, CA.; London: Academic Press.

Kaplan, R. and J. Bresnan, 1982. *The Mental Representation of Grammatical Relations*, chapter Lexical-Functional Grammar: A Formal System for Grammatical Representation, pages 173–281. MIT Press, Cambridge, MA.

Kaplan, R., S. Riezler, T. King, J. Maxwell, A. Vasserman, and R. Crouch. 2004. Speed and Accuracy in Shallow and Deep Stochastic Parsing. In *Proceedings of the Human Language Technology Conference and the 4th Annual Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 97–104, Boston, MA. printed.

King, T. H., R. Crouch, S. Riezler, M. Dalrymple, and R. M. Kaplan. 2003. The PARC 700 Dependency Bank. In *Proceedings of the 4th International Workshop on Linguistically Interpreted Corpora at the 10th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1–8, Budapest, Hungary. printed.

Magerman, D. 1994. *Natural Language Parsing as Statistical Pattern Recognition*. Ph.D. Thesis, Department of Computer Science, Stanford University, CA. printed.

Marcus, M., G. Kim, M. A. Marcinkiewicz, R. MacIntyre, M. Ferguson, K. Katz, and B. Schasberger. 1994. The Penn Treebank: Annotating Predicate Argument Structure. In *Proceedings of ARPA Human Language Technology Workshop*, pages 114–119, Plainsboro, NJ. printed.

Maxwell III, J. and R. Kaplan. 1993. The Interface between Phrasal and Structural Constraints. *Computational Linguistics*, 19(4):571–589. printed.

O'Donovan, R., M. Burke, A. Cahill, J. van Genabith, and A. Way. 2004. Large-Scale Induction and Evaluation of Lexical Resources from the Penn-II Treebank. In *Proceedings of 42nd Conference of the Association for Computational Linguistics*, pages 367–374, Barcelona, Spain.

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