Evaluating Automatic LFG F-Structure Annotation for the Penn-II Treebank

Michael Burke, Aoife Cahill, Mairéad McCarthy, Ruth O'Donovan, Josef van Genabith and Andy Way
({mburke,acahill,mccarthy,rodonovan,josef,away}@computing.dcu.ie)
School of Computing, Dublin City University, Dublin, Ireland

Abstract.
Lexical-Functional Grammar (LFG: Kaplan and Bresnan, 1982; Bresnan, 2001; Dalrymple, 2001) f-structures represent abstract syntactic information approximating to basic predicate-argument-modifier (dependency) structure or simple logical form (van Genabith and Crouch, 1996; Cahill *et al.*, 2003a). A number of methods have been developed (van Genabith *et al.*, 1999a,b; Sadler *et al.*, 2000; Frank, 2000; van Genabith *et al.*, 2001; Frank *et al.*, 2003) for automatically annotating treebank resources with LFG f-structure information. Until recently, however, most of this work on automatic f-structure annotation has been applied only to limited data sets, so while it may have shown 'proof of concept', it has not yet demonstrated that the techniques developed scale up to much larger data sets. More recent work (Cahill *et al.*, 2002a,b) has presented efforts in evolving and scaling techniques established in these previous papers to the full Penn-II Treebank (Marcus *et al.*, 1994). In this paper, we present a number of quantitative and qualitative evaluation experiments which provide insights into the effectiveness of the techniques developed to automatically derive a set of f-structures for the more than 1,000,000 words and 49,000 sentences of Penn-II. Currently we obtain 94.85% Precision, 95.4% Recall and 95.09% F-Score for preds-only f-structures against a manually encoded gold standard.

Keywords: LFG, unification grammar, automatic annotation, evaluation, corpora, treebanks

1. Introduction

A number of methods have been developed (van Genabith *et al.*, 1999a,b; Sadler *et al.*, 2000; Frank, 2000; van Genabith *et al.*, 2001; Frank *et al.*, 2003) for automatically annotating treebank resources with Lexical-Functional Grammar (LFG) (Kaplan and Bresnan, 1982; Bresnan, 2001; Dalrymple, 2001) f-structure information. F-structures are feature structures which represent abstract syntactic information, approximating to basic predicate-argument-modifier (dependency) structure or simple logical form (van Genabith and Crouch, 1996; Cahill *et al.*, 2003a). Until recently, however, most of the work on automatic f-structure annotation outlined above has been applied only to the 100 publicly available sentences of the AP treebank (Leech & Garside, 1991) and the first 166 sentences of the Susanne corpus

(Sampson, 1995). As a consequence, this research has had to face the criticism that while it may have shown 'proof of concept', it has not yet demonstrated that the techniques developed scale up to much larger data sets.

More recent work (Cahill *et al.*, 2002a,b) has presented efforts in evolving and scaling up techniques established in these previous papers to a different treebank, Penn-II (Marcus *et al.*, 1994). Of course, Penn-II is many orders of magnitude larger than the AP and Susanne treebank fragments covered in earlier work—more than 1,000,000 words in over 49,000 sentences—so this newer work is an attempt to demonstrate that sets of f-structures can indeed be successfully annotated automatically on a much larger scale than has been the case heretofore.

In the earlier work on automatic annotation, f-structure annotations are defined either in terms of manually annotating automatically extracted CFG rule sets which are then automatically rematched against the treebank trees (van Genabith *et al.*, 1999a,b), or in terms of regular expression-based annotation principles applied to CFG rules extracted from the treebank (Sadler *et al.*, 2000; van Genabith *et al.*, 2001), or in terms of a rewriting system that rewrites flat sets of term descriptions of treebank trees (Frank, 2000). These approaches are compared in greater detail in (Frank *et al.*, 2003).

For the research presented here, we have developed a novel automatic annotation algorithm comprising four basic procedures which, compared to our earlier research, are slightly more coarse-grained, as our main aim is that the methodology scales to complete, large-scale treebanks. The algorithm associates nodes in treebank trees with functional annotations (attribute-value structure equations describing f-structures) from which f-structures can be computed by a constraint solver. The algorithm is described in detail in the next section.[1]

In the earlier papers cited above, the automatically produced f-structure annotations were compared against a set of 'gold standard' annotations produced by a linguist. Impressive results were presented for Precision and Recall (greater than 91% for both). Given the scale of the task involved in automatically annotating Penn-II, annotation results cannot be evaluated against a manually constructed set of f-structures for the *entire* set of over 49,000 sentences in the treebank.

In this paper we report the results for a number of automatic evaluation techniques for assessing the effectiveness of the techniques we have developed. The evaluation includes quantitative and qualitative

[1] Note expressly that we are not arguing that the earlier approaches (regular expression-based annotation or set rewriting) cannot be scaled to complete treebanks. Indeed, in related, ongoing research we are working on scaling these to full treebanks.

metrics. Quantitative metrics do not involve a 'gold standard', while qualitative metrics do. For the quantitative evaluation, we demonstrate the coverage of our annotation algorithm with respect to rule types and tokens, and we also provide details of f-structure fragmentation, as well as annotation failure where a set of automatically generated f-structure descriptions is unresolvable and results in no f-structure being produced. The qualitative measures compare the f-structure annotations generated by our automatic annotation procedure against those contained in a manually constructed 'gold standard' set of f-structures. 105 trees from section 23 of the Wall Street Journal (WSJ) part of the Penn-II treebank were randomly selected and manually annotated with f-structure descriptions by a linguist. We then use two measures to compare the automatically generated set of equations against the gold standard set: firstly, we use the standard `evalb` test on the automatically and manually annotated trees, and secondly, following (Crouch *et al.*, 2002, Riezler *et al.*, 2002) we calculate Precision and Recall on flat sets of term based descriptions of the f-structures generated.

Currently 48,175 sentences (99.48% of the Penn-II trees without FRAG and X constituents) receive a single, complete f-structure, and 28 trees are assigned two f-structure fragments. 221 trees are not associated with any f-structure due to inconsistent annotations. Using `evalb` and the manually annotated gold-standard, we obtain 90.37% Precision and Recall, with 35 out of the 105 trees receiving a completely correct set of annotations under automatic annotation. We also calculate Precision and Recall directly on sets of term descriptions of the automatically derived and the reference f-structures. For the preds-only set of equations (i.e. with the lexical annotations excluded), we obtain a Precision of 94.85%, Recall of 95.4% and F-Score of 95.09%. For the complete f-structure term descriptions the results are: Precision 96.4%, Recall 97.4% and F-Score 96.9%.

Finally, we provide concluding remarks and present some avenues for further work.

2. Automatic F-Structure Annotation for Penn-II

The task of our project is to provide a set of f-structures corresponding to the complete set of sentences in Penn-II. Given the scale of the task, the overriding concern is that our automatic annotation algorithm be robust while simultaneously maintaining quality. The algorithm, written in Java and Perl, recursively annotates the nodes in the Penn-II trees with f-structure descriptions (in the form of attribute-value structure equations). From these a constraint solver then generates f-structures which encode basic predicate-argument-modifier structures for the more than 49,000 sentences in the treebank. However, some of these automatically generated f-structures may be partial or unconnected, in that in a few cases a sentence may be associated with two or more unconnected f-structure fragments (rather than a single f-structure).

In comparison to the regular expression-based and set-rewriting-based methods of automatic annotation developed in previous work, our current methodology is slightly more coarse-grained, both with respect to the construction of f-structures and the formulation of the linguistic principles which guide the annotation process. Certain reentrancies (e.g. distribution of subjects into coordinate structures) are not yet encoded.

In our current work, automatic annotation is defined in terms of an annotation algorithm based on the following four components:

1. Left/right context annotation principles (L/R): these are based on an automatic partition of local trees of depth one (corresponding to CFG rules) into head (we adapt head-lexicalisation rules from Magerman, 1994; Collins, 1996), left- and right-context. Annotations then depend on categorial, positional and functional tag information in the automatically head-lexicalised Penn-II trees. Arguments, adjuncts and co-heads (e.g. determiners) are distinguished in left and right contexts, and nodes in the trees are annotated accordingly. These L/R principles do not apply if the local tree RHS contains coordination.

2. Coordination annotation principles (CP): these are used for coordinate structures. We identify the coordinating categories and the coordinate constituents.

3. "Catch-All" annotation principles (CA): these apply defaults to previously unannotated nodes (usually based on categorial and functional Penn-II tags irrespective of positional information) or, in some cases, rewrite previous annotations where L/R and CC principles overgenerate.

4. Trace annotation principles (TR): the Penn-II treebank employs a rich arsenal of traces to encode both long-distance and more local dependencies to relate "moved" material to where it should be interpreted semantically.

The advantage of designing the algorithm in terms of these four separate components, rather than simply hard coding the linguistic generalisations directly into an algorithm, is that this design enables us to ensure that both the linguistic information and the algorithm itself are reusable and easily maintained. We shall describe each of these components, which work roughly in the sequence outlined (the CP component reinvokes the L/R component to annotate local constituents that do not belong to the coordination set), in the next four sections.

Finally, we encode a set of general lexical macros for each Penn-II leaf category (POS) which provides the appropriate f-structure annotations at the word level (with the lemma corresponding to the surface form as the value of the `pred` attribute).[2] As an example, a 3rd person singular verb such as *has* receives the annotations ↑`pred='have'`, ↑`tense=pres`, ↑`pers=3`, ↑`num=sg` from the lexical macro for VBZ POS tags.

2.1. L/R Annotation Principles

Our annotation algorithm is a recursive procedure which traverses trees top-down. In almost all cases, annotation is driven by categorial, basic configurational and functional Penn-II tag information in local subtrees of depth one (i.e. CFG-rules).[3]

The annotation algorithm itself is dependent on locating the head daughter. In an automatic pre-processing step, we transform the treebank into head-lexicalised form. Initially, we used Collins' (1996) head lexicalised grammar annotation scheme, but following experiments which for our purposes generated improved results, we adopted Magerman's (1994) scheme (except for coordinate structures—see next section for details).[4] We revised the order of categories in the lists provided by Magerman to improve the results. As an example, for VP rules, Magerman gives the following (left-to-right) order of likely candidates for the local head:

(1) VP: VBD, VBN, MD, VBZ, TO, VB, VP, VBG, VBP, ADJP

That is, the most likely head of a VP is a VBD (simple past verb), and the least likely (yet plausible in case there is no other match) category for headedness is an ADJP. However, note that for Magerman, if, inside

[2] We lemmatise the Penn-II treebank in a preprocessing step using the XLE English morphological analyser `http://www2.parc.com/istl/groups/nltt/xle/`.

[3] Possessive NPs are an exception, in that we annotate the mother node with the annotation ↑`poss`=↓. This departs from standard LFG theory (e.g. Kaplan and Bresnan, 1982). Another example are prenominal adjectival phrases headed by DT-like POS tags such as *almost all*. In the Penn-II annotation scheme such constituents are annotated as adjectival phrases but should be analysed as complex `spec` attributes rather than elements of an adjunct set in f-structure.

[4] The two schemes are similar: Collins' (1996) scheme builds on Magerman (1994).

a VP a VP daughter is found together with a VBG (present participle) or a VBP (non-3rd person present tense), then the VP daughter rather than the VBG or the VBP daughter will be designated as the head of the VP. We have reversed this order (VBP, VBG, VP) and improved results have been achieved. One other simple, successful change here was to place MD (modal) as the first candidate to be tested for headedness: if a modal occurs at all, it is *always* the head of a VP.

During automatic annotation, it is easy to identify the head constituent in a head-lexicalised local tree as that constituent which bears the same terminal string as the mother of the local tree.[5] Once this is identified, we provide linguistic generalisations over the left (the prefix) and right (suffix) context of the head for each preterminal category occurring with such heads.

Table I. Simplified, partial annotation matrix for NP rules

NP	left context	head	right context
subcat	DT,CD: $\uparrow\mathtt{spec}=\downarrow$	NN,NNS,NNP: $\uparrow=\downarrow$	...
non-subcat	ADJP: $\downarrow\in\uparrow\mathtt{adjn}$		SBAR,VP: $\uparrow\mathtt{relmod}=\downarrow$
	NN,NNS,NP: $\downarrow\in\uparrow\mathtt{adjn}$		PP: $\downarrow\in\uparrow\mathtt{adjn}$
	...		NN,NNS,NP: $\downarrow\in\uparrow\mathtt{app}$
	...		...

We follow the traditional distinction in LFG between subcategorisable (subj, obj, obj2, obl, xcomp, comp, poss) and non-subcategorisable (adjn, xadjn, relmod, app ...) grammatical functions. Essentially, subcategorisable grammatical functions are arguments and non-subcategorisable functions are modifiers. Knowing the mother, and with our tripartite distinction between head, left context and right context, we construct an 'annotation matrix' for each LHS category in the Penn-II CFG-rule set which our algorithm uses to annotate nodes in the treebank. To illustrate the basic idea, a much simplified, partial example matrix is given in Table I for NPs. In English, the rightmost nominal (NN, NNS, NNP etc.) on the RHS is (usually) the head. Heads are annotated $\uparrow=\downarrow$. Any DT (determiner) or CD (quan-

[5] While this method works almost flawlessly, we recently uncovered an example for which it failed, namely *Then, just as the Tramp is given a blind girl to cure in "City Lights," the Artist is put in charge of returning a two-year-old waif (Nicole Alysia), whose father has been murdered by thugs, to her mother.* Note that the sentence contains the word *is* twice—we found that the wrong token was assigned the annotation $\uparrow=\downarrow$. This problem was fixed locally, but we are currently engaged in discovering a general solution to examples of this type.

tifier/numeral) constituent in the left context is annotated $\uparrow$`spec`$=\downarrow$, while an ADJP in the left context receives the annotation $\downarrow\in\uparrow$`adjn`. Any nominal in the left context (in noun noun sequences) is annotated $\downarrow\in\uparrow$`adjn`. Regarding non-subcategorisable functions in the right context for NPs, an SBAR or VP (relative clause) is annotated $\uparrow$`relmod`$=\downarrow$, a PP receives the annotation $\downarrow\in\uparrow$`adjn`, while nominal phrases following the head (often separated by commas) are identified as appositions and are assigned the annotation $\downarrow\in\uparrow$`app`.

Note that we have adopted a conservative approach to the assignment of f-structure annotations. For example, with respect to NPs, all internal PPs (unless marked by the Penn-II tag -CLR for closely related) are considered to be adjuncts rather than oblique arguments $\uparrow$`obl`$=\downarrow$. In order to reduce errors to a bare minimum, we assign subcategorisable grammatical functions (i.e. arguments) only where there is no doubt, e.g. an NP following a preposition in a PP is assigned $\uparrow$`obj`$=\downarrow$; a NP following a verb in a VP constituent is assigned the annotation $\uparrow$`obj`$=\downarrow$, a VP following a verb in a VP $\uparrow$`xcomp`$=\downarrow$, and so on.

We noted above that our annotation matrices are constructed on the basis of the CFG rules in the Penn-II grammar. However, the fact that there are over 19,000 distinct rule types prevents us from considering all rules for the construction of the annotation matrices. Instead, in order to construct the matrices, we select only the most frequently used rule types for each different rule LHS to populate the matrix tables. To be precise, we analyse those most frequent rule types which together cover at least 85% of the *token* occurrences of all rules expanding any particular LHS category. Table II shows for a subset of rule LHSs how many rule *types* there are in total for each LHS, and how many most frequent rule types we actually analyse to populate our matrices. For instance, while there are over 6500 distinct NP rules types in Penn-II, we analyse only the 102 most frequent of them (only 1.5% of the total number of NP rule types) to populate our NP annotation matrix, as sketched in Table I.

The fact that we can sample so few rules and still achieve good results is due to an interesting property of treebanks. For each rule LHS category, a small number of very frequently occurring rules expand those categories, while there exists a large number of much less frequent rules, many of which may occur only once or twice in the whole treebank.[6]

[6] Indeed, in (Cahill *et al.*, 2002a), we design a number of probabilistic parsers where, following (Krotov *et al.*, 1998), thresholding is applied—all rules appearing fewer than five times are excluded. For the basic probabilistic parser trained on sections 02–21, and tested on section 23 of the WSJ, using all rules, we obtain

Table II. No. of most frequent rule types analysed to construct annotation matrices

LHS Category	Total No. Rule Types	No. Rule Types Used
ADJP	525	41
ADVP	234	6
NP	6595	102
PP	345	3
PRN	371	54
QP	379	18
S	2602	20
SBAR	131	3
SINV	302	32
VP	10239	307
WHNP	131	2

In addition, despite being constructed on the basis of very few most frequent rule types, during the automatic annotation process the annotation matrices generalise to the less frequent, unseen rule types in two ways: firstly, any constituents in the unseen rules which match the left/right context and head specifications receive the appropriate annotations during the recursive application of the annotation algorithm to a treebank tree; and secondly, the Penn-II treebank contains a number of categories which differ from their monadic counterparts only with respect to the addition of functional information tags (e.g. NP-TMP (temporal), NP-LOC (locative), etc.). Accordingly, our algorithm annotates not only the rules corresponding to the monadic categories, but also their functional tag-annotated counterparts. With respect to NP, for example, all daughters of NP-LOC, NP-TMP and so on are annotated using the same NP annotation matrix.

The only 'constituents' in the Penn-II treebank that remain uncovered in our work to date are FRAG(ment) and X (unknown constituents). These occur in 743 of the 49,167 trees. Note also that all L/R context annotation principles apply only if the local tree does not contain any instance of a coordinating conjunction CC. These constructions are handled by the second component in our annotation algorithm.

Precision of 78.39% and Recall of 75.54%. For the threshold-5 grammar, results deteriorate only slightly: Precision is 74.7% and Recall is 74.36%.

2.2. Coordination Annotation Principles CP

In the Penn-II treebank, there are two types of coordinate structures, namely coordination of 'like' and 'unlike' constituents. The latter carry the category label UCP. It is often the case that treebank grammars contain much less recursion than manually constructed grammars, so that the resulting flat RHSs pose problems, particularly so for coordinate structures. Integrating a treatment of coordinate structures into the L/R principles described in the previous section unduly complicates those principles and makes them harder to maintain and extend. Accordingly, we treat coordinate structures in a separate component in our algorithm.

As before, annotation is predicated on finding the local head. In cases where like constituents are coordinated, where just one CC (coordinating element `and, or, ...`) is found, this is always the head and receives the annotation ↑=↓. This differs from both Magerman's (1994) and Collins' (1996) head schemas which do not provide clauses specific to rules with CC's in their RHSs. A wide range of different RHS configurations are possible, of course, but to give a simple example, if both the immediate left and right sisters of the CC are of the same category, then all such constituents are assigned as elements of the set of conjuncts, i.e. ↓∈↑`conj`. Where more than one CC is found, we designate the rightmost such constituent to be the head, and where CC is the first daughter on the RHS (e.g. *And if rain doesn't fall soon* ...), we annotate the CC with ↓∈↑`adjn`. After application of the coordination annotation principles, all remaining unannotated constituents in a coordinate construction receive annotations from the other three components of the automatic annotation algorithm.

With respect to UCPs, again, if there is just one CC, this is assumed to be the head. Then the following principles apply:

1. if there is only one non-punctuation constituent to the left or right of the head, mark it as an element of the coordination set (i.e. ↓∈↑`conj`),

2. if (1) applies, search for constituents similar to the one found in (1) to the right or left and mark them as an elements of the coordination set (similarity is defined in terms of similarity sets related to head rules in (Magerman, 1994) and (Collins, 1996)),

3. if (1) and hence (2) does not apply, search for nominal sequences and make them elements of the coordination set,

4. if (1), (2) and (3) do not apply assign non-punctuation constituents as elements of the coordination set.

2.3. Catch-All Annotation Principles CA

Despite the previous two annotation steps, some categories on the RHS of some rules may not have been assigned any annotation. In a catch-all operation, therefore, we mark all remaining unannotated components bearing Penn-II functional tags with appropriate annotations: -SBJ receive the annotation $\uparrow\texttt{subj}{=}\downarrow$, -PRD $\uparrow=\downarrow$ and -CLR $\uparrow\texttt{obl}{=}\downarrow$ for the first such item, and $\downarrow\in\uparrow\texttt{adjn}$ for subsequent cases, and so on.

In a certain number of cases the CA annotation principles overwrite (i.e. correct) annotations generated by the L/R and CC components. Accepting a limited amount of overgeneration in these components and catching the relevant exceptions in CA often allows for a simpler statement of the L/R and CC annotation principles.

2.4. Trace Annotation Principles TP

The first three components of the automatic annotation algorithm completely ignore the rich system of empty productions and traces in the Penn-II treebank employed to coindex “displaced” linguistic material with the locations where such material should be interpreted semantically. The first three components of the annotation algorithm produce “proto-” as opposed to “proper” f-structures. Proto-f-structures interpret linguistic material purely locally where it occurs in the tree. In order to obtain proper f-structures we need to translate the traces in the treebank trees into corresponding coindexations (reentrancies) in the f-structures. The trace annotation principles translate traces for A and A$'$ movement (movement to argument and non-argument positions) including traces for wh-constructions, relative clauses, fronted elements and subjects of participal clauses, gerunds and infinitival clauses (including both controlled and arbitrary PRO) into corresponding reentrancies in the f-structure representations (cf. Figure. 1).

The trace component also handles the passive construction. In the treebank passives encoded in terms of combinations of an optional *by* PP with an NP-LGS argument marked as logical subject by the functional tag, an empty NP production in object position coindexed with the surface subject and sequences of forms of *be* or *get* followed by a past participle form. The trace component of the automatic f-structure annotation algorithm reflects such combinations in terms of a $\uparrow$ `passive=+` annotation in the local f-structure.

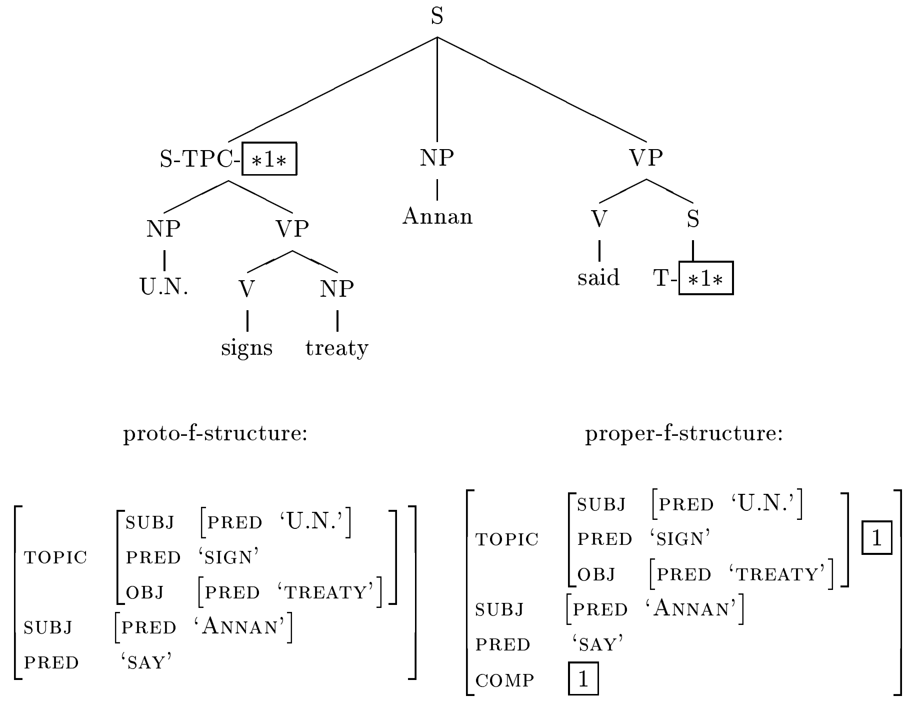

Figure 1. Tree with Traces and Proto- vs. Proper F-Structure

3. Running the Automatic Annotation Algorithm

Our automatic annotation algorithm is implemented in terms of a Java/Perl program. Annotation of the complete WSJ section of the Penn-II treebank takes less than 5 minutes on a Pentium IV PC. Once annotated, for each tree we collect the feature structure annotations and feed them into a constraint solver implemented in Prolog. The constraint solver can cope with equality constraints, disjunction and simple set-valued feature constraints. In previous automatic f-structure annotation work (van Genabith *et al.*, 1999a,b; Sadler *et al.*, 2000; van Genabith *et al.*, 2001; Frank *et al.*, 2003), we permitted disjunctive constraints, but in our current work, this has so far been avoided. Accordingly, therefore, for each tree in the treebank we either get a single f-structure, or in the case of partially annotated trees (where certain nodes do not receive an f-structure annotation), a number of unconnected f-structure fragments, or in the case of feature structure clashes, no f-structure is produced.

We pointed out that annotation matrices for FRAG(ments) and X (unknown) constituents have not been constructed. Currently, our

Compound yields assume reinvestment of dividends and that the current yield continues for a year.

```
(S (NP-SBJ (NN Compound)
           (NNS yields))
   (VP (VBP assume)
       (UCP (NP (NP (NN reinvestment))
                (PP (IN of)
                    (NP (NNS dividends))))
            (CC and)
            (SBAR (IN that)
                  (S (NP-SBJ (DT the)
                             (JJ current)
                             (NN yield))
                     (VP (VBZ continues)
                         (PP-TMP (IN for)
                                 (NP (DT a)
                                     (NN year))))))))
   (. .)
)
```

Figure 2. Tree from file wsj_0004 the Penn-II treebank

annotation scheme does not involve subsumption constraints. Such constraints would allow us to, e.g. distribute material into coordinate structures. This is currently beyond the capabilities of our constraint solver, but other avenues using the Xerox Linguistic Environment (XLE)[7] will be pursued in future work.

In order to illustrate our automatic annotation methodology, we provide in Figure 2 a sentence from file wsj_0004 of the WSJ part of the Penn-II treebank. Figure 2 shows the string *Compound yields assume reinvestment of dividends and that the current yield continues for a year* and the Penn-II parse tree, and Figure 3 the f-structure derived automatically via our method. The annotated tree resulting from the application of the automatic annotation algorithm is given in Figure 4.

An evaluation of the results appears in the next section, where we measure coverage and fragmentation over the whole treebank, and precision and recall against a set of gold standard annotations constructed manually for a test set of 105 trees randomly selected from section 23 of the WSJ section of the Penn-II treebank.[8]

[7] http://www2.parc.com/istl/groups/nltt/xle/

[8] The full set of gold-standard reference f-structures are available at: http://www.computing.dcu.ie/research/nclt/.

```
subj : adjunct : 1 : pred : compound
                     num : sing
                     pers : 3
       pred : yield
       num : pl
       pers : 3
pred : assume
tense : pres
obj : conj : 2 : pred : reinvestment
                 num : sing
                 pers : 3
                 adjunct : 3 : pform : of
                               obj : pred : dividend
                                     num : pl
                                     pers : 3
             4 : that : +
                 comp : subj : spec : det : pred : the
                               adjunct : 5 : pred : current
                               pred : yield
                               num : sing
                               pers : 3
                        pred : continue
                        tense : pres
                        adjunct : 6 : pform : for
                                      obj : spec : det : pred : a
                                            pred : year
                                            num : sing
                                            pers : 3
      pred : and
```

Figure 3. F-structure automatically generated for tree in Figure 2

4. Automatic Evaluation

In this section, we report a number of automatic evaluation experiments for assessing the effectiveness of the techniques we have developed. They involve quantitative and qualitative metrics, each of which will be addressed in turn in the following sections.

4.1. Quantitative Evaluation

For the quantitative evaluation, we demonstrate the coverage of our annotation algorithm with respect to rule types and tokens, and we also provide details of fragmentation, as well as complete annotation failure where a set of unresolvable descriptions results in no f-structure being produced by the constraint solver.

4.1.1. *Coverage*

The automatic f-structure annotation algorithm recursively traverses the Penn-II trees and applies annotations at all appropriate nodes. As an example, applying our automatic annotation to the tree-structure in Figure 2 produces the annotated tree-structure in Figure 4.

```
(S
   (NP-SBJ[up-subj=down]
      (NN[down-elem=up:adjunct] Compound[up-pred='compound',
                                         up-num=sing,up-pers=3])
      (NNS[up=down] yields[up-pred='yield',
                           up-num=pl,up-pers=3]))
   (VP[up=down]
      (VBP[up=down] assume[up-pred='assume',up-tense=pres])
      (UCP[up-obj=down]
         (NP[down-elem=up:conj]
            (NP[up=down]
               (NN[up=down] reinvestment[up-pred='reinvestment',
                                         up-num=sing,up-pers=3]))
               (PP[down-elem=up:adjunct]
                  (IN[up=down] of[up-pform='of'])
                  (NP[up-obj=down]
                     (NNS[up=down] dividends[up-pred='dividend',
                                             up-num=pl,up-pers=3]))))
         (CC[up=down] and[up-pred='and'])
         (SBAR[down-elem=up:conj]
            (IN[up=down] that[up-that='+'])
            (S[up-comp=down]
               (NP-SBJ[up-subj=down]
                  (DT[up-spec:det=down] the[up-pred='the'])
                  (JJ[down-elem=up:adjunct]
                      current[up-pred='current'])
                  (NN[up=down] yield[up-pred='yield',
                                     up-num=sing,up-pers=3]))
               (VP[up=down]
                  (VBZ[up=down] continues[up-pred='continue',
                                          up-tense=pres])
                  (PP-TMP[down-elem=up:adjunct]
                     (IN[up=down] for[up-pform='for'])
                     (NP[up-obj=down]
                        (DT[up-spec:det=down] a[up-pred='a'])
                        (NN[up=down] year[up-pred='year',
                                          up-num=sing,up-pers=3]))))))))
   (. .))
```

Figure 4. Automatically annotated tree-structure for sentence in Figure 2

In order to evaluate the coverage of our method, we extract the CFG grammar rule types with the automatically generated f-structure annotations. From the annotated tree-structure in Figure 4, the following thirteen annotated rules are extracted:

$$\begin{array}{llll}
\text{S} \longrightarrow & \text{NP-SBJ} & \text{VP} & \\
 & \uparrow\texttt{subj}=\downarrow & \uparrow=\downarrow & \\[1em]
\text{SBAR} \longrightarrow & \text{IN} & \text{S} & \\
 & \uparrow=\downarrow & \uparrow\texttt{comp}=\downarrow & \\[1em]
\text{NP-SBJ} \longrightarrow & \text{NN} & \text{NNS} & \\
 & \downarrow\in\uparrow\texttt{adjunct} & \uparrow=\downarrow & \\[1em]
\text{NP-SBJ} \longrightarrow & \text{DT} & \text{JJ} & \text{NN} \\
 & \uparrow\texttt{spec}=\downarrow & \downarrow\in\uparrow\texttt{= adjunct} & \uparrow=\downarrow \\[1em]
\text{NP} \longrightarrow & \text{NP} & \text{PP} & \\
 & \uparrow=\downarrow & \downarrow\in\uparrow\texttt{adjunct} & \\[1em]
\text{NP} \longrightarrow & \text{DT} & \text{NN} & \\
 & \uparrow\texttt{spec}=\downarrow & \uparrow=\downarrow & \\[1em]
\text{NP} \longrightarrow & \text{NN} & & \\
 & \uparrow=\downarrow & & \\[1em]
\text{NP} \longrightarrow & \text{NNS} & & \\
 & \uparrow=\downarrow & & \\[1em]
\text{VP} \longrightarrow & \text{VBP} & \text{UCP} & \\
 & \uparrow=\downarrow & \uparrow\texttt{obj}=\downarrow & \\[1em]
\text{VP} \longrightarrow & \text{VBZ} & \text{PP-TMP} & \\
 & \uparrow=\downarrow & \downarrow\in\uparrow\texttt{adjunct} & \\[1em]
\text{UCP} \longrightarrow & \text{NP} & \text{CC} & \text{SBAR} \\
 & \downarrow\in\uparrow\texttt{conj} & \uparrow=\downarrow & \downarrow\in\uparrow\texttt{conj} \\[1em]
\text{PP} \longrightarrow & \text{IN} & \text{NP} & \\
 & \uparrow=\downarrow & \uparrow\texttt{obj}=\downarrow & \\[1em]
\text{PP-TMP} \longrightarrow & \text{IN} & \text{NP} & \\
 & \uparrow=\downarrow & \uparrow\texttt{obj}=\downarrow &
\end{array}$$

Figure 5. Annotated rules extracted from Figure 4

This procedure is repeated for all 19,000 rule types in the treebank. We then measure the percentage of constituents in annotated rule type RHSs which do not receive an annotation against the total number of rule type RHS elements. These results are given in Table III for the major rule types. Note that with two exceptions, our automatic method annotates over 99.6% of the RHS constituents for these types. It is unsurprising that one of the two outliers here is PRN (parenthetical material), as it is difficult to predict exactly what material will be included in these rule RHSs, so while our method still ensures that over 93% of such material is annotated, our linguistic generalisations are less successful than for other categories which exhibit more predictable behaviour. The other LHS category whose constituents are annotated less than 99.6% of the time is QP (Quantifier Phrase) with 98.9%. As a matter of ongoing work, we are continuing to analyse these cases in order to improve coverage.

Table III. Percentage of annotated constituents in rule type RHSs

LHS Category	No. RHS Constituents	No. RHS Annotated	Percentage Annotated
ADJP	1641	1639	99.87
ADVP	605	603	99.66
NP	30735	30726	99.66
PP	1073	1071	99.81
PRN	1373	1283	93.44
QP	1555	1538	98.90
S	14817	14815	99.98
SBAR	409	409	100.00
SBARQ	256	256	100.00
SINV	1644	1643	99.93
SQ	650	648	99.69
UCP	649	647	99.69
VP	40824	40822	99.99
WHNP	367	367	100.00

In order to factor in the frequency of rule type use in the corpus, we also compute the number of constituents which occur on rule RHSs tokens (not types) and do not receive an annotation. These results are given in Table IV for all categories occurring more than 20,000 times in the treebank. Note that annotations are obtained more than 99% of the time by our method. We already pointed out that due to the flat treebank tree representations, coordinate structures in particular are difficult to annotate. In order to test whether annotation coverage is affected adversely by the presence of coordinate structures, for one experiment (*Without Coordination*) we factored out CONJP and UCP rules as well as all rules containing a CC constituent on their RHS and measured annotation coverage on the remaining rule token RHS constituents. The results show that annotation coverage is not adversely affected by the presence of coordinate structures.

Table IV. Percentage of unannotated constituents in rule token RHSs

	All Rules		*Without Coordination*	
Category	No. RHS Constituents	No. RHS Unannotated	No. RHS Constituents	No. RHS Unannotated
CD	44936	1	44068	1
DT	101135	11	99349	11
IN	121760	7	121213	6
JJ	75257	6	72234	4
NN	163923	8	157887	7
NNP	114041	3	107031	3
NNS	73955	0	69972	0
NP	277909	27	257704	26
PP	64128	22	62528	20
PRP	21357	1	21343	1
RB	38125	23	36958	16
S	58890	3	52052	2
SBAR	25703	1	24847	1
TO	27443	1	27424	0
VB	32545	0	32094	0
VBD	37468	0	37329	0
VBN	24854	1	24537	1
VBZ	26423	0	26254	0
VP	179992	1	168166	1

While the results in Table III and Table IV measure the coverage of our automatic annotation algorithm, they do not inform us as to how many of the assigned annotations are correct. Of course, the frequency of occurrence of a rule is an important factor here: it is far more serious if we do not fully (and correctly) annotate all elements in a rule which applies often, compared to a rule which occurs only rarely.

4.1.2. *Fragmentation and Feature-Value Clashes*

The results provided in the previous section measure the coverage of our algorithm but not the errors made in assigning annotations. The results given in this section are a first step towards trying to identify some of these mistakes. Some f-structure annotations are inconsistent, in that full f-structures cannot be built from the set of equations generated owing to feature clashes. To give a simple, invented example, if both *Annan* and *a mandate* are assigned the equation $\uparrow$`obj`$=\downarrow$ given the string *UN gave Annan a mandate*, the constraint resolution algorithm will not be able to unify the paths `obj:pred:annan` and `obj:pred:mandate`. Consequently, for this simple example, it is imperative that (as is indeed the case in our annotation algorithm) *a mandate* be assigned the annotation $\uparrow$`obj2`$=\downarrow$, which permits the construction of an appropriate f-structure for this string.

Unannotated nodes in a tree will cause the generation of a number of "dangling" f-structure fragments (n unannotated nodes will generate $n+1$ fragments) rather than a single, complete f-structure for that tree. In each such case, the sub-f-structure below the unannotated node will not be integrated into the encompassing f-structure.

The f-structure fragmentation and failure results obtained are summarised in Table V. Note again that the Penn-II treebank consists of 49,167 trees, but the results in Table V exclude 743 trees containing FRAG and X constituents, these being the only constituents which have so far been excluded from our annotation procedure. This leaves us with 48,424 trees.

We generate 48,231 f-structure fragments altogether, with an average of 1.0006 fragments per tree, excluding the 221 trees which currently receive no f-structure (due to feature clashes). In an earlier paper (Cahill *et al.*, 2002b), we reported that 78.836% of the Penn-II trees received one f-structure, but this has been improved considerably to currently 99.486%. (Cahill *et al.*, *op cit.*) reported 2701 sentences receiving 0 f-structures, which has been reduced tenfold in the figures presented here. Furthermore, over 10% of the trees previously were assigned two f-structure fragments, compared to just 0.058% here. Finally, the previously reported figures contained trees which received up to 11 f-structure fragments, whereas fragmentation has been reduced to a maximum of 2 f-structure fragments for in total 28 trees.

The results given in Table V provide us with our first handle on the quality (or rather lack of quality) of our automatic annotation in terms of detecting fatal annotation errors made by our automatic procedure. We can identify attribute-value clashes, cyclical f-structures, uninstantiated variables and the like, but it is a good deal harder to spot cases of wrong grammatical function assignment, the oblique/adjunct

Table V. Automatic Annotation Results: F-Structure Fragmentation and Failure

No. f-structure (fragments)	No. sentences	Percentage
0	221	0.456
1	48175	99.486
2	28	0.058

distinction and so on. In this respect, fragmentation is predominantly a measure of coverage.

4.2. Qualitative Evaluation

This section describes a set of experiments designed to provide insights into the quality of the f-structures constructed by our automatic annotation methodology. The qualitative measures compare the f-structure annotations generated by our automatic annotation procedure against those contained in a manually constructed gold standard set of f-structures. These consist of 105 trees selected at random from section 23 of the WSJ section of the Penn-II treebank. The average string length is 23.98 words, with the shortest string 2 words, and the longest 45 words. These were manually annotated, and after a number of iterations, refined to provide a set of complete, correct annotations. The task that our automatic annotation method is confronted with is to match as many of these correct annotations as possible. We use two measures to evaluate the success of our procedure: we perform the standard `evalb` test to compare the automatically annotated trees with the manually annotated reference trees, as well as calculating Precision and Recall on flat set descriptions of the f-structures following (Crouch *et al.*, 2002).

4.2.1. *Evaluation with* `evalb`

In this section, we describe experiments using `evalb` on the annotated Penn-II trees in our testset. `evalb`[9] is a bracket scoring program designed by Sekine & Collins which reports precision, recall, non-crossing and tagging accuracy for given data. The main advantage for us in using `evalb` is that it provides a quick and cheap way of evaluating the automatically annotated trees against the manually annotated ones in the gold standard. In order to use `evalb`, we treat tree nodes consisting of a CFG category followed by one or more f-structure annotations (e.g.

[9] Available at: `http://www.cs.nyu.edu/cs/projects/proteus/evalb/`

VP[up-xcomp=down,up-subj=down-subj]) as atomic node identifiers. The results are presented in Table VI.

Table VI. Evaluation of automatically annotated trees using evalb

No. sentences	105
Bracketing Recall	90.37%
Bracketing Precision	90.37%
Complete match	33.33%
Tagging accuracy	98.65%

For the 105 test strings, we obtain 90.37% labelled Precision and Recall (P&R), with 35 out of the 105 trees (33.33%) being completely correct. The reason why results for P&R are identical is because the configurational structure and number of nodes in both test and gold-standard trees is the same. They can only differ with respect to node labels. Accordingly, therefore, if we find an annotation and it is correct (or not), the figures for P&R must be the same.

The major disadvantage with using evalb is that for our purposes it is an extremely severe evaluation metric, in that for any given node in a tree, the representation of the set of equations produced automatically must be identical to the representation of the set of manually created annotations. That is, if the f-structure equations in two sets of equations do not appear in the same order, then evalb would state that the two sets are not identical, even if the individual members of those sets are identical. For evalb, therefore, the sets {2,1,3} and {1,3,2} are different (here 1, 2 and 3 represent f-structure annotations). Similarly, partial but correct annotations (e.g. {1,3} against {1,2,3}) are scored as full mistakes by evalb.

4.2.2. *Precision and Recall Evaluation on F-Structure Descriptions*

In order to calculate P&R directly on descriptions of f-structures, we use the evaluation methodology and software presented in (Crouch *et al.*, 2002; Riezler *et al.*, 2002) for f-structures. Each f-structure is represented as a set of terms of the form: relation(argument,argument). As an example, let us assume the simplified f-structure for *Annan signed treaty* in (2):

(2) $$1:\left[\begin{array}{ll}\textsc{subj} & 2:\left[\begin{array}{ll}\textsc{pred} & \text{‘}\textsc{annan}\text{’}\\ \textsc{num} & \textsc{sg}\\ \textsc{pers} & 3\textsc{rd}\end{array}\right]\\ \textsc{pred} & \text{‘}\textsc{sign}\text{’}\\ \textsc{obj} & 3:\left[\begin{array}{ll}\textsc{pred} & \text{‘}\textsc{treaty}\text{’}\\ \textsc{num} & \textsc{sg}\\ \textsc{pers} & 3\textsc{rd}\end{array}\right]\end{array}\right]$$

Given this f-structure, we automatically ‘translate’ it into a flat set of terms corresponding to f-structure descriptions, as in (3):

(3)
```
subj(sign::1,annan::2), obj(sign::1,treaty::3),
num(annan::2,sg), pers(annan::3rd), num(treaty::3,sg),
pers(treaty::3,3rd)
```

We calculate P, R and F-score ($\frac{2PR}{P+R}$) for complete f-structures and preds-only f-structures encoding basic predicate-argument-modifier relations. A preds-only f-structure contains only paths that end in a `pred:value` pair:

(4) $$1:\left[\begin{array}{ll}\textsc{subj} & 2:\left[\textsc{pred}\ \ \text{‘}\textsc{annan}\text{’}\right]\\ \textsc{pred} & \text{‘}\textsc{sign}\text{’}\\ \textsc{obj} & 3:\left[\textsc{pred}\ \ \text{‘}\textsc{treaty}\text{’}\right]\end{array}\right]$$

Our approach differs from that of Liakata and Pulman (2002) in that all precision and recall violations are counted and always carry the same weight.

Table VII. Precision and Recall on descriptions of f-structures

	All annotations	Preds-only annotations
Precision	96.435%	94.848%
Recall	97.373%	95.329%
F-Score	96.902%	95.088%

The results are presented in Table VII. For the full set of 5445 annotations in the gold standard f-structures and the 5498 equations in the automatically generated set, we obtain Precision of over 96.4% and Recall of over 97.3%. For the preds-only set of equations (i.e. with other annotations excluded), Precision goes down to 94.8%, and Recall to 95.3%. The number of preds-only annotations in the manually annotated trees is 2762, with 2776 for the automatically annotated

Table VIII. Precision and Recall on descriptions of f-structures by grammatical function

DEPENDENCY	PRECISION	RECALL	FSCORE
adjunct	974/1053 = 92	974/999 = 97	95
app	23/25 = 92	23/23 = 100	96
comp	69/71 = 97	69/80 = 86	91
conj	177/181 = 98	177/183 = 97	97
focus	1/1 = 100	1/1 = 100	100
obj	439/457 = 96	439/450 = 98	97
obj2	1/1 = 100	1/2 = 50	67
obl	20/21 = 95	20/55 = 36	53
part	8/9 = 89	8/11 = 73	80
poss	69/72 = 96	69/74 = 93	95
relmod	50/53 = 94	50/56 = 89	92
spec	271/275 = 99	271/277 = 98	98
subj	338/348 = 97	338/353 = 96	96
topic	12/12 = 100	12/13 = 92	96
topicrel	21/23 = 91	21/22 = 95	93
xcomp	160/174 = 92	160/163 = 98	95

trees. Results broken down by grammatical function (for preds-only f-structures) are presented in Table VIII:

Not surprisingly, OBL(ique) arguments are the hardest to annotate reliably. The results show, however, that with respect to obliques the annotation algorithm is conservative (i.e. more partial than incorrect): 95% of the time it annotates an oblique, the annotation is correct. The gold standard currently does not contain enough instances of FOCUS and OBJ2 to allow reliable interpretation of the P&R results for these grammatical functions.

5. Conclusions and Further Work

This paper has presented an algorithm for automatically annotating the Penn-II treebank with LFG f-structure information and a number of quantitative and qualitative evaluation experiments. Quantitative evaluation does not involve a gold-standard, while qualitative evaluation does.

Our quantitative experiments measure the coverage of our automatic annotation algorithm. In the simplest case, we compute the percentage of rule type RHS elements that do/do not receive an f-structure annotation. In order to factor in frequency of use of rule types in the corpus, in our second experiment we compute the percentage of rule token RHSs (nodes in the treebank) that do/do not receive an annotation. F-structure fragmentation (number of trees in the treebank that receive a single, complete f-structure, number of trees that receive two unconnected f-structure fragments and so on, as well as average f-structure fragmentation for a tree) co-varies with the measures reported above and is indicative of the coverage of our automatic f-structure annotation procedure. In addition, we measure the number of trees associated with unresolvable f-structure annotations (that is, inconsistent or cyclical sets of f-annotations that fail to generate an f-structure). In contrast to counting annotated versus unannotated nodes and f-structure fragmentation, measuring unresolvable f-annotations gives a first indication of the quality (or rather: lack of quality) as opposed to mere coverage of the automatic annotation algorithm.

Quantitative evaluation is cheap and easy to implement. Qualitative evaluation involves the manual construction of a 'gold-standard' fragment against which the output of automatic annotation is evaluated. We have constructed a reference fragment consisting of 105 manually annotated trees randomly selected from section 23 of the WSJ section of the Penn-II treebank. We have presented two variants for qualitative evaluation. The first reuses the standard `evalb` evaluation software available from the probabilistic parsing community. In order to use `evalb`, we treat annotated nodes in the treebank (i.e. strings with a CFG category followed by one or more f-structure annotations) as monadic categories. Evaluation involving `evalb` in this manner is simple and easy to implement but extremely strict: for a node to match, complete string identity is required with partial annotations and annotations in non-canonical order (of f-structure annotations) counted as mistakes. The evaluation results obtained using `evalb` provide lower bounds. In order to provide a more fine-grained evaluation, our second qualitative evaluation variant involves the translation of f-structures into a flat set of terms describing test and reference f-structures using the method and software presented in (Crouch *et al.*, 2002; Riezler *et al.*, 2002).

To summarise the particular results obtained in our automatic f-structure annotation experiments to date, 48,175 sentences (99.49% of the Penn-II trees without FRAG and X constituents) receive a single, complete f-structure, and 28 trees are assigned two f-structure fragments. 221 trees are not associated with any f-structure due to inconsistent annotations. For the purposes of qualitative evaluation we randomly selected 105 trees from section 23 and manually annotated these with gold standard reference f-structure annotations. Using evalb, we obtained 90.37% Precision and Recall, with 35 out of the 105 trees receiving a completely correct set of annotations under automatic annotation. We also calculated Precision and Recall directly on sets of term descriptions of the automatically derived and the reference f-structures. For the preds-only set of equations, we obtain a Precision of 94.85%, and Recall of 95.3%, F-Score 95.09%. For the complete f-structure term descriptions the results are Precision of 96.4%, and Recall of 97.4%, F-Score 96.9%.

We are working to further reduce the amount of f-structure fragmentation and unresolvable f-structures by providing more complete and better annotation principles. Furthermore, we hope to provide a treatment for FRAG and X 'constituents' which are not currently treated. We hope to improve the quality of the annotation principles to achieve a closer match with our manually annotated gold standard.

We are planing to use the XLE constraint solver as this would allow us to express subsumption constraints in our annotations, thereby enabling the percolation of subjects into coordinate structures.

We are currently working on evaluating the results of our automatic annotation algorithm against external manually annotated reference corpora (Crouch *et al.*, 2002; Carroll *et al.*, 1999) to facilitate inter-system comparisons. Perhaps surprisingly this turns out to be less than straightforward: (Carroll *et al.*, 1999) is based on the Susanne corpus with tree bracketing structures substantially different from what is provided by Penn-II (to give one example, Susanne does not provide VP constituents) so that in addition to mapping different feature sets (LFG grammatical functions (GFs) to what (Carroll *et al.*, 1999) refer to as grammatical relations (GRs)), extra effort is required to port our automatic annotation algorithm to the new treebank to be able to carry out meaningful evaluation. The reference corpus provided by (Crouch *et al.*, 2002) is based on the Penn-II treebank but uses complex predicate names (such as *Merrill_Lynch_Capital_Markets*, wsj_2349) while our approach currently generates "fully parsed" f-structures (with the head noun predicate modified by the preceding nominal predicates) for such constituents. Again this requires extra conversion work to achieve meaningful comparison. We hope to report on this work in a subsequent publication.

An f-structure annotated version of the Penn-II treebank is an extremely valuable resource. In (Cahill *et al.*, 2002) we show how wide-coverage, probabilistic LFG grammars for English can be extracted automatically from such a resource while (Cahill *et al.*, 2003b) shows how such a grammar can be extracted for German based on automatic f-structure annotation of the TIGER treebank (Brants *et al.*, 2002).

These grammars parse new text into proto-f-structures (i.e. they do not resolve long distance dependencies). In current research we have extended these grammars to deep grammars that automatically resolve long distance dependency probabilistically at f-structure and parse new text into proper f-structures. In order to do this we automatically extract subcategorisation frames (LFG semantic forms) from the f-structure annotated treebank. We hope to report on this research in a subsequent publication.

We are currently not aware of any other approaches to associate treebank resources automatically with deep unification grammar representations (such as LFG f-structures) other than the ones cited here. Of related interest are the approaches of (Clement and Kinyon, 2003) and (Hockenmaier and Steedman, 2002). (Hockenmaier and Steedman, 2002) derive a probabilistic CCG (Combinatory Categorial Grammar) from the Penn-II resource while (Clement and Kinyon, 2003) show how grammars can be synthesised from general hand-coded principles encoded in a MetaGrammar. We generate deep, wide-coverage, probabilistic unification grammars from treebank resources. A detailed comparison is unfortunately beyond the confines of the present paper.

A sample of automatically generated f-structures for the first 1000 sentences of the Penn-II treebank is available for inspection at at `http://www.computing.dcu.ie/research/nclt/`.

Acknowledgements

The research reported here was supported by the Enterprise Ireland Basic Research grant SC/2001/186. We would like to thank our three anonymous referees and Anette Frank, Ron Kaplan, Tracy King, Mary Dalrymple for feedback and discussion and in particular Dick Crouch for his evaluation software.

References

Brants, S, S. Dipper, S. Hansen, W. Lezius and G. Smith (2002): 'The TIGER Treebank', in Proceedings of the Workshop on Treebanks and Linguistic Theories, Sozopol, Bulgaria.

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

Cahill, A., M. McCarthy, J. van Genabith and A. Way (2002a): 'Parsing Text with PCFGs and Automatic F-Structure Annotation', in *Proceedings of the Seventh International Conference on Lexical-Functional Grammar*, Athens, Greece.

Cahill, A., M. McCarthy, J. van Genabith and A. Way (2002b): '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, Las Palmas, Canary Islands, Spain, pp.8–15.

Cahill, A., M. McCarthy, M. Burke, J. van Genabith and A. Way (2003a): 'Deriving Quasi-Logical Forms for the Penn Treebank'; Computing Meaning, Volume 3, Studies in Linguistics and Philosophy, (eds.) Harry Bunt, Reinhard Muskens and Elias Thijsse, Kluwer Academic Publishers, Dordrecht/Boston/London, in press.

Cahill, A., M. Forst, M. McCarthy, R. O'Donovan, C. Rohrer, J. van Genabith and A. Way (2003b): 'Treebank-Based Multilingual Unification Grammar Development'; in the Proceedings of the Workshop on Ideas and Strategies for Multilingual Grammar Development, at the 15th European Summer School in Logic Language and Information, Vienna, Austria, 18th - 29th August 2003

Carroll, J., G. Minnen and T. Briscoe (1999): 'Corpus Annotation for Parser Evaluation', in *Proceedings of the EACL Workshop on Linguistically Interpreted Corpora (LINC-99)*, Bergen, Norway, pp.35–41.

Clement L. and A. Kinyon (2003): 'Generating parallel multilingual LFG-TAG grammars using a MetaGrammar'. Proc. ACL'03. Sapporo

Collins, M. (1996): 'A New Statistical Parser Based on Bigram Lexical Dependencies', in *Proceedings of 34th Conference of the Association of Computational Linguistics*, Santa Cruz, CA, pp.184–192.

Crouch, R., R. Kaplan, T. King and S. Riezler (2002): *A comparison of evaluation metrics for a broad coverage parser*, Beyond PARSEVAL workshop at 3rd Int. Conference on Language Resources and Evaluation (LREC'02), Las Palmas.

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

Frank, A. (2000): 'Automatic F-Structure Annotation of Treebank Trees', in *Proceedings of the The Fifth International Conference on Lexical-Functional Grammar*, CSLI Publications, Stanford, CA, pp.139–160.

Frank, A., L. Sadler, J. van Genabith and A. Way (2003): 'From Treebank Resources To LFG F-Structures' in A. Abeillé (ed.) *Treebanks: Building and Using Syntactically Annotated Corpora*, Kluwer Academic Publishers, Dordrecht/Boston/London (in press).

Hockenmaier J. and M. Steedman (2002): 'Generative Models for Statistical Parsing with Combinatory Categorial Grammar', in Proceedings of 40th Annual Meeting of the Association for Computational Linguistics, Philadelphia, 2002.

Kaplan, R. and J. Bresnan (1982): 'Lexical-Functional Grammar: a Formal System for Grammatical Representation', in J. Bresnan (ed.) *The Mental Representation of Grammatical Relations*, MIT Press, Cambridge, MA.

Krotov, A., M. Hepple, R. Gaizauskas and Y. Wilks (1998): 'Compacting the Penn Treebank Grammar', in *Proceedings of the 17th International Conference on Computational Linguistics and 36th Conference of the Association for Computational Linguistics*, Montreal, Canada, **1**:699–703.

Leech, G. and R. Garside (1991): 'Running a Grammar Factory: On the Compilation of Parsed Corpora, or 'Treebanks', in S. Johansson and A. Stenström (eds) *English Computer Corpora*, Mouton de Gruyter, Berlin, pp.15–32.

Liakata, M. and S. Pulman (2002): 'From trees to Predicate-Argument Structures', COLING'02, Proceedings of the Conference, Taipei, 24 Aug. – 1.Sept 2002

Magerman, D. (1994): *Natural Language Parsing as Statistical Pattern Recognition*, PhD Thesis, Stanford University, CA.

Marcus, M., G. Kim, M. A. Marcinkiewicz, R. MacIntyre, M. Ferguson, K. Katz and B. Schasberger (1994): 'The Penn Treebank: Annotating Predicate Argument

Structure', in *Proceedings of the ARPA Human Language Technology Workshop*, Princeton, NJ.

Riezler, S., R. Kaplan, T. King, M. Johnson, R. Crouch, and J. Maxwell III (2002): 'Parsing the Wall Street Journal using a Lexical-Functional Grammar and Discriminative Estimation Techniques', in *Proceedings of 40th Conference of the Association for Computational Linguistics*, Philadelphia, PA.

Sadler, L., J. van Genabith and A. Way (2000): 'Automatic F-Structure Annotation from the AP Treebank; LFG-2000', in *Proceedings of the Fifth International Conference on Lexical-Functional Grammar*, CSLI Publications, Stanford, CA, pp.226–243.

Sampson, G. (1995): *English for the Computer: The Susanne Corpus and Analytic Scheme*, Clarendon Press, Oxford, UK.

van Genabith, J., A. Frank and A. Way (2001): 'Treebank versus X-Bar based Automatic F-Structure Annotation', in *Proceedings of the Sixth International Conference on Lexical-Functional Grammar*, CSLI Publications, Stanford, CA, pp.127–146.

van Genabith, J., L. Sadler and A. Way (1999a): 'Data-Driven Compilation of LFG Semantic Forms', in *Proceedings of the EACL Workshop on Linguistically Interpreted Corpora (LINC-99)*, Bergen, Norway, pp.69–76.

van Genabith, J., A. Way and L. Sadler (1999b): 'Semi-Automatic Generation of f-structures from Treebanks', in *Proceedings of the Fourth International Conference on Lexical-Functional Grammar*, CSLI Publications, Stanford, CA.

van Genabith, J. and D. Crouch (1996): 'Direct and Underspecified Interpretations of LFG f-Structures', in *COLING 96*, Copenhagen, Denmark, Proceedings of the Conference, pp.262–267.