C-STRUCTURES AND F-STRUCTURES FOR THE BRITISH NATIONAL CORPUS

Joachim Wagner[†], Djamé Seddah[‡], Jennifer Foster[†] and Josef van Genabith[†]

[†] National Centre for Language Technology (NCLT)
School of Computing
Dublin City University

[‡] Laboratoire Langages, Logiques, Informatique, Cognition (LaLIC)
Université Paris-Sorbonne

Proceedings of the LFG07 Conference

Miriam Butt and Tracy Holloway King (Editors)

2007

CSLI Publications

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

Abstract

We describe how the British National Corpus (BNC), a one hundred million word balanced corpus of British English, was parsed into Lexical Functional Grammar (LFG) c-structures and f-structures, using a treebank-based parsing architecture. The parsing architecture uses a state-of-the-art statistical parser and reranker trained on the Penn Treebank to produce context-free phrase structure trees, and an annotation algorithm to automatically annotate these trees into LFG f-structures. We describe the pre-processing steps which were taken to accommodate the differences between the Penn Treebank and the BNC. Some of the issues encountered in applying the parsing architecture on such a large scale are discussed. The process of annotating a gold standard set of 1,000 parse trees is described. We present evaluation results obtained by evaluating the c-structures produced by the statistical parser against the c-structure gold standard. We also present the results obtained by evaluating the f-structures produced by the annotation algorithm against an automatically constructed f-structure gold standard. The c-structures achieve an f-score of 83.7% and the f-structures an f-score of 91.2%.

1 Introduction

We describe a parsing experiment involving the British National Corpus (BNC) (Burnard, 2000) and a treebank-based parsing architecture for Lexical-Functional Grammar (LFG) (Kaplan and Bresnan, 1982) that reuses a lexicalised, history-based, generative, probabilistic parser (Charniak, 2000), a discriminative reranker (Charniak and Johnson, 2005) and an f-structure annotation algorithm (Cahill et al., 2002, 2004; Burke, 2006) in a pipeline process: the parser and reranker produce c-structures from which f-structures are produced via the annotation algorithm. We show how this technology can be scaled to parse the 100 million word BNC into both c-structure and f-structure representations. We investigate the effect on performance when moving from the domain upon which the LFG parsing resources have been trained (financial newspaper text) to the more varied text of the BNC.

The paper is structured as follows: In Section 2 we describe the LFG parsing architecture. In Section 3 we review related work. Section 4 provides a brief introduction to the BNC. The f-structures constructed in our pipeline processing architecture can only be as good as the c-structures produced by the parsers in the first phase of our parsing architecture. Section 5 presents the c-structure parsing experiments, including BNC preprocessing, parsing issues, gold standard tree construction and evaluation against the gold standard trees. Section 6 presents the f-structure annotation experiments including coverage, the automatic construction of an f-structure gold standard and evaluation against the f-structure gold standard. Section 7 summarises and outlines ongoing and future research.

[†]We are grateful to IRCSET (basic research grant SC/02/298 and postdoctoral fellowship P/04/232) and Science Foundation Ireland (04/IN/I527) for funding this research. We wish to acknowledge the SFI/HEA Irish Centre for High-End Computing for the provision of computational facilities and support. We would like to thank Aoife Cahill and Gzregorz Chrupała for their help.

2 Treebank-Based LFG Parsing

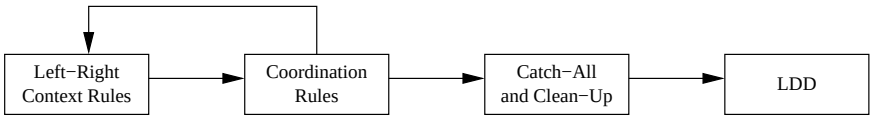

Figure 1: F-Structure Annotation Algorithm Modules

Cahill et al. (2002, 2004) present an LFG f-structure annotation algorithm that annotates Penn-II-style treebank trees (Marcus et al., 1994) with f-structure information (equations), from which a constraint solver can produce an f-structure. The algorithm is modular with four components (Figure 1) taking Penn-II trees as input and automatically adding LFG f-structure equations to each node in the tree. The annotation algorithm is described at length in Burke (2006). Here we will be brief. First, head finding rules are used to head-lexicalise the treebank trees. This partitions the daughters in local subtrees of depth one into a left context, followed by the head, followed by a right context. Left-right annotation matrices state generalisations over nodes occurring in these contexts: e.g. DET nodes in the left context of local subtrees rooted in an NP receive the LFG f-structure annotation $\uparrow$ SPEC:DET = $\downarrow$. Heads are annotated $\uparrow$=$\downarrow$. The leftmost sister NP to a V head rooted in a VP is annotated $\uparrow$ OBJ = $\downarrow$ etc. In order to keep annotation matrices perspicuous and concise, coordination is treated by a separate component. A Catch-All and Clean-Up component corrects overgeneralisations of the previous modules and provides default annotations for nodes which did not receive an annotation. The LDD component translates traces and coindexation representing long-distance dependencies in Penn-II treebank trees into reentrancies in f-structure. Lexical information is provided automatically in terms of macros associated with Penn-II part-of-speech (POS) tags.

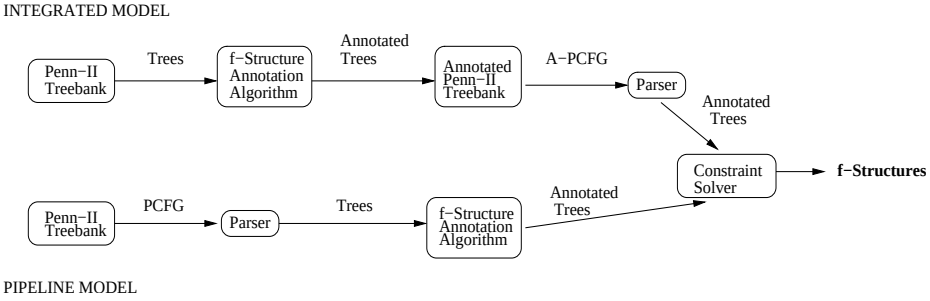

Figure 2: Treebank-based LFG Parsing: Parsing Architectures

Cahill et al. (2002, 2004) provide two parsing architectures exploiting the f-structure annotation algorithm: in the *pipeline* architecture a probabilistic context-

free grammar (PCFG) or a lexicalised history-based Markov grammar (Collins, 1999; Charniak, 2000) is extracted from the Penn-II treebank (WSJ sections 02-21), new text is parsed into trees, these trees are passed to the f-structure annotation algorithm and f-structures are produced. In the *integrated* architecture, the Penn-II treebank is first annotated with f-structure equations using the f-structure annotation algorithm and an annotated PCFG (an A-PCFG) is extracted (from WSJ sections 02-21). An A-PCFG carries f-structure equations associated with CFG categories in the rules. The A-PCFG is used to parse new text into trees with f-structure annotations from which f-structures are produced. The two parsing architectures are shown in Figure 2.

Because c-structure parser output in general does not produce traces and coindexation in parse trees to represent long distance dependencies, in both parsing architectures long distance dependencies are resolved at the level of f-structure using finite approximations of LFG functional uncertainty equations and subcategorisation frames automatically learned from the f-structure annotated Penn-II treebank (Cahill et al., 2004; O'Donovan et al., 2004). In this paper, the pipeline parsing architecture is employed with Charniak and Johnson's reranking parser (Charniak and Johnson, 2005) to parse the BNC.

3 Related Work

In the field of information extraction, there have been attempts to obtain predicate-argument structures from the output of statistical parsers. Pasca and Harabagiu (2001) use head finding rules to "read off" binary dependencies from the output of Collins' parser (Collins, 1999) to help analyse questions into corresponding answer types in a high-performance QA system. Surdeanu et al. (2003) present a Propbank (Kingsbury et al., 2002) trained role labeling system to label the output of Collins' parser for use in an information extraction system. These systems do not resolve long distance dependencies, although Surdeanu et al. (2003) integrate an anaphora resolution component into their IE processing pipeline.

The LFG annotation algorithm takes as input a c-structure and produces an f-structure from which dependency relationships between words can be obtained. Thus, it is related to the constituent-to-dependency conversion methods used to prepare English training material for dependency parsing (Yamada and Matsumoto, 2003; Johansson and Nugues, 2007). The dependencies produced by the conversion method of Yamada and Matsumoto (2003), which is based on the head-finding rules of Collins (1999), do not attempt to capture long-distance dependencies. They are handled by the conversion method of Johansson and Nugues (2007).

Deep linguistic grammars have been automatically acquired from treebanks for Head-Driven Phrase Structure Grammar (HPSG) (Pollard and Sag, 1994) and Combinatory Categorial Grammar (CCG) (Steedman, 2000). Hockenmaier and Steedman (2002) and Clark and Curran (2004) show how wide-coverage probabilistic CCGs can be extracted from the Penn-II treebank and how these resources

can be used in a number of probabilistic parsing models, resolving long-distance dependencies. Their CCG-based parsing system has been used to analyse the one billion word Gigaword corpus (Curran et al., 2007). Miyao and Tsujii (2002, 2004) show how wide-coverage, HPSG resources can be acquired from the Penn-II treebank and how the feature forest model can be used for efficient parsing and parameter estimation. The HPSG resources resolve long distance dependencies. Miyao et al. (2006) show how these resources can be adapted to the medical domain.

To our knowledge, we are the first to parse the BNC in its entirety *and* to evaluate the parsing on a hand-annotated subset. Briscoe and Carroll (2002) report that almost all of the written section of the BNC (90 million words, see Section 4) has been parsed using the RASP parser, which combines a hand-crafted grammar with a probabilistic parse selection model. Baldwin et al. (2004) describe how the Lingo ERG (Copestake and Flickinger, 2000), a hand-crafted HPSG English grammar, was tested by using it to parse BNC sentences. The aim of their work was not to parse the BNC but to test the grammar's coverage using corpus data, and only a small subsection of the written section of the BNC was parsed.

4 British National Corpus

The BNC is a one hundred million word corpus of written and spoken English from a variety of sources. It is a balanced corpus and is designed to be a representative sample of British English from the late twentieth century. Written text comprises 90% of the BNC: 75% of this is non-fiction. The written text is taken from newspapers, published and unpublished letters, school and university essays, academic journals and novels. The spoken component of the BNC consists of transcriptions of spontaneous unscripted dialogue with participants of various ages, regions and social classes, and transcriptions of more formal speech, e.g. business meetings, speeches or radio shows. The BNC is automatically tagged for part-of-speech using the CLAWS4 tagger (Garside et al., 1987), with an estimated tagging accuracy of 97.5%.[1] A two million word subset has been manually tagged using a richer tagset. It is encoded in SGML, with metadata expressed at the document (e.g. document source, genre, id) and sentence (e.g. sentence id) level.

5 C-Structure Parsing

In this section we describe how the BNC was parsed using Charniak and Johnson's two-stagereranking parser (Charniak and Johnson, 2005). The first stage of the parser is a generative, history-based, lexicalised parser (Charniak, 2000) which outputs a list of *n*-best parse trees. The second stage is a discriminative reranker which re-orders the *n*-best list produced by the first stage parser. This parser achieves a

[1]This figure was obtained from `http://www.natcorp.ox.ac.uk/docs/bnc2error.htm`.

labelled Parseval f-score of 91.3% on Section 23 of the Wall Street Journal corpus, and 85.2% on the Brown corpus standard test set (McClosky et al., 2006b).

Section 5.1 details the preprocessing steps which were applied to the BNC sentences to facilitate parsing. Section 5.2 discusses some of the issues involved in actually parsing the BNC sentences, Section 5.3 describes how a gold standard set of 1,000 parse trees was constructed, and, finally, Section 5.4 presents the results of evaluating the parses produced by the parser against the gold standard trees.

5.1 Preprocessing Steps

> *"Cleaning is a low-level, unglamorous task, yet crucial: The better it is done, the better the outcomes. All further layers of linguistic processing depend on the cleaniness of the data."*
> (Kilgarriff, 2007, p.149)

In our approach we use a Penn-II American English trained parser to parse British English as represented by the BNC. In order to achieve optimal results, we carry out a number of preprocessing steps to adapt BNC coding conventions to what is expected by a Penn-II trained parser. This includes SGML entities, soft hyphens, quotes, currency symbols and spelling differences between American and British English. Adaptations carried out to more closely match the Penn Treebank (PTB) encoding conventions can be expected to improve the parse results because the number of unknown tokens for the parser is reduced. This section describes the preprocessing steps we applied to the BNC in order to obtain cleaner input for the c-structure parsing described in Section 5.2.

5.1.1 Extraction of Sentences

In the original BNC, the start but not the end of each sentence is marked. Usually, the end of a sentence is indicated by the start of the next sentence or the end of the document. Very occasionally (eighteen cases), other tags such as paragraph markers were used to detect the end of a sentence. While processing the BNC SGML files, various tags were exploited to annotate the sentences with additional information, for example whether they belong to headers, list items, spoken utterances, poems, etc. A tag that needs special attention is the `<gap>` tag. It marks omissions due to anonymisation and replaces various material including formulae and figures. To facilitate parsing, we automatically re-inserted text for gaps according to Table 1. The gap substitutions are recorded and are recoverable. In total, 51,827 gap substitutions were performed in 38,452 sentences (0.617 %).

5.1.2 UTF-8 Encoding of SGML Entities

The BNC uses a large number of SGML entities to represent special characters, symbols, fractions, etc. A mapping to UTF-8 was manually created based on the description in the file `bncents.dtd` included in the BNC distribution and Unicode

Gap Description	Substitution String
last or full name	Jon1234es
list of names	Jon1234es , Smi1234th and Mur1234phy
date	29/12/1970
list of dates	29/12/1970 , 30/12/1970 and 31/12/1970
list of countries	Germ1234any , Ire1234land and Spa1234in
address	11234 Sun1234set Avenue
name and address	Mur1234phy , 11234 Sun1234set Avenue
telephone number	0123/4561234
number	1231234
formula	1231234

Table 1: Gap substitutions: 1234 is replaced by a random number drawn from an exponential distribution.

character code charts[2] and other web resources. UTF-8 is not used in the PTB which only contains ASCII characters. Nevertheless, the conversion is useful as it also affects ASCII characters that are represented by an SGML entity in the BNC, for example, the dollar sign. For other characters, UTF-8 serves more as an intermediate format that allows us to keep as much information as possible and at the same time to visualise the intended symbols in normal text editors. 1,255,316 (20.156 %) BNC sentences contain UTF-8 characters. Quote and currency conversions (see below) reduce this number to 45,828 sentences (0.736%).

5.1.3 Disambiguation of Soft Hyphens

Inspection of the frequency tables of special characters revealed that soft hyphens occur in the BNC. They are supposed to mark hyphens inserted by pagination processes at the end of a line. Often, they are also used to mark possible hyphenation points.[3] As the PTB does not contain soft hyphens at all, we decided to replace them with the following simple strategy. We create three candidate substitutions (deletion, space, normal hyphen) and vote based on the frequency of the respective tokens and bigrams in the BNC. Manual evaluation of this strategy on 100 randomly extracted instances showed 6 clear errors and 12 unclear cases. Only 4,190 BNC sentences (0.067%) contain soft hyphens.

5.1.4 Normalisation of Quotes

The PTB uses and the PTB-trained parsers expect sequences of two single left or right quotes to represent left and right quotes. In most cases, distinct quotes in the

[2] `http://www.unicode.org/charts/` accessed during 2005 and 2006

[3] The correct usage is controversial – compare for instance the Wikipedia article on hyphens and the detailed discussion on the web-page `http://www.cs.tut.fi/~jkorpela/shy.html`

BNC can be easily converted to PTB-style. However, some sections of the BNC use neutral quotes. Very rarely, single quotes are used as well. In order to achieve optimal results, a more elaborate conversion is necessary. We disambiguate neutral quotes by replacing them with alternating left and right quotes. Existing unambiguous quotes are respected, so that a neutral quote after a left quote becomes a right quote. Single quotes are not changed as there would be a high risk of accidently damaging apostrophes. The total number of BNC sentences containing ambiguous neutral double quotes is 68,020 (1.092%).

5.1.5 Currency and Other Normalisations

The PTB uses individual tokens for currency and number, for example US$ 2,000, while the BNC amalgamates them into a single token. Furthermore, the pound sign is the dominant currency symbol in the BNC while the PTB does not provide (much) training data for it.[4] Therefore, we map pound, yen and euro symbols to the dollar sign and, in a second step, insert a token boundary after each dollar sign to separate a possibly attached amount. The currency symbols are restored after parsing. A total of 69,459 BNC sentences (1.115%) contain currency symbols.

Additionally, dashes are replaced by PTB-style sequences of minus signs. Horizontal ellipsis is replaced by three full stops. Many fractions are represented by single entities in the BNC, and consequently mapped to single characters in Unicode (if possible), e.g. frac23 and U+2154 for two-thirds. The common fractions 1/4, 1/2, and 3/4 are re-written with normal numbers and a forward slash. Prime and double prime are encoded as single and double (neutral) quotes. The multiplication sign is replaced by 'x'. The bullet and micro signs that are quite frequent in the BNC are not replaced because we could not find suitable examples in the PTB.

5.1.6 Translation to American English

The varcon package (`http://wordlist.sf.net`) was used to translate the BNC to American English. According to the source code and vocabulary file, the varcon translation process is only a matter of different spelling and words substitutions. Word order and tokenisation are not changed. If required, the original British English tokens can be written into the leaf nodes of parse trees. The varcon tool has been modified to not change "For" to "Four" because this substitution would also apply to the preposition "for" at the start of a sentence or in headings. The total number of BNC sentences that are changed by varcon is 333,745 (5.359%).

[4] Recently, we found out that the pound sign is represented by the # sign in the PTB, see `http://www.ldc.upenn.edu/Catalog/docs/treebank2/cl93.html`. Still, a substitution with the dollar sign can be justified by the larger amount of instances that provide more reliable statistics for the parser.

5.2 Parsing the BNC sentences

5.2.1 N-Best Parsing and Reranking

To carry out the BNC c-structure parsing, we used Charniak and Johnson's reranking parser (Charniak and Johnson, 2005) taken off the shelf from the June 2006 package.[5] We implemented a wrapper that supervises the parsing process because the earlier August 2005 version we used during development sometimes aborts parsing, leaving some sentences unparsed, or fails to terminate. The order of sentences is randomised to spread the effect of bugs evenly over the corpus and to make the duration of each task-farming package more similar (see Section 5.2.2)

The parsing and reranking phases were carried out separately. Charniak's parser allows for an ID string to be present in the `<s>` tag, but this string cannot contain any whitespace. Therefore, the SGML attribute-value list expressing the annotation was encoded with underscores instead of spaces and restored after parsing. The first-stage parser was instructed to respect the input tokenization and to output the 50-best parse trees. Of the 6,228,111 BNC input sentences, 6,218,384 (99.844%) were parsed successfully. After parsing, we applied the reranker to the output of the first stage. The reranker succeeded in reranking all sentences that had been parsed by the first-stage parser.

5.2.2 Time and Space Requirements

Parsing the preprocessed BNC is not a trivial task as it would require several months of computation time on a single PC and substantial disk space for the results. We decided to avail ourselves of a high-end computing facility to carry out this task. On 31 CPUs (AMD Opteron 250, 2.4 GHz single core), parsing the 6,228,111 sentences took 79.5 hours walltime, i.e. roughly 2,500 CPU hours or 1.425 seconds per sentence. x Table 2 shows the parsing time for 50-best parsing (and also for 2-best parsing that we considered initially), this time on a 2.8 GHz Pentium 4 and with an older version of the parser. There is a huge variance in parsing time and (unsurprisingly) sentence length is an important factor. Because of this, the observed parsing speed may not translate well to other corpora. The re-ranking process is faster than the parsing process – sentences were re-ranked at the rate of 0.15 seconds per sentence.

The space required for the output of parsing tends to get big compared to the raw sentences, especially for n-best parsing. However, the output can be compressed with very high compression ratios as the set of categories is small and n-best parses are similar to each other. We measured a compression ratio of 27.5 for GZip and 38.0 for BZip2 on the 8,000 sentences used in Table 2 for time measurements. The actual size of the 50-best parses including our SGML markup is 3.4 GB after compression with BZip2.

[5] reranking-parserJune06.tar with SHA1 832e63ce87196d5d0e54b6414020d1c786217936 downloaded from `ftp://ftp.cs.brown.edu/pub/nlparser/`

Length	$n = 2$	$n = 50$	**Increase** $2 \rightarrow 50$
00–04	0.407	0.414	1.72%
05–09	0.687	0.696	1.31%
10–14	1.153	1.169	1.39%
15–19	1.914	1.946	1.67%
20–24	2.559	2.577	0.70%
25–29	3.594	3.630	1.00%
30–34	4.683	4.664	-0.41%
35–39	6.116	6.139	0.38%

Table 2: Parsing time for n-best parsing in seconds per sentence, measured with 1,000 random BNC sentences in each length range

5.3 Constructing BNC Gold Standard C-Structures

A gold standard set of 1,000 BNC sentences was constructed by one annotator who manually corrected the output of the first stage parser of Charniak and Johnson's reranking parser. The sentences included in the gold standard were chosen at random from the BNC, subject to the condition that they contain a token which occurs as a verb in the BNC but not in the training sections of the WSJ section of the PTB. A decision was made to select sentences for the gold standard set which differ from the sentences in the WSJ training sections, and one way of finding different sentences is to focus on verbs which are not attested in the WSJ Sections 2-21. The gold standard sentences serve a dual purpose: as a set of test sentences (and it is in this role that they are being used in the research described here) and as a potential set of training sentences (for future research). Because they contain verbs which do not occur in the parser's training set, they are likely to represent a hard test for WSJ-trained parsers.

The following steps were carried out to obtain the gold standard sentences:

1. Using the BNC tag set, a list of all verbs occurring in the BNC was generated. A frequency count was associated with each verb.

2. All verbs in the BNC verb list were converted to their root form, duplicates were merged and frequency counts adjusted.[6]

3. The BNC verb root forms were converted to American English using the varcon tool (see Section 5.1.6).

4. Using the PTB tag set, a list of all verbs occurring in Sections 2-21 of the WSJ corpus was generated.

5. All verbs in the WSJ verb list were converted to their root form, and duplicates merged.

[6]Lemmatisation was carried out using the Xerox XLE xfst tool (Maxwell and Kaplan, 1996).

	LP	LR	F-Score	%100 Match
All Sentences	83.8	83.7	83.7	25.2
Less than 41 words	86.4	86.2	86.3	30.3

Table 3: BNC C-Structure Evaluation: Labelled Parseval

6. A list of all verb root forms in the BNC verb root form list which are not in the WSJ verb root form list was compiled (BNC-WSJ). This resulted in 25,874 root forms. Of these, 537 occurred more than one hundred times within the BNC, and 14,787 occurred only once.[7]

7. 1,000 forms were selected from the BNC-WSJ verb root form list, respecting the frequency of the verb forms so that root forms which occur frequently within the BNC were favoured over root forms occurring only once.

8. For each of the 1,000 verb roots, a sentence containing a conjugated form of this token was randomly selected from the BNC.

The PTB bracketing guidelines (Bies et al., 1995) and the PTB itself were used as references by the BNC annotator. Functional tags and traces were not annotated. The annotator noticed that the PTB parse trees sometimes violate the PTB annotator guidelines, and in these cases, the annotator chose the analysis set out in the guidelines. An example is the noun phrase *almost certain death* which occurred in a sentence in the BNC gold standard. According to Bies et al. (1995, p.179), this should be analysed as *(NP (ADJP almost certain) death)*, but a search for the word *almost* in the PTB yielded a similar example (in WSJ Section 9, 0946.prd) *almost unimaginable speed*, which was parsed as *(NP almost unimaginable speed)*. The BNC annotator analysed the phrase *almost certain death* as *(NP (ADJP almost certain) death)*, according to the guidelines. If a structure was encountered which was not mentioned in the PTB bracketing guidelines and no example of which could be found in the PTB, the annotator decided how it should be analysed and documented this decision. An example is the phrase *day in day out* which was analysed as a flat adverbial phrase. It took approximately sixty hours to construct the gold standard.

5.4 C-Structure Evaluation

Table 3 shows the results of evaluating the parses produced by Charniak and Johnson's parser against the gold standard parses described in Section 5.3 using the Parseval labelled precison/recall measures (Black et al., 1991). The results were calculated using the *evalb* software and the *new.prm* parameter file, which are distributed with the parser. The precision figure represents the number of correct constituents divided by the total number of constituents produced by the parser.

[7]The most frequently occurring verb lemma in the BNC which does not appear (as a verb) in WSJ2-21 is *mutter*, which occurs 1,871 times.

	LP	LR	F-Score	%100 Match
All Sentences	85.5	85.4	85.4	27.9
Less than 41 words	88.2	88.0	88.1	33.5

Table 4: BNC C-Structure Evaluation: Unlabelled Parseval

Constituent Type	Precision	Recall	F-Score
NP	86.8	88.4	87.6
VP	81.6	81.8	81.7
S	80.0	81.8	80.9
PP	80.2	82.1	81.1
SBAR	75.8	77.6	76.7
ADVP	80.3	77.4	78.8
ADJP	67.2	69.5	68.3
WHNP	91.9	96.8	94.3
PRT	61.4	84.3	71.1
WHADVP	97.3	95.5	96.4

Table 5: BNC C-Structure Evaluation: 10 Most Frequent Constituents

The recall figure represents the number of correct constituents divided by the total number of constituents in the gold standard set. The f-score is the harmonic mean of precision and recall. A constituent in a test parse tree is considered to be correct if it spans the same sequence of words and has the same label as a constituent in the corresponding gold standard parse tree. Unlabelled precision/recall figures are shown in Table 4: these figures reflect a more relaxed notion of correctness, whereby a constituent in a test parse is considered correct if a constituent spanning the same sequence of words in the corresponding gold tree can be found, regardless of whether the labels on both constituents match. Tables 3 and 4 both also contain the percentage of sentences which achieve an f-score of 100%.

The results in Tables 3 and 4 show that Charniak and Johnson's parser performs quite well on BNC data with a labelled f-score of 83.7%, considering that it achieves an f-score of 91.3% on Section 23 of the Wall Street Journal corpus and 85.2% on the Brown Corpus (McClosky et al., 2006a). This is quite encouraging because the BNC sentences represent a different domain to the Wall Street Journal and the sentences in the gold standard contain verbs which do not occur as verbs in the parser's training data. The quality of the c-structure trees means that the f-structures which are generated from them are more likely to be reliable: previous research (Judge et al., 2006) has shown that, given good CFG trees, the f-structure annotation algorithm can produce good f-structures. Table 5 shows the labelled precision and recall figures for the ten most frequent constituents in the BNC test set (in descending order of their frequency). Among the frequently oc-

curring constituents, ADJP, SBAR, ADVP and PRT are the categories with the most room for improvement for Charniak and Johnson's parser. The parser performs well on NP and WH constituents.

6 F-Structure Annotation

In Section 6.1 we describe how the LFG Annotation Algorithm (see Section 2) was applied to the BNC c-structures produced by Charniak and Johnson's reranking parser (see Section 5) to yield BNC f-structures. In Section 6.2 we describe how the BNC f-structures were evaluated, we present evaluation results and, in an error analysis, discuss some low-scoring examples.

6.1 Annotation Process

The same high-end computing facility which was used to perform the first stage of the c-structure parsing (see Section 5.2.2) was employed to apply the annotation algorithm to the c-structures. The output of the reranking parser is a list of 50 parse trees, and the highest ranked of these is passed as input to the annotation algorithm. The annotation is faster than the c-structure parsing, with an annotation rate of approximately 16 sentences per second. The annotation algorithm fails for just one sentence:

> *And , and , you know , they 've got paid youth officer 's working in Harlow , now they are , there are , they 're over they 're over stretched it 's true and , but we , I mean what were doing here is actually supplementing there service and were not meeting all , we would n't of erm meeting all the demands , but the important thing I think is that were continuing to erm , you know , were trying to do something about it , and one of the things that were trying to do as officer 's in the Local Government Unit is work with Leisure Services and get them to put more resources into doing things for young people* . (BNC D95.477)

6.2 F-Structure Evaluation

6.2.1 F-Structure Evaluation Procedure

In order to evaluate the f-structures generated by the annotation algorithm, it is necessary to have a set of reference or gold standard f-structures. Unfortunately, due to time constraints, we do not have a hand-corrected set of gold standard f-structures. Instead we followed the established procedure (Hockenmaier and Steedman, 2002; Miyao and Tsujii, 2004) of automatically constructing a reference set of f-structures by applying the annotation algorithm to the manually corrected gold standard c-structures (see Section 5.3). We then evaluate the f-structures produced by applying the annotation algorithm to the c-structure parser output against

Attribute	Precision	Recall	F-Score	WSJ
OVERALL	91.1	91.4	**91.2**	94.3
PRED-ONLY	86.5	86.1	**86.3**	91.1

Table 6: BNC F-Structure Evaluation

the reference f-structures by computing precision and recall on the f-structures as sets of term descriptions (following Crouch et al. (2002)).

6.2.2 F-Structure Evaluation Results

Attribute	Precision	Recall	F-Score	WSJ
adjunct	83.3	83.6	**83.4**	89
num	96.6	97.3	**96.9**	97
pers	97.2	97.9	**97.5**	98
obj	90.1	90.4	**90.2**	94
subj	89.6	87.4	**88.5**	93
tense	97.4	96.3	**96.8**	95
det	96.5	96.4	**96.4**	98
pron_form	98.5	99.0	**98.7**	99
coord	84.0	82.1	**83.0**	89
xcomp	87.7	85.6	**86.6**	91

Table 7: BNC F-Structure Evaluation: Ten Most Frequent Attributes

The f-structure evaluation results for the 1,000 BNC test sentences are shown in Table 6. Evaluated in this way, the BNC sentences receive an f-score of 91.2%. When attributes with atomic values (e.g. `num`, `tense` and `pers`) are omitted, the f-score goes down to 86.3%. The fourth column in Table 6 shows the results of performing the same evaluation on Section 23 of the WSJ. The annotation algorithm is applied to Section 23 gold standard parse trees (stripped of functional tags and traces) and the resulting f-structures are compared to those produced by applying the annotation algorithm to Charniak and Johnson parser output.

Table 7 shows the individual evaluation results for the ten most frequently occurring attributes, and Table 8 for the remaining less frequent pred-only attributes. Again, the WSJ Section 23 results are shown for comparison. It is clear from Table 7 that atomic-valued attributes such as `tense`, `num` and `pers` attributes suffer little when moving from the WSJ to the BNC domain, unlike the arguably more important attributes of `subj`, `adjunct`, `obj` and `xcomp`. Table 8 shows that there is significant room for improvement for the attributes `relmod`, `topicrel`, `comp`, `quant`, `app`, `obj2` and `topic`.

Attribute	Precision	Recall	F-Score	WSJ
poss	95.7	94.7	**95.2**	96
comp	72.6	73.7	**73.1**	87
topicrel	77.3	79.8	**78.5**	88
relmod	64.5	69.6	**67.0**	80
quant	87.6	82.7	**85.1**	95
obl	69.0	61.9	**65.3**	71
obl_ag	91.5	91.5	**91.5**	96
app	42.9	43.8	**43.3**	86
obj2	47.1	55.8	**51.1**	71
topic	50.0	75.0	**60.0**	87
focus	100.0	75.0	**85.7**	59
obl2	50.0	33.3	**40.0**	22

Table 8: BNC F-Structure Evaluation: Other Pred-Only Attributes

6.2.3 Two Low-Scoring Examples

Consider the BNC examples (1) and (2):

(1) *They've been digging coal in small private mines in the area for centuries* (BNC K1J.1996)

(2) *Grey-haired, stooping, shabbily dressed* (BNC GVT.0268)

The top part of Figure 3 shows the gold standard parse tree and Charniak and Johnson parser output tree for example (1). Notice that the parser has incorrectly tagged *digging* as a noun and has incorrectly adjoined the prepositional phrase *for centuries* to the noun *area*. The dependency triples in Figure 3 show how mistakes in c-structure are propagated to f-structure. The mistagging of *digging* leads to the misanalysis of *coal* as a verb. The identification of *for* as an adjunct of *area* is a straightforward consequence of the c-structure mistake. Notice also that the "gold" f-structure in Figure 3 is not completely accurate: it incorrectly analyses the sentence as being passive. This is a consequence of the *unsupervised* creation of the reference f-structure set. Figure 4 shows the reference and test c-structures and f-structures for example (2). This example shows again that mistakes at the c-structure level are propagated to the f-structure level, and that mistakes can be introduced by the annotation algorithm, resulting in less than perfect reference f-structures.

7 Conclusions and Future Work

In this paper we have described in some detail how Lexical-Functional Grammar c-structures and f-structures were extracted from the one-hundred million word

Gold
(S (NP They) (VP 've (VP been (VP (VBG digging) (NP coal) (PP in (NP (NP small private mines) (PP in (NP the area)))) (PP for (NP centuries))))))
Test
*(S (NP They) (VP 've (VP been (VP **(NN digging)** (NP coal) (PP in (NP (NP small private mines) (PP in **(NP (NP the area) (PP for (NP centuries)))**)))))))*

Gold	Test
adjunct(dig,in)	**adjunct(coal,in)**
subj(dig,pro)	**subj(coal,pro)**
subj(be,pro)	subj(be,pro)
adjunct(mine,in)	adjunct(mine,in)
passive(be,+)	passive(be,+)
obj(in,area)	**adjunct(area,for)**
tense(be,past)	tense(be,past)
adjunct(dig,for)	**num(digging,sg)**
obj(dig,coal)	**pers(digging,3)**
xcomp(be,dig)	**xcomp(be,coal)**
participle(dig,pres)	**obj(coal,digging)**

Figure 3: Gold and Test C-Structures and F-Structures for Example (1)

British National Corpus using a treebank-based parsing architecture. Two steps were involved in this process: the first step was the parsing of the sentences into context-free trees or LFG c-structure; the second step was the annotation of these trees to produce LFG f-structures, from which semantic dependencies can be extracted. A thorough description of the preprocessing required to parse a corpus as large as the BNC was provided. This paper also described a c-structure gold standard for the BNC, presented Parseval results for Charniak and Johnson's reranking parser on this gold standard, and provided an evaluation of the f-structures against an automatically generated set of gold standard f-structures. The c-structures achieve an f-score of 83.7% and the f-structures an f-score of 91.2%. Our research demonstrates that it is feasible to provide a reasonably accurate LFG analysis of a very large body of sentences in a robust, non-labour-intensive way.

Our next goal is to investigate to what extent parsing accuracy can be improved by performing self-training experiments. We have already begun work on this: following McClosky et al. (2006a), we have re-trained Charniak and Johnson's first-stage parser on BNC parse trees produced by the two-stage reranking parser, and have obtained a statistically significant f-score increase of 1.7% (Foster et al., 2007). When the annotation algorithm is applied to the output of this self-trained parser, accuracy goes up from 91.2% to 91.7%. We aim to build on these small improvements by applying more sophisticated domain adaptation methods.

The f-structure annotation algorithm is sensitive to the presence of Penn-II

Gold
(FRAG (ADJP (ADJP Grey-haired), (ADJP stooping), (ADJP (ADVP shabbily) dressed)))
Test
(FRAG ***(ADJP Grey-haired , stooping)*** *, (VP (ADVP shabbily) dressed))*

Gold	Test
tense(dress,past) adjunct(dress,shabbily) **adjunct(dress,grey-haired)** **adjunct(dress,stooping)** participle(stooping,pres)	tense(dress,past) adjunct(dress,shabbily) **adjunct(stooping,grey-haired)**

Figure 4: Gold and Test C-Structures and F-Structures for Example (2)

functional tags and will use such information to assign f-structure functional equations, backing off to simpler categorical and configurational information if functional tags are not present. Here we use the annotation algorithm on raw parser output trees with no functional tags. In the future we aim to apply a post-parsing Penn-II functional tag labeller, e.g. Chrupała et al. (2007), to raw parser output prior to the application of the annotation algorithm. Other aims include the manual correction of our reference f-structures so that more confidence can be placed in the f-structure evaluation results and the application of our pipeline parsing architecture to the BNC using other c-structure parsers, e.g. Bikel (2004).

References

Baldwin, Timothy, Bender, Emily M., Flickinger, Dan, Kim, Ara and Oepen, Stephan. 2004. Road-testing the English Resource Grammar over the British National Corpus. In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC-04)*, volume Six, pages 2047–2050, Lisbon, Portugal.

Bies, Ann, Ferguson, Mark, Katz, Karen and MacIntyre, Robert. 1995. Bracketing Guidelines for Treebank II Style, Penn Treebank Project. Technical Report Tech Report MS-CIS-95-06, University of Pennsylvania, Philadelphia, PA.

Bikel, Daniel. 2004. Intricacies of Collins Parsing Model. *Computational Linguistics* 30(4), 479–511.

Black, Ezra, Abney, Steve, Flickinger, Dan, Gdaniec, Claudia, Grishman, Robert, Harrison, Philip, Hindle, Donald, Ingria, Robert, Jelinek, Fred, Klavans, Judith, Liberman, Mark, Marcus, Mitchell, Roukos, Salim, Santorini, Beatrice and Strzalkowski, Tomek. 1991. A Procedure for Quantitatively Comparing the

Syntactic Coverage of English Grammars. In *Proceedings of the 1991 DARPA Speech and Natural Language Workshop*, pages 306–311.

Briscoe, Ted and Carroll, John. 2002. Robust Accurate Statistical Annotation of General Text. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC-02)*, pages 1499–1504, Las Palmas, Gran Canaria.

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

Burnard, Lou. 2000. User Reference Guide for the British National Corpus. Technical Report, Oxford University Computing Services.

Cahill, Aoife, Burke, Michael, O'Donovan, Ruth, van Genabith, Josef and Way, Andy. 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 (ACL-04)*, pages 320–327, Barcelona, Spain.

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

Charniak, Eugene. 2000. A Maximum Entropy Inspired Parser. In *Proceedings of the First Annual Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL-00)*, pages 132–139, Seattle, WA.

Charniak, Eugene and Johnson, Mark. 2005. Course-to-fine n-best-parsing and MaxEnt discriminative reranking. In *Proceedings of the 43rd Annual Meeting of the ACL (ACL-05)*, pages 173–180, Ann Arbor, Michigan.

Chrupała, Grzegorz, Stroppa, Nicolas, van Genabith, Josef and Dinu, Georgiana. 2007. Better Training for Function Labeling. In *Proceedings of the Recent Advances in Natural Language Processing Conference (RANLP-07)*, pages 133–138, Borovets, Bulgaria.

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

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

Copestake, Ann and Flickinger, Dan. 2000. An Open-source Grammar Development Environment and Broad-coverage English Grammar using HPSG. In *Proceedings of the 2nd International Conference on Language Resources and Evaluation (LREC-00)*, Athens, Greece.

Crouch, Richard, Kaplan, Ron, King, Tracy Holloway and Riezler, Stefan. 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, Gran Canaria.

Curran, James R., Clark, Stephen and Bos, Johan. 2007. Linguistically Motivated Large-Scale NLP with C&C and Boxer. In *Proceedings of the Demonstrations Session of the 45th Annual Meeting of the Association for Computational Linguistics (ACL-07)*, pages 29–32, Prague, Czech Republic.

Foster, Jennifer, Wagner, Joachim, Seddah, Djamé and van Genabith, Josef. 2007. Adapting WSJ-Trained Parsers to the British National Corpus using In-domain Self-training. In *Proceedings of the Tenth International Workshop on Parsing Technologies (IWPT-07)*, pages 33–35, Prague, Czech Republic.

Garside, Roger, Leech, Geoffrey and Sampson, Geoffrey (eds.). 1987. *The Computational Analysis of English: a Corpus-Based Approach*. Longman, London.

Hockenmaier, Julia and Steedman, Mark. 2002. Generative Models for Statistical Parsing with Combinatory Categorial Grammar. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL-02)*, pages 335–342, Philadelphia.

Johansson, Richard and Nugues, Pierre. 2007. Extended Constituent-to-Dependency Conversion for English. In Joakim Nivre, Heiki-Jaan Kaalep, Kadri Muischnek and Mare Koit (eds.), *Proceedings of NODALIDA 2007*, pages 105–112, Tartu, Estonia.

Judge, John, Cahill, Aoife and van Genabith, Josef. 2006. QuestionBank: Creating a Corpus of Parse-Annotated Questions. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL (COLING-ACL-06)*, pages 497–504, Sydney, Australia.

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

Kilgarriff, Adam. 2007. Googleology is Bad Science. *Computational Linguistics* 33(1), 147–151.

Kingsbury, Paul, Palmer, Martha and Marcus, Mitch. 2002. Adding Semantic Annotation to the Penn TreeBank. In *Proceedings of the Human Language Technology Conference (HLT-02)*, San Diego, CA.

Marcus, Mitchell, Kim, Grace, Marcinkiewicz, Mary Ann, MacIntyre, Robert, Bies, Ann, Ferguson, Mark, Katz, Karen and Schasberger, Britta. 1994. The

Penn Treebank: Annotating Predicate Argument Structure. In *Proceedings of the ARPA Workshop on Human Language Technology*, pages 110–115, Princeton, NJ.

Maxwell, John and Kaplan, Ron. 1996. An Efficient Parser for LFG. In *Proceedings of the First LFG Conference*, Grenoble, France.

McClosky, David, Charniak, Eugene and Johnson, Mark. 2006a. Effective Self-Training for Parsing. In *Proceedings of the Human Language Technology Conference and North American chapter of the ACL annual meeting (HLT-NAACL-06)*, pages 152–159, New York.

McClosky, David, Charniak, Eugene and Johnson, Mark. 2006b. Reranking and Self-Training for Parser Adaptation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL (COLING-ACL-06)*, pages 337–344, Sydney, Australia.

Miyao, Yusuke, Ohta, Tomoko, Masuda, Katsuya, Tsuruoka, Yoshimasa, Yoshida, Kazuhiro, Ninomiya, Takashi and Tsujii, Jun'ichi. 2006. Semantic Retrieval for the Accurate Identification of Relational Concepts in Massive Textbases. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL (COLING-ACL-06)*, pages 1017–1024, Sydney, Australia.

Miyao, Yusuke and Tsujii, Jun'ichi. 2002. Maximum Entropy Estimation for Feature Forests. In *Proceedings of the Human Language Technology Conference (HLT-02)*, San Diego, CA.

Miyao, Yusuke and Tsujii, Jun'ichi. 2004. Deep Linguistic Analysis for the Accurate Identification of Predicate-Argument Relations. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING-04)*, pages 1392–1397, Geneva, Switzerland.

O'Donovan, Ruth, Burke, Michael, Cahill, Aoife, van Genabith, Josef and Way, Andy. 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 (ACL-04)*, pages 368–375, Barcelona, Spain.

Pasca, Marius A. and Harabagiu, Sandra M. 2001. High Performance Question/Answering. In *The 25th Annual International ACM SIGIR Conference (SIGIR-01)*, pages 366–374, New Orleans, Louisiana.

Pollard, Carl and Sag, Ivan A. 1994. *Head-Driven Phrase Structure Grammar*. University of Chicago Press and CSLI Publications.

Steedman, Mark. 2000. *The Syntactic Process*. MIT Press.

Surdeanu, Mihai, Harabagiu, Sandra, Williams, John and Aarseth, Paul. 2003. Using Predicate-Argument Structure for Information Extraction. In *Proceedings of the 41st Annual Meeting of the ACL (ACL-03)*, pages 8–15, Sapporo, Japan.

Yamada, Hiroyasu and Matsumoto, Yuji. 2003. Statistical Dependency Analysis with Support Vector Machines. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT-03)*, pages 195–206, Nancy, France.