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

Aoife Cahill, Mairéad McCarthy, Josef Van Genabith and Andy Way*

School of Computer Applications,
Dublin City University,
Dublin, Ireland.
Email: away@computing.dcu.ie

Abstract

Methodologies have been developed (van Genabith *et al.*, 1999a,b; Sadler *et al.*, 2000; Frank, 2000; van Genabith *et al.*, 2001; Frank *et al.*, 2002) for automatically annotating treebank resources with Lexical-Functional Grammar (LFG: Kaplan and Bresnan, 1982) f-structure information. Until recently, however, most of this work on automatic annotation has been applied only to limited datasets, so while it may have shown 'proof of concept', it has not been demonstrated that the techniques developed scale up to much larger data sets (Liakata and Pulman, 2002). 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 and assess a number of quantitative and qualitative evaluation methodologies which provide insights into the effectiveness of the techniques developed to derive automatically a set of f-structures for the more than 1,000,000 words and 49,000 sentences of Penn-II.

1 Introduction

Methodologies have been developed (van Genabith *et al.*, 1999a,b; Sadler *et al.*, 2000; Frank, 2000; van Genabith *et al.*, 2001; Frank *et al.*, 2002) for automatically annotating treebank resources with Lexical-Functional Grammar (LFG: Kaplan and Bresnan, 1982) f-structure information. F-structures are feature structures which represent abstract syntactic information, approximating to basic predicate-argument-modifier structures. These automatic f-structure annotation methodologies are fine-grained and produce detailed f-structure representations. Until recently, however, most of the work on automatic 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 been 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

*This research was part-funded by Enterprise Ireland Basic Research grant SC/2001/186.

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 (van Genabith *et al.*, 1999a,b; Sadler *et al.*, 2000; Frank, 2000; van Genabith *et al.*, 2001; Frank *et al.*, 2002), f-structure annotations are defined either in terms of regular expression-based annotation principles applied to CFG-rules extracted from the treebank, or in terms of a rewriting system that rewrites flat sets of term descriptions of treebank trees.

For the research presented here, we have developed an automatic annotation algorithm comprising three basic procedures which, compared to our earlier research, are somewhat more coarse-grained, as our main aim is that the methodology scales to complete, large-scale treebanks. These techniques are discussed in detail in the next section.

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 huge scale of the task involved in annotating Penn-II, automatic 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 present a number of automatic evaluation methodologies for assessing the effectiveness of the techniques we have developed. They include quantitative and qualitative 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 fragmentation, as well as annotation failure where a set of unresolvable descriptions results in no unified 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 treebank were randomly selected and manually annotated with f-structure descriptions. 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 annotated trees, and secondly calculate Precision and Recall on the flat set descriptions of f-structures generated.

Finally, we summarise our findings, provide some concluding remarks and offer some avenues for further work.

2 Automatic F-Structure Annotation for Penn-II

Ultimately, we are attempting 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, recursively annotates the nodes in the Penn-II trees with f-structure descriptions. This gives rise to a set of 'proto-f-structures' which encode basic predicate-argument-modifier structures for the more than 49,000 sentences in the treebank. However, currently some of these proto-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). Furthermore, certain reentrancies (for wh- and other cases of movement

phenomena, and distribution of subjects into auxiliary strings in VPs and in coordinate structures) are not yet encoded.

In comparison to the regular expression-based and set-rewriting-based methods of automatic annotation developed in previous work, our current methodology is more coarse-grained, both with respect to the construction of f-structures and the formulation of the linguistic principles which guide the annotation process.

Automatic annotation is defined in terms of an annotation algorithm based on the following three components:

1. Left/right (L/R) context annotation principles: these partition local trees into head (cf. Magerman, 1994; Collins, 1996), left- and right-context. Annotations depend on categorial, positional and functional tag information in the 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: these are used for coordinate structures. We identify the coordinating categories and the coordinate constituents.

3. "Catch-All" annotation principles: these are based on categorial and functional annotation tags in the Penn-II trees that apply irrespective of positional information.

The advantage of incorporating these three separate components into the algorithm, rather than hard coding the linguistic generalisations directly into the algorithm, is that we ensure that both the linguistic information and the algorithm itself are reusable and easily maintained. We shall describe each of these components, which work in the sequence outlined, in the next three sections. Finally, we encode a set of lexical macros for each Penn-II leaf category which provides the appropriate f-structure annotations at the word level. As an example, a 3rd person singular verb such as *has* receives the annotations ↑`pred='has'`, ↑`tense=pres`, ↑`pers=3`, and ↑`num=sing` from the VBZ macro.

2.1 L/R Annotation Principles

Our annotation algorithm is a recursive procedure which traverses trees top-down. In almost all cases,[1] annotation is driven by categorial and basic configurational information in local subtrees of depth one (i.e. CFG-rules).

[1] Possessive NPs are an exception, in that whenever we encounter a POS (possessive) tag, we annotate the mother node with the annotation ↑`poss`=↓. This deparTs from standard LFG theory (e.g. Kaplan and Bresnan, 1982).

The annotation procedure itself is dependent on locating the head daughter. In a preprocessing step, we transform the treebank into head-lexicalised form. During automatic annotation, it is easy to identify the head constituent in a local tree as that constituent which bears the same terminal string as the mother of the local tree.[2] 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. Initially, we used Collins' (1996) head lexicalised grammar annotation scheme, but following experiments which generated improved results, we currently use Magerman's (1994) scheme (except for coordinate structures—see next section for details). In ongoing work, we are revising the order of lists of categories provided by Magerman to improve the results still further. As an example, for VP rules, Magerman gives the following order of possible 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), and the least likely (yet plausible) category for headedness is an ADJP. However, note that for Magerman, if a VP daughter is found together with a VBG (present participle) or a VBP (non-3rd person present tense), then neither of the latter two categories 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.

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

Table 1: Simplified, partial annotation matrix for NP rules

We follow the traditional distinction in LFG between subcategorisable (`subj, obj, obj2, obl, xcomp, comp, poss`) and non-subcategorisable (`adjn, xadjn` ...) grammatical functions. Essentially, the set of subcategorisable grammatical functions are arguments, while the non-subcategorisable set are modifiers. Knowing the mother, and with our tripartite distinction between head, left and right context, we construct an 'annotation matrix' for each LHS category in the Penn-II CFG-ruleset which our algorithm uses to annotate nodes in the treebank. To illustrate the basic idea, a partial example matrix is given in Table 1 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 (quantifier/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 $\downarrow\in\uparrow$`headmod`. Regarding non-subcategorisable functions in the right context for NPs, an SBAR or VP (relative clauses) is annotated $\uparrow$`relmod`$=\downarrow$, a PP receives the annotation $\downarrow\in\uparrow$`adjn`, while

[2] 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.

LHS Category	Total Rule Types	No. Rules 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

Table 2: No. of most frequent rule types analysed to construct annotation matrices

nominal phrases separated by commas following the head are identified as appositions and are assigned the annotation ↑`app`=↓.

Note that we have adopted a relatively conservative approach to the assignment of f-structure annotations at this stage. For example, with respect to NPs, *all* PPs are considered to be adjuncts—no PP in an NP would receive the annotation ↑`obl`=↓ given our current implementation. In order to reduce errors to a bare minimum, we assign subcategorisable grammatical functions only where there is no doubt, e.g. an NP following a preposition in a PP is assigned ↑`obj`=↓; a VP following a verb in a VP constituent is assigned the annotation ↑`xcomp`=↓, ↑`subj`=↑`xcomp:subj`, and so on. At this stage, however, it should be stressed that we are aiming at producing a set of proto-f-structures; considerable refinement will be necessary when we publish our final set of f-structures.

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 such rule types mediates against all rules having some input into the construction of the annotation matrices. Instead, we select only the most frequently used rule types for each different rule LHS to populate the annotation matrices. To be precise, we analyse those rule types which together cover at least 85% of the *token* occurrences of all rules expanding any particular LHS category. Table 2 shows for a subset of rule LHSs how many rules *types* there are in total for each LHS, and how many most frequent rules we actually use to cover at least 85% of the token occurrences. For instance, while there are over 6500 distinct NP rules 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 the NP annotation matrix given in Table 1.

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. Indeed, in (Cahill *et al.*, 2002a), we design a number of probabilistic parsers where thresholding (cf. Krotov *et al.*, 1998) 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 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%.

In addition, despite being constructed on the basis of very few most frequent rule types, the annotation matrices constructed generalise to 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; 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. -TMP (temporal), -LOC (locative), etc.). Accordingly, our algorithm annotates not only the rules corresponding to the monadic categories, but also their functional equivalents. With respect to NP, for example, all daughters of NP-LOC, NP-TMP and so on are also annotated using the same annotation schema as shown in Table 1.

The only 'constituents' in the Penn-II treebank that remain uncovered in our work to date are FRAG(ment) and X (unknown constituents). 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.

2.2 Coordination Annotation Principles

In the Penn-II treebank, there are two types of coordinate structures, namely coordination of 'like' and 'unlike' constituents. Indeed, the latter set have their own 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 particular problems. Integrating a treatment of coordinate structures into the L/R principles described in the previous section unduly complicates principles and makes than harder to maintain and extend. Accordingly, we treat coordinate structures in a separate, subsequent component in our algorithm.

As before, annotation is predicated on our finding the local head. In cases where like constituents are coordinated, where just one CC is found, this is always the head and receives the annotation $\uparrow=\downarrow$. A number 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. $\downarrow\in\uparrow$`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 $\downarrow\in\uparrow$`adjn`.

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

1. Mark everything followed by a comma as a conjunct (i.e. $\downarrow\in\uparrow$`conj`).
2. Check for noun sequences to the left and right of the head. Annotate the rightmost noun with $\downarrow\in\uparrow$`conj`, and the others as adjuncts ($\downarrow\in\uparrow$`adjn`).
3. Mark left and right of CC with $\downarrow\in\uparrow$`conj` (unless otherwise annotated).
4. Mark DT to the left of CC with $\uparrow$`spec`$=\downarrow$ (unless otherwise annotated).
5. Mark SBAR and SBAR-TMP to the right of CC with $\uparrow$`comp`$=\downarrow$ (unless otherwise annotated).

2.3 Catch-All Annotation Principles

Despite the previous two steps, some categories on the RHS of the rules are not assigned any annotation. As a final clean-up operation, therefore, we mark all unannotated components bearing the Penn-II functional tags with the appropriate annotations: -SBJ receive the annotation ↑`subj`=↓, -PRD ↑= ↓ and -CLR ↑`obl`=↓ for the first such item, and ↓∈↑`adjn` for subsequent cases, and so on.

3 Running the Automatic Annotation Algorithm

Our automatic annotation algorithm is implemented in terms of a Java program. Annotation of the complete WSJ section of the Penn-II treebank takes less than 30 minutes on a Pentium IV PC. Once annotated, for each tree we collect the feature structure annotations and feed them into a simple constraint solver implemented in Prolog. Our constraint solver can cope with equality constraints, disjunction and simple set-valued feature constraints. In previous work (van Genabith *et al.*, 1999a,b; Sadler *et al.*, 2000; van Genabith *et al.*, 2001; Frank *et al.*, 2002), we permitted disjunctive constraints, but in our current work to develop proto-f-structures, this has so far been unnecessary. Accordingly, therefore, for each tree in the treebank we either get a single f-structure, or in the case of partially annotated trees, a number of unconnected f-structure fragments, or in the case of feature structure clashes, no f-structure.

We have already pointed out that annotation matrices for FRAG and X constituents have not been constructed. In addition, we currently ignore movement, dislocation and long distance control phenomena marked in the Penn-II trees with coindexation indices. Our proto-f-structures therefore omit certain re-entrancies which would be captured in a more fine-grained approach. These are all areas for future work. One potential solution to modelling the 'movement' phenomena just described would be to use LFG functional uncertainty constraints (regular expression-based constraints over paths in f-structure). This is currently beyond the capabilities of our constraint solver, but other avenues using the Xerox Linguistic Environment (XLE)[3] have presented themselves. One further area of improvement which the XLE would solve would be to express subsumption constraints in our annotations. This would enable us to percolate subjects into coordinate VP structures, among other things.

In order to illustrate our methodology, we provide in Figure 1 the first of our 105 testset sentences from section 23 of the WSJ part of the Penn-II treebank, as annotated by our automatic annotation algorithm. We show the string *The investment community, for one, has been anticipating a speedy resolution*, the Penn-II parse tree, and the resulting f-structure derived automatically via our method. Note the percolation of subjects into auxiliary strings (XCOMPs in LFG) in this instance owing to the ↑`subj`=↓`subj` annotations on the VP daughters in the VP rules in (2).

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 transcribed manually for a testset of 105 trees randomly selected from section 23 of the WSJ section of the Penn-II treebank.[4]

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

[4] The full set of f-structures are available at: `http://www.computing.dcu.ie/~away/Treebank/treebank.html`.

The investment community, for one, has been anticipating a speedy resolution.

```
(S (NP-SBJ (DT The)
           (NN investment)
           (NN community))
   (, ,)
   (PP (IN for)
       (NP (CD one)))
   (, ,)
   (VP (VBZ has)
       (VP (VBN been)
           (VP (VBG anticipating)
               (NP (DT a)
                   (JJ speedy)
                   (NN resolution)))))
   (. .))
```

```
subj : spec : det : pred : The
       headmod : 1 : num : sing
                     pers : 3
                     pred : investment
       num : sing
       pers : 3
       pred : community
adjunct : 1 : obj : pred : one
              pred : for
xcomp : subj : spec : det : pred : The
               headmod : 1 : num : sing
                             pers : 3
                             pred : investment
               num : sing
               pers : 3
               pred : community
        xcomp : subj : spec : det : pred : The
                       headmod : 1 : num : sing
                                     pers : 3
                                     pred : investment
                       num : sing
                       pers : 3
                       pred : community
                obj : spec : det : pred : a
                      adjunct : 2 : pred : speedy
                      pred : resolution
                      num : sing
                      pers : 3
                participle : pres
                pred : anticipating
        pred : been
        tense : past
pers : 3
num : sing
pred : has
tense : pres
```

Figure 1: F-structure automatically generated for the first of our 105 Penn-II test sentences

4 Automatic Evaluation

In this section, we present a number of automatic evaluation methodologies for assessing the effectiveness of the techniques we have developed. They include 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 unified f-structure being produced.

4.1.1 Coverage

The methodology behind our automatic annotation procedure is to recursively traverse the Penn-II trees and apply annotations at all appropriate nodes. As an example, applying our automatic annotation to the tree-structure in Figure 1 produces the annotated tree-structure in Figure 2.

```
(S
   (NP-SBJ[up-subj=down]
      (DT[up-spec:det=down] The[up-pred='The'])
      (NN[down-elem=up:headmod] investment[up-pred='investment',up-num=sing,up-pers=3])
      (NN[up=down] community[up-pred='community',up-num=sing,up-pers=3]))
   (, ,)
   (PP[down-elem=up:adjunct]
      (IN[up=down] for[up-pred='for'])
      (NP[up-obj=down]
         (CD[up=down] one[up-pred='one'])))
   (, ,)
   (VP[up=down]
      (VBZ[up=down] has[up-pred='has',up-tense=pres,up-pers=3,up-num=sing])
      (VP[up-xcomp=down,up-subj=down:subj]
         (VBN[up=down] been[up-pred='been',up-tense=past])
         (VP[up-xcomp=down,up-subj=down:subj]
            (VBG[up=down] anticipating[up-pred='anticipating',up-participle=pres])
            (NP[up-obj=down]
               (DT[up-spec:det=down] a[up-pred='a'])
               (JJ[down-elem=up:adjunct] speedy[up-pred='speedy'])
               (NN[up=down] resolution[up-pred='resolution',up-num=sing,up-pers=3])))))
   (. .))
```

Figure 2: Annotated tree-structure for the first of our 105 Penn-II test sentences

In order to evaluate the coverage of our method, we then extract the CFG grammar rule types with the appropriate annotations. From the annotated tree-structure in Figure 2, the

following eight rules would be extracted:

(2)

S $\longrightarrow$	NP-SBJ	PP	VP	
	$\uparrow$subj=$\downarrow$	$\downarrow\in\uparrow$adjn	$\uparrow=\downarrow$	
NP-SBJ $\longrightarrow$	DT	NN	NN	
	$\uparrow$spec=$\downarrow$	$\downarrow\in\uparrow$headmod	$\uparrow=\downarrow$	
PP $\longrightarrow$	IN	NP		
	$\uparrow=\downarrow$	$\uparrow$obj=$\downarrow$		
NP $\longrightarrow$	CD			
	$\uparrow=\downarrow$			
VP $\longrightarrow$	VBZ	VP		
	$\uparrow=\downarrow$	$\uparrow$xcomp=$\downarrow$ $\uparrow$subj=$\downarrow$subj		
VP $\longrightarrow$	VBN	VP		
	$\uparrow=\downarrow$	$\uparrow$xcomp=$\downarrow$ $\uparrow$subj=$\downarrow$subj		
VP $\longrightarrow$	VBG	NP		
	$\uparrow=\downarrow$	$\uparrow$obj=$\downarrow$		
NP $\longrightarrow$	DT	JJ	NN	
	$\uparrow$spec=$\downarrow$	$\downarrow\in\uparrow$adjn	$\uparrow=\downarrow$	

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 3 for the major rule types. Note that with two exceptions, our automatic method annotates over 93% of the RHS constituents for these types, and with most considerably higher. 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 85% 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 90% of the time is QP, namely quantified phrases. Almost all unannotated nodes share the same tag, CD, representing numerals. As a matter of ongoing work, we continue to analyse these cases in order to improve coverage.

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 4 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. However, in examining the circumstances under which these categories failed to receive any annotation, we observed that quite often, an NP under NP would not receive an annotation, nor would an S under S, and so on. That is, we suspected that most of this failure to provide an annotation was due to instances in coordinate structures. As can be seen from the rightmost

LHS Category	No. RHS Constituents	No. RHS annotated	Percentage annotated
ADJP	1641	1556	94.82
ADVP	605	591	97.68
NP	30735	28801	93.70
PP	1073	1011	94.22
PRN	1373	1177	85.72
QP	1555	1388	89.26
S	14817	14235	96.07
SBAR	409	386	94.37
SBARQ	256	254	99.21
SINV	1644	1583	96.28
SQ	650	605	93.07
UCP	649	615	94.76
VP	40824	40489	99.17
WHNP	367	349	95.09

Table 3: Percentage of annotated constituents in rule type RHSs

columns in Table 4, this proved to be the case. Omitting categories CONJP, CC and UCP saw major improvements in coverage for all categories in Table 4. These results confirm that writing linguistic generalisations for non-coordinate structures appears to be simpler. In addition, however, they also show that despite distinguishing between annotation principles for non-coordinate structures and cases of coordination, more work needs to be done for the latter set.

While the results in Table 3 and Table 4 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 quite often, compared to a rule which occurs only rarely. Given the scale of the task, it is currently impossible for us to state categorically how successful we are here, but the qualitative experiments described in section 4.3 provide some insights in this regard.

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 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 *Mary* and *a cake* are assigned the equation $\uparrow$`obj`=$\downarrow$ given the string *John baked Mary a cake*, the constraint resolution algorithm will not be able to unify the paths `obj:pred:cake` and `obj:pred:mary`. Consequently, for this simple example, it is imperative that (as is indeed the case in our annotation algorithm) *a cake* be assigned the annotation $\uparrow$`obj2`=$\downarrow$, which undoes the unification clash and permits the construction of an f-structure for this string.

	All Rules		*Without Conjunctions*	
Category	No. RHS Constituents	No. RHS Unannotated	No. RHS Constituents	No. RHS Unannotated
CD	44936	110	44068	1
DT	101135	240	99349	37
IN	121760	35	121213	1
JJ	75257	376	72234	19
NN	163923	772	157887	10
NNP	114041	305	107031	40
NNS	73955	461	69972	1
NP	277909	481	257704	33
PP	64128	212	62528	25
PRP	21357	2	21343	1
RB	38125	152	36958	12
S	58890	463	52052	14
SBAR	25703	42	24847	4
TO	27443	5	27424	1
VB	32545	33	32094	4
VBD	37468	17	37329	8
VBN	24854	32	24537	1
VBZ	26423	33	26254	20
VP	179992	226	168166	8

Table 4: Percentage of unannotated constituents in rule token RHSs

The general results obtained with our automatic annotation algorithm are summarised in Table 5. Note again that the Penn-II treebank consists of 49,167 trees, but the results in Table 5 exclude 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 48,440 trees.

We generate 51,501 f-structure fragments altogether, with an average of 1.067 fragments per tree, excluding the 166 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 in the interim to 96.48%. (Cahill *et al.*, *op cit.*) reported 2701 sentences receiving 0 f-structures, which has been improved sixteenfold in the figures presented here. Furthermore, over 10% of the trees previously were assigned two f-structure fragments, compared to just 0.799% here. Finally, the previously reported figures contained trees which received up to 11 f-structure fragments, whereas the largest fragmentation has currently been reduced to 8 fragments for 6 trees.

The results given in Table 5 provide us with our first handle on the fatal annotation errors made by our automatic procedure. We can identify through rigorous linguistic inspection 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 distinction and so on. In this respect, fragmentation is predominantly just another measure of coverage.

No. f-structure (fragments)	No. sentences	Percentage
0	166	0.343
1	46802	96.648
2	387	0.799
3	503	1.039
4	465	0.960
5	70	0.145
6	17	0.035
7	8	0.017
8	6	0.012

Table 5: Automatic Annotation results: Coverage

4.2 Discussion

The quantitative experiments described above inform us as to the coverage obtained by our system in annotating the trees of the Penn-II treebank with f-structure information. For the major rule LHSs in the grammar extracted from the trees, we showed that for the major rule types, we cover over 85% of their constituents, and for all but two LHS categories (PRN and QP), over 93% of the constituents recieve an annotation.

In order to factor in the frequency of rule type use in the corpus, we also calculated the number of constituents (tokens, not types) which occur on rule RHSs which do not receive an annotation. For all RHS categories occurring more than 20,000 times in the treebank, we observed that annotations are obtained more than 99% of the time by our method. In addition, we noted that omitting cases of coordination brings about major improvements in coverage for all categories.

These quantitative evaluations provide us with some preliminary insight into the success rate of our method. They do not, however, tell us how many of the annotations assigned are correct. The first step towards identifying the errors made by our automatic annotation procedure was to show the level of fragmentation and the number of feature-value clashes involved in this process. Regarding this latter point, if two (or more) f-structure annotations are inconsistent, no f-structure can be built for the string in question. So far, just 166 out of the 48,440 trees (ignoring the 727 trees containing FRAG and X constituents) currently receive no f-structure. In ongoing research, we are confident that this number can be made smaller still. Regarding fragmentation, we observed that for just 1456 of the 48,274 trees (3.02%) which received some f-structure, more than one f-structure fragment was produced. Currently, the highest number of fragments for any one tree is eight; again, this number is coming down all the time. In sum, we take heart from the fact that for the trees which receive some f-structure fragment(s), 48157 (99.76%) of them are assigned between one and four fragments only.

Nevertheless, while these results are encouraging, they show only that annotations are assigned which cause no clash in features in the constraint solver. Quantitative evaluation gives us no clear picture as to whether incorrect grammatical functions have been assigned, nor can we distinguish between oblique arguments and adjuncts, for example. In the next section, therefore, we describe a suite of experiments designed to evaluate the quality of the f-structures we construct with respect to a set of manually created annotations.

4.3 Qualitative Evaluation

This section describes a set of experiments designed to provide insights into the quality of the f-structures constructed by our 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 Penn-II trees selected at random from section 23 of the WSJ section of the treebank. The average string length was 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 the flat set descriptions of f-structures generated.

4.3.1 Evaluation with `evalb`

In this section, we describe experiments using `evalb` on the annotated Penn-II trees in our testset. `evalb`[5] 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 order to use `evalb`, we treat a string consisting of a CFG category followed by one or more f-structure annotations (e.g. `VP[up-xcomp=down,up-subj=down-subj]`) as an atomic node identifier. The results are presented in Table 6.

No. sentences	105
Bracketing Recall	89.62%
Bracketing Precision	89.62%
Complete match	23.81%
Tagging accuracy	95.35%

Table 6: Evaluation of automatically annotated trees using `evalb`

That is, for all 105 test strings, we obtain 89.62% Precision and Recall (P&R), with 25 out of the 105 strings (23.81%) being completely correct. For strings of less than 40 words, very similar results for P&R are achieved (89.52%), with the automatic annotations in 25 of the 99 strings matching the manual annotations completely. 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. 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 it is an extremely severe evaluation metric, in that the set of equations produced automatically must be identical to the set of manually created annotations. That is, if two sets of equations do not appear in canonical form, then `evalb` would state that the two sets are not identical, even if the members of those sets are the same, but appear in different orders. For `evalb`, therefore, the sets {2,1,3} and {1,3,2}

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

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 mistakes by `evalb`.

We conjecture that some good results may not be recognised by `evalb`. Given this, in the next section we calculate P&R *directly* on descriptions of f-structures.

4.3.2 Precision and Recall Evaluation on F-Structure Descriptions

In order to calculate P&R directly on descriptions of f-structures, we use a methodology similar to the one reported in (Riezler *et al.*, 2002) for f-structures and (Liakata and Pulman, 2002) for Quasi Logical Forms. Each f-structure is represented as a set of terms of the form: `relation(predicate,argument)`. Our notation differs to theirs in that our terms contain only node numbers, except for the case where the node is a terminal f-structure node, when we state the values of the `pred, number, person` etc. features. Let us assume the simple f-structure in (3):

(3)
$$1:\begin{bmatrix} \text{SUBJ} & 2:\begin{bmatrix} \text{PRED} & \text{`John'} \\ \text{NUM} & \text{SG} \\ \text{PERS} & \text{3rd} \end{bmatrix} \\ \text{PRED} & \text{`love}\langle(\uparrow\text{SUBJ})(\uparrow\text{OBJ})\rangle\text{'} \\ \text{OBJ} & 3:\begin{bmatrix} \text{PRED} & \text{`Mary'} \\ \text{NUM} & \text{SG} \\ \text{PERS} & \text{3rd} \end{bmatrix} \end{bmatrix}$$

Given this f-structure, we automatically 'translate' it into a flat set of terms corresponding to f-structure descriptions, as in (4):

(4)
```
subj(1,2), pred(2,john), num(2,sg), pers(2,3rd),
pred(1,love),
obj(1,3), pred(2,mary), num(3,sg), pers(3,3rd)
```

We also calculate results for the preds-only f-structures encoding basic predicate-argument-modifier relations. This is the same set of f-structures as before, except that this time the lexical attributes such as person and number are stripped out. We differ from Riezler *et al.* (*op cit.*) in one final respect: while they also map their set of f-structures to the coarser set of grammatical relations contained in the evaluation corpus of (Carroll *et al.*, 1999), we do not, although this remains an area for further research. Our approach differs from Liakata and Pulman's (2002) in that precison and recall violations in our metric always carry the same weight.

	All annotations	Preds-only annotations
Precision	94.915%	93.862%
Recall	93.888%	88.739%

Table 7: Precision and Recall on descriptions of f-structures

In order to calculate P&R from sets of terms such as those in (4), we simply consume matching terms from the set of terms given by each f-structure from the gold standard to the set of

terms given by the corresponding f-structure which is generated automatically. The results are presented in Table 7. For the full set of 8715 annotations in the gold standard f-structures, we obtain Precision and Recall of over 93.8%. There are 8400 equations in the automatically generated set. For the preds-only set of equations (i.e. with the lexical annotations excluded), Precision remains around 94%, while Recall dips by about 5%. The number of preds-only annotations in the manually annotated trees is 3150, exactly the same number as in the automatically annotated trees. 27 out of the 105 trees receive a complete and correct set of annotations. We consider these results to be very promising. They show that our automatic annotation methodology is more often (slightly more) partial than incorrect. Furthermore, these results confirm our hypothesis that some correct automatic annotations are discounted by `evalb`.

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 methods and experiments. Quantitative evaluation does not involve a gold-standard, while qualitative evaluation does.

Our quantitative methods 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 from the annotations, number of sentences that receive two unconnected f-structure fragments and so on, as well as average f-structure fragmentation for a tree) co-vary with the measures reported above and are indicative of the coverage of our automatic f-structure annotation procedure. In addition, we also measure the number of trees associated with unresolvable f-structure descriptions (that is, inconsistent or cyclical sets of f-descriptions that fail to generate an f-structure). In contrast to counting annotated versus unannotated nodes and f-structure fragmentation, measuring unresolvable f-descriptions gives a first indication of the quality (or rather: lack of quality) as opposed to mere coverage of the automatic annotation results.

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 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` certainly provide lower bounds. In order to provide a more fine-grained evaluation, our second qualitative evaluation variant involves the translation of generated f-structures into a flat set of terms describing test and reference f-structures. Precision and Recall are then computed on these term descriptions,

in a similar manner to (Riezler *et al.*, 2002) and (Liakata and Pulman, 2002).

To summarise the particular results obtained in our automatic f-structure annotation experiments to date, 46802 sentences (96.65% of the 48,440 trees without FRAG and X constituents) receive a complete f-structure, and 48157 (99.76%) trees are assigned between one and four fragments only. 166 trees are not associated with any f-structure. Currently, there are an average of 1.067 fragments per tree. Using `evalb`, we obtained 89.62% Precision and Recall, with 25 out of the 105 strings receiving a completely correct set of annotations. We also calculated Precision and Recall directly on sets of term descriptions of f-structures. For the preds-only set of equations (i.e. with the lexical annotations excluded), we obtain a Precision of 93.862%, and Recall of 88.739%. 27 out of the 105 trees receive a complete and correct set of annotations. These results show that our automatic annotation methodology is more often partial than incorrect.

A number of avenues for further work present themselves:

- We continue to reduce further the amount of f-structure fragmentation by providing more complete annotation principles, especially for rules expressing coordination, for performance is currently adversely affected when such phenomena are encountered. Furthermore, we hope to provide a treatment for FRAG and X 'constituents' which are not currently treated in the research presented here.
- We intend to improve the order in which categories are evaluated for headedness in Magerman's (1994) scheme still further. One possibility we have in mind is to automatically learn the best order in which constituents are evaluated for headedness using probabilistic techniques.
- Our generated f-structures do not currently contain root forms for `pred` values. Instead, surface, inflected forms appear, but making the output from a morphological tool available to our lexical macros will enable `pred` values to appear in the f-structures produced by our automatic annotation algorithm.
- We do not currently avail of the Penn-II encoding of 'moved' elements to facilitate reentrancies in the generated f-structures. We hope to use functional uncertainty equations in the XLE to model such phenomena. Using the XLE would also allow us to express subsumption constraints in our annotations, thereby enabling the percolation of subjects into coordinate structures.
- We hope to subject the f-structures produced by our method to the Completeness and Coherence grammaticality checks of LFG. This entails obtaining a large lexicon with subcategorisation frames for verbs and other argument-bearing categories.
- We intend to evaluate the results of our automatic annotation against the reference corpora constructed by (Riezler *et al.*, 2002) and (Carroll *et al.*, 1999) to facilitate inter-system comparisons.

References

[1] Cahill, A., M. McCarthy, J. van Genabith and A. Way (2002a): 'Parsing Text with a PCFG derived from Penn-II with an Automatic F-Structure Annotation Procedure', in *Proceedings of the Seventh International Conference on Lexical-Functional Grammar*, Athens, Greece.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] Frank, A., L. Sadler, J. van Genabith and A. Way (2002): '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).

[7] 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.

[8] 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.

[9] 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.

[10] Liakata, M. and S. Pulman (2002): 'From trees to predicate-argument structures', Unpublished working paper, Centre for Linguistics and Philology, University of Oxford, UK.

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

[12] 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.

[13] 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 of Computational Linguistics*, Philadelphia, PA.

[14] 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.

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

[16] 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.

[17] 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.

[18] 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.