EVALUATING AUTOMATICALLY ACQUIRED F-STRUCTURES AGAINST PROPBANK

Michael Burke, Aoife Cahill, Josef van Genabith and Andy Way

Proceedings of the LFG05 Conference
University of Bergen
Miriam Butt and Tracy Holloway King (Editors)
2005
CSLI Publications
http://csli-publications.stanford.edu/

— Abstract —

An automatic method for annotating the Penn-II Treebank (Marcus et al., 1994) with high-level Lexical Functional Grammar (Kaplan and Bresnan, 1982; Bresnan, 2001; Dalrymple, 2001) f-structure representations is presented by Burke et al. (2004b). The annotation algorithm is the basis for the automatic acquisition of wide-coverage and robust probabilistic approximations of LFG grammars (Cahill et al., 2004) and for the induction of subcategorisation frames (O'Donovan et al., 2004; O'Donovan et al., 2005). Annotation quality is, therefore, extremely important and to date has been measured against the DCU 105 and the PARC 700 Dependency Bank (King et al., 2003). The annotation algorithm achieves f-scores of 96.73% for complete f-structures and 94.28% for preds-only f-structures against the DCU 105 and 87.07% against the PARC 700 using the feature set of Kaplan et al. (2004). Burke et al. (2004a) provides detailed analysis of these results. This paper presents an evaluation of the annotation algorithm against PropBank (Kingsbury and Palmer, 2002). PropBank identifies the semantic arguments of each predicate in the Penn-II treebank and annotates their semantic roles. As PropBank was developed independently of any grammar formalism it provides a platform for making more meaningful comparisons between parsing technologies than was previously possible. PropBank also allows a much larger scale evaluation than the smaller DCU 105 and PARC 700 gold standards. In order to perform the evaluation, first, we automatically converted the PropBank annotations into a dependency format. Second, we developed conversion software to produce PropBank-style semantic annotations in dependency format from the f-structures automatically acquired by the annotation algorithm from Penn-II. The evaluation was performed using the evaluation software of Crouch et al. (2002) and Riezler et al. (2002). Using the Penn-II Wall Street Journal Section 24 as the development set, currently we achieve an f-score of 76.58% against PropBank for the Section 23 test set.

1 Introduction

Recent research (Burke et al., 2004b) has presented a method for automatically annotating the Penn-II treebank (Marcus et al., 1994) with Lexical Functional Grammar (LFG) (Kaplan and Bresnan, 1982; Bresnan, 2001; Dalrymple, 2001) f-structure representations. The automatic f-structure annotation algorithm is a central component in a larger project which automatically acquires wide-coverage and robust probabilistic approximations of LFG grammars (Cahill et al., 2004) and induces LFG lexical resources (O'Donovan et al., 2004; O'Donovan et al., 2005). Annotation quality is, therefore, extremely important and to date has been evaluated, using the methodology and software presented in (Crouch et al., 2002) and (Riezler et al., 2002), against the DCU 105[1] and the PARC 700 Dependency Bank (King et al., 2003). The annotation algorithm achieves f-scores of 96.73% for complete f-structures and 94.28% for preds-only[2] f-structures against the DCU 105 and 87.07% against the PARC 700 using the feature set of Kaplan et al. (2004). Burke et al. (2004a) provides further analysis of these results and describes the conversion software used in the PARC 700 evaluation process.

In this paper we present an evaluation of the f-structures produced by the annotation algorithm for Penn-II treebank trees against PropBank (Kingsbury and Palmer, 2002). PropBank was developed independently of any grammar formalism and provides a platform for making more meaningful comparisons between parsing technologies than was previously possible. PropBank has been used for the evaluation of CCG

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

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

(Gildea and Hockenmaier, 2003) and HPSG (Miyao and Tsujii, 2004) parsers. The methodology presented in this paper will allow the parsing technology of Cahill et al. (2004) to eventually be evaluated against PropBank and for direct comparisons with CCG, HPSG and other parsers to be made. Whereas previous evaluations of the annotation algorithm have been against syntax-based gold standards (DCU 105 and PARC 700), evaluating against PropBank provides a semantic evaluation of the automatically acquired f-structures. Using PropBank also allows a much larger scale evaluation than was previously possible. The quality of the f-structure annotation algorithm can eventually be evaluated against PropBank data for the *entire* Penn-II treebank.

PropBank adds semantic information to the syntax trees of Penn-II, identifying predicates and their semantic arguments. To give a simple example, for the sentence *Both companies rejected the offers*, PropBank identifies *rejected* as the predicate with *both companies* as ARG0 and *the offers* as ARG1. Figure 1 provides the f-structure produced by the annotation algorithm for the example sentence, (a subset of the) triples extracted from that f-structure and the corresponding PropBank triples. A simple mapping of SUBJ to ARG0 and OBJ to ARG1 is sufficient to obtain the semantic annotations provided by PropBank in this example, but clearly a more elaborate mapping is required to extract PropBank-style semantic annotations from more complex automatically f-structure-annotated Penn-II trees.

$$\begin{bmatrix} \text{SUBJ} & \begin{bmatrix} \text{SPEC} & \begin{bmatrix} \text{QUANT} & \begin{bmatrix} \text{PRED} & \text{both} \end{bmatrix} \end{bmatrix} \\ \text{PRED} & \text{company} \\ \text{NUM} & \text{pl} \end{bmatrix} \\ \text{PRED} & \text{reject} \\ \text{TENSE} & \text{past} \\ \text{OBJ} & \begin{bmatrix} \text{SPEC} & \begin{bmatrix} \text{DET} & \begin{bmatrix} \text{PRED} & \text{the} \end{bmatrix} \end{bmatrix} \\ \text{PRED} & \text{offer} \\ \text{NUM} & \text{pl} \end{bmatrix} \end{bmatrix}$$

SUBJ(reject, company) ARG0(reject, company)
OBJ(reject, offer) ARG1(reject, offer)

Figure 1: F-structure produced by annotation algorithm for *Both companies rejected the offers* with some extracted LFG triples and the expected PropBank triples

Section 2 introduces the automatic f-structure annotation algorithm. Section 3 provides an overview of PropBank and the process of converting the PropBank semantic annotations into a dependency format. Section 4 describes the conversion software required to systematically convert the triples extracted from the automatically generated f-structures for evaluation against PropBank. Section 5 presents and analyses the results of the evaluation process. Using the Penn-II Wall Street Journal Section 24 as the development set, currently we achieve an f-score of 76.58% against PropBank for the Section 23 test set. Section 6 summarises and provides possibilities for future work.

2 Automatic F-Structure Annotation Algorithm

The automatic f-structure annotation algorithm (Burke et al., 2004b; Cahill et al., 2004; O'Donovan et al., 2004; O'Donovan et al., 2005) is modular (Figure 2). The first module, *Left-Right Context Rules*, head-lexicalises the treebank using a modified version of Magerman's (1994) scheme. This process creates a bi-

partition of each local subtree, with nodes lying in either the left or right context of the head. An annotation matrix is manually constructed for each parent category in the treebank by analysing the most frequent CFG rule types that together give at least 85% coverage of rule tokens for that parent category in the treebank. For example, only the most frequent 102 NP rule types were analysed to produce the NP annotation matrix which generalises to provide default annotations for the complete set of 6,595 NP rule types in the treebank. Default annotations are read from these matrices by the annotation algorithm to annotate nodes in the left and right context of each subtree.

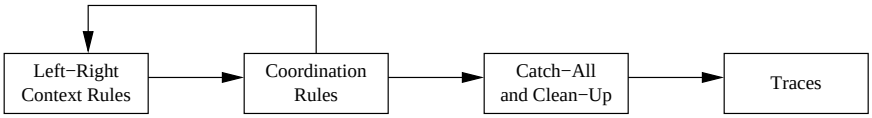

Figure 2: Annotation Algorithm modules

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

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

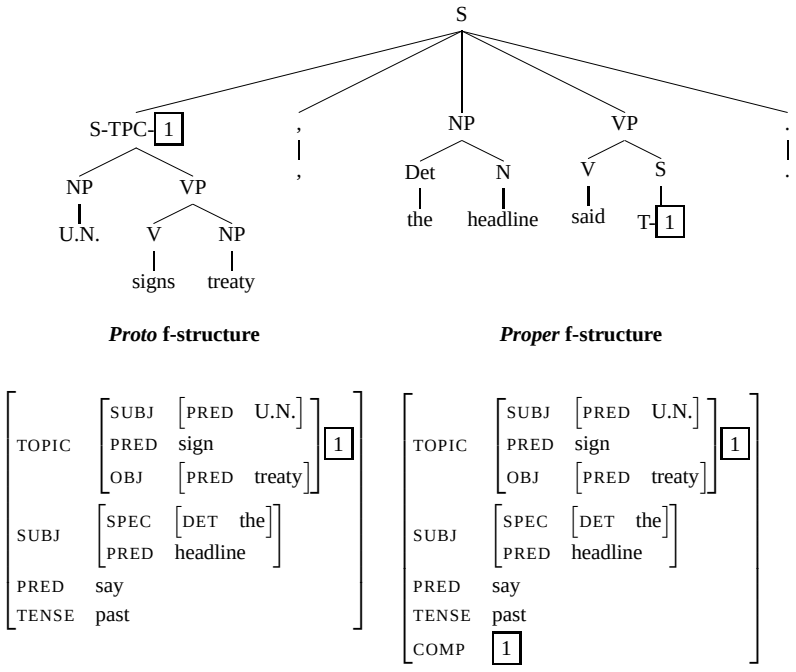

Figure 3: Penn-II style tree with LDD trace and corresponding re-entrancy in proper f-structure

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

The annotation algorithm achieves near complete coverage for the WSJ section of Penn-II with 99.82% of the 48K sentences receiving a single connected and covering f-structure. Table 1 provides a quantitative evaluation of the f-structures produced by the annotation algorithm. Feature clashes in the annotation of 85 trees result in no f-structure being produced for those sentences. Nodes left unannotated by the annotation algorithm in two trees caused two separate f-structure fragments for both sentences.

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

Table 1: Quantitative Evaluation

While achieving such wide coverage is important, the annotation quality must be of a high standard, particularly as the annotation algorithm plays a vital role in the generation of wide-coverage, probabilistic LFG parsing technology (Cahill et al., 2004) and lexical resources (O'Donovan et al., 2004; O'Donovan et al., 2005). Annotation quality has been measured in terms of precision, recall and f-score[3] against the DCU 105 and PARC 700 Dependency Bank using the evaluation methodology and software presented in (Crouch et al., 2002) and (Riezler et al., 2002). The DCU 105 is a set of gold standard f-structures for 105 randomly selected sentences from Section 23 of the WSJ section of Penn-II. To create the gold standard f-structures the Penn-II trees were first automatically annotated and the annotations were then manually corrected and extended. The PARC 700 consists of dependency structures for 700 randomly selected sentences from Section 23 of the WSJ section of Penn-II. These sentences were automatically parsed by a hand-coded, deep LFG grammar of English using the XLE system (Maxwell and Kaplan, 1993). In cases where multiple parses were generated, the best parse was manually chosen. The f-structures of the best parses were then automatically converted to dependency format (triples) and manually extended and corrected by two independent reviewers.

The f-structure annotation algorithm currently achieves an f-score of 96.73% for complete f-structures and 94.28% for preds-only f-structures against the DCU 105 (Table 2). Burke et al. (2004a) presents conversion software developed to overcome some of the systematic differences in linguistic analysis, feature geometry and nomenclature between the automatically acquired f-structures and the PARC 700 dependency structures. The f-structures automatically acquired by the annotation algorithm and mapped by conversion software achieves an f-score of 87.07% for the feature set of Kaplan et al. (2004) against the PARC 700. Burke et al. (2004a) provides a detailed analysis of the evaluation process and the results.

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

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

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

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

	DCU 105		PARC 700
	All grammatical functions	*Preds only*	*Feature set of Kaplan et al. (2004)*
Precision	96.77	94.32	87.95
Recall	96.69	94.24	86.21
F-score	96.73	94.28	87.07

Table 2: Annotation quality evaluated against DCU 105 and PARC 700

3 PropBank

3.1 Overview

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

3.2 Semantic Frames

PropBank assigns a set of semantic frames for every verb in Penn-II. Each semantic frame provides a definition for the semantic role labels relevant to that particular instance of the verb. Table 3 provides the three semantic frames for the predicate *yield*. The first semantic frame for *yield* defines the semantic role labels for the meaning *to result in*: ARG0 is the "thing yielding" and ARG1 is the "thing yielded".

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

Table 3: PropBank semantic frame set for the predicate *yield*

An example sentence, annotated with semantic role labels, for this semantic frame is: [$_{ARG0}$ *A single acre of grapes*] *yielded* [$_{ARG1}$ *a mere 75 cases*] [$_{ARGM-TMP}$ *in 1987*]. The semantic role label annotations indicate that in this example sentence *a single acre of grapes* is the "thing yielding" while *a mere 75 cases* is the "thing yielded". The phrase *in 1987* is annotated as an optional modifier ARGM-TMP. Annotated example sentences for the three semantic frames for *yield* are:

(1) Frame 1: "To result in"
[$_{ARG0}$ *A single acre of grapes*] *yielded* [$_{ARG1}$ *a mere 75 cases*] [$_{ARGM-TMP}$ *in 1987*].

Frame 2: "To give way"
[$_{ARG0}$ *John*] *yielded* [$_{ARG1}$ *the right-of-way*] *to* [$_{ARG2}$ *the Mack truck*].

Frame 3: "To give a dividend"

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

3.3 Semantic Argument Annotation

PropBank provides a file of semantic annotations for Penn-II in the following format. The annotations first identify the relevant Penn-II tree by providing the Penn-II file name and line number, e.g. line 12 in `wsj/00/wsj_0004.mrg` identifies the tree shown in Figure 4 for the sentence *The top money funds are currently yielding well over 9%*. The annotation then identifies the verb being annotated and the relevant semantic frame for this occurrence of the verb, which in this case is "yield.01", the frame "to result in" as outlined in Table 3. The semantic arguments are then listed in the form `terminal number:node height-semantic role`. Terminals are numbered from left to right starting with zero.

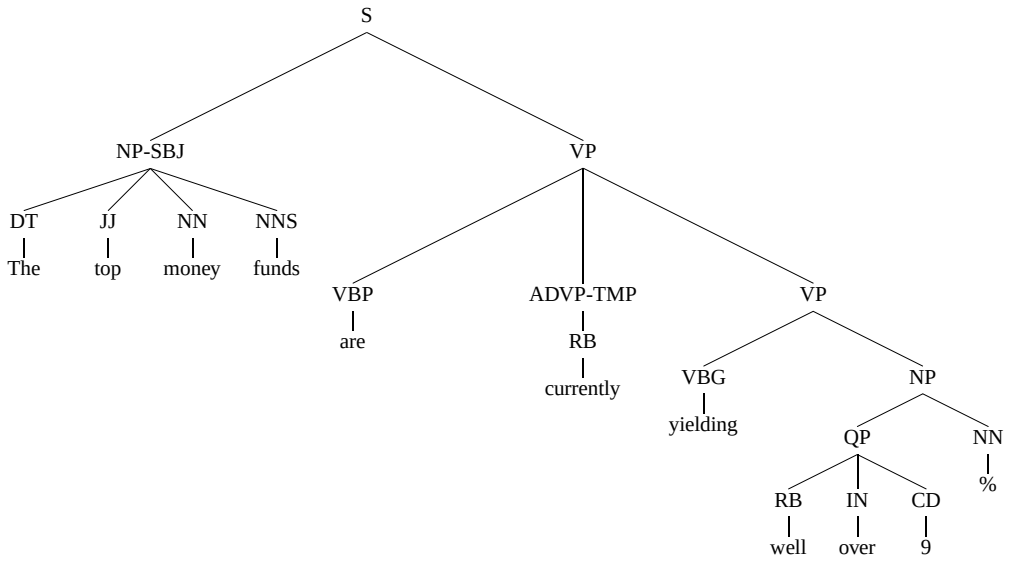

Figure 4: Penn-II tree for the sentence *The top money funds are currently yielding well over 9%*

PropBank annotates the semantic arguments of the verb *yield* in the example sentence as: `0:1-ARG0`, `5:1-ARGM-TMP` and `7:2-ARG1`. The annotation `0:1-ARG0` indicates that the node NP-SBJ which governs the noun phrase *The top money funds* is a semantic argument of the verb *yield* with the semantic role ARG0. This node is found in the tree of Figure 4 by starting with the POS tag of terminal 0 in the tree, i.e. DT, and traversing one node, i.e. `0:1`, upwards from that node. Similarly, the argument paths `5:1-ARGM-TMP` and `7:2-ARG1` indicate that the semantic roles ARGM-TMP and ARG1 are played by the nodes ADVP-TMP and NP governing *currently* and *well over 9%*, respectively.

3.4 Creating Gold Standard PropBank Dependencies

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

The PropBank semantic annotations were automatically converted into triples of the form: SEMANTIC ROLE(verb, argument). The Penn-II nodes representing the semantic roles were identified by automatically traversing the argument paths as outlined in Section 3.3. For each node, the head word of the subtree governed by that node was identified using the head-lexicalisation rules of the annotation algorithm, which are a modified version of the rule set of Magerman (1994). The verbs and head words were lemmatised with the XLE lemmatiser also used by the annotation algorithm. The PropBank semantic roles were conflated, removing the different subtypes of ARGM modifiers (Table 4), to the subset: ARG0, ARG1, ARG2, ARG3, ARG4, ARG5 and ARGM.

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

Table 4: PropBank ARGM subtypes

To create PropBank triples for the sentence *The top money funds are currently yielding well over 9%*, the head words of the nodes NP-SBJ, ADVP-TMP and NP (Figure 4) were automatically identified as *funds*, *currently* and *%*, respectively. After lemmatising all words and conflating the semantic roles, the triples ARG0(yield, fund), ARG1(yield, percent) and ARGM(yield, currently) were created. This process was applied to all trees in the treebank.

4 Converting F-Structures into Semantic Roles

We developed conversion software to produce PropBank-style semantic role annotations in the dependency format introduced in Section 3.4 from the f-structures automatically acquired by the annotation algorithm from Penn-II trees. Triples are extracted from the f-structures generated by the annotation algorithm and then post-processed by the conversion software to produce semantic role annotations. The conversion procedure employs default mappings from LFG feature names to PropBank semantic roles before handling the following phenomena which require more complex mappings:

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

4.1 Default mappings

Default mappings are used to map LFG feature names to PropBank semantic role annotations. Table 5 lists these mappings for active verbs. Passive voice is identified by the annotation algorithm which results in

PASSIVE triples being extracted from the automatically generated f-structures. These triples are used by the conversion software to map the SUBJ triple of passive verbs to ARG1 (subjects of active verbs are mapped by default to ARG0), while oblique agents are mapped to ARG0.

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

Table 5: Default mappings from LFG feature names to PropBank semantic roles for active voice

The default mappings of Table 5 were applied to the automatically generated LFG triples for the active verb *yield* in the sentence *The top money funds are currently yielding well over 9%*. The resulting mapped PropBank-style triples and the original LFG triples are provided in Table 6. The default mappings are successful for this sentence, producing the required PropBank triples.

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

Table 6: Default mappings applied to automatically generated triples for *The top money funds are currently yielding well over 9%*

4.2 Particles

PropBank annotates phrasal verbs by grouping all nodes representing the phrasal verb and providing their semantic arguments as normal. When creating the gold standard PropBank triples we combined the grouped nodes to form a complex predicate for the phrasal verb. Phrasal verbs have a single triple for each semantic argument as with all other verbs. The third column of Table 7 provides the gold standard triples we extracted from PropBank for the phrasal verb *snap up* in the sentence *Earlier this year, Japanese investors snapped up a similar fund*. The first column provides the triples produced by the f-structure annotation algorithm for the same sentence, while the second column shows the PropBank-style triples produced by the application of the default mappings to the triples of column one.

An f-score of zero will be achieved for this sentence unless the complex predicate analysis is adopted for the phrasal verb. The Penn-II PRT (particle) tag is automatically annotated $\uparrow$PART=$\downarrow$, which results in the triple PART(snap, up) in this example. The conversion software uses the PART triple to create the complex predicate which replaces all occurrences of the bare verb in the mapped triples. This allows the desired gold standard triples to be produced by the mapping module.

4.3 Modal verbs

Modal verbs are represented in PropBank as optional arguments of the main verb. This treatment differs markedly from the cascading XCOMP analysis of the automatically generated f-structures and triples. Table

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

Table 7: Triples for *Earlier this year, Japanese investors snapped up a similar fund*

8 provides the automatically generated LFG triples and gold standard PropBank triples for the sentence *France can boast the lion's share of high-priced bottles*.

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

Table 8: Automatically generated LFG triples and gold standard PropBank triples for the sentence *France can boast the lion's share of high-priced bottles*.

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

4.4 Relative clauses

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

The automatically generated LFG triples indicate the presence of a relative clause through RELMOD and TOPICREL triples. The first column of Table 9 provides the automatically generated LFG triples for the fragment *The rights, which expire Nov. 21*. The RELMOD triple indicates that the noun (lemmatised as) *right* is modified by a relative clause which has *expire* as its main verb. The value *pro* represents the relative pronoun, whose surface form *which* is provided by the PRON_FORM triple. The TOPICREL triple links the *pro* value to the verb, indicating which pronoun is the fronted element of the relative clause. The SUBJ triple indicates that the relative pronoun is the subject of the relative clause.

Applying the default mappings to the triple SUBJ(expire, pro) would produce the incorrect PropBank triple ARG0(expire, pro). To overcome this problem, the conversion software first locates RELMOD triples. A RELMOD triple indicates that a noun is modified by a relative clause and provides the main verb of that clause. The TOPICREL triple associated with that main verb is then found. This triple provides the relative pronoun. Every occurrence of that relative pronoun, in all triples for that sentence, is replaced with the noun from the RELMOD triple (Table 9, second column). With this step in place, the default mappings (in this case from SUBJ to ARG0) are used to achieve the correct analysis.

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

Table 9: Automatically generated LFG triples and mapped PropBank triples for the fragment *The rights, which expire Nov. 21*

4.5 Mapping to ARG3, ARG4 and ARG5

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

4.6 Mappings for specific verbs

In many cases, even when the annotation algorithm generates a correct f-structure, there are no syntactic cues which can be used to produce the expected PropBank triples. The syntactic information available through the automatically generated f-structures and triples is insufficient for mapping the semantic roles of, for example, *climb*. Table 10 provides three sets of triples for the sentence *Net profit climbed to 30%*; (i) the triples produced by the f-structure annotation algorithm, (ii) the mapped triples produced using the conversion software described so far and (iii) the expected PropBank triples.

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

Table 10: Automatically generated LFG triples, mapped triples and PropBank triples for *Net profit climbed to 30%*

Applying the default mappings to the automatically generated triples produces ARG0 and ARG2 triples which should actually be ARG1 and ARG4, respectively. Having reviewed the development set, this is the normal expected behaviour for the verb *climb*. There is no further syntactic information available which could be used in a general mapping rule to produce the correct triples in this case, without degrading the overall performance of the conversion software for most verbs. Instead of introducing a general rule to deal with this case, a specific rule was introduced for the verb *climb* mapping SUBJ to ARG1, OBJ to ARG2, OBL to ARG3 for prepositional phrases headed by *from* and to ARG4 for PPs headed by *to*.

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

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

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

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

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

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

4.7 Filtering

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

The original version of the conversion software used the PropBank gold standard triples to overcome this problem. The gold standard triples were consulted to indicate which words were annotated as verbs in PropBank. The conversion software only produced PropBank-style triples for those lemmas. This procedure has since been removed and the conversion software no longer refers to the gold-standard triples, relying instead on Penn-II POS tagging and bracketing only.

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

The new procedure is preferred to the original consultation of the gold-standard PropBank triples to identify the annotated verbs as it is more methodologically sound and the results presented in Table 11 are derived with the new procedure. The new procedure achieves an f-score which is only 0.32% lower than the original procedure.

5 Evaluation

5.1 Results

The 2,416 trees in the Wall Street Journal Section 23 of Penn-II were annotated by the automatic f-structure annotation algorithm. Triples were extracted from the resulting f-structures and passed through the conversion software outlined in this paper. These triples were evaluated against the gold standard triples extracted

from the PropBank annotations for the same sentences using the methodology and software presented in (Crouch et al., 2002) and (Riezler et al., 2002). Without specific verb mappings an f-score of 73.42% is achieved, with precision and recall at 75.14% and 71.77%, respectively. Including specific verb mappings sees the overall f-score increase to 76.58% as a result of improved precision and recall scores of 78.44% and 74.81%. Table 11 provides the results in terms of precision, recall and f-score for each semantic role both without and with specific verb mappings.

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

Table 11: Annotation quality measured against PropBank for WSJ Section 23 of Penn-II, with and without mappings for specific verbs

5.2 Analysis

The overall f-score of 76.58% achieved by the annotation algorithm against PropBank for WSJ section 23 of Penn-II is lower than the results in previous evaluation experiments. Against the DCU 105 an f-score of 96.73% was achieved for complete f-structures and 94.28% for preds-only f-structures, while against the PARC 700 Dependency Bank using the feature set of Kaplan et al. (2004) the f-score was 87.07%. When evaluating the automatically generated f-structures — a syntax-based resource — against a gold standard of semantic relations, lower results should be expected than in experiments evaluating the f-structures against syntax-based gold standards, such as the DCU 105 and PARC 700.

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

A further, albeit less significant, explanation for the lower precision score for ARG0 is the failure of the annotation algorithm in some cases to identify a verb occurrence as having passive voice. In a syntax-based evaluation, this results in a missing PASSIVE triple which lowers recall slightly and leaves precision unchanged. The impact is not so significant as there are a far more triples per sentence than in the semantic evaluation. A missing passive marker in this semantic evaluation means that the SUBJ triple is mapped by default to ARG0 instead of ARG1. This results in lower precision for ARG0 and lower recall for ARG1, which is reflected in the scores for ARG1; precision 82%, recall 74%.

The best results are achieved for the semantic roles ARG0, ARG1 and ARGM with f-scores of 82%, 77% and 79%, respectively. As these semantic roles are the most frequently occurring, accounting for 90% of all gold standard triples, the development of mappings for these triples was the main focus of this research.

However, when the conversion software does produce the less frequently occurring ARG3 and ARG4 triples they are usually correct, as shown by the high precision scores of 90% and 93%, respectively. The low recall scores of 31% and 49% indicate that far too few ARG3 and ARG4 triples are being mapped.

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

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

6 Summary and Future Work

This paper has presented an evaluation of the automatic f-structure annotation algorithm (Burke et al., 2004b) against PropBank for the test set, WSJ section 23 of Penn-II. A dependency-format gold standard was extracted from PropBank to facilitate the evaluation process. The Penn-II trees were automatically annotated to produce LFG f-structures, from which triples were extracted. Conversion software was developed to map these triples to produce PropBank-style semantic annotations in dependency format. Section 24 of the WSJ section of Penn-II and PropBank was used as the development set for the mapping software. An f-score of 76.58% was achieved against PropBank for the test set. These results are lower than those achieved in previous syntax-based qualitative evaluation experiments. A detailed analysis of the results was provided.

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

Evaluating the parsing technology of Cahill et al. (2004) against PropBank is one obvious area for the development of this research. However, the mapping software will have to be improved significantly in order to provide a fair evaluation of this technology. An alternative approach to the mapping process may be required, as there are clear limitations to the improvements which can be made to the current mapping software.

The evaluation process provides useful feedback on the quality of the automatic f-structure annotations. Greater focus needs to be placed on the analysis of the evaluation results for the purpose of improving the annotation algorithm itself and not just the mapping software. The analysis of the results to date has shown that the identification of passive voice is one area which needs to be improved. Further research into this area will allow improvements to be made to the annotation algorithm and parsing technology.

References

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

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

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

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

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

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

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

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

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

King, T. H., R. Crouch, S. Riezler, M. Dalrymple, and R. Kaplan. 2003. The PARC700 Dependency Bank. In *Proceedings of the EACL03: 4th International Workshop on Linguistically Interpreted Corpora (LINC-03)*, pages 1–8, Budapest, Hungary.

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

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

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

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

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

O'Donovan, R., M. Burke, A. Cahill, J. van Genabith, and A. Way. 2004. Large-Scale Induction and Evaluation of Lexical Resources from the Penn-II Treebank. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 368–375, Barcelona, Spain.

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

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