Wide-Coverage Deep Statistical Parsing using Automatic Dependency Structure Annotation

Aoife Cahill
Dublin City University*

Michael Burke
Dublin City University**
IBM Center for Advanced Studies†

Ruth O'Donovan
Dublin City University**

Stefan Riezler
Palo Alto Research Center‡

Josef van Genabith
Dublin City University**
IBM Center for Advanced Studies†

Andy Way
Dublin City University**
IBM Center for Advanced Studies†

A number of researchers (Lin 1995; Carroll, Briscoe, and Sanfilippo 1998; Carroll et al. 2002; Clark and Hockenmaier 2002; King et al. 2003; Preiss 2003; Kaplan et al. 2004; Miyao and Tsujii 2004) have convincingly argued for the use of dependency (rather than CFG-tree) representations for parser evaluation. Preiss (2003) and Kaplan et al. (2004) conducted a number of experiments comparing "deep" hand-crafted wide-coverage with "shallow" treebank- and machine-learning-based parsers at the level of dependencies, using simple and automatic methods to convert tree output generated by the shallow parsers into dependencies. In this article, we revisit the experiments in Preiss (2003) and Kaplan et al. (2004), this time using the sophisticated automatic LFG f-structure annotation methodologies of Cahill et al. (2002b, 2004) and Burke (2006), with surprising results. We compare various PCFG and history-based parsers (based on Collins, 1999; Charniak, 2000; Bikel, 2002) to find a baseline parsing system that fits best into our automatic dependency structure annotation technique. This combined system of syntactic parser and dependency structure annotation is compared to two hand-crafted, deep constraint-based parsers (Carroll and Briscoe 2002; Riezler et al. 2002). We evaluate using dependency-based gold standards (DCU 105, PARC 700, CBS 500 and dependencies for WSJ Section 22) and use the Approximate Randomization Test (Noreen 1989) to test the statistical significance of the results. Our experiments show that machine-learning-based shallow grammars augmented with sophisticated automatic dependency annotation technology outperform hand-crafted, deep, wide-coverage constraint grammars. Currently our best system achieves an f-score of 82.73% against the PARC 700 Dependency Bank (King et al. 2003), a statistically significant improvement of 2.18% over the most recent results of 80.55% for the hand-crafted LFG grammar and XLE parsing system of Riezler et al. (2002), and an f-score of 80.23% against the CBS 500 Dependency

∗ Now at the Institüt für Maschinelle Sprachverarbeitung, Universität Stuttgart, Germany
∗∗ National Centre for Language Technology, Dublin City University, Dublin 9, Ireland
† IBM Dublin Center for Advanced Studies (CAS), Dublin 15, Ireland
‡ Now at Google Inc., Mountain View, CA

Submission received: 24 August 2005; revised submission received: 20 March 2007; accepted for publication 2 June 2007

Bank (Carroll, Briscoe, and Sanfilippo 1998), a statistically significant 3.66% improvement over the 76.57% achieved by the hand-crafted RASP grammar and parsing system of Carroll and Briscoe (2002).

1. Introduction

Wide-coverage parsers are often evaluated against gold-standard CFG trees (e.g., Penn-II WSJ Section 23 trees) reporting traditional PARSEVAL metrics (Black et al. 1991) of labeled and unlabeled bracketing precision, recall and f-score measures, number of crossing brackets, complete matches etc. While tree-based parser evaluation provides valuable insights into the performance of grammars and parsing systems, it is subject to a number of (related) draw-backs:

1. Bracketed trees do not always provide NLP applications with enough information to carry out the required tasks: Many applications involve a deeper analysis of the input in the form of semantically motivated information such as deep dependency relations, predicate-argument structures or simple logical forms.
2. A number of alternative, but equally valid tree representations can potentially be given for the same input. To give just a few examples: In English, VPs containing modals and auxiliaries can be analyzed using (predominantly) binary branching rules (Penn-II (Marcus et al. 1994)) or employ flatter analyses where modals and auxiliaries are sisters of the main verb (AP treebank (Leech and Garside 1991)) or indeed do without a designated VP constituent at all (SUSANNE (Sampson 1995)). Treebank bracketing guidelines can use 'traditional' CFG categories such as S, NP etc. (Penn-II) or a maximal projection-inspired analysis with IPs and DPs (Chinese Penn Treebank (Xue et al. 2004)).
3. Since a tree-based gold standard for parser evaluation must adopt a particular style of linguistic analysis (reflected in the geometry and nomenclature of the nodes in the trees), evaluation of statistical parsers and grammars that are derived from particular treebank resources (as well as hand-crafted grammars/parsers) can suffer unduly if the gold standard deviates systematically from the (possibly) equally valid style of linguistic analysis provided by the parser.

Problems such as these have motivated research on more abstract, dependency-based parser evaluation (e.g., Lin (1995), Carroll, Briscoe, and Sanfilippo (1998), Carroll et al. (2002), Clark and Hockenmaier (2002), King et al. (2003), Preiss (2003), Kaplan et al. (2004), Miyao and Tsujii (2004)). Dependency-based linguistic representations are approximations of abstract predicate-argument-adjunct (or more basic head-dependent) structures, providing a more normalized representation abstracting away from the particulars of surface realization or CFG-tree representation, which enables meaningful cross-parser evaluation.

A related contrast holds between shallow and deep grammars and parsers.[1] In addition to defining a language (as a set of strings), deep grammars relate strings to information/meaning, often in the form of predicate-argument structure, dependency relations[2] or logical forms. By contrast, a shallow grammar simply defines a language and may associate syntactic (e.g., CFG tree) representations with strings. Natural languages do not always interpret linguistic material locally where the material is encountered in the string (or tree). In order to obtain accurate and complete predicate-argument, dependency or logical form representations, a hallmark of deep grammars is that they usually involve a long-distance dependency (LDD) resolution mechanism.

Traditionally, deep grammars are hand-crafted (cf. the ALVEY Natural Language Tools (Briscoe et al. 1987), the Core Language Engine (Alshawi and Pulman 1992), the Alpino Dutch dependency parser (Bouma, van Noord, and Malouf 2000), the Xerox Linguistic Environment (Butt et al. 2002), the RASP dependency parser (Carroll and Briscoe 2002) and the LinGO English Resource Grammar (Flickinger 2000; Baldwin et al. 2004)). Wide-coverage, deep grammar development, particularly in rich formalisms such as LFG (Kaplan and Bresnan 1982; Bresnan 2001; Dalrymple 2001) and HPSG (Pollard and Sag 1994), is knowledge-intensive, time-consuming and expensive, constituting an instance of the (in-)famous "knowledge acquisition bottleneck" familiar from other areas in traditional, rule-based AI and NLP. Very few hand-crafted deep grammars (Briscoe and Carroll (1993), Bouma, van Noord, and Malouf (2000), Riezler et al. (2002)) have, in fact, been successfully scaled to unrestricted input.

The last 15 years have seen extensive efforts on treebank-based automatic grammar acquisition using a variety of machine-learning techniques (e.g., Gaizauskas (1995), Charniak (1996), Collins (1999), Johnson (1999), Charniak (2000), Bikel (2002), Bod (2003), Klein and Manning (2003)). These grammars are wide-coverage and robust and in contrast to manual grammar development, machine-learning-based grammar acquisition incurs relatively low development cost. With few notable exceptions,[3] however, these treebank-induced wide-coverage grammars are shallow: They usually do not attempt to resolve LDDs nor do they associate strings with meaning representations.

Over the last few years, addressing the knowledge acquisition bottleneck in deep constraint-based grammar development, a growing body of research has emerged to automatically acquire wide-coverage deep grammars from treebank resources (TAG (Xia 1999), CCG (Hockenmaier and Steedman 2002), HPSG (Miyao, Ninomiya, and Tsujii 2003), LFG (Cahill et al. 2002b, 2004)). To a first approximation, these approaches can be classified as 'conversion'- and/or 'annotation'-based. TAG-based approaches convert treebank trees into (lexicalized) elementary or adjunct trees. CCG-based approaches convert trees into CCG derivations from which CCG categories can be extracted. HPSG- and LFG-based grammar induction methods automatically annotate treebank trees

1 Our use of the terms "shallow" and "deep" parsers/grammars follows Kaplan et al. (2004) where a "shallow parser" does not relate strings to meaning representations. This deviates from a more common use of the terms where, for example, a "shallow parser" refers to (often finite-state-based) parsers (or chunkers) that may produce partial bracketings of input strings.

2 By dependency relations we mean deep, fine-grained, labeled dependencies that encode long-distance dependencies and passive information, for example. These differ to the types of unlabeled dependency relations in other work such as (McDonald and Pereira 2006)

3 Both Collins Model 3 (1999) and Johnson (2002) output CFG tree representations with traces. Collins Model 3 performs LDD resolution for wh-relative clause constructions, Johnson (2002) for a wide range of LDD phenomena in a post-processing approach based on Penn-II tree fragments linking displaced material with where it is to be interpreted semantically. The work of Dienes and Dubey (2003) and Levy and Manning (2004) is similar to that of Johnson (2002), recovering empty categories on top of CFG-based parsers. None of them map strings into dependencies.

with (typed) attribute-value structure information for the extraction of constraint-based grammars and lexical resources.

Two recent papers (Preiss 2003; Kaplan et al. 2004) have started tying together the research strands sketched above: They use dependency-based parser evaluation to compare wide-coverage parsing systems using hand-crafted, deep, constraint-based grammars with systems based on a simple version of treebank-based deep grammar acquisition technology in the conversion paradigm. In the experiments, tree output generated by Collins' Model 1, 2 and 3 (1999) and Charniak's (2000) parsers, for example, are automatically translated into dependency structures and evaluated against gold-standard dependency banks.

Preiss (2003) uses the grammatical relations and the CBS 500 Dependency Bank described in Carroll, Briscoe, and Sanfilippo (1998) to compare a number of parsing systems (Briscoe and Carroll (1993), Collins' (1997) models 1 and 2 and Charniak (2000)) using a simple version of the conversion-based deep grammar acquisition process (i.e., reading off grammatical relations from CFG parse trees produced by the treebank-based shallow parsers). The paper also reports on a task-based evaluation experiment to rank the parsers using the grammatical relations as input to an anaphora resolution system. Preiss concluded that parser ranking using grammatical relations reflected the absolute ranking (between treebank-induced parsers) using traditional tree-based metrics, but that the difference between the performance of the parsing algorithms narrowed when they carried out the anaphora resolution task. Her results show that the hand-crafted deep unification parser (Briscoe and Carroll 1993) outperforms the machine-learned parsers (Collins 1997; Charniak 2000) on the f-score derived from weighted precision and recall on grammatical relations.[4] Kaplan et al. (2004) compare their deep, hand-crafted, LFG-based XLE parsing system (Riezler et al. 2002) with Collins' (1999) model 3 using a simple conversion-based approach, capturing dependencies from the tree output of the machine-learned parser, and evaluating both parsers against the PARC 700 Dependency Bank (King et al. 2003). They conclude that the hand-crafted, deep grammar outperforms the state-of-the-art treebank-based shallow parser on the level of dependency representation, at the price of a small decrease in parsing speed.

Both Preiss (2003) and Kaplan et al. (2004) emphasize that they use rather *basic* versions of the conversion-based deep grammar acquisition technology outlined above. In this article we revisit the experiments carried out by Preiss (2003) and Kaplan et al. (2004), this time using the *sophisticated* and *fine-grained* treebank- and annotation-based, deep, probabilistic Lexical-Functional Grammar (LFG) grammar acquisition methodology developed in Cahill et al. (2002b), Cahill et al. (2004), O'Donovan et al. (2004) and Burke (2006) with a number of surprising results:

1. Evaluating against the PARC 700 Dependency Bank (King et al., 2003) using a retrained version of Bikel's (2002) parser, the best automatically induced, deep LFG resources achieve an f-score of 82.73%. This is an improvement of 3.13% over the previously best published results established by Kaplan et al. (2004) who use a hand-crafted, wide-coverage, deep LFG and the XLE parsing system. This is also a statistically significant improvement of 2.18% over the most recent improved results presented in this article for the XLE system.

4 The numbers given are difficult to compare as the results for the Briscoe and Carroll (1993) parser were captured for a richer set of grammatical relations than those for Collins (1997) and Charniak (2000).

2. Evaluating against the Carroll, Briscoe, and Sanfilippo (1998) CBS 500 gold-standard dependency bank using a retrained version of Bikel's (2002) parser, the best Penn-II treebank-based, automatically acquired, deep LFG resources achieve an f-score of 80.23%. This is a statistically significant improvement of 3.66% over Carroll and Briscoe (2002), who use a hand-crafted, wide-coverage, deep, unification grammar and the RASP parsing system.

Evaluation results on a reannotated version (Briscoe and Carroll 2006) of the PARC 700 Dependency Bank were recently published in Clark and Curran (2007), reporting f-scores of 81.9% for the CCG parser, and 76.3% for RASP. As Briscoe and Carroll (2006) point out, these evaluations are not directly comparable with the Kaplan et al. (2004) style evaluation against the original PARC 700 Dependency Bank, since the annotation schemes are different.

The article is structured as follows: In Section 2, we outline the automatic LFG f-structure annotation algorithm and the pipeline parsing architecture of Cahill et al. (2002b), Cahill et al. (2004) and Burke (2006). In Section 3, we present our experiment design. In Section 4, using the DCU 105 Dependency Bank as our development set, we evaluate a number of treebank-induced LFG parsing systems against the automatically generated Penn-II WSJ Section 22 Dependency Bank test set. We use the Approximate Randomization Test (Noreen 1989) to test for statistical significance and choose the best parsing system for the evaluations against the wide-coverage, hand-crafted RASP and LFG grammars of Carroll and Briscoe (2002) and Kaplan et al. (2004) using the CBS 500 and PARC 700 Dependency Banks in Section 5. In Section 6, we discuss results and issues raised by our methodology, outline related and future research and conclude in Section 7.

2. Methodology

In this section, we briefly outline Lexical Functional Grammar (LFG) and present our automatic f-structure annotation algorithm and parsing architecture. The parsing architecture enables us to integrate PCFG- and history-based parsers, which allows us to compare these parsers at the level of dependency structures, rather than just trees.

2.1 Lexical Functional Grammar

Lexical Functional Grammar (LFG) (Kaplan and Bresnan 1982; Bresnan 2001; Dalrymple 2001) is a constraint-based theory of grammar. It (minimally) posits two levels of representation, c(onstituent)-structure and f(unctional)-structure. C-structure is represented by context-free phrase-structure trees, and captures surface grammatical configurations such as word order. The nodes in the trees are annotated with functional equations (attribute-value structure constraints, for example ($\uparrow$OBJ)=$\downarrow$) which are resolved (in the case of well-formed strings) to produce an f-structure. F-structures are recursive attribute-value matrices, representing abstract syntactic functions, which approximate to basic predicate-argument-adjunct structures or dependency relations.[5]

5 van Genabith and Crouch (1996) and van Genabith and Crouch (1997) provide translations between f-structures, Quasi-Logical Forms (QLFs) and Underspecified Discourse Representation Structures (UDRSs).

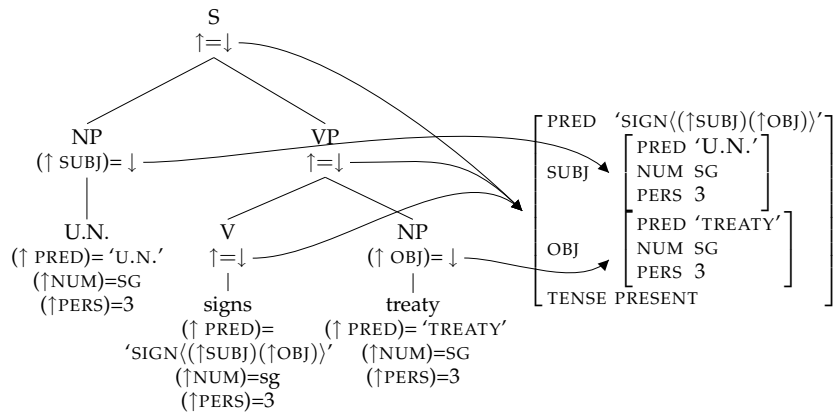

Figure 1
C- and f-structures for the sentence *U.N. signs treaty*

Figure 1 shows the c- and f-structures for the string *U.N.signs treaty*. Each node in the c-structure is annotated with f-structure equations, for example (↑ SUBJ)= ↓. The uparrows (↑) point to the f-structure associated with the mother node, downarrows (↓) to that of the local node. In a complete parse tree, these ↑ and ↓ meta variables are instantiated to unique tree node identifiers and a set of constraints (a set of terms in an equality logic) is generated which (if satisfiable) generates an f-structure.

2.2 Automatic F-Structure Annotation Algorithm

Deep grammars can be induced from treebank resources if the treebank encodes enough information to support the derivation of deep grammatical information, such as predicate-argument structures, deep dependency relations or logical forms. Many second generation treebanks such as Penn-II provide information to support the compilation of meaning representations, for example in the form of traces relating displaced linguistic material to where it should be interpreted semantically. The f-structure annotation algorithm exploits configurational and categorial information, as well as traces and the Penn-II functional tag annotations (Table 1) to automatically associate Penn-II CFG trees with LFG f-structure information.

Given a tree, such as the Penn-II-style tree in Figure 2, the algorithm will traverse the tree and deterministically add f-structure equations to the phrasal and leaf nodes of the tree, resulting in an f-structure annotated version of the tree. The annotations are then collected and passed on to a constraint solver which generates an f-structure (if the constraints are satisfiable). We use a simple graph-unification-based constraint solver (Eisele and Dörre 1986), extended to handle path, set-valued, disjunctive and existential constraints. Given parser output without Penn-II style annotations and traces, the same algorithm is used to assign annotations to each node in the tree, while a separate module is applied at the level of f-structure to resolve any long-distance dependencies (see Section 2.3).

The f-structure annotation algorithm is described in detail in Cahill et al. (2002a), McCarthy (2003), Cahill et al. (2004) and Burke (2006). In brief, the algorithm is modular with four components (Figure 3), taking Penn-II trees as input and automatically adding LFG f-structure equations to each node in the tree.

Table 1
A complete list of the Penn-II functional labels

Tag	Description
	Form/Function Discrepancies
-ADV	clausal and NP adverbials
-NOM	non NPs that function as NPs
	Grammatical Role
-DTV	dative
-LGS	logical subjects in passives
-PRD	non VP predicates
-PUT	locative complement of put
-SBJ	surface subject
-TPC	topicalized and fronted constituents
-VOC	vocatives
	Adverbials
-BNF	benefactive
-DIR	direction and trajectory
-EXT	extent
-LOC	location
-MNR	manner
-PRP	purpose and reason
-TMP	temporal phrases
	Miscellaneous
-CLR	closely related to verb
-CLF	true clefts
-HLN	headlines and datelines
-TTL	titles

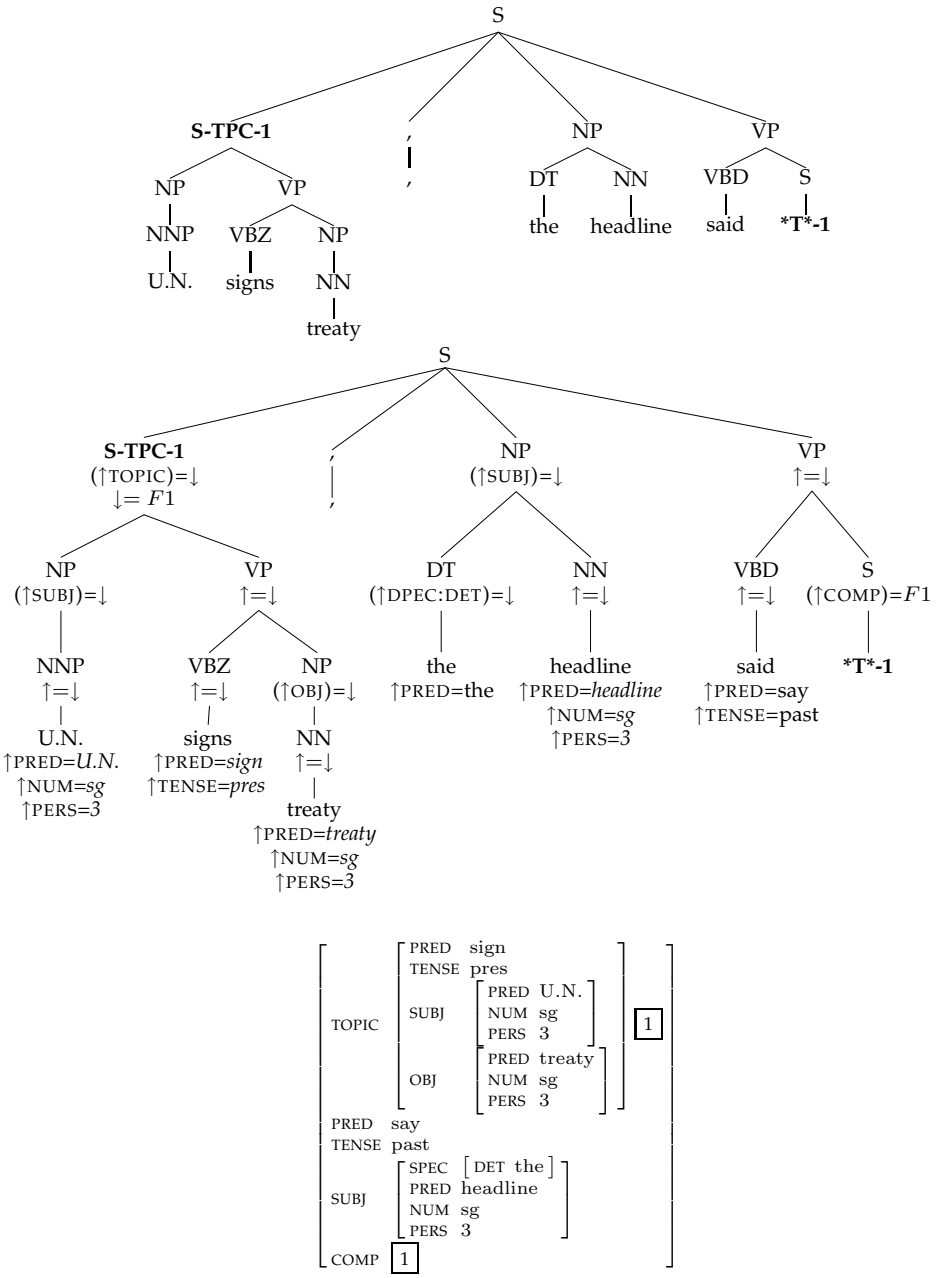

Figure 2
Trees for the sentence *U.N. signs treaty, the headline said* before and after automatic f-structure annotation, with the f-structure automatically produced.

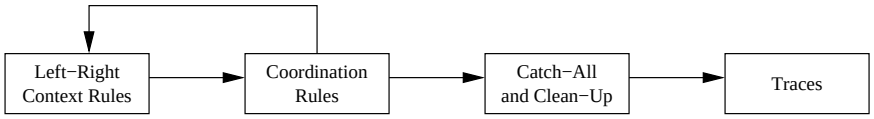

Figure 3
F-Structure Annotation Algorithm modules

Lexical Information:. Lexical information is generated automatically by macros for each of the POS classes in Penn-II. To give a simple example, third person plural noun Penn-II POS-word sequences of the form `NNS word` are automatically associated with the equations (↑PRED) = `word′`, (↑NUM) = pl and (↑PERS) = 3rd, where `word′` is the lemmatized `word`.

Left-Right Context Annotation:. The Left-Right Context Annotation component identifies the heads of Penn-II trees using a modified version of the head finding rules of Magerman (1994). This partitions each local subtree (of depth one) into a local head, a left context (left sisters) and a right context (right sisters). The contexts together with information about the local mother and daughter categories and (if present) Penn-II functional tag labels (Table 1) are used by the f-structure annotation algorithm. For each Penn-II mother (i.e., phrasal) category an "Annotation Matrix" expresses generalizations about how to annotate immediate daughters dominated by the mother category relative to their location in relation to the local head. To give a (much simplified) example, the head finding rules for NPs state that the rightmost nominal (NN, NNS, NNP, ...) not preceded by a comma or "-"[6] is likely to be the local head. The annotation matrix for NPs states (inter alia) that heads are annotated ↑=↓, that DTs (determiners) to the left of the head are annotated (↑ SPEC DET) = ↓, NPs to the right of the head as ↓∈(↑ APP) (appositions). Table 2 provides a sample extract from the NP Annotation Matrix. Figure 4 provides an example of the application of the NP and PP annotation matrices to a simple tree.

Table 2
Sample from NP Annotation Matrix

Left Context	Head	Right Context
DT: (↑SPEC DET)=↓ CD: (↑SPEC QUANT)=↓ ADJP, JJ, NN, NNP: ↓∈(↑ADJUNCT)	NN, NNS, NNP, NNPS, NP: ↑=↓	RRC, SBAR: (↑RELMOD)=↓ PP: ↓∈(↑ADJUNCT) NP: ↓∈(↑APP)

For each phrasal category, annotation matrices are constructed by inspecting the most frequent Penn-II rule types expanding the category such that the token occurrences of these rule types cover more than 85% of all occurrences of expansions of that category in Penn-II. For NP rules, for example, this means that we analyze the most frequent 102 rule types expanding NP, rather than the complete set of more than 6,500 Penn-II NP rule types, in order to populate the NP annotation matrix. Annotation matrices generalize to unseen rule types as, in the case of NPs, these may also feature DTs to the

6 If the rightmost nominal is preceded by a comma or "-", it is likely to be an apposition to the head.

left of the local head and NPs to the right and similarly for rule types expanding other categories.

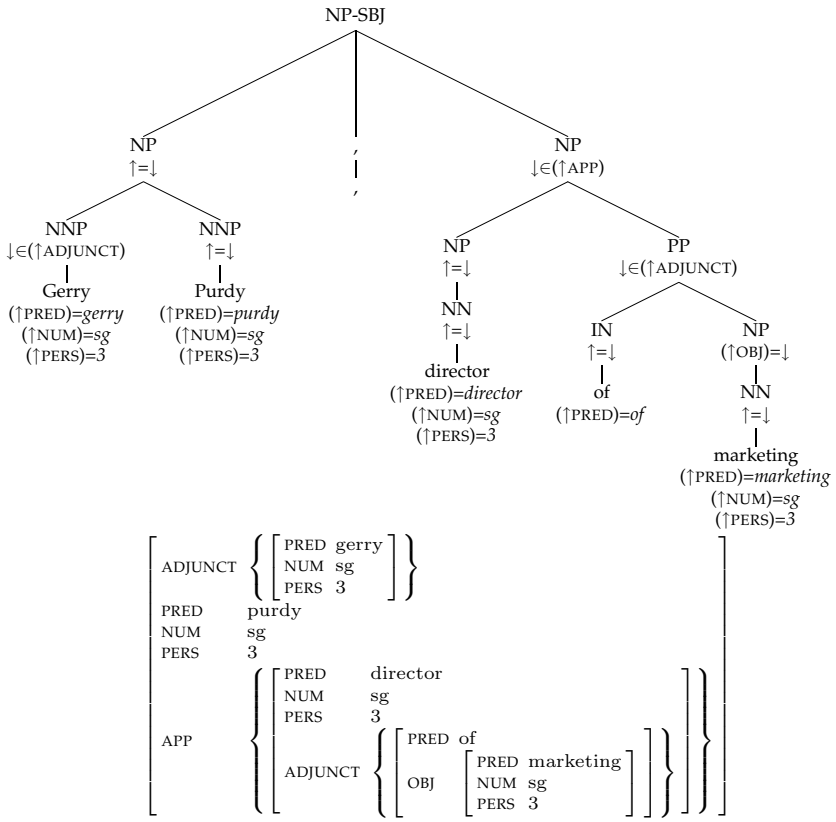

Figure 4
Automatically annotated Penn-II tree (fragment) and f-structure (simplified) for *Gerry Purdy, director of marketing*

Coordination:. In order to support the modularity, maintainability and extendability of the annotation algorithm, the Left-Right Annotation Matrices apply only to local trees of depth one which do not feature coordination. This keeps the statement of Annotation Matrices perspicuous and compact. The Penn-II treatment of coordination is (intentionally) flat. The annotation algorithm has modules for like- and unlike-constituent coordination. Coordinated constituents are elements of a COORD set and annotated $\downarrow \in (\uparrow$ COORD). The Coordination module reuses the Left-Right context annotation matrices to annotate any remaining nodes in a local subtree containing a coordinating conjunction. Figure 5 provides a VP-coordination example (with right-node-raising).

Catch-All and Clean-Up:. The Catch-All and Clean-Up module provides defaults to capture remaining unannotated nodes (Catch-All) and corrects (Clean-Up) overgeneralizations resulting from the application of the Left-Right Context Annotation Matrices. The Left-Right Annotation Matrices are allowed a certain amount of overgeneralization as this facilitates the perspicuous statement of generalizations and a separate statement of exceptions, supporting the modularity and maintainability of the annotation algorithm.

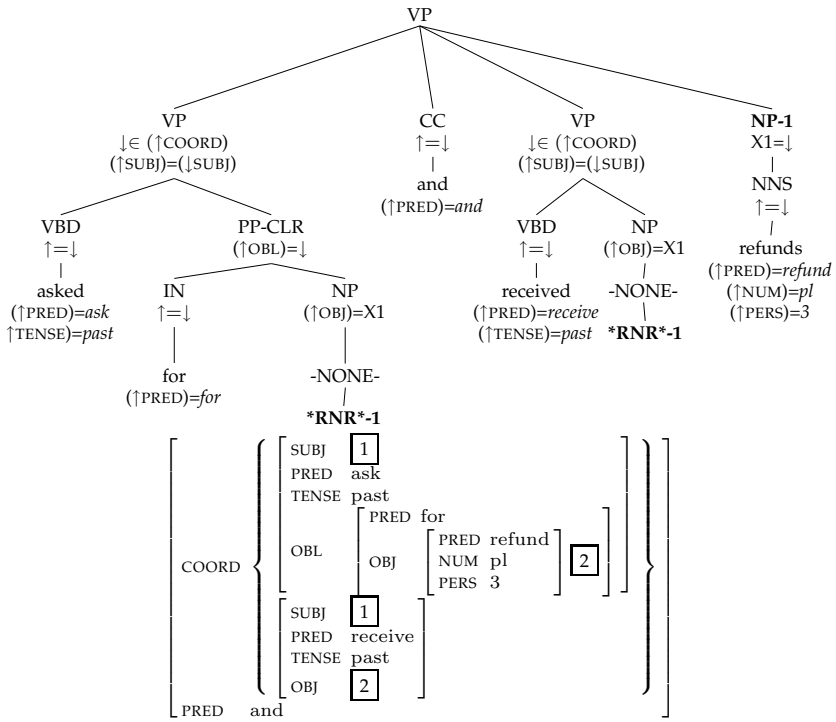

Figure 5
Automatically annotated Penn-II tree (fragment) and resulting f-structure for *asked for and received refunds.*

PPs under VPs are a case in point. The VP Annotation Matrix analyses PPs to the right of the local VP head as adjuncts: $\downarrow \in (\uparrow \text{ADJUNCT})$. The Catch-All and Clean-Up module uses Penn-II functional tag (Table 1) information (if present), for example -CLR (closely related to local head), to replace the original adjunct analysis by an oblique argument analysis: $(\uparrow \text{OBL})=\downarrow$. An example of this is provided by the PP-CLR in the left VP-conjunct in Figure 5. In other cases, argument–adjunct distinctions are encoded configurationally in Penn-II (without the use of -CLR tags). To give a simple example, the NP annotation matrix indiscriminately associates SBARs to the right of the local head with $(\uparrow \text{RELMOD}) = \downarrow$. However, some of these SBARs are actually arguments of the local NP head and, unlike SBAR relative clauses which are Chomsky-adjoined to NP (i.e., relative clauses are daughters of an NP mother and sisters of a phrasal NP head), SBAR arguments are sisters of non-phrasal NP heads.[7] In such cases, the Catch-All and Clean-Up module rewrites the original relative clause analysis into the correct complement argument analysis $(\uparrow \text{COMP})=\downarrow$. Figure 6 shows the COMP f-structure analyses for an example NP containing an internal SBAR argument (rather than relative clause) node.

Traces:. The Traces Module translates traces and coindexed material in Penn-II trees representing long-distance dependencies into corresponding reentrancies at f-structure. Penn-II provides a rich arsenal of trace types to relate "displaced" material to where it

7 Structural information of this kind is not encoded in the Annotation Matrices, c.f. Table 2.

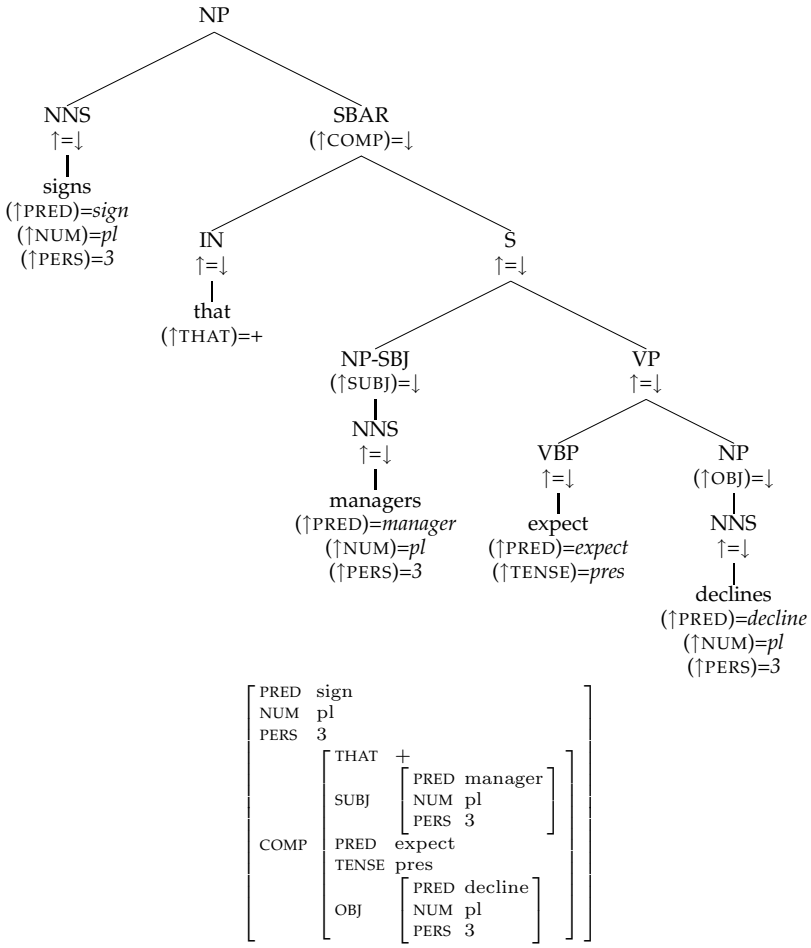

Figure 6
Automatically annotated Penn-II tree (fragment) and f-structure for *signs that managers expect declines*

should be interpreted semantically. The f-structure annotation algorithm covers wh- and wh-less relative clause constructions, interrogatives, control and raising constructions, right-node-raising and general ICH (interpret constituent here) traces. Figure 5 gives an example that shows the interplay between coordination, right-node-raising traces and the corresponding automatically generated reentrancies at f-structure.

2.3 Parsing Architecture

The *pipeline* parsing architecture of Cahill et al. (2004) and Cahill (2004) for parsing raw text into LFG f-structures is shown in Figure 7. In this model, PCFGs or history-based lexicalized parsers are extracted from the unannotated treebank and used to parse raw

text into trees. The resulting parse trees are then passed to the automatic f-structure annotation algorithm to generate f-structures.[8]

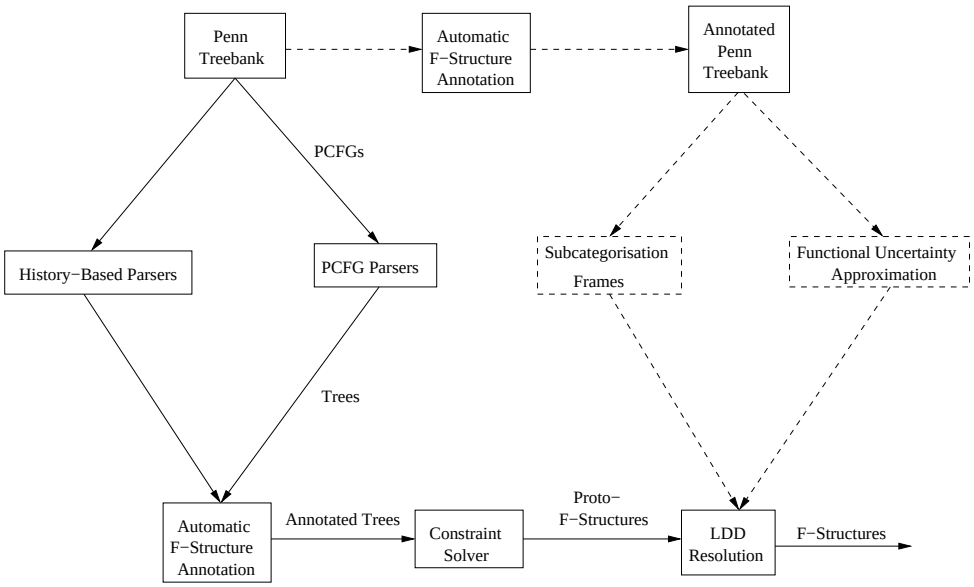

Figure 7
Treebank-based LFG parsing architecture

Compared to full Penn-II treebank trees, the output of standard probabilistic parsers is impoverished: Parsers do not normally output Penn-II functional tag annotations (Table 1) nor do they indicate/resolve long-distance dependencies, recorded in terms of a fine-grained system of empty productions (traces) and coindexation in the full Penn-II treebank trees. The f-structure annotation algorithm, as described in Section 2.2, makes use of Penn-II functional tag information (if present) and relies on traces and coindexation to capture LDDs in terms of corresponding reentrancies at f-structure.

Penn-II functional labels are used by the annotation algorithm to discriminate between adjuncts and (oblique) arguments. PP-sisters to a head verb are analyzed as arguments iff they are labeled `-CLR, -PUT, -DTV` or `-BNF`, for example. Conversely, functional labels (e.g., `-TMP`) are also used to analyze certain NPs as adjuncts, and `-LGS` labels help to identify logical subjects in passive constructions. In the absence of functional labels, the annotation algorithm will default to decisions based on simple structural, configurational and CFG-category information (and e.g., conservatively analyze a PP sister to a head verb as an adjunct, rather than as an argument).

In Sections 3 and 4 below we present a number of treebank-based parsers (in particular the PCFGs and a version of Bikel's history-based, lexicalized generative parser) trained to output CFG categories with Penn-II functional tags. We achieve this through

8 In the *integrated* model (Cahill et al. 2004; Cahill 2004), we extract f-structure annotated PCFGs (A-PCFGs) from the f-structure annotated treebank, where each non-terminal symbol in the grammar has been augmented with LFG functional equations, such as NP[↑OBJ=↓] → DT[↑SPEC=↓] NN[↑=↓]. We treat a non-terminal symbol followed by annotations as a monadic category for grammar extraction and parsing. Parsing with A-PCFGs results in annotated parse trees, from which an f-structure can be generated. In this article we only use the pipeline parsing architecture.

a simple masking and un-masking operation where functional tags are joined with their local CFG category label to form a new (larger) set of (monadic) CFG category labels (e.g., `PP-CLR` goes to `PP_CLR`) for training and parsing (for Bikel, the parser head-finding rules are also adjusted to the expanded set of categories). After parsing, the Penn-II functional tags are unmasked and available to the f-structure annotation algorithm.

The Traces component in the f-structure annotation algorithm (Figure 3) translates LDDs represented in terms of traces and coindexation in the original Penn-II treebank trees into corresponding reentrancies at f-structure. Most probabilistic treebank-based parsers, however, do not indicate/resolve LDDs, and the Traces component of the annotation algorithm does not apply. Initially, the f-structures produced for parser output trees in the architecture in Figure 7 are therefore LDD-unresolved: They are incomplete (or proto) f-structures, where displaced material (e.g., the values of FOCUS, TOPIC and TOPICREL attributes (wh- and wh-less relative clauses, topicalization and interrogative constructions) at f-structure) is not yet linked to the appropriate argument grammatical functions (or elements of adjunct sets) for the governing local PRED. A dedicated LDD Resolution component in the architecture in Figure 7 turns parser output proto-f-structures into fully LDD-resolved proper f-structures, without traces and coindexation in parse trees.

Consider the following fragment of a proper Penn-II treebank tree (Figure 8), where the LDD between the WHNP in the relative clause and the embedded direct object position of the verb `reward` is indicated in terms of the trace *T*-3 and its coindexation with the antecedent WHNP-3. Note further that the control relation between the subject of the verbs `wanted` and `reward` is similarly expressed in terms of traces (*T*-2) and coindexation (NP-SBJ-2). From the treebank tree, the f-structure annotation algorithm is able to derive a fully resolved f-structure where the LDD and the control relation are captured in terms of corresponding reentrancies (Figure 9).

Now consider the corresponding "impoverished" (but otherwise correct) parser output tree (Figure 10) for the same string: The parser output does not explicitly record the control relation nor the LDD.

Given this parser output tree, prior to the LDD resolution component in the parsing architecture (Figure 7), the f-structure annotation algorithm would initially construct the partial (proto-) f-structure in Figure 11, where the LDD indicated by the TOPICREL function is unresolved (i.e., the value of TOPICREL is not coindexed with the OBJ grammatical function of the embedded verb `reward`). The control relation (shared subject between the two verbs in the relative clause) is in fact captured by the annotation algorithm in terms of a default annotation (↑ SUBJ) = (↓ SUBJ) on sole argument VPs to the right of head verbs (as often, even in the full Penn-II treebank trees, control relations are not consistently captured through explicit argument traces).

In Lexical-Functional Grammar, LDD resolution operates at the level of f-structure, using functional uncertainty equations (regular expressions over paths in f-structure (Kaplan and Zaenen 1989) relating f-structure components in different parts of an f-structure), obviating traces and coindexation in c-structure trees. For the example in Figure 10, a functional uncertainty equation of the form (↑TOPICREL) = (↑[COMP | XCOMP]* [SUBJ | OBJ]) would be associated with the WHNP daughter node of the SBAR relative clause. The equation states that the value of the TOPICREL attribute is token-identical (re-entrant) with the value of a SUBJ or OBJ function, reached through a path along any number (including zero) of COMP or XCOMP attributes. This equation, together with subcategorization frames (LFG semantic forms) for the local PREDs and the usual LFG

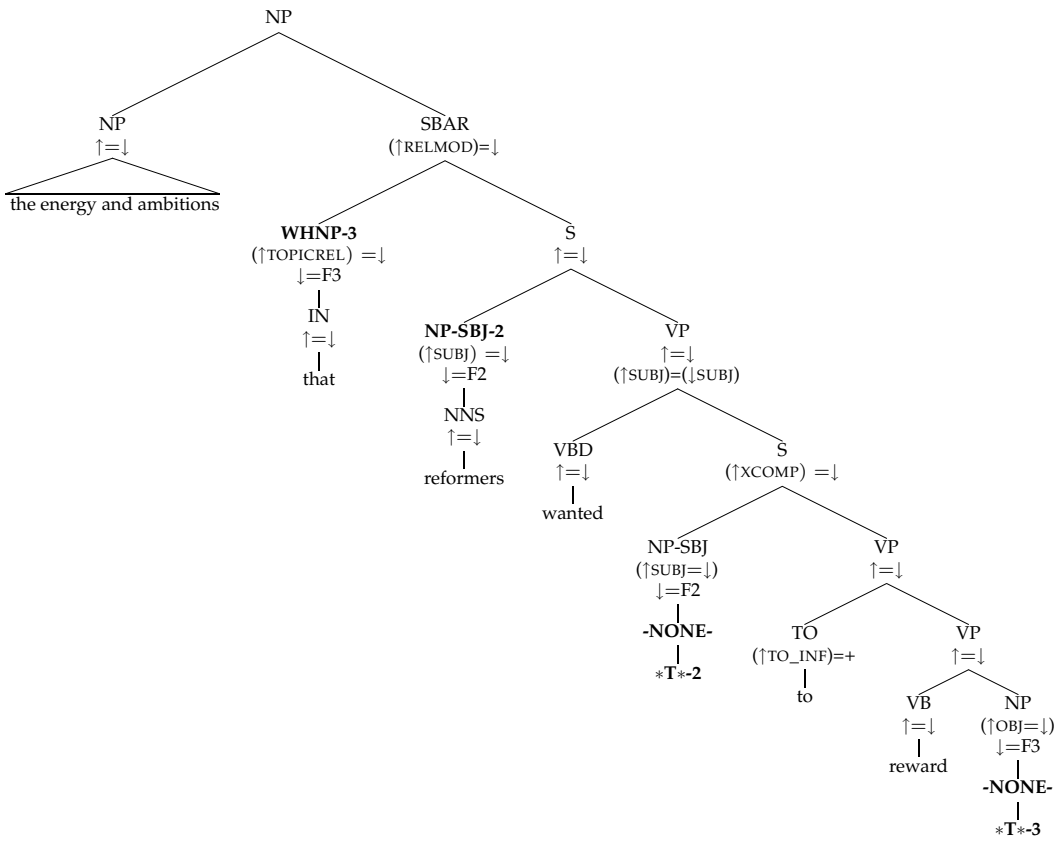

Figure 8
Penn-II treebank tree with LDD indicated in terms of traces (empty productions) and coindexation and f-structure annotations generated by the annotation algorithm.

completeness and coherence conditions, resolve the partial proto-f-structure in Figure 11 into the fully LDD-resolved proper f-structure in Figure 9.

Following Cahill et al. (2004), in our parsing architecture (Figure 7) we model LFG LDD resolution using automatically induced finite approximations of functional-uncertainty equations and subcategorization frames from the f-structure-annotated Penn-II treebank (O'Donovan et al. 2004) in an LDD resolution component. From the fully LDD resolved f-structures from the Penn-II training section treebank trees we learn probabilistic LDD resolution paths (reentrancies in f-structure), conditional on LDD type (Table 3), and subcategorization frames, conditional on lemma (and voice) (Table 4). Table 3 lists the 8 most probable TOPICREL paths (out of a total of 37 TOPICREL paths acquired). The totality of these paths constitutes a finite subset of the reference language definde by the full functional uncertainty equation (↑TOPICREL) = (↑[COMP | XCOMP]* [SUBJ | OBJ]). Given an unresolved LDD type (such as TOPICREL in the parser output for the relative clause example in Figure 11 above), admissible LDD resolutions assert a reentrancy between the value of the LDD trigger (here TOPICREL) and a grammatical function (or adjunct set element) of an embedded local predicate, subject to the conditions that (i) the local predicate can be reached from the LDD trigger

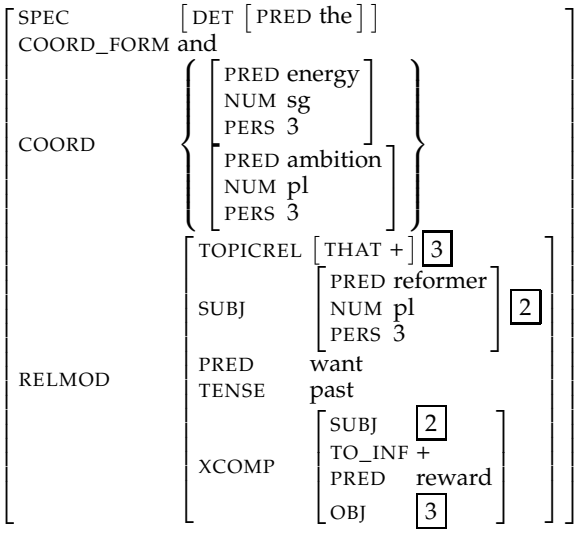

Figure 9
Fully LDD resolved f-structure.

using the LDD path; (ii) the grammatical function terminates the LDD path, (iii) the grammatical function is not already present (at the relevant level of embedding in the local f-structure) and (vi), the local predicate subcategorizes for the grammatical function in question.[9] Solutions satisfying (i) - (iv) are ranked using the product of LDD path and subcategorization frame probabilities and the highest ranked solution (possibly involving multiple interacting LDDs for a single f-structure) is returned by the algorithm (for details and comparison against alternative LDD resolution methods, see Cahill et al. (2004)).[10]

Table 3
Most frequent wh-TOPICREL paths

wh-TOPICREL	**prob.**	wh-TOPICREL	**prob.**
subj	0.7583	xcomp	0.0830
obj	0.0458	xcomp:obj	0.0338
xcomp:xcomp	0.0168	xcomp:subj	0.0109
comp	0.0097	comp:subj	0.0073

For our example (Figure 11), the highest ranked LDD resolution is for LDD path (↑TOPICREL) = (↑ XCOMP OBJ) and the local subcat frame REWARD⟨↑ SUBJ, ↑ OBJ⟩. This (together with the subject control equation described above) turns the parser-output proto-f-structure (in Figure 11) into the fully LDD resolved f-structure in (Figure 9).

9 Conditions (i) – (iv) are suitably adapted for LDD resolutions terminating in adjunct sets.
10 In our experiments we do not use the limited LDD resolution for wh-phrases provided by Collins' Model 3 parser as better results are achieved using the purely f-structure-based LDD resolution as shown in Cahill et al. (2004).

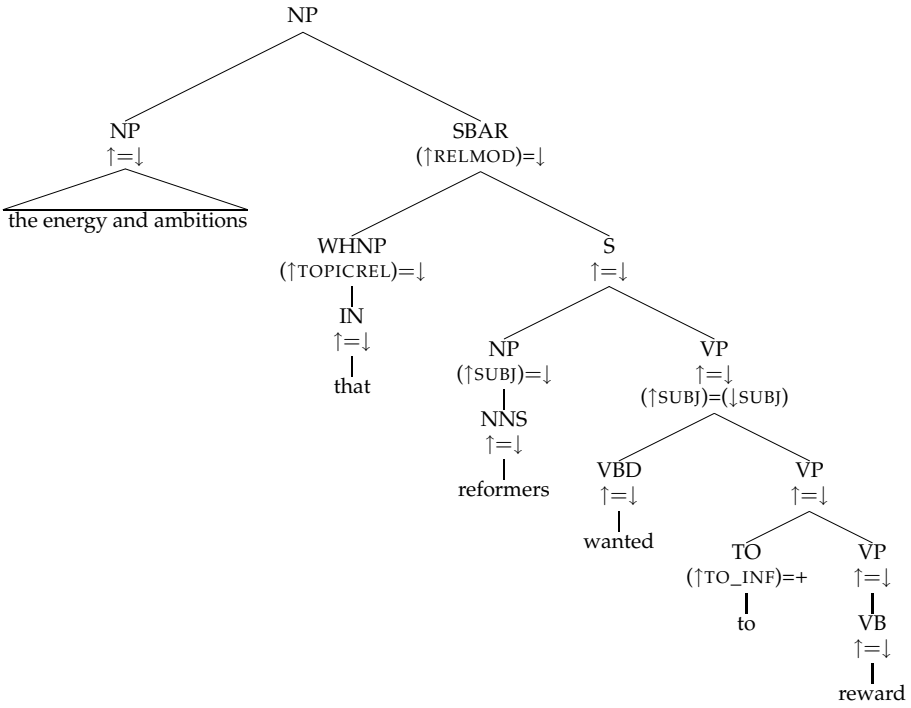

Figure 10
Impoverished parser output tree: LDDs not captured.

Table 4
Most frequent semantic forms for active and passive (**p**) occurrences of the verb `want` and `reward`

Semantic Form	Probability
`want([subj,xcomp])`	0.6208
`want([subj,obj])`	0.2496
`want([subj,obj,xcomp])`	0.1008
`want([subj])`	0.0096
`want([subj,obj,obl])`	0.0048
`want([subj,obj,part]),p)`	0.5000
`want([subj,obl]),p)`	0.1667
`want([subj,part]),p)`	0.1667
`want([subj]),p)`	0.1667
`reward([subj,obj])`	0.8000
`reward([subj,obj,obl])`	0.2000
`reward([subj]),p)`	1.0000

The full pipeline parsing architecture with the LDD resolution (rather than the Traces component for LDD resolved Penn-II treebank trees) component (and the LDD path and subcategorization frame extraction) is given in Figure 7.

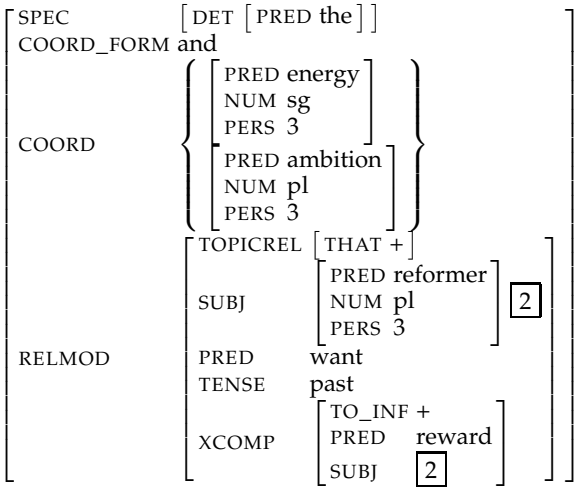

Figure 11
Proto-f-structure: LDDs not captured.

The pipeline architecture supports flexible integration of treebank-based PCFGs or state-of-the-art, history-based and lexicalized parsers (Collins 1999; Charniak 2000; Bikel 2002) and enables dependency-based evaluation of such parsers.

3. Experiment Design

In our experiments we compare 4 history-based parsers for integration into the pipeline parsing architecture described in Section 2.3.

- Collins Models 3 (Collins 1999)[11]
- Charniak's maximum-entropy inspired parser (Charniak 2000)[12]
- Bikel's emulation of Collins Model 2 (Bikel 2002)[13]
- A retrained version of Bikel's (2002) parser which retains Penn-II functional tags

Input for Collins' and Bikel's parsers was pre-tagged using the MXPOST POS tagger (Ratnaparkhi 1996). Charniak's parser provides its own POS tagger. The combined system of best history-based parser and automatic f-structure annotation is compared to two probabilistic parsing systems based on hand-crafted, wide-coverage, constraint-based, deep grammars:

- The RASP parsing system (Carroll and Briscoe 2002)
- The XLE parsing system (Riezler et al. 2002; Kaplan et al. 2004)

Both hand crafted-grammars perform their own POS tagging, resolve LDDs and associate strings with dependency relations (in the form of grammatical relations or LFG f-structures).

We evaluate the parsers against a number of gold-standard dependency banks. We use the DCU 105 Dependency Bank (Cahill et al. 2002a) as our development set for the treebank-based LFG parsers. We use the f-structure annotation algorithm to automatically generate a gold standard test set from the original Section 22 treebank trees (the f-structure annotated WSJ Section 22 Dependency Bank) to choose the best treebank-based LFG parsing systems for the PARC 700 and CBS 500 experiments. Following the experimental setup in Kaplan et al. (2004), we use the Penn-II Section 23-based PARC 700 Dependency Bank (King et al. 2003) to evaluate the treebank-induced LFG resources against the hand-crafted XLE grammar and parsing system of Riezler et al. (2002) and Kaplan et al. (2004). Following Preiss (2003), we use the SUSANNE Based CBS 500 Dependency Bank (Carroll, Briscoe, and Sanfilippo 1998) to evaluate the treebank-induced LFG resources against the hand-crafted RASP grammar and parsing system (Carroll and Briscoe 2002) as well as against the XLE system (Riezler et al. 2002).

For each gold standard, our experiment design is as follows: We parse automatically tagged input[14] sentences with the treebank- and machine-learning-based parsers trained on WSJ Sections 02-21 in the pipeline architecture, pass the resulting parse trees to our automatic f-structure annotation algorithm, collect the f-structure equations, pass them to a constraint-solver which generates an f-structure, resolve long-distance

11 Downloaded from `ftp://ftp.cis.upenn.edu/pub/mcollins/PARSER.tar.gz`
12 Downloaded from `ftp://ftp.cs.brown.edu/pub/nlparser/`
13 Developed at the University of Pennsylvania by Dan Bikel and is freely available to download from `http://www.cis.upenn.edu/~dbikel/software.html`
14 Tags were automatically assigned either by the parsers themselves or by the MXPOST tagger (Ratnaparkhi 1996)

dependencies at f-structure following Cahill et al. (2004) and convert the resulting LDD-resolved f-structures into dependency representations using the formats and software of Crouch et al. (2002) (for the DCU 105, PARC 700 and WSJ Section 22 evaluations) and the formats and software of Carroll, Briscoe, and Sanfilippo (1998) (for the CBS 500 evaluation). In the experiments we did not use any additional annotations such as -A (for argument) that can be generated by some of the history-based parsers (Collins 1999) as the f-structure annotation algorithm is designed for Penn-II trees (which do not contain such annotations). We also did not use the limited LDD resolution for wh-relative clauses provided by Collins (1999) Model 3 as better results are achieved by LDD resolution on f-structure (Cahill et al. 2004). A complete set of parameter settings for the parsers is provided in the Appendix.

In order to evaluate the treebank-induced LFG resources against the PARC 700 and the CBS 500 dependency banks, a certain amount of automatic mapping is required to account for systematic differences in linguistic analysis, feature geometry and nomenclature at the level of dependencies. This is discussed in Sections 5.1 and 5.2. Throughout, we use the Approximate Randomization Test (Noreen 1989) to test the statistical significance of the results.

4. Choosing a Treebank-Based LFG Parsing System

In this section, we choose the best treebank-based LFG parsing system for the comparisons with the hand-crafted XLE and RASP resources in Section 5. We use the DCU 105 Dependency Bank as our development set and carry out comparative evaluation and statistical significance testing on the larger, automatically generated WSJ Section 22 Dependency Bank as a test set. The system based on Bikel's (2002) parser retrained to retain Penn-II functional tags (Table 1) achieves overall best results.

4.1 Tree-Based Evaluation against WSJ Section 23

For reference, we include the traditional CFG-tree-based comparison for treebank-induced parsers. The parsers are trained on Sections 02 to 21 of the Penn-II Treebank and tested on Section 23. The published results[15] on these experiments for the history-based parsers are given in Table 5. We also include figures for a PCFG and a Parent-PCFG (a PCFG which has undergone the parent transformation (Johnson 1999)). These PCFGs are induced following standard treebank preprocessing steps, including elimination of empty nodes, but following Cahill et al. (2004), they do include Penn-II functional tags (Table 1), as these tags contain valuable information for the automatic f-structure annotation algorithm (Section 2.2). These tags are removed for the tree-based evaluation.

The results show that the history-based parsers produce considerably better trees than the more basic PCFGs (with and without parent transformations). Charniak's (2000) parser scores best with an f-score of 89.73 on all sentences in Section 23. The vanilla PCFG achieves the lowest f-score of 73.03%, a difference of 16.7%. The hand-crafted XLE and RASP grammars achieve around 80% coverage (measured in terms of complete spanning parse) on Section 23 and use a variety of (longest) fragments combining techniques to generate dependency representations for the remaining 20% of Section 23 strings. By contrast, the treebank-induced PCFGs and history-based parsers

15 Where there were no published results available for Section 23, we calculated them using the downloadable versions of the parsers.

Table 5
Results of tree-based evaluation on all sentences WSJ section 23, Penn-II

Parser	Labeled F-Score(%)
PCFG	73.03
Parent-PCFG	78.05
Collins M3	88.33
Charniak	**89.73**
Bikel	88.32
Bikel+Tags	87.53

all achieve coverage of over 99.9%. Given that the history-based parsers score considerably better than PCFGs on trees, we would also expect them to produce dependency structures of substantially higher quality.

4.2 Using DCU 105 as a Development Set

The DCU 105 (Cahill et al. 2002a) is a hand-crafted gold standard dependency bank for 105 sentences, randomly chosen from Section 23 of the Penn-II Treebank.[16] This is a relatively small gold standard, initially developed to evaluate the automatic f-structure annotation algorithm. We parse the 105 tagged sentences into LFG f-structures with each of the treebank-induced parsers in the pipeline parsing and f-structure annotation architecture. The f-structures of the gold standard and the f-structures returned by the parsing systems are converted into dependency triples following Crouch et al. (2002) and Riezler et al. (2002) and we also use their software for evaluation. The following dependency triples are produced by the f-structure in Figure 1:

```
subj(sign~0,U.N.~1)
obj(sign~0,treaty~2)
num(U.N.~1,sg)
pers(U.N.~1,3)
num(treaty~2,sg)
pers(treaty~3,3)
tense(sign~0,present)
```

We evaluate preds-only f-structures (i.e., where paths in f-structures end in a PRED value: the predicate-argument-adjunct structure skeleton) and all grammatical functions (GFs) including number, tense, person etc. The results are given in Table 6.

With one main exception, Tables 5 and 6 confirm the general expectation that the better the trees produced by the parsers, the better the f-structures automatically generated for those trees. The exception is Bikel+Tags. The automatic f-structure annotation algorithm will exploit Penn-II functional tag information if present to generate appropriate f-structure equations (see Section 2.2). It will default to possibly less reliable configurational and categorial information if Penn-II tags are not present in the trees.

In order to test whether the retention of Penn-II functional labels in the history-based parser output will improve LFG f-structure-based dependency results, we use

16 It is publicly available to download from: http://nclt.computing.dcu.ie/gold105.txt

Bikel's (2002) training software,[17] and retrain the parser on a version of the Penn-II treebank (Sections 02 to 21) with the Penn-II functional tag labels (Table 1) annotated in such a way that the resulting history-based parser will retain them (Section 2.3). The retrained parser (Bikel+Tags) then produces CFG-trees with Penn-II functional labels and these are used by the f-structure annotation algorithm. We evaluate the f-structure dependencies against the DCU 105 (Table 6) and achieve an f-score of 82.92% preds-only and 88.3% all GFs. A detailed breakdown by dependency is given in Table 7. The system based on the retrained parser is now much better able to identify oblique arguments and overall preds-only accuracy has improved by 3.53% over the original Bikel experiment and 3.31% over Charniak's parser, even though Charniak's parser performs more than 2% better on the tree-based scores in Table 5 and even though the retrained parser drops 0.79% against the original Bikel parser on the tree-based scores.[18]

Table 6
Treebank-induced parsers: results of dependency-based evaluation against DCU 105

Parser	Preds Only F-Score(%)	All GFs F-Score(%)
PCFG	70.24	79.90
Parent-PCFG	75.84	83.58
Collins M3	77.84	85.08
Charniak	79.61	85.66
Bikel	79.39	86.56
Bikel+Tags	**82.92**	**88.30**

Inspection of the results broken down by grammatical function (Table 7) for the preds-only evaluation against the DCU 105 shows that just over one third of all dependency triples in the gold standard are adjuncts. SUBJ(ects) and OBJ(ects) together make up a further 30%.

Table 7 shows that the treebank-based LFG system using Collins' Models 3 is unable to identify APP(osition). This is due to Collins' treatment of punctuation and the fact that punctuation is often required to reliably identify apposition.[19] None of the original history-based parsers produced trees which enabled the annotation algorithm to identify second oblique dependencies (OBL2), and they generally performed considerably worse than Parent-PCFG when identifying OBL(ique) dependencies. This is because the automatic f-structure annotation algorithm is cautious to the point of undergeneralisation when identifying oblique arguments. In many cases, the algorithm relies on the presence of a, for exampl, -CLR Penn-II functional label (indicating that the phrase is closely related to the verb), and the history-based (Collins M3, Charniak and Bikel) parsers do not produce these labels, while Parent-PCFG (as well as PCFG) are trained to retain Penn-II functional labels. Parent-PCFG, by contrast, performs poorly for oblique

17 We use Bikel's software rather than Charniak's for this experiment as the former proved more stable during the retraining phase.

18 The figures suggest that retraining Charniak's (2000) parser to retain Penn-II functional tags is likely to produce even better dependency scores than those achieved by Bikel's retrained parser.

19 The annotation algorithm relies on Penn-II-style punctuation patterns where a NP apposition follows a nominal head separated by a comma ([NP [NP Bush] , [NP the president]]), all three sisters of the same mother node, while the trees produced by Collins' parser attach the comma low in the tree ([NP [NP Bush ,] [NP the president]]). While it would be trivial to carry out a tree transformation on the Collins output to raise the punctuation to the expected level, we have not done this here.

Table 7
Treebank induced parsers: breakdown by dependency relation of preds-only evaluation against DCU 105

Dep.	% of total	F-Score(%)				
		Parent-PCFG	Collins M3	Charniak	Bikel	Bikel+Tags
ADJUNCT	33.73	71	72	76	73	79
APP	0.68	61	0	55	70	65
COMP	2.31	60	61	66	61	73
COORD	5.73	64	73	77	67	76
DET	9.58	91	93	96	96	96
FOCUS	0.04	100	100	0	100	100
OBJ	16.42	82	84	86	85	90
OBJ2	0.07	80	57	50	57	50
OBL	2.17	58	24	27	23	63
OBL2	0.07	50	0	0	0	67
OBL_AG	0.43	40	96	96	92	92
POSS	2.88	80	83	82	82	79
QUANT	1.85	70	67	69	70	70
RELMOD	1.78	50	78	67	78	73
SUBJ	14.74	80	81	83	85	85
TOPIC	0.46	85	87	96	96	89
TOPICREL	1.85	61	80	80	79	74
XCOMP	5.20	90	92	79	93	93

agents (OBL_AG, agentive *by*-phrases in passive constructions), whereas the history-based parsers are able to identify these with considerable accuracy. This is because Parent-PCFG often erroneously finds oblique agents, even when the preposition is not *by*, as it never has enough context in which to distinguish *by* prepositional phrases from other PPs. The history-based parsers produce trees from which the automatic f-structure annotation algorithm can better identify RELMOD and TOPICREL dependencies than Parent-PCFG. This, in turn, leads to improved long distance dependency resolution which improves overall accuracy.

The DCU 105 development set is too small to support reliable statistical significance testing of the performance ranking of the 6 treebank-based LFG parsing systems. In order to carry out significance testing to select the best treebank-based LFG parsing system for comparative evaluation against the hand-crafted deep XLE and RASP resources, we move to a larger dependency-based evaluation data set: the gold standard dependency bank automatically generated from WSJ Section 22.

4.3 Evaluation against WSJ Section 22 Dependencies

In an experimental setup similar to that of Hockenmaier and Steedman (2002),[20] we evaluate each parser against a large automatically generated gold standard. The gold standard dependency bank is automatically generated by annotating the *original* 1700

20 This corresponds to experiments where the original Penn-II Section 23 treebank trees are automatically converted into CCG derivations, which are then used as a gold standard to evaluate the CCG parser trained on Sections 02-21. A similar methodology is used for the evaluation of treebank-based HPSG resources (Miyao, Ninomiya, and Tsujii 2003) where Penn-II treebank trees are automatically annotated with HPSG typed-feature structure information.

Table 8
Results of dependency-based evaluation against the automatically generated gold standard for WSJ Section 22

Parser	Preds Only F-Score(%)	All GFs F-Score(%)
PCFG	70.76	80.44
Parent-PCFG	74.92	83.04
Collins M3	79.30	86.00
Charniak	81.35	86.96
Bikel	81.40	87.00
Bikel+Tags	**83.06**	**87.63**

treebank trees from WSJ Section 22 of the Penn-II Treebank with our f-structure annotation algorithm. We then evaluate the f-structures generated from the tree output of the 6 parsers trained on Sections 02 to 21 resulting from parsing the Section 22 strings against the *automatically* produced f-structures for the *original* Section 22 Penn-II treebank trees. The results are given in Table 8.

Compared to Table 6 for the DCU 105 gold standard, most scores are up, particularly so for the history-based parsers. This trend is possibly due to the fact that the WSJ Section 22 gold standard is generated automatically from the original "perfect" Penn-II treebank trees using the automatic f-structure annotation algorithm, while the DCU 105 has been created manually without regard as to whether or not the f-structure annotation algorithm could ever generate the f-structures, even given the "perfect" trees.

The LFG system based on Bikel's retrained parser achieves the highest f-score of 83.06% preds-only and 87.63% all GFs. Parent-PCFG achieves an f-score of 74.92% preds-only and 83.04% all GFs. Table 9 provides a breakdown by feature of the preds-only evaluation.

Table 9 shows that, once again, the automatic f-structure annotation algorithm is not able to identify any cases of apposition from the output of Collins' Model 3 parser. Apart from Bikel's retrained parser, none of the history-based parsers are able to identify OBJ2, OBL or OBL2 dependencies very well, although Parent-PCFG is able to produce trees from which it is easier to identify obliques (OBL), because of the Penn-II functional -CLR label. The automatic annotation algorithm is unable to identify RELMOD dependencies satisfactorily from the trees produced by parsing with Parent-PCFG, while the history-based parsers score reasonably well for this function. While Charniak's parser is able to identify some dependencies better than Bikel's retrained parser, overall the system based on Bikel's retrained parser performs better when evaluating against the dependencies in WSJ Section 22.

In order to determine whether the results are statistically significant, we use the Approximate Randomization Test (Noreen 1989).[21]

The Approximate Randomization test is an example of a computer-intensive statistical hypothesis test. Such tests are designed to assess result differences with respect to a test statistic in cases where the sampling distribution of the test statistic is unknown.

21 Applications of this test to natural language processing problems can be found in Chinchor et al. (1993) and Yeh (2000).

Table 9
Breakdown by dependency of results of preds-only evaluation against the automatically generated Section 22 gold standard

Dep.	% of total	F-Score(%)				
		Parent-PCFG	Collins M3	Charniak	Bikel	Bikel+Tags
ADJUNCT	33.77	70	75	78	78	80
APP	0.74	61	0	77	77	71
COMP	1.35	60	72	70	70	80
COORD	5.11	74	78	82	82	81
DET	10.72	88	91	92	92	91
FOCUS	0.02	27	67	88	88	71
OBJ	16.17	80	85	87	87	88
OBJ2	0.07	15	30	32	32	71
OBL	1.92	50	19	21	21	73
OBL2	0.07	47	3	3	3	69
OBL_AG	0.31	50	90	89	89	85
POSS	2.47	86	91	91	91	91
QUANT	2.12	89	89	93	93	92
RELMOD	1.84	51	71	72	72	69
SUBJ	15.45	75	80	81	81	82
TOPIC	0.44	81	85	84	84	76
TOPICREL	1.82	61	72	74	75	69
XCOMP	5.62	81	87	81	81	88

Comparative evaluations of outputs of parsing systems according to test statistics, such as differences in F-score, are examples of this situation. The test statistics are computed by accumulating certain count variables over the sentences in the test set. In the case of F-score, variable tuples consisting of the number of dependency-relations in the parse for the system translation, the number of dependency-relations in the parse for the reference translation, and the number of matching dependency-relations between system and reference parse, are accumulated over the test set.

Under the null hypothesis, the compared systems are not different, thus any variable tuple produced by one of the systems could have been produced just as likely by the other system. So shuffling the variable tuples between the two systems with equal probability, and recomputing the test statistic, creates an approximate distribution of the test statistic under the null hypothesis. For a test set of S sentences there are 2^S different ways to shuffle the variable tuples between the two systems. Approximate randomization produces shuffles by random assignments instead of evaluating all 2^S possible assignments. Significance levels are computed as the percentage of trials where the pseudo statistic, that is the test statistic computed on the shuffled data, is greater than or equal to the actual statistic, that is the test statistic computed on the test data. A sketch of an algorithm for approximate randomization testing is given in Figure 12.

Table 10 gives the p-values (the smallest fixed level at which the null hypothesis can be rejected) for comparing each parser against all of the other parsers. We test for significance at the 95% level. Since we are doing a pairwise comparison of 6 systems, giving 15 comparisons, the p-value needs to be below 0.0034 for there to be a significant

Set $c = 0$
Compute actual statistic of score differences $|S_{\mathrm{X}} - S_{\mathrm{Y}}|$ on test data
For random shuffles $r = 0, \ldots, R$
    For sentences in test set
        Shuffle variable tuples between system X and Y with probability 0.5
    Compute pseudo-statistic $|S_{\mathrm{X_r}} - S_{\mathrm{Y_r}}|$ on shuffled data
    If $|S_{\mathrm{X_r}} - S_{\mathrm{Y_r}}| \geq |S_{\mathrm{X}} - S_{\mathrm{Y}}|$
        $c++$
$p = (c + 1)/(R + 1)$
Reject null hypothesis if p is less than or equal to specified rejection level.

Figure 12
Approximate Randomization Test for Statistical Significance Testing

difference at the 95% level.[22] For each parser, the values in the row corresponding to that parser represent the p-values for those parsers that achieve a lower f-score than that parser. This shows that the system based on Bikel's retrained parser is significantly better than those based on the other parsers with a statistical significance of >95%. For the XLE and RASP comparisons, we will use the f-structure-annotation algorithm and Bikel retrained-based LFG system.

Table 10
Comparing parsers evaluated against Section 22 dependencies (preds-only): p-values for approximate randomization test for 10,000,000 randomizations

	PCFG	Parent-PCFG	Collins M3	Charniak	Bikel	Bikel+Tags
PCFG	-	-	-	-	-	-
Parent-PCFG	<0.0001	-	-	-	-	-
Collins M3	<0.0001	<0.0001	-	-	-	-
Charniak	<0.0001	<0.0001	<0.0001	-	-	-
Bikel	<0.0001	<0.0001	<0.0001	0.0003	-	-
Bikel+Tags	<0.0001	<0.0001	<0.0001	<0.0001	<0.0001	-

5. Cross-Formalism Comparison of Treebank-Induced and Hand-Crafted Grammars

From the experiments in Section 4, we choose the treebank-based LFG system using the retrained version of Bikel's parser (which retains Penn-II functional tag labels) to compare against parsing systems using deep, hand-crafted, constraint-based grammars at the level of dependencies. We report on two experiments. In the first experiment (Section 5.1), we evaluate the f-structure annotation algorithm and Bikel retrained parser-based LFG system against the hand-crafted, wide-coverage LFG and XLE parsing system (Riezler et al. 2002; Kaplan et al. 2004) on the PARC 700 Dependency Bank (King et al. 2003). In the second experiment (Section 5.2), we evaluate against the hand-

22 Based on Cohen (1995:190): $\alpha_e \approx 1$ - (1 - $\alpha_c)^m$, where m is the number of pairwise comparisons, α_e is the experiment-wise error and α_c is the per-comparison error.

crafted, wide-coverage unification grammar and RASP parsing system of Carroll and Briscoe (2002) on the CBS 500 Dependency Bank (Carroll, Briscoe, and Sanfilippo 1998).

5.1 Evaluation against PARC 700

The PARC 700 Dependency Bank (King et al. 2003) provides dependency relations (including LDD relations) for 700 sentences randomly selected from WSJ Section 23 of the Penn-II Treebank. In order to evaluate the parsers, we follow the experimental setup of Kaplan et al. (2004) with a split of 560 dependency structures for the test set and 140 for the development set. The set of features (Table 12) evaluated in the experiment form a proper superset of preds-only, but 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. There are systematic differences between the PARC 700 dependencies and the f-structures generated in our approach as regards feature-geometry, feature-nomenclature and the treatment of named-entities. In order to evaluate against the PARC 700 test set, we automatically map the f-structures produced by our parsers to a format similar to that of the PARC 700 Dependency Bank. This is done with conversion software in a post-processing stage on the f-structure annotated trees (Figure 13).

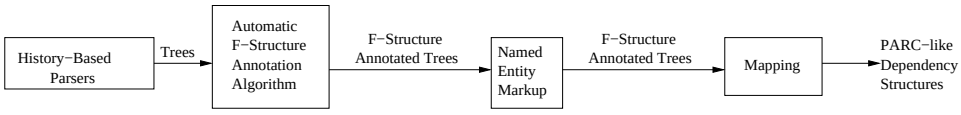

Figure 13
PARC 700 Conversion Software

The conversion software is developed on the 140-sentence development set of the PARC 700, except for the Multi-Word Expressions section. Following the experimental setup of Kaplan et al. (2004), we mark up multi-word expression predicates based on the gold standard PARC 700 Dependency Bank.

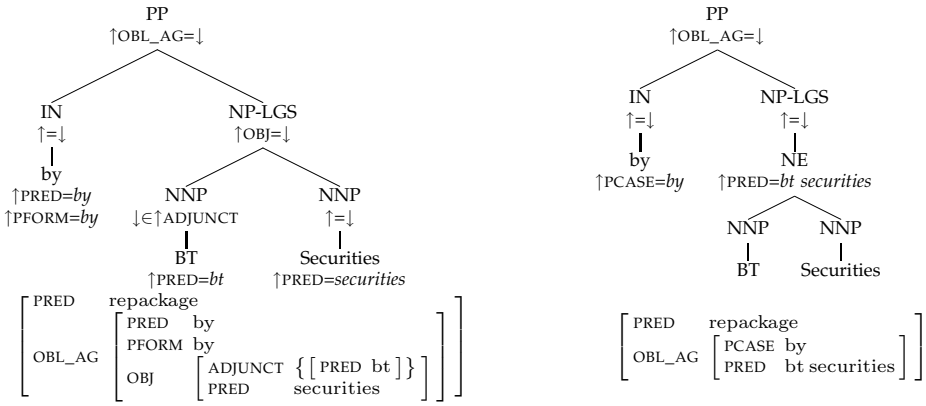

Figure 14
Named entity and OBL_AG feature geometry mapping

Multi-Word Expressions The f-structure annotation algorithm analyzes the internal structure of all noun phrase fully. In Figure 14 for example, *BT* is analyzed as an ADJUNCT modifier of the head *securities*, while PARC 700 analyzes this and other (more complex) named entities as multi-word expression predicates. The conversion software transforms the output of the f-structure annotation algorithm into the multi-word expression predicate format.

Feature Geometry In constructions such as Figure 2, the f-structure annotation algorithm analyzes *say* as the main PRED with what is said as the value of a COMP argument. In the PARC 700, these constructions are analyzed in such a way that what is said/reported etc. provides the top level f-structure while other material (who reported etc.) is analyzed in terms of ADJUNCTs modifying the top level f-structure. A further systematic structural divergence is provided by the analysis of passive oblique agent constructions (Figure 14): The f-structure annotation algorithm generates a complex internal analysis of the oblique agent PP, while the PARC analysis encodes a flat representation. The conversion software adjusts the output of the f-structure annotation algorithm to the PARC-style encoding of linguistic information.

Feature Nomenclature There are a number of systematic differences between feature names used by the automatic annotation algorithm and PARC 700: For example, DET is DET_FORM in the PARC 700, COORD is CONJ, FOCUS is FOCUS_INT. Nomenclature differences are treated in terms of a simple relabeling by the conversion software.

Additional Features A number of features in the PARC 700 are not produced by the automatic annotation algorithm. These include: AQUANT for adjectival quantifiers, MOD for NP-internal modifiers and STMT_TYPE for statement type (declarative, interrogative etc.). Additional features (and their values) are automatically generated by the mapping software, using categorial, configurational and already produced f-structure annotation information, extending the original annotation algorithm.

XCOMP Flattening The automatic annotation algorithm treats both auxiliary and modal verb constructions in terms of hierarchically cascading XCOMPs, while in PARC 700 the temporal and aspectual information expressed by auxiliary verbs is represented in terms of a flat analysis and features (Figure 15). The conversion software automatically flattens the f-structures produced by the automatic annotation algorithm into the PARC-style encoding.

For full details of the mapping, see Burke et al. (2004).

In our parsing experiments, we used the most up-to-date version of the hand-crafted, wide-coverage, deep LFG resources and XLE parsing system with improved results over those reported in Kaplan et al. (2004): This latest version achieves 80.55% f-score, a 0.95% improvement on the previous 79.6%. The XLE parsing system combines a large-scale, hand-crafted LFG for English and a statistical disambiguation component to choose the most likely analysis among those returned by the symbolic parser. The statistical component is a log-linear model trained on 10,000 partially labeled structures from the WSJ. The results of the parsing experiments are presented in Table 11. We also include a figure for the upper bound of each system.[23] Using Bikel's retrained

23 The upper bound for the treebank-based LFG system is determined by taking the original Penn-II WSJ Section 23 trees corresponding to the PARC 700 strings, automatically annotating them with the f-structure annotation algorithm and evaluating the f-structures against the PARC 700 dependencies. The

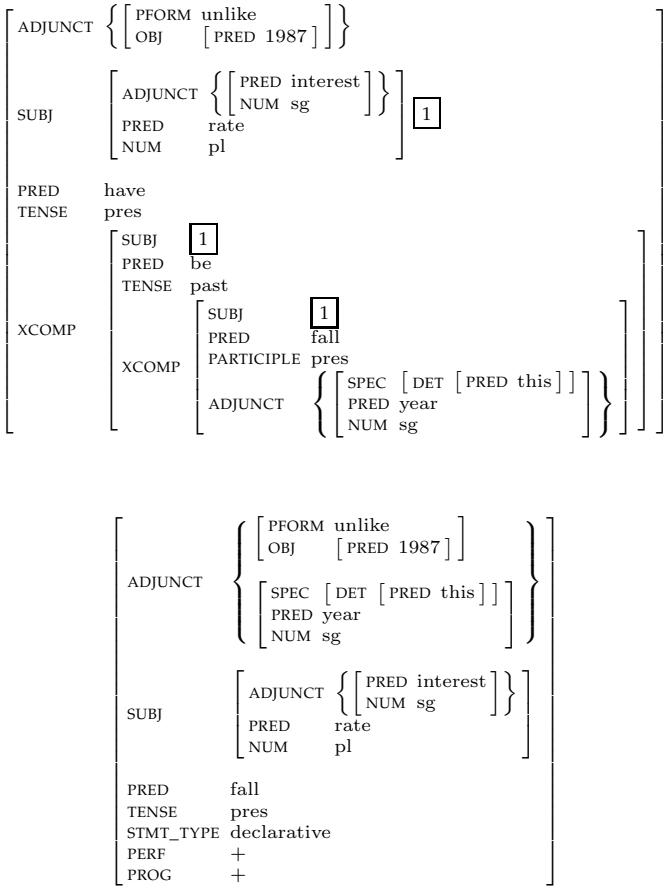

Figure 15
DCU 105 and PARC 700 analyses for the sentence *Unlike 1987, interest rates have been falling this year*

parser, the treebank-based LFG system achieves an f-score of 82.73%, while the hand-crafted grammar and XLE-based system achieves an f-score of 80.55%. The approximate randomization test produced a p-value of 0.0054 for this pairwise comparison, showing that this result difference is statistically significant at the 95% level. Evaluation results on a reannotated version (Briscoe and Carroll 2006) of the PARC 700 Dependency Bank were recently published in Clark and Curran (2007), reporting f-scores of 81.9% for the CCG parser, and 76.3% for RASP. As Briscoe and Carroll (2006) point out, these evaluations are not directly comparable with the Kaplan et al. (2004) style evaluation against the original PARC 700 Dependency Bank, since the annotation schemes are

upper bound for the XLE system is determined by selecting the XLE parse that scores best against the PARC 700 dependencies for each of the PARC 700 strings. It is interesting to note that the upper bound for the treebank-based system is only 1.18 percentage points higher than that for the XLE system. Apart from the two different methods for establishing the upper bounds, this is most likely due to the fact that the mapping required for evaluating the treebank-based LFG system against PARC 700 is lossy (cf. the discussion in Section 6).

different. Kaplan et al. (2004) and our experiments use a fine-grained feature set of 34 features (Table 12), while the Briscoe and Carroll (2006) scheme uses 17 features.

Table 11
Results of evaluation against the PARC 700 Dependency Bank following the experimental setup of Kaplan et al. (2004)

	Bikel+Tags	XLE	P-Value
F-Score	82.73	80.55	0.0054
Upper Bound	86.83	85.65	-

Table 12
Breakdown by dependency relation of results of evaluation against PARC 700

Dep.	% of Deps	F-Score(%)	
		Bikel+Tags	XLE
ADEGREE	6.17	80	82
ADJUNCT	14.32	68	66
AQUANT	0.06	78	61
COMP	1.23	80	74
CONJ	2.64	73	69
COORD_FORM	1.20	83	90
DET_FORM	4.61	97	91
FOCUS	0.02	0	36
MOD	2.74	74	67
NUM	19.82	91	89
NUMBER	1.42	89	83
NUMBER_TYPE	2.10	94	86
OBJ	8.92	87	78
OBJ_THETA	0.05	43	31
OBL	0.83	55	69
OBL_AG	0.22	82	76
OBL_COMPAR	0.07	38	56
PASSIVE	1.14	80	88
PCASE	0.25	79	68
PERF	0.41	89	90
POSS	0.98	88	80
PRECOORD_FORM	0.03	0	91
PROG	0.97	89	81
PRON_FORM	2.54	92	94
PRON_INT	0.03	0	33
PRON_REL	0.57	74	72
PROPER	3.56	83	93
PRT_FORM	0.22	80	41
QUANT	0.34	77	80
STMT_TYPE	5.23	87	80
SUBJ	8.51	78	78
SUBORD_FORM	0.93	47	42
TENSE	5.02	95	90
TOPIC_REL	0.57	56	73
XCOMP	2.29	80	78

A breakdown by dependency relation for each system is given in Table 12. The treebank-induced grammar system can better identify DET_FORM, SUBORD_FORM and

PRT_FORM dependencies[24] and achieves higher f-scores for OBJ and POSS. However, the hand-crafted parsing system can better identify FOCUS, OBL(ique) arguments, PRECO-ORD_FORM and TOPICREL relations.

5.2 Evaluation against CBS 500

We also compare the hand-crafted, deep, probabilistic unification grammar-based RASP parsing system of Carroll and Briscoe (2002) to our treebank- and retrained Bikel parser-based LFG system. The RASP parsing system is a domain-independent, robust statistical parsing system for English, based on a hand-written, feature-based unification grammar. A probabilistic parse selection model conditioned on the structural parse context, degree of support for a subanalysis in the parse forest, and lexical information (when available) chooses the most likely parses. For this experiment, we evaluate against the CBS 500,[25] developed by Carroll, Briscoe, and Sanfilippo (1998) in order to evaluate a precursor of the RASP parsing resources. The CBS 500 contains dependency structures (including some long distance dependencies[26]) for 500 sentences chosen at random from the SUSANNE corpus (Sampson 1995), but subject to the constraint that they are parsable by the parser in Carroll, Briscoe, and Sanfilippo (1998). As with the PARC 700, there are systematic differences between the f-structures produced by our methodology and the dependency structures of the CBS 500. In order to be able to evaluate against the CBS 500, we automatically map our f-structures into a format similar to theirs. We did not split the data into a heldout and a test set when developing the mapping, so that a comparison could be made with other systems that report evaluations against the CBS 500. The following CBS 500-style grammatical relations are produced from the f-structure in Figure 1:

```
(ncsubj sign U.N.)
(dobj sign treaty)
```

Some mapping is carried out (as in the evaluation against the PARC 700) on the f-structure annotated trees, and the remaining mapping is carried out on the f-structures (Figure 16). As with the PARC 700 mapping, all mappings are carried out automatically.

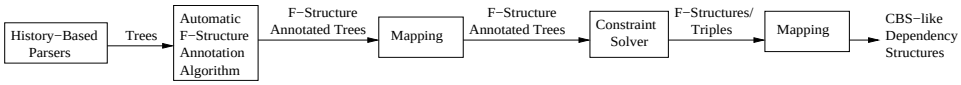

Figure 16
CBS 500 Conversion Software

The following phenomena were dealt with on the f-structure annotated trees:

Auxiliary verbs (xcomp flattening) XCOMPS were flattened to promote the main verb to the top level, while maintaining a list of auxiliary and modal verbs and their relation to one another.

24 DET_FORM, SUBORD_FORM and PRT_FORM (and in general X_FORM) dependencies record (semantically relevant) surface forms in f-structure for X-type closed class categories.
25 Downloaded from
http://www.informatics.susx.ac.uk/research/nlp/carroll/greval.html
26 The long distance dependencies include passive, wh-less relative clauses, control verbs, etc.

Treatment of topicalized sentences The predicate of the topicalized sentence became the main predicate and any other top level material became an adjunct.

Multi-word expressions Multi-word expressions (such as *according to*) were not marked up in the parser input, but captured in the annotated trees and the annotations adjusted accordingly.

Treatment of the verbs *be* and *become* Our automatic annotation algorithm does not treat the verbs *be* and *become* differently to any other verbs when they are used transitively. This analysis conflicted with the CBS 500 analysis, so was changed to match theirs.

The following are the main mappings carried out on the f-structures:

Encoding of Passive We treat passive as a feature in our automatic f-structure annotation algorithm, while the CBS 500 triples encode this information indirectly.

Objects of Prepositional Phrases No dependency was generated for these objects, as there was no corresponding dependency in the CBS 500 analyses.

Nomenclature Differences There were some trivial mappings to account for differences in nomenclature, for example OBL in our analyses became IOBJ in the mapped dependencies.

Encoding of wh-less relative clauses These are encoded by means of reentrancies in f-structure, but were encoded in a more indirect way in the mapped dependencies to match the CBS 500 annotation format.

To carry out the experiments, we POS-tagged the tokenized CBS 500 sentences with the MXPOST tagger (Ratnaparkhi 1996) and parsed the tag sequences with our Penn-II and Bikel retrained-based LFG system. We use the evaluation software of Carroll, Briscoe, and Sanfilippo (1998)[27] to evaluate the grammatical relations produced by each parser. The results are given in Table 13.

Table 13
Results of dependency evaluation against the CBS 500 (Carroll, Briscoe and Sanfillipo, 1998)

	Bikel+Tags	RASP	P-Value
F-Score	80.23	76.57	$<$0.0001

Our LFG system based on Bikel's retrained parser achieves an f-score of 80.23%, while the hand-crafted RASP grammar and parser achieves an f-score of 76.57%. Crouch et al. (2002) report that their XLE system achieves an f-score of 76.1% for the same experiment. A detailed breakdown by dependency is given in Table 14. The LFG system based on Bikel's retrained parser is able to better identify MOD(ifier) dependency relations, ARG_MOD (the relation between a head and a semantic argument which is syntactically realized as a modifier, for example *by*-phrases), IOBJ (indirect object) and AUXiliary relations. RASP is able to better identify XSUBJ (clausal subjects controlled from without), CSUBJ (clausal subjects) and COMP (clausal complement) relations. Again we use the Approximate Randomization Test to test the parsing results for statistical significance. The p-value for the test comparing our system using Bikel's retrained parser against RASP is $<$0.0001. The treebank-based LFG system using Bikel's retrained

27 Downloaded from
`http://www.informatics.susx.ac.uk/research/nlp/carroll/greval.html`

parser is significantly better than the hand-crafted, deep, unification grammar-based RASP parsing system with a statistical significance of >95%.

Table 14
Breakdown by grammatical relation for results of evaluation against CBS 500

Dep.	% of Deps	F-Score(%)	
		Bikel+Tags	RASP
DEPENDANT	100.00	80.23	76.57
MOD	48.96	80.30	75.29
NCMOD	30.42	85.37	72.98
XMOD	1.60	70.05	55.88
CMOD	2.61	75.60	53.08
DETMOD	14.06	95.85	91.97
ARG_MOD	0.51	80.00	64.52
ARG	43.70	78.28	77.57
SUBJ	13.10	79.84	83.57
NCSUBJ	12.99	87.84	84.32
XSUBJ	0.06	0.00	88.89
CSUBJ	0.04	0.00	22.22
SUBJ_OR_DOBJ	18.22	81.21	83.84
COMP	12.38	76.73	71.87
OBJ	7.34	76.05	69.53
DOBJ	5.11	84.55	84.57
OBJ2	0.25	48.00	43.84
IOBJ	1.98	59.04	47.60
CLAUSAL	5.04	77.74	75.37
XCOMP	4.03	80.00	84.11
CCOMP	1.01	69.61	75.14
AUX	4.76	94.94	88.27
CONJ	2.06	68.84	69.09

6. Discussion and Related Work

At the moment, we can only speculate as to why our treebank-based LFG resources outperform the hand-crafted XLE and RASP grammars.

In Section 4 above, we observed that the treebank-induced LFG resources have considerably wider coverage (>99.9% measured in terms of complete spanning parse) than the hand-crafted grammars (∼80% for XLE and RASP grammars on unseen treebank text). Both XLE and RASP use a number of (largest) fragment combining techniques to achieve full coverage. If coverage is a significant component in the performance difference observed between the hand-crafted and treebank-induced resources, then it is reasonable to expect that the performance difference is more pronounced with increasing sentence length (with shorter sentences being simpler and more likely to be within the coverage of the hand-crafted grammars). In other words, we expect the hand-crafted, deep, precision grammars to do better on shorter sentences (more likely to be within their coverage), while the treebank-induced grammars should show better performance on longer strings (less likely to be within the coverage of the hand-crafted grammars).

In order to test this hypothesis, we carried out a number of experiments:

First, we plot the sentence length distribution for the 560 PARC 700 test sentences and the 500 CBS 500 sentences (Figures 17 and 18). Both gold standards are approxi-

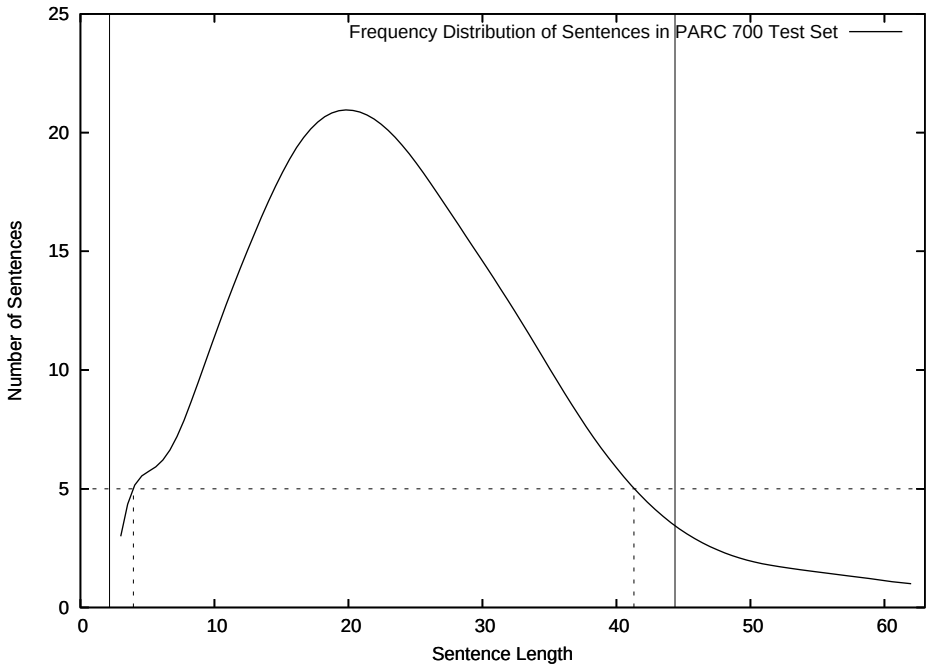

Figure 17
Distribution of sentence frequency by sentence length in the PARC 700 test set with Bezier interpolation, vertical lines mark two standard deviations from the mean.

mately normally distributed, with the CBS 500 distribution possibly showing the effects of being chosen subject to the constraint that the strings are parsable by the parser in Carroll, Briscoe, and Sanfilippo (1998). For each case we use the mean μ and 2 standard deviations 2σ to the left and right of the mean to exclude sentence lengths not supported by sufficient observations: For PARC 700, $\mu = 23.27$, $\mu - 2\sigma = 2.17$ and $\mu + 2\sigma = 44.36$, while for CBS 500, $\mu = 17.27$, $\mu - 2\sigma = 1.59$ and $\mu + 2\sigma = 32.96$. Both the PARC 700 and the CB 500 distributions are positively skewed. For the PARC 700, $\mu - 2\sigma$ is actually outside the observed data range, while for CB 500, $\mu - 2\sigma$ almost coincides with the left border. It is therefore useful to further constrain the $\pm 2\sigma$ range by a sentence count threshold of ≥ 5.[28] This results in a sentence length range of 4–41 for PARC 700 and 4–32 for CBS 500.

Second, in order to test whether fragment parses increase with sentence length, we plot the percentage of fragment parses over sentence length for the XLE parses of the 560 sentence test set of the PARC 700 (we did not do this for the CBS 500 as its strings are selected to be fully parsable by RASP). Figure 19 shows that the number of fragment parses tends to increase with sentence length.

Third, we plot the average dependency f-score for the hand-crafted and the treebank-induced resources against sentence lengths. Figure 20 shows the results for PARC 700, Figure 21 for CBS 500.

28 Note that since the distributions in Figures 17 and 18 are Bezier interpolated, the constraint does not guarantee that every sentence length within the range occurs 5 or more times.

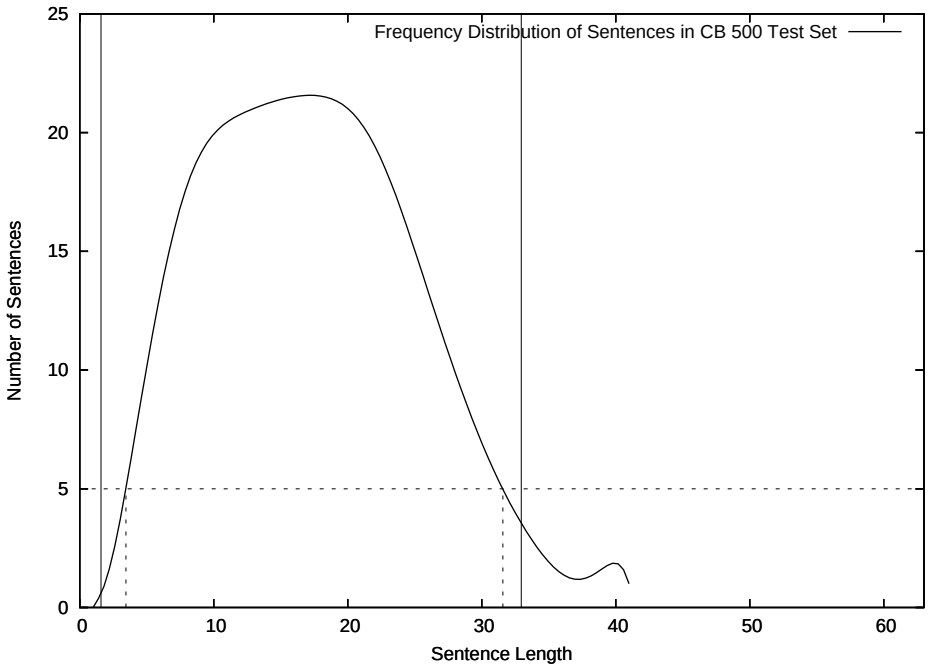

Figure 18
Distribution of sentence frequency by sentence length in the CB 500 test set with Bezier interpolation, vertical lines mark two standard deviations from the mean.

In both cases, contrary to our (perhaps somewhat naive) assumption, the graphs show that the treebank-induced resources outperform the hand-crafted resources within (most of) the 4–41 and 4–32 sentence length bounds, with the results for the very short and the very long strings outside those bounds not being supported by sufficient data points.

In the parsing literature, results are often also provided for strings with lengths ≤ 40. Below we give those results and statistical significance testing for the PARC 700 and CBS 500 (Tables 15 and 16). The results show that the Bikel retrained-based LFG system achieves a higher dependency f-score on sentences of length ≤ 40 than on all sentences, while the XLE system achieves a slightly lower score on sentences of length ≤ 40. The Bikel retrained system achieves an f-score of 83.18%, a statistically significant improvement of 2.67% over the XLE system on sentences of length ≤ 40. Against the CBS 500, Bikel's retrained system achieves a weighted f-score of 82.58%, a statistically significant improvement of 3.87% over the RASP system which achieves a weighted f-score of 78.81% on sentences of length ≤ 40.

Finally, we test whether the strict preds-only dependency evaluation has an effect on the results for the PARC 700 experiments. Recall that following Kaplan et al. (2004) for the PARC 700 evaluation we used a set of semantically relevant grammatical functions that is a superset of preds-only and a subset of all-GFs. A preds-only based evaluation is "stricter" and tends to produce lower scores as it directly reflects the effects of predicate-argument/adjunct misattachments in the resulting dependency representations (while local functions such as NUM(ber), for example, can score properly even

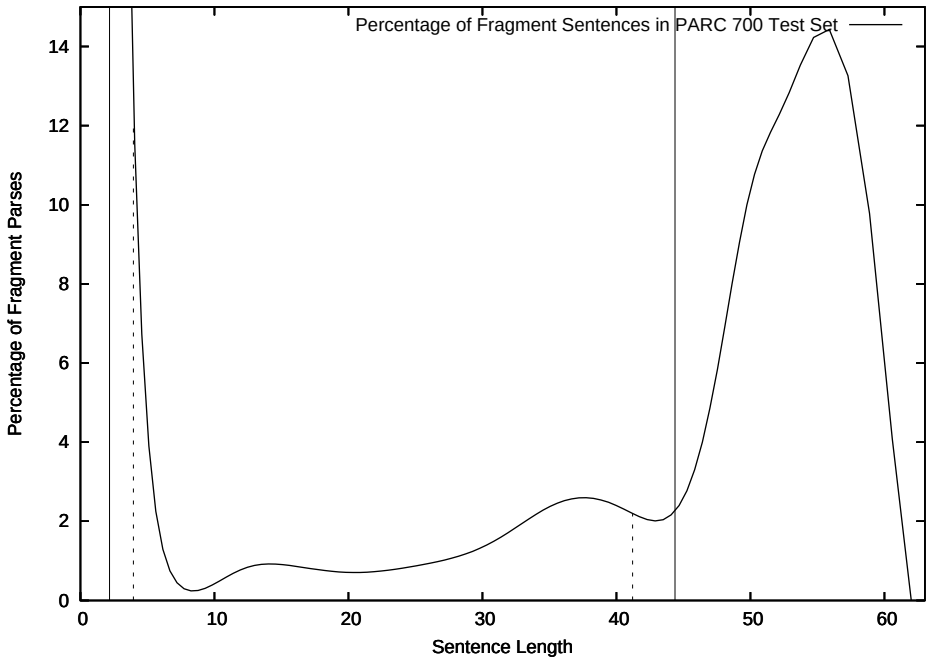

Figure 19
Percentage of fragment sentences for XLE parsing system per sentence length with Bezier interpolation.

Table 15
Evaluation and significance testing of sentences length ≤ 40 against the PARC 700

	All sent. lengths	length ≤ 40
	F-Score	F-Score
Bikel + Tags	82.73	83.18
XLE	80.55	80.51
P-Value	0.0054	0.0010

if the local predicate is misattached). Table 17[29] below gives the results for preds-only evaluation[30] on the PARC 700 for all sentence lengths. The results show that the Bikel retrained treebank-based LFG resource achieves an f-score of 77.40%, 5.33% lower than the score for all the PARC dependencies. The XLE system achieves an f-score of 74.31%, 6.24% lower than the score for all the PARC dependencies and a 3.09% drop against the results obtained by the Bikel retrained treebank-based LFG resources. The performance of the Bikel retrained-based LFG system suffers less than the XLE system when preds-only dependencies are evaluated.

29 We do not include a p-value here as the breakdown by function per sentence was not available to us for the XLE data.

30 The dependency relations we include in preds-only evaluation are: ADJUNCT, AQUANT, COMP, CONJ, COORD_FORM, DET_FORM, FOCUS_INT, MOD, NUMBER, OBJ, OBJ_THETA, OBL, OBL_AG, OBL_COMPAR, POSS, PRON_INT, PRON_REL, PRT_FORM, QUANT, SUBJ, SUBORD_FORM, TOPIC_REL, XCOMP.

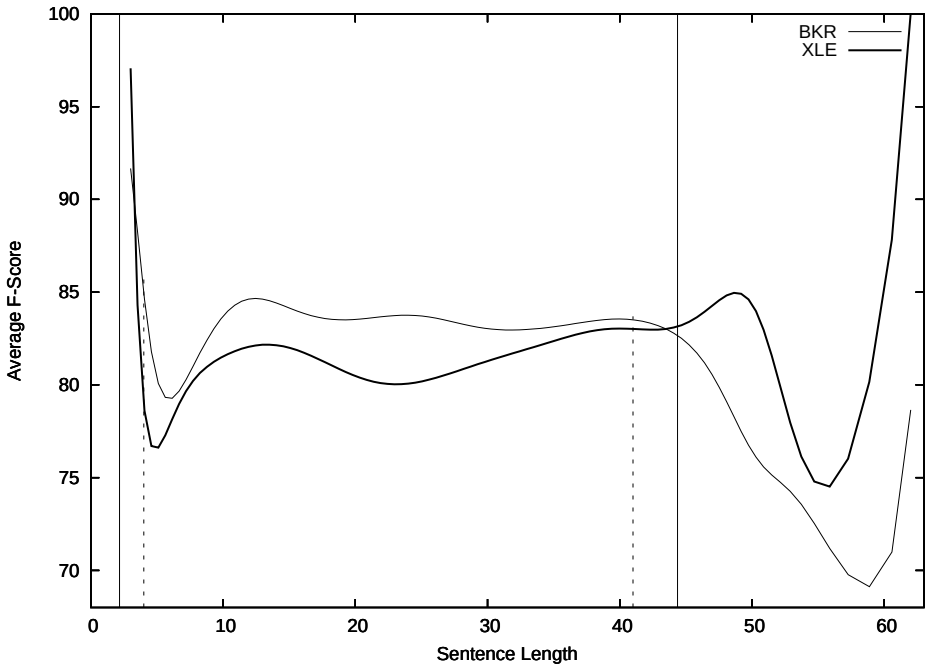

Figure 20
Average f-score by sentence length for PARC 700 test set with Bezier interpolation.

Table 16
Evaluation and significance testing of sentences length ≤ 40 against the CBS 500

	All sent. lengths	length ≤ 40
	F-Score	F-Score
Bikel + Tags	80.23	82.58
RASP	76.57	78.81
P-Value	<0.0001	<0.0001

Table 17
Preds-only evaluation against the PARC 700 Dependency Bank

	All GFs F-Score	Preds Only F-Score
Bikel + Tags	82.73	77.40
XLE	80.55	74.31

It is important to note that in our f-structure annotation algorithm and treebank-based LFG parsing architectures, we do not claim to provide fully adequate statistical models. It is well known (Abney 1997) that PCFG- or history-based parser approximations to general constraint-based grammars can yield inconsistent probability models due to loss of probability mass: The parser successfully returns the highest ranked parse tree but the constraint solver cannot resolve the f-structure equations and the

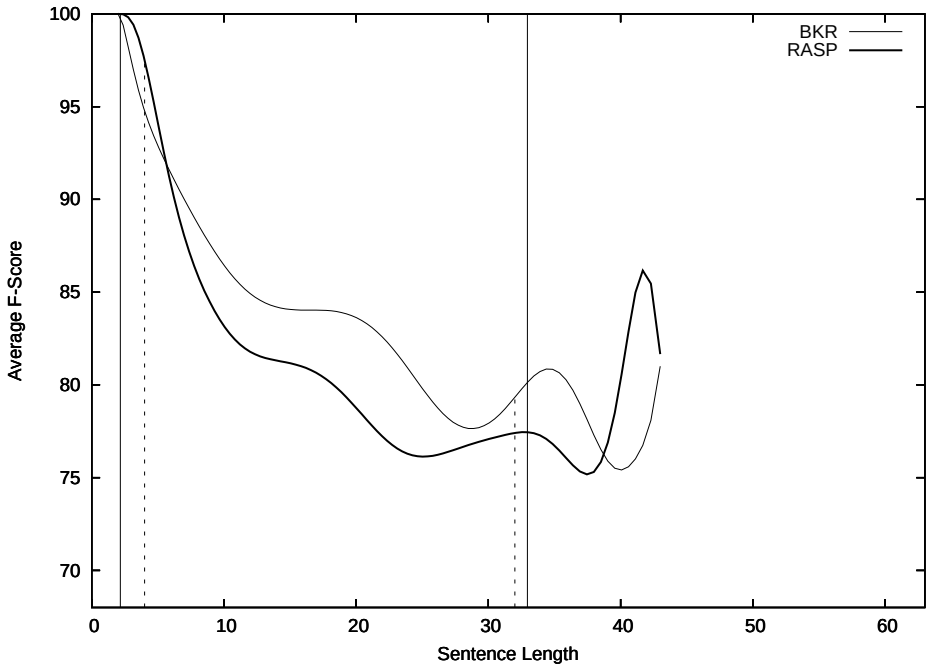

Figure 21
Average f-score by sentence length for CB 500 test set with Bezier interpolation.

probability mass associated with that tree is lost. Research on adequate probability models for constraint-based grammars is important (Bouma, van Noord, and Malouf 2000; Miyao and Tsujii 2002; Riezler et al. 2002; Clark and Curran 2004). In this context, it is interesting to compare parser performance against upper bounds. For the PARC 700 evaluation, the upper bound for the XLE-based resources is 85.65%, against 86.83% for the treebank-based LFG resources. XLE-based parsing currently achieves 94.05% (f-score 80.55%) of its upper bound using a discriminative disambiguation method, while the treebank-based LFG resource achieves 95.28% (f-score 82.73%) of its upper bound.

While this seems to indicate that the two disambiguation models achieve similar results, the figures are actually very difficult to compare. In the case of the XLE, the upper-bound is established by unpacking all parses for a string and scoring the best match against the gold standard (rather than letting the probability model select a parse). By contrast, in the case of the treebank-based LFG resources, we use the original "perfect" Penn-II treebank trees (rather than the trees produced by the parser), automatically annotate those trees with the f-structure annotation algorithm and score the results against the PARC 700 (it is not feasible to generate all parses for a string, there are simply too many for treebank-induced resources). The upper-bound computed in this fashion for the treebank-based LFG resource (86.83%) is relatively low. The main reason is that the automatic mapping required to relate the f-structures generated by the treebank-based LFG resources to the PARC 700 dependencies is lossy. This is indicated by comparing the upper-bound for the treebank-based LFG resources for the PARC 700 against the upper-bound for the DCU 105 gold standard, where little or no mapping (apart from the feature-structure to dependency-triple conversion) is required: scoring

the f-structure annotations for the original treebank trees results in 86.83% against PARC 700 vs. 96.80% against DCU 105.

Our discussion shows how delicate it can be to compare parsing systems and their disambiguation models. Ultimately what is required is an evaluation strategy that separates out and clearly distinguishes between the grammar, parsing algorithm and disambiguation model and is capable of assessing different combinations of these core components. Of course, this will not always be possible and moving towards it is part of a much more extended research agenda, well beyond the scope of the research reported in the present article. Our approach, and previous approaches, evaluate systems at the highest level of granularity, that of the complete package: the combined grammar-parser-disambiguation model. The results show that machine-learning-based resources can outperform deep, state-of-the-art hand-crafted resources with respect to the quality of dependencies generated.

Treebank-based, deep and wide-coverage constraint-based grammar acquisition has become an important research topic: Starting with the seminal TAG-based work of (Xia 1999), there are now also HPSG-, CCG- and LFG-based approaches. Miyao, Ninomiya, and Tsujii (2003) and Tsuruoka, Miyao, and Tsujii (2004) present research on inducing Penn-II treebank-based HPSGs with log-linear probability models. Hockenmaier and Steedman (2002) and Hockenmaier (2003) provide treebank-induced CCG-based models including LDD resolution. It would be interesting to conduct a comparative evaluation involving treebank-based HPSG, CCG and LFG resources against the PARC 700 and CBS 500 Dependency Banks to enable more comprehensive comparison of machine-learning-based with hand-crafted, deep, wide-coverage resources such as those used in the XLE or RASP parsing systems.

Gildea and Hockenmaier (2003) and Miyao and Tsujii (2004) present PropBank (Kingsbury, Palmer, and Marcus 2002) based evaluations of their automatically induced CCG and HPSG resources. PropBank-based evaluations provide valuable information to rank parsing systems. Currently, however, PropBank-based evaluations are somewhat partial: They only represent (and hence score) verbal arguments and disregard a raft of other semantically important dependencies (e.g., temporal and aspectual information, dependencies within nominal clusters etc.) as captured in the PARC 700 or CBS 500 Dependency Banks.[31]

7. Conclusions

Parser comparison is a non-trivial and time-consuming exercise. We extensively evaluate 4 machine-learning-based shallow parsers and two hand-crafted, wide-coverage deep probabilistic parsers involving four gold standard dependency banks, using the Approximate Randomization Test (Noreen 1989) to test for statistical significance. We use a sophisticated method for automatically producing deep dependency relations from Penn-II-style CFG-trees (Cahill et al. 2002b, 2004) to compare shallow parser output at the level of dependency relation and revisit experiments carried out by Preiss (2003) and Kaplan et al. (2004).

Our main findings are twofold:

31 In a sense, PropBank does not yet provide a single agreed upon gold standard: Role information is provided indirectly and an evaluation gold-standard has to be computed from this. In doing so, choices have to be made as regards the representation of shared arguments, the analysis of coordinate structures (and so forth) and it is not clear that currently the same choices are made for evaluations carried out by different research groups.

1. Using our CFG-tree-to-dependency annotation technology, together with treebank- and machine-learning-based parsers can outperform hand-crafted, wide-coverage, deep, probabilistic, constraint-based grammars and parsers.

This result is surprising for two reasons. First, it is established against two externally provided dependency banks (the PARC 700 and the CBS 500 gold standards), originally designed using the hand-crafted, wide-coverage, probabilistic grammars for the XLE (Riezler et al. 2002) and RASP (Carroll and Briscoe 2002) parsing systems to evaluate those systems. What is more, the SUSANNE corpus-based CBS 500 constitutes an instance of domain variation for the Penn-II-trained LFG resources, likely to adversely affect scores. Second, the treebank- and machine-learning-based LFG resources require automatic mapping to relate f-structure output of the treebank-based parsing systems to the representation format in the PARC 700 and CBS 500 Dependency Banks. These mappings are partial and lossy: That is, they do not cover all of the systematic differences between f-structure and dependency bank representations and introduce a certain amount of error in what they are designed to capture, that is they both over- and undergeneralize, again adversely affecting scores. Improvements of the mappings should lead to a further improvement in the dependency scores.

2. Parser evaluation at the level of dependency representation still requires non-trivial mappings between different dependency representation formats.

Earlier we criticized tree-based parser evaluation on the grounds that equally valid different tree-typologies can be associated with strings, and identified this as a major obstacle to fair and unbiased parser evaluation. Dependency-based parser evaluation, as it turns out, is not entirely free from this criticism either: There are significant systematic differences between the PARC 700 dependency and the CBS 500 dependency representations; there are significant systematic differences between the LFG f-structures generated by the hand-crafted, wide-coverage grammars of Riezler et al. (2002) and Kaplan et al. (2004) and those of the treebank-induced and f-structure annotation algorithm based resources of Cahill et al. (2004). These differences require careful implementation of mappings if parsers are not to be unduly penalized for systematic and motivated differences at the level of dependency representation. By and large, these differences are, however, less pronounced than differences on CFG tree representations, making dependency-based parser evaluation a worthwhile and rewarding exercise.

Summarizing our results, we find that against the DCU 105 development set, the treebank- and f-structure annotation algorithm-based LFG system using Bikel's parser retrained to retain Penn-II functional tag labels performs best, achieving f-scores of 82.92% preds-only and 88.3% all grammatical functions. Against the automatically generated WSJ Section 22 Dependency Bank, the system using Bikel's retrained parser achieves the highest results, achieving f-scores of 83.06% preds-only and 87.861% all GFs. This is statistically significantly better than all other parsers. In order to evaluate against the PARC 700 and CBS 500 gold standards, we automatically map the dependencies produced by our treebank-based LFG system into a format compatible with the gold standards. Against the PARC 700 Dependency Bank, the treebank-based LFG system using Bikel's retrained parser achieves an f-score of 82.73%, a statistically significant improvement of 2.18% against the most up-to-date results of the hand-crafted XLE-based parsing resources. Against the CBS 500, the treebank-based LFG system using Bikel's retrained parser achieved the highest f-score of 80.23%, a statistically significant

improvement of 3.66% on the highest previously published results for the same experiment with the hand-crafted RASP resources in Carroll and Briscoe (2002).

Appendix A: Parser Parameter Settings

This section provides a complete list of the parameter settings used for each of the parsers described in this article.

Parser	Parameters
Collins Model 3	We used the Collins parser with its default settings of a beam size of 10,000 and where the values of the following flags are set to 1: punctuation-flag, distaflag, distvflag and npbflag. Input was pre-tagged using the MXPOST POS tagger (Ratnaparkhi 1996). We parse the input file with all three models and use the scripts provided to merge the outputs into the final parser output file. Note that this file has been cleaned of all -A functional tags and trace nodes.
Charniak	We used the parser dated August 2005 and ran the parser using the data provided in the download on pre-tokenized sentences of length ≤ 200. Input was automatically tagged by the parser.
Bikel Emulation of Collins Model 2	We used version 0.9.9b of the parser trained on the file of observed events made available on the downloads page. We used the collins.properties file and a maximum heap size of 1500MB. Input was pre-tagged using the MXPOST POS tagger (Ratnaparkhi 1996).
Bikel + Functional Tags	We used version 0.9.9b of the parser trained on a version of sections 02–21 of the Penn-II treebank where functional tags were not ignored by the parser. We updated the default head-finding rules to deal with the new categories. We also trained on all sentences from the training set (the default collins.properties file is set to ignore trees of more than 500 tokens). Input was pre-tagged using the MXPOST POS tagger (Ratnaparkhi 1996) and the maximum heap size was 1500MB.
RASP	We used a file of parser output provided through personal communication with John Carroll. (Tagging is carried out automatically by the parser)
XLE	We used a file of parser output provided by the Natural Language Theory and Technology group at the Palo Alto Research Center. (Tagging is carried out automatically by the parser)

Acknowledgments

We are grateful to our anonymous reviewers whose insightful comments have improved the article significantly. We would like to thank John Carroll for discussion and help with reproducing the RASP parsing results, Michael Collins, Eugene Charniak, Dan Bikel and Helmut Schmid for making their parsing engines available and Ron Kaplan and the team at PARC for discussion, feedback and support. Part of the research presented here has been supported by Science Foundation Ireland grants 04/BR/CS0370 and 04/IN/I527, Enterprise Ireland Basic Research Grant SC/2001/0186, an Irish Research Council for Science, Engineering and Technology PhD studentship, an IBM PhD studentship and support from IBM's Centre for Advanced Studies (CAS) in Dublin.

References

Abney, Stephen. 1997. Stochastic attribute-value grammars. *Computational Linguistics*, 23(4):597–618.

Alshawi, Hiyan and Stephen Pulman, 1992. *Ellipsis, comparatives, and generation*, chapter 13, pages 251–275. The MIT Press, Cambridge, Massachusetts and London, England.

Baldwin, Timothy, Emily Bender, Dan Flickinger, Ara Kim, and Stephan Oepen. 2004. Road-testing the English Resource Grammar over the British National Corpus. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC 2004)*, pages 2047–2050, Lisbon, Portugal.

Bikel, Dan. 2002. Design of a Multi-lingual, Parallel-processing Statistical Parsing Engine. In *Proceedings of HLT 2002*, pages 24–27, San Diego, CA.

Black, Ezra, Steven Abney, Dan Flickenger, Claudia Gdaniec, Ralph Grishman, Philip Harrison, Donald Hindle, Robert Ingria, Fred Jelineck, Judith Klavans, Mark Liberman, Mitchell Marcus, Salim Roukos, Beatrice Santorini, and Tomek Strzalkowski. 1991. A procedure for quantitatively comparing the syntactic coverage of english grammars. In *Proceedings of the Speech and Natural Language Workshop*, pages 306–311, Pacific Grove, CA.

Bod, Rens. 2003. An Efficient Implementation of a New DOP Model. In *Proceedings of the Tenth Conference of the European Chapter of the Association for Computational Linguistics (EACL'03)*, pages 19–26, Budapest, Hungary.

Bouma, Gosse, Gertjan van Noord, and Robert Malouf. 2000. Alpino: Wide-coverage Computational Analysis of Dutch. In Walter Daelemans, Khalil Sima'an, Jorn Veenstra, and Jakub Zavrel, editors, *Computational Linguistics in The Netherlands 2000*. Rodopi, Amsterdam, pages 45–59.

Bresnan, Joan. 2001. *Lexical-Functional Syntax*. Blackwell, Oxford.

Briscoe, Edward and John Carroll. 1993. Generalized probabilistic LR parsing of natural language (corpora) with unification-based grammars. *Computational Linguistics*, 19(1):25–59.

Briscoe, Edward, Claire Grover, Bran Boguraev, and John Carroll. 1987. A formalism and environment for the development of a large grammar of English. In *Proceedings of the 10th International Joint Conference on AI*, pages 703–708, Milan, Italy.

Briscoe, Ted and John Carroll. 2006. Evaluating the accuracy of an unlexicalized statistical parser on the parc depbank. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 41–48, Sydney, Australia.

Burke, Michael. 2006. *Automatic Treebank Annotation for the Acquisition of LFG Resources*. Ph.D. thesis, School of Computing, Dublin City University, Dublin, Ireland.

Burke, Michael, Aoife Cahill, Ruth O'Donovan, Josef van Genabith, and Andy Way. 2004. 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.

Butt, Miriam, Helge Dyvik, Tracy Holloway King, Hiroshi Masuichi, and Christian Rohrer. 2002. The Parallel Grammar Project. In *Proceedings of COLING 2002, Workshop on Grammar Engineering and Evaluation*, pages 1–7, Taipei, Taiwan.

Cahill, Aoife. 2004. *Parsing with Automatically Acquired, Wide-Coverage, Robust, Probabilistic LFG Approximations*. Ph.D. thesis, School of Computing, Dublin City University, Dublin, Ireland.

Cahill, Aoife, Michael Burke, Ruth O'Donovan, Josef van Genabith, and Andy Way. 2004. 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, Aoife, Mairéad McCarthy, Josef van Genabith, and Andy Way. 2002a. Automatic Annotation of the Penn Treebank with LFG F-Structure Information. In *Proceedings of the LREC Workshop on Linguistic Knowledge Acquisition and Representation: Bootstrapping Annotated Language Data*, pages 8–15, Las Palmas, Canary Islands, Spain.

Cahill, Aoife, Mairéad McCarthy, Josef van Genabith, and Andy Way. 2002b. Parsing with PCFGs and Automatic F-Structure Annotation. In Miriam Butt and Tracy Holloway King, editors, *Proceedings of the Seventh International Conference on LFG*, pages 76–95, Stanford, CA. CSLI Publications.

Carroll, John and Edward Briscoe. 2002. High precision extraction of grammatical relations. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING)*, pages 134–140, Taipei, Taiwan.

Carroll, John, Edward Briscoe, and Antonio Sanfilippo. 1998. Parser evaluation: A survey and new proposal. In *Proceedings of the International Conference on Language Resources and Evaluation*, pages 447–454, Granada, Spain.

Carroll, John, Anette Frank, Dekang Lin, Detlef Prescher, and Hans Uszkoreit, editors. 2002. *HLT Workhop: 'Beyond PARSEVAL — Towards improved evaluation measures for parsing systems'*, Las Palmas, Canary Islands, Spain.

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

Charniak, Eugene. 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.

Chinchor, Nancy, Lynette Hirschman, and David D. Lewis. 1993. Evaluating Message Understanding Systems: An Analysis of the Third Message Understanding Conference (MUC-3). *Computational Linguistics*, 19(3):409–449.

Clark, Stephen and James Curran. 2004. Parsing the WSJ using CCG and Log-Linear Models. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 104–111, Barcelona, Spain.

Clark, Stephen and James R. Curran. 2007. Formalism-Independent Parser Evaluation with CCG and DepBank. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (to appear)*, Prague, Czech Republic.

Clark, Stephen and Julia Hockenmaier. 2002. Evaluating a Wide-Coverage CCG Parser. In *Proceedings of the LREC 2002 Beyond Parseval Workshop*, pages 60–66, Las Palmas, Canary Islands, Spain.

Cohen, Paul R. 1995. *Empirical Methods for Artificial Intelligence*. The MIT Press, Cambridge, MA.

Collins, Michael. 1997. Three Generative, Lexicalized Models for Statistical Parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pages 16–23, Madrid, Spain.

Collins, Michael. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA.

Crouch, Richard, Ron Kaplan, Tracy Holloway King, and Stefan Riezler. 2002. A comparison of evaluation metrics for a broad coverage parser. In *Proceedings of the LREC Workshop: Beyond PARSEVAL – Towards Improved Evaluation Measures for Parsing Systems*, pages 67–74, Las Palmas, Canary Islands, Spain.

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

Dienes, Péter and Amit Dubey. 2003. Antecedent recovery: Experiments with a trace tagger. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pages 33–40, Sapporo, Japan.

Eisele, Andreas and Jochen Dörre. 1986. A Lexical Functional Grammar System in Prolog. In *Proceedings of the 11th International Conference on Computational Linguistics (COLING 1986)*, pages 551–553.

Flickinger, Dan. 2000. On building a more efficient grammar by exploiting types. *Natural Language Engineering*, 6(1):15–28.

Gaizauskas, Rob. 1995. Investigations into the Grammar Underlying the Penn Treebank II . Research Memorandum CS-95-25, Department of Computer Science, Univeristy of Sheffield.

Gildea, Daniel and Julia Hockenmaier. 2003. Identifying semantic roles using combinatory categorial grammar. In Michael Collins and Mark Steedman, editors, *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pages 57–64, Sapporo, Japan.

Hockenmaier, Julia. 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.

Hockenmaier, Julia and Mark Steedman. 2002. Generative Models for Statistical Parsing with Combinatory Categorial Grammar. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 335–342, Philadelphia, PA.

Johnson, Mark. 1999. PCFG models of linguistic tree representations. *Computational Linguistics*, 24(4):613–632.

Johnson, Mark. 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*, pages 136–143, Philadelphia, PA.

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

Kaplan, Ron, Stefan Riezler, Tracy Holloway King, John T. Maxwell, Alexander Vasserman, and Richard 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 (HLT-NAACL'04)*, pages 97–104, Boston, MA.

Kaplan, Ronald and Annie Zaenen. 1989. Long-distance Dependencies, Constituent Structure and Functional Uncertainty. In Mark Baltin and Anthony Kroch, editors, *Alternative Conceptions of Phrase Structure*, pages 17–42, Chicago. Chicago University Press. Reprinted in Dalrymple et al. (editors), *Formal Issues in Lexical-Functional Grammar*. CSLI Publications, 1995.

King, Tracy Holloway, Richard Crouch, Stefan Riezler, Mary Dalrymple, and Ron Kaplan. 2003. The PARC 700 dependency bank. In *Proceedings of the EACL03: 4th International Workshop on Linguistically Interpreted Corpora (LINC-03)*, pages 1–8, Budapest, Hungary.

Kingsbury, Paul, Martha Palmer, and Mitch Marcus. 2002. Adding Semantic Annotation to the Penn TreeBank. In *Proceedings of the Human Language Technology Conference*, pages 252–256, San Diego, CA.

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

Leech, Geoffrey and Roger Garside. 1991. Running a Grammar Factory: On the compilation of Parsed Corpora, or 'Treebanks'. In Stig Johansson and Anna-Brita Stenström, editors, *English Computer Corpora: selected papers*. Mouton de Gruyter, Berlin, pages 15–32.

Levy, Roger and Christopher D. Manning. 2004. Deep dependencies from context-free statistical parsers: correcting the surface dependency approximation. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL 2004)*, pages 328–335, Barcelona, Spain.

Lin, Dekang. 1995. A dependency-based method for evaluating broad-coverage parsers. In *Proceedings of the International Joint Conference on AI*, pages 1420–1427, Montréal, Canada.

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

Marcus, Mitchell, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The Penn Treebank: Annotating Predicate Argument Structure. In *Proceedings of the ARPA Workshop on Human Language Technology*, pages 110–115, Princeton, NJ.

McCarthy, Mairéad. 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.

McDonald, Ryan and Fernando Pereira. 2006. Online Learning of Approximate Dependency Parsing Algorithms. In *In the Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 81–88, Trento, Italy.

Miyao, Yusuke, Takashi Ninomiya, and Jun'ichi Tsujii. 2003. Probabilistic modeling of argument structures including non-local dependencies. In *Proceedings of the Conference on Recent Advances in Natural Language Processing (RANLP)*, pages 285–291, Borovets, Bulgaria.

Miyao, Yusuke and Jun'ichi Tsujii. 2002. Maximum Entropy Estimation for Feature Forests. In *Proceedings of Human Language Technology Conference (HLT 2002)*, pages 292–297, San Diego, CA.

Miyao, Yusuke and Jun'ichi Tsujii. 2004. Deep Linguistic Analysis for the Accurate Identification of Predicate-Argument Relations. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING 2004)*, pages 1392–1397, Geneva, Switzerland.

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

O'Donovan, Ruth, Michael Burke, Aoife Cahill, Josef van Genabith, and Andy Way. 2004. Large-Scale Induction and Evaluation of Lexical Resources from the Penn-II Treebank. In *Proceedings of the 42nd*

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

Pollard, Carl and Ivan Sag. 1994. *Head-driven Phrase Structure Grammar*. CSLI Publications, Stanford, CA.

Preiss, Judita. 2003. Using Grammatical Relations to Compare Parsers. In *Proceedings of the Tenth Conference of the European Chapter of the Association for Computational Linguistics (EACL'03)*, pages 291–298, Budapest, Hungary.

Ratnaparkhi, Adwait. 1996. A Maximum Entropy Part-Of-Speech Tagger. In *Proceedings of the Empirical Methods in Natural Language Processing Conference*, pages 133–142, Philadelphia, PA.

Riezler, Stefan, Tracy King, Ronald Kaplan, Richard Crouch, John T. Maxwell, and Mark Johnson. 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.

Sampson, Geoffrey. 1995. *English for the Computer: The SUSANNE Corpus and analytic scheme*. Clarendon Press, Oxford.

Tsuruoka, Yoshimasa, Yusuke Miyao, and Jun'ichi Tsujii. 2004. Towards efficient probabilistic HPSG parsing: integrating semantic and syntactic preference to guide the parsing. In *Proceedings of IJCNLP-04 Workshop: Beyond shallow analyses - Formalisms and statistical modeling for deep analyses*, Hainan Island, China. [No page numbers].

van Genabith, Josef and Richard Crouch. 1996. Direct and Underspecified Interpretations of LFG f-Structures. In *16th International Conference on Computational Linguistics (COLING 96)*, pages 262–267, Copenhagen, Denmark.

van Genabith, Josef and Richard Crouch. 1997. On Interpreting f-Structures as UDRSs. In *Proceedings of ACL-EACL-97*, pages 402–409, Madrid, Spain.

Xia, Fei. 1999. Extracting Tree Adjoining Grammars from Bracketed Corpora. In *Proceedings of the 5th Natural Language Processing Pacific Rim Symposium (NLPRS-99)*, pages 398–403, Beijing, China.

Xue, Nianwen, Fei Xia, Fu-Dong Chiou, and Martha Palmer. 2004. The Penn Chinese TreeBank: Phrase Structure Annotation of a Large Corpus. *Natural Language Engineering*, 10(4):1–30.

Yeh, Alexander. 2000. More accurate tests for the statistical significance of result differences. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING, 2000)*, pages 947–953, Saarbrücken, Germany.