Evaluating a Wide-Coverage CCG Parser

Stephen Clark and Julia Hockenmaier

Division of Informatics, University of Edinburgh
2 Buccleuch Place, Edinburgh. EH8 9LW
Scotland, UK
{stephenc,julia}@cogsci.ed.ac.uk

Abstract

This paper compares three evaluation metrics for a CCG parser trained and tested on a CCG version of the Penn Treebank. The standard Parseval metrics can be applied to the output of this parser; however, these metrics are problematic for CCG, and a comparison with scores given for standard Penn Treebank parsers is uninformative. As an alternative, we consider two evaluations based on head-dependencies; one considers local dependencies defined in terms of the derivation tree, and one considers dependencies defined in terms of the CCG categories. The latter set of dependencies includes long-range dependencies such as those inherent in coordination and extraction phenomena.

1. Introduction

In this paper, we compare the advantages and shortcomings of three evaluation metrics for a statistical parser based on Combinatory Categorial Grammar (CCG, Steedman (2000)). The parser (described in Hockenmaier and Steedman (2002b)) is trained and tested on a treebank of CCG normal-form derivations which has been derived (semi)-automatically from the Penn Treebank (Marcus et al., 1993).

We apply the standard Parseval metrics to compare the derivation trees produced by the parser with those in the gold standard. However, CCG derivation trees are binary-branching, and the set of CCG categories is much larger than the set of nonterminal labels in the Penn Treebank. Therefore, a comparison with Parseval figures given for standard Penn Treebank parsers is uninformative. Furthermore, in the presence of left and right modifiers to the same constituent, there are equivalent normal-form derivations, which Parseval does not take into account.

We also consider two dependency evaluations. Like the standard Penn Treebank parsers of Collins (1999) and Charniak (2000), the CCG parser models word-word dependencies defined in terms of local rule applications. Collins (1999) proposes an evaluation based on these dependencies, which we apply to our parser. This allows a direct comparison with Collins' parser and overcomes the problem of equivalent normal-form derivations.

Unlike the phrase-structure trees returned by standard Penn Treebank parsers, CCG derivations encode the long range dependencies involved in constructions such as raising, control, extraction and coordination. In order to evaluate another CCG parser, Clark et al. (2002) introduce an evaluation which incorporates the long range, as well as local, dependencies. This evaluation is applied to the output of the normal-form parser, using the Clark et al. parser to extract the relevant dependencies from the derivation trees. This evaluation is much closer to the dependency-based evaluations of Lin (1995) and Carroll et al. (1998).

2. Combinatory Categorial Grammar

A CCG grammar consists of a lexicon, which pairs words with lexical categories, and a set of combinatory rules, which specify how categories combine. Categories are either atomic or complex. Examples of atomic categories include S[dcl] (declarative sentence), NP (noun phrase), N (noun) and PP (prepositional phrase).

Complex categories are functors which express the type and directionality of their arguments, and the type of the result. For example, the category for the transitive verb *bought* specifies that one NP is required to the right of the verb, and one NP to the left, resulting in a sentence:

(1) bought := (S[dcl]\NP)/NP

Other examples of complex categories expressing subcategorisation are as follows ([*pt*] denotes a past participle and [pss] denotes a passive):

(2) has := (S[dcl]\NP)/(S[pt]\NP)
been := (S[pt]\NP)/(S[pss]\NP)
bought := (S[pt]\NP)/NP
bought := S[pss]\NP

Complex categories of the form X/X or X\X can express modification:

(3) big := N/N
quickly := (S\NP)\(S\NP)

Constituents combine according to a set of combinatory rules, including function application, function composition and type-raising (see Steedman (2000) for the details). For example, the following derivation uses forward (>) and backward (<) application:

$$
(4)\quad
\begin{array}{cccc}
\textit{IBM} & \textit{quickly} & \textit{bought} & \textit{Lotus} \\
\hline
\mathsf{NP} & \mathsf{(S\backslash NP)/(S\backslash NP)} & \mathsf{(S[dcl]\backslash NP)/NP} & \mathsf{NP} \\
 & & \multicolumn{2}{c}{\overline{\mathsf{S[dcl]\backslash NP}}\;>} \\
 & \multicolumn{3}{c}{\overline{\mathsf{S[dcl]\backslash NP}}\;>} \\
\multicolumn{4}{c}{\overline{\mathsf{S[dcl]}}\;<}
\end{array}
$$

Composition and type-raising are necessary for certain types of extraction and coordination phenomena. In the following object-extraction example, type-raising (>*T*) first turns the NP for *IBM* into a functor looking for a verb-phrase, which then combines with the category for *bought* using forward composition (>*B*):

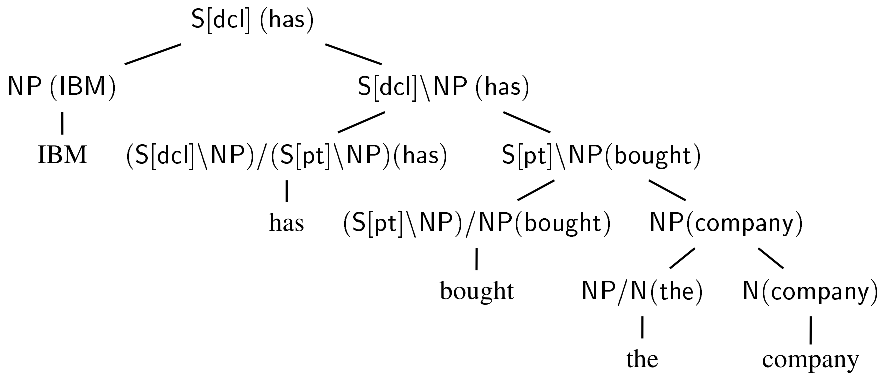
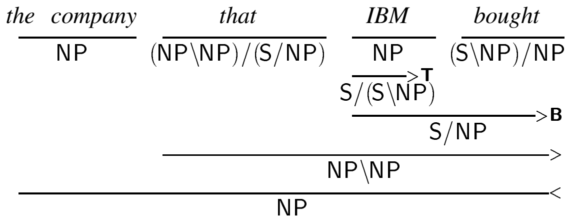


Figure 1: A derivation tree marked with heads

(5) 

Note that the use of composition introduces so-called "spurious ambiguity", in which distinct derivations for a sentence lead to the same semantic interpretation. Even a simple sentence such as *IBM bought Lotus* has several derivations, one using only function application, and the others using type-raising and composition. However, all derivations lead to the same interpretation: that *IBM* is the buyer and *Lotus* is the buyee.

One solution to the problem of spurious ambiguity is to only apply function composition when syntactically necessary; such a derivation is called *normal-form*. The corpus that we use to train and test the parser described here contains only normal-form derivations.

3. The parser

The parser that we evaluate is described in Hockenmaier and Steedman (2002b), and is based on a generative model of CCG derivation trees. Like most recent work in statistical parsing – including the generative models of Collins (1997) and Charniak (2000) – the parser models the word-word dependencies defined by local subtrees. Each constituent is assumed to have one lexical head (a word and its lexical category). The example derivation in Figure 1 shows how heads are percolated through the derivation tree.

The statistical model assumes a top-down tree-generating process in which heads are generated at the maximal projection of a constituent. Unless this maximal projection is the root of the entire tree, the constituent is a complement or adjunct of another constituent, and there is a dependency between the the heads of both constituents. This dependency is expressed in the statistical model by conditioning the head of complements or adjuncts on the head of the parent node and the local tree which defines the dependency relation. For example, in Figure 1, *bought* is not only conditioned on its lexical category (S[pt]\NP)/NP, but also on the fact that it appears within a local tree with head word *has*, parent S[dcl]\NP, left (head) daughter (S[dcl]\NP)/(S[pt]\NP) and right (non-head) daughter S[pt]\NP.

The parser is trained and tested on a treebank of CCG normal-form derivations. This corpus, which we call CCGbank, has been derived (semi)-automatically from the Penn Treebank (Marcus et al., 1993), using sections 02-21 for training and section 23 for testing. For further details of CCGbank we refer readers to Hockenmaier and Steedman (2002a).

4. Evaluation metrics

This section describes the different evaluation metrics, which we illustrate by evaluating the (fictitious) output tree in the bottom of figure 2 against the correct derivation given in the top of figure 2.

4.1. Parseval

The first measures are the standard Parseval metrics bracketed precision/recall and labelled precision/recall used to compare the normal-form derivation trees produced by the parser with those in the gold standard (section 23 of the CCGbank).

Following common practice, we disregard punctuation marks. Since CCG derivation trees are at most binary branching, punctuation marks introduce a separate level into the tree, which we also disregard in the evaluation.

Consider the trees given in figure 2. Discarding the lexical categories (but not their unary projections), the gold standard has six nodes, three of which are correctly identified in the output tree. The output tree has seven nonterminal nodes. Hence, labelled and bracketed precision are both 3/7; labelled and bracketed recall are both 3/6. Note that Parseval does not take the correctness of lexical categories into account, which is important for CCG since categories encode subcategorisation information. Therefore, we also give the accuracy of lexical categories (again disregarding punctuation marks), which in this case is 4/6.

4.2. Dependency evaluation 1

Collins (1999) gives an alternative evaluation to Parseval, measuring the recovery of word-word dependencies. According to his definition, there is a dependency between two words w and w' if the parse contains a local tree such that w' is the head of this tree and w is the head of a non-

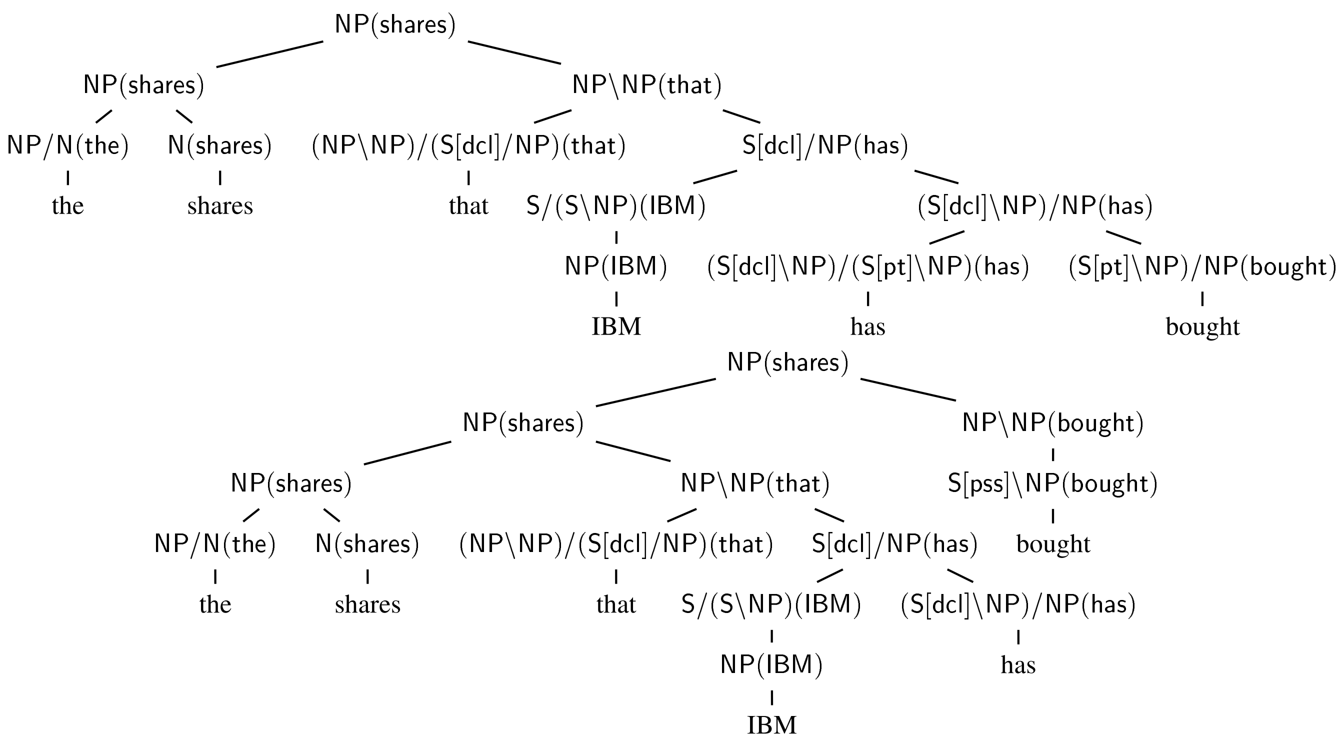
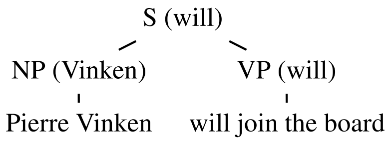


Figure 2: Example trees for evaluation: the top tree is the gold standard.

head daughter. The following tree defines a dependency between *Vinken* and *will*:



The dependency relation is determined by the label of the parent node (S), the label of the head daughter (VP), the label of the non-head daughter (NP), and the direction of the non-head daughter (left): ⟨S, VP, NP, left⟩. Furthermore, if the non-head daughter is a complement, its category carries a complement feature −C. In Collins' original evaluation, coordinate constructions are distinguished by a further element CC. We adapt this evaluation to CCG; however, since the directionality of the head is directly encoded in the categories, there is no need for this feature. A similar comment applies to the complement feature. Also, in CCGbank, binary nodes within a coordinate construction carry a special coordination feature, and so the CC-feature is redundant as well.

The way in which these dependencies are defined means there is exactly one relation to be determined for each word. There is a special relation for the head of the sentence (which is not dependent on any other word). Collins gives scores for labelled and unlabelled dependencies. Unlabelled dependency scores only take into account whether there is a relation between w and w' such that w' is the head and w its modifier or complement, but not whether the local tree which defines this dependency is correctly labelled.

Returning to our example, the gold standard in figure 2 defines the following dependencies:

Relation ⟨*Parent*, *Head*, *Sister*⟩	**Head**	**Dep**
⟨NP, N, NP/N⟩	*shares*	*the*
⟨Head⟩		*shares*
⟨NP, NP, NP\NP⟩	*shares*	*that*
⟨NP\NP, (NP\NP)/(S[dcl]/NP), S[dcl]/NP⟩	*that*	*has*
⟨S[dcl]/NP, (S[dcl]\NP)/NP, S/(S\NP)⟩	*has*	*IBM*
⟨(S[dcl]\NP)/NP, (S[dcl]\NP)/(S[pt]\NP), (S[pt]\NP)/NP⟩	*has*	*bought*

These are the dependencies in the incorrect analysis:

Relation ⟨*Parent*, *Head*, *Sister*⟩	**Head**	**Dep**
⟨NP, N, NP/N⟩	*shares*	*the*
⟨Head⟩		*shares*
⟨NP, NP, NP\NP⟩	*shares*	*that*
⟨NP\NP, (NP\NP)/(S[dcl]/NP), S[dcl]/NP⟩	*that*	*has*
⟨S[dcl]/NP, (S[dcl]\NP)/NP, S/(S\NP)⟩	*has*	*IBM*
⟨NP, NP, NP\NP⟩	*shares*	*bought*

Thus, according to this measure, five out of six dependencies are correct. Note that this measure is not always affected by errors in the lexical categories. For example, the dependency between *has* and *that* is considered correct, even though the gold standard analyses *has* as an auxiliary and the incorrect derivation analyses *has* as a transitive verb.

4.3. Dependency evaluation 2

The parser in Clark et al. (2002) can be used to yield a third measure. This parser uses CCG categories extended with head and dependency information and captures the "deep" dependencies inherent in cases such as raising, control, and extraction and coordination phenomena, as well as the standard local dependencies. Figure 3 is an example from Clark et al. (2002), with the links expressing dependencies. (The labels are omitted for clarity.) Note that *investors* and *managers* are both subjects of *want*, and subjects of *lock*.

An example of an extended category for the transitive verb *bought* is as follows:

(6) bought := $(\mathsf{S}_{bought}\backslash\mathsf{NP}_1)/\mathsf{NP}_2$

There are two dependency relations encoded: the subject of the transitive verb – here marked 1 – and the direct object – here marked 2. The subscript on the S category indicates that the head of the resulting sentence is *bought*. Since the argument slots in CCG categories correspond closely to the grammatical relations used by Carroll et al. (1998), this dependency evaluation is very much in the spirit of the Carroll et al. evaluation (and that of Lin (1995)).

A dependency is formally defined as a 4-tuple: $\langle h_f, f, s, h_a \rangle$, where h_f is the head word of the functor, f is the functor category (extended with dependency information), s is the argument slot, and h_a is the head word of the argument. For example, in the sentence *IBM bought Lotus*, the subject-verb dependency is as follows:

(7) $\langle \mathit{bought}, (\mathsf{S}\backslash\mathsf{NP}_1)/\mathsf{NP}_2, 1, \mathit{IBM} \rangle$

The category set used by the parser consists of 398 category types (chosen according to frequency), derived automatically from the CCGbank. Each category has been manually marked-up with head and dependency information, and at this stage we encode every argument slot as a dependency relation. In future work we may use only a subset of the argument slots.

In order to recover such dependencies from the trees produced by the normal-form parser, the Clark et al. (2002) parser is run over the trees output by the normal-form parser, tracing out the derivations and outputting the dependencies. This method can also be applied to the trees in the test set, in order to provide a set of gold standard dependency structures. Note that the marked-up categories used by the Clark et al. parser are necessary to obtain these dependencies; without this information, they cannot be derived from the local dependencies used in the first dependency evaluation.

The evaluation metrics we use are precision and recall over the dependencies (labelled and unlabelled). To obtain a point for a labelled dependency, the head, dependent, functor category, and slot must all be correct. To obtain a point for an unlabelled dependency, the head and dependent must have appeared together in some relation (in any order) in the gold standard. The dependencies obtained from the trees in Figure 2 are given in table 1. The scores for the incorrect tree are 3/6 labelled precision, 3/7 labelled recall, 5/6 unlabelled precision, and 5/7 unlabelled recall.

Gold standard
$\langle \mathit{the}, \mathsf{NP}/\mathsf{N}_1, 1, \mathit{shares} \rangle$
$\langle \mathit{that}, (\mathsf{NP}\backslash\mathsf{NP}_1)/(\mathsf{S[dcl]}_2\backslash\mathsf{NP}), 1, \mathit{shares} \rangle$
$\langle \mathit{that}, (\mathsf{NP}\backslash\mathsf{NP}_1)/(\mathsf{S[dcl]}_2\backslash\mathsf{NP}), 2, \mathit{has} \rangle$
$\langle \mathit{has}, (\mathsf{S[dcl]}\backslash NP_1)/(\mathsf{S[pt]}_2\backslash\mathsf{NP}), 1, \mathit{IBM} \rangle$
$\langle \mathit{has}, (\mathsf{S[dcl]}\backslash\mathsf{NP}_1)/(\mathsf{S[pt]}_2\backslash\mathsf{NP}), 2, \mathit{bought} \rangle$
$\langle \mathit{bought}, (\mathsf{S[pt]}\backslash\mathsf{NP}_1)/\mathsf{NP}_2, 1, \mathit{IBM} \rangle$
$\langle \mathit{bought}, (\mathsf{S[pt]}\backslash\mathsf{NP}_1)/\mathsf{NP}_2, 2, \mathit{shares} \rangle$
Example tree
$\langle \mathit{the}, \mathsf{NP}/\mathsf{N}_1, 1, \mathit{shares} \rangle$
$\langle \mathit{that}, (\mathsf{NP}\backslash\mathsf{NP}_1)/(\mathsf{S[dcl]}_2\backslash\mathsf{NP}), 1, \mathit{shares} \rangle$
$\langle \mathit{that}, (\mathsf{NP}\backslash\mathsf{NP}_1)/(\mathsf{S[dcl]}_2\backslash\mathsf{NP}), 2, \mathit{has} \rangle$
$\langle \mathit{has}, (\mathsf{S[dcl]}\backslash NP_1)/\mathsf{NP}_2, 1, \mathit{IBM} \rangle$
$\langle \mathit{has}, (\mathsf{S[dcl]}\backslash NP_1)/\mathsf{NP}_2, 2, \mathit{shares} \rangle$
$\langle \mathit{bought}, \mathsf{S[pss]}\backslash NP_1, 1, \mathit{shares} \rangle$

Table 1: Dependencies for the trees in Figure 2

Accuracy of lexical categories			
92.0%			
Parseval			
LP	LR	BP	BR
81.6%	81.9%	85.5%	85.9%
Tree dependencies			
Labelled recall		Unlabelled recall	
84.0%		90.1%	
"Deep" dependencies			
LP	LR	UP	UR
83.7%	84.2%	90.5%	91.1%

Table 2: Results for the three evaluation metrics

5. Results and discussion

The results for the three evaluation metrics on Section 23 of CCGbank are given in Table 2. BP is bracketed precision; LP is labelled precision; UP is unlabelled precision. BR, LR, UR are defined similarly for recall. The scores for each evaluation are accumulated over all sentences in the test set, rather than averaged per sentence. We also give the score for accuracy of the lexical categories.

5.1. The Parseval scores

It is hard to draw conclusions from the Parseval scores because of the difficulty in comparing results across different tree representations. Our figures are below the 88.3%/88.0% labelled precision/recall of Collins (1999). However, a direct comparison of the Parseval result with Penn Treebank parsers is not informative, even for the same set of sentences. Because Penn Treebank trees are very flat, they contain far fewer brackets than CCG derivation trees; hence the rate of crossing brackets (and bracketed precision and recall) will automatically be much lower than for a grammar which produces at most binary-branching trees. The flat trees also mean that Parseval is too lenient towards mis-attachments produced by Penn Treebank parsers (Manning and Schütze, 1999). Furthermore, the set of node labels for Penn Treebank trees and the set of CCG categories are not comparable.

Hockenmaier (2001) notes a further problem with applying Parseval metrics to CCG derivation trees. Consider verb phrases, $(\mathsf{S}\backslash\mathsf{NP})$, which can have left and right modifiers ($(\mathsf{S}\backslash\mathsf{NP})/(\mathsf{S}\backslash\mathsf{NP})$ and $(\mathsf{S}\backslash\mathsf{NP})\backslash(\mathsf{S}\backslash\mathsf{NP})$) with the following two rule instantiations:

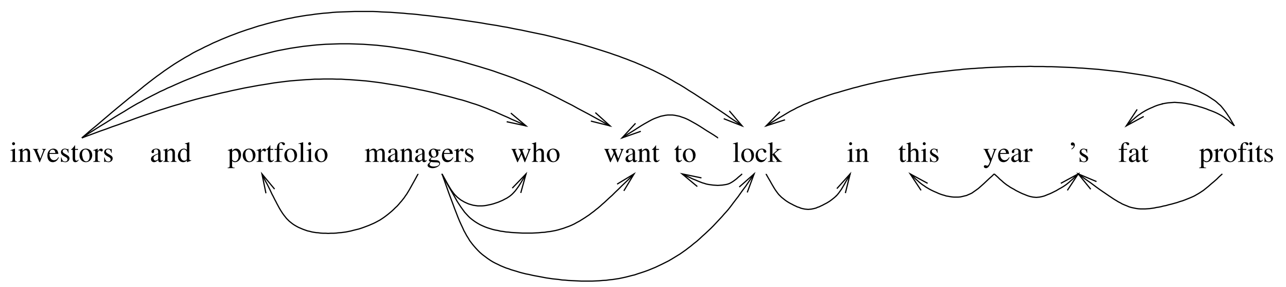


Figure 3: Example dependency structure

(8) $\mathsf{S\backslash NP} \rightarrow \mathsf{(S\backslash NP)/(S\backslash NP)}\ \ \mathsf{S\backslash NP}$
$\quad\ \mathsf{S\backslash NP} \rightarrow \mathsf{S\backslash NP}\ \ \mathsf{(S\backslash NP)\backslash(S\backslash NP)}$

For any parsing model which is defined in terms of (possibly headed or lexicalized) local trees, the following two trees are equivalent:

```
                S\NP
              /      \
(S\NP)/(S\NP)         S\NP
                    /      \
                S\NP        (S\NP)\(S\NP)

                      S\NP
                    /      \
                S\NP        (S\NP)\(S\NP)
              /      \
(S\NP)/(S\NP)         S\NP
```

This is also the case for the normal-form parser described above. A similar problem arises with coordinations/lists involving more than two conjuncts. The dependency evaluations described below do not suffer from this problem because the same dependencies are produced for each derivation.

5.2. Dependency evaluation 1

As expected, the results for the tree dependencies are higher than the Parseval scores. Unlike Parseval, the dependency measure is neutral with respect to the branching factor of the trees produced by the grammar. In particular, for a given sentence, the number of dependencies is identical to the number of words in the sentence. Since this is the same for any parser, unlabelled recovery of dependencies can be used to indicate how parsers based on different grammars compare. Note that our unlabelled figures (90.1% recall) are similar to those of Collins (90.9%).

However, a direct comparison with the labelled figures given by Collins (88.3% recall) is again problematic. First, the sets of labels are very different. In order for labelled dependencies as defined by a CCG derivation tree to be correct, complement-adjunct distinctions as well as extraction cases have to be correctly recovered. Extraction is not indicated in the trees returned by Collins' parser, and can therefore not be evaluated. Mistaking a complement daughter for an adjunct or vice versa has a much greater effect on the labelled scores for CCG than for Penn Treebank parse trees. In Collins' parser, the complement-adjunct distinction is only expressed in the label of the particular node in question. However, in CCG this can affect the entire tree below the parent – both the subtree underneath the head daughter and the subtree underneath the non-head daughter.

In addition, Collins performs the following preprocessing steps on the output of his parser and the Gold standard: all POS tags are replaced by a single token "TAG". All complement markings on the parent and head node are removed, so that one attachment decision made higher up in the tree does not affect the evaluation of its daughter. We cannot readily perform the same preprocessing steps: the choice of lexical categories can affect the tree at several levels, not just at the leaf nodes; furthermore, complement-adjunct distinctions are also encoded in all intermediate categories, not just a constituent's maximal projection.

5.3. Dependency evaluation 2

One of the advantages of a dependency-style evaluation is that the scores can be broken down by relation, as shown in Table 4, which gives scores for some of the most frequent types.[1] The table also gives some indication of the kinds of relations used in the evaluation.

The relations are defined in terms of CCG categories, which raises the question of how these compare with a more generic set such as that proposed by Carroll et al. (1998). First note that there are many more relations in our scheme: around 700 in total compared with 20 for Carroll et al. We have so many relations because each argument slot in each category (of which there are 398) encodes a separate relation.

Clearly there is room for generalisation in our scheme. For example, Carroll et al. have one relation for subjects, whereas we have a different relation for each category type encoding a subject. Examples of two categories encoding subject relations are $\mathsf{(S[dcl]\backslash NP_{\mathit{1}})/NP_{\mathit{2}}}$ and $\mathsf{(S[b]\backslash NP_{\mathit{1}})/NP_{\mathit{2}}}$. In future work we will investigate mapping our relations onto Carroll et al.'s.

One potential weakness of our evaluation (which follows from encoding all argument slots as relations) is that some relations are effectively counted more than once. For

[1]#ref is the number of dependencies with the given relation type in the gold standard; #test is the number of dependencies with the given relation type produced by the parser; LP/LR are labelled precision/recall; and the F-score is calculated as (2*LP*LR)/(LP+LR).

$\langle \mathsf{P,H,S} \rangle$	#ref	#test	LP%	LR%
$\langle \mathsf{NP,NP,NP\backslash NP} \rangle$	3,765	3,626	75.2	72.4
$\langle \mathsf{HEAD} \rangle$	2371	2367	94.5	94.4
$\langle \mathsf{NP,NP,NP[conj]} \rangle$	935	1,075	61.3	70.5
$\langle \mathsf{S[dcl]\backslash NP,S[dcl]\backslash NP,(S\backslash NP)\backslash(S\backslash NP)} \rangle$	914	905	60.9	60.3
$\langle \mathsf{S[dcl]\backslash NP,(S[dcl]\backslash NP)/NP,NP} \rangle$	880	858	86.7	84.6
$\langle \mathsf{S[pss]\backslash NP,S[pss]\backslash NP,(S\backslash NP)\backslash(S\backslash NP)} \rangle$	442	470	70.6	75.1

Table 3: Some dependency relations in evaluation 1

Functor	Slot	Category description	LP %	# test	LR %	# ref	F-score
$\mathsf{N}_X/\mathsf{N}_{X,1}$	1	*nominal modifier*	94.4	7,856	93.2	7,955	93.8
$\mathsf{NP}_X/\mathsf{N}_{X,1}$	1	*determiner*	96.7	4,548	96.4	4,566	96.5
$(\mathsf{NP}_X\backslash\mathsf{NP}_{X,1})/\mathsf{NP}_2$	2	*np modifying preposition*	82.1	2,659	81.2	2,690	81.6
$(\mathsf{NP}_X\backslash\mathsf{NP}_{X,1})/\mathsf{N}P_2$	1	*np modifying preposition*	76.0	2,449	76.2	2,443	76.1
$(\mathsf{S}_X\backslash\mathsf{NP}_Y)\backslash(\mathsf{S}_{X,1}\backslash\mathsf{NP}_Y)/\mathsf{NP}_2$	2	*vp modifying preposition*	68.7	1,327	66.1	1,379	67.4
$(\mathsf{S}_X\backslash\mathsf{NP}_Y)\backslash(\mathsf{S}_{X,1}\backslash\mathsf{NP}_Y)/\mathsf{NP}_2$	1	*vp modifying preposition*	66.2	1,247	65.0	1,271	65.6
$(\mathsf{S[dcl]}\backslash\mathsf{NP}_1)/\mathsf{NP}_2)$	1	*transitive verb*	83.2	885	82.0	898	82.6
$(\mathsf{S[dcl]}\backslash\mathsf{NP}_1)/\mathsf{NP}_2)$	2	*transitive verb*	80.3	885	78.4	907	79.3
$(\mathsf{S}_X\backslash\mathsf{NP}_Y)\backslash(\mathsf{S}_{X,1}\backslash\mathsf{NP}_Y)$	1	*adverbial modifier*	81.5	961	82.2	953	81.8
$(\mathsf{PP}/\mathsf{NP}_1)$	1	*preposition complement*	61.5	993	75.7	807	67.9
$(\mathsf{S[b]}\backslash\mathsf{NP}_1)/\mathsf{NP}_2)$	2	*infinitival transitive verb*	86.6	719	85.2	731	85.9
$(\mathsf{S[dcl]}\backslash\mathsf{NP}_{X,1})/(\mathsf{S[b]}_2\backslash\mathsf{NP}_X)$	2	*auxiliary*	97.6	631	98.6	625	98.1
$(\mathsf{S[dcl]}\backslash\mathsf{NP}_{X,1})/(\mathsf{S[b]}_2\backslash\mathsf{NP}_X)$	1	*auxiliary*	92.2	638	95.0	619	93.6
$(\mathsf{S[b]}\backslash\mathsf{NP}_1)/\mathsf{NP}_2$	1	*infinitival transitive verb*	80.6	566	83.1	549	81.8
$(\mathsf{NP}_X/\mathsf{N}_{X,1})\backslash\mathsf{NP}_2$	1	*s genitive*	96.6	472	95.2	479	95.9
$(\mathsf{NP}_X/\mathsf{N}_{X,1})\backslash\mathsf{NP}_2$	2	*s genitive*	92.5	482	95.3	468	93.9
$(\mathsf{S[dcl]}\backslash\mathsf{NP}_1)/\mathsf{S[dcl]}_2$	1	*sentential complement verb*	93.0	431	95.5	420	94.2
$(\mathsf{NP}_X\backslash\mathsf{NP}_{X,1})/(\mathsf{S[dcl]}_2\backslash\mathsf{NP}_X)$	1	*subject relative pronoun*	71.9	295	72.6	292	72.2
$(\mathsf{NP}_X\backslash\mathsf{NP}_{X,1})/(\mathsf{S[dcl]}_2\backslash\mathsf{NP}_X)$	2	*subject relative pronoun*	94.5	289	95.5	286	95.0

Table 4: Results for dependency evaluation 2 by relation; only a subset of the relations are shown

example, in the sentence *John has been eating beans*, *John* is evaluated as a subject three times: as the subject of *has*, *been* and *eating*. But if the subject of *eating* is correct in this example, then the subjects of the auxiliary verbs will be correct as well.

We would also like to make a distinction between arguments that have been extracted from a predicate, and those that are "in situ". Currently the direct object of a verb, for example, is the same relation whether it has been extracted or not. It would be useful to at least have the option to make this distinction.

5.4. Comparing the dependency evaluations

The dependencies expressed in dependency evaluation 1 are not simply a subset of the relations used in the second dependency evaluation. When the relations are broken down individually, this leads to an interesting comparison.

In dependency evaluation 2, it is possible to determine how well nominal prepositions have been recovered, whereas in dependency evaluation 1, we can only evaluate how well NP postmodifiers have been recovered.

In contrast to dependency evaluation 2, dependency evaluation 1 includes a separate relation for the head of a sentence $\langle \mathsf{HEAD} \rangle$ (assuming a single head for each sentence, including coordinate structures).

In dependency evaluation 1, it can be seen for each type of constituent whether coordination is recovered properly, e.g. $\langle \mathsf{NP,NP,NP[conj]} \rangle$. In dependency evaluation 2, coordination relations are not represented explicitly.

Some relations in dependency evaluation 1 (like the direct object of transitive declaratives, $\langle \mathsf{S[dcl]\backslash NP,(S[dcl]\backslash NP)/NP,NP} \rangle$) seem to be the same as in evaluation 2. However, in dependency evaluation 1 only non-extracted cases are taken into account.

6. Conclusion

We have presented three evaluations for a wide-coverage CCG parser. Of these, Parseval seems the least appropriate, especially if a comparison is to be made with existing Penn Treebank parsers. In an attempt to compare with the Collins parser, we adopted a dependency evaluation in which dependencies are defined in terms of local trees; however, the different labelling used in the CCG derivation tree compared to the Penn Treebank made the comparison of labelled dependencies problematic. The comparison of unlabelled dependencies was more appropriate, however.

One of the features of CCG is its analysis of long-range dependencies. In an attempt to incorporate such dependencies into the evaluation, we proposed a second dependency evaluation, in which the dependency relations are defined in terms of the CCG categories. This is closer to evaluations based on grammatical relations, although if a comparison is to be made with parsers using such an evaluation, a mapping is required between the CCG dependencies and the set of grammatical relations.

7. Acknowledgements

This research was funded by EPSRC grant GR/M96889/01 and an EPSRC studentship to the second author. We would like to thank Mark Steedman for his guidance and expert help with this work.

8. References

John Carroll, Ted Briscoe, and Antonio Sanfilippo. 1998. Parser evaluation: a survey and a new proposal. In *Proceedings of the 1st LREC Conference*, pages 447–454, Granada, Spain.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st Meeting of the NAACL*, pages 132–139, Seattle, WA.

Stephen Clark, Julia Hockenmaier, and Mark Steedman. 2002. Building deep dependency structures using a wide-coverage CCG parser. In *Proceedings of the 40th Meeting of the ACL (to appear)*, Philadelphia, PA.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Meeting of the ACL*, pages 16–23, Madrid, Spain.

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

Julia Hockenmaier and Mark Steedman. 2002a. Acquiring compact lexicalized grammars from a cleaner treebank. In *Proceedings of the Third International Conference on Language Resources and Evaluation (to appear)*, Las Palmas, Spain.

Julia Hockenmaier and Mark Steedman. 2002b. Generative models for statistical parsing with Combinatory Categorial Grammar. In *Proceedings of the 40th Meeting of the ACL (to appear)*, Philadelphia, PA.

Julia Hockenmaier. 2001. Statistical parsing for CCG with simple generative models. In *Proceedings of Student Research Workshop, 39th Meeting of the ACL*, Toulose, France.

Dekang Lin. 1995. A dependency-based method for evaluating broad-coverage parsers. In *Proceedings of IJCAI-95*, pages 1420–1425, Montreal, Canada.

Christopher Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, MA.

Mitchell Marcus, Beatrice Santorini, and Mary Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

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