Formalism-Independent Parser Evaluation with CCG and DepBank

Stephen Clark
Oxford University Computing Laboratory
Wolfson Building, Parks Road
Oxford, OX1 3QD, UK
stephen.clark@comlab.ox.ac.uk

James R. Curran
School of Information Technologies
University of Sydney
NSW 2006, Australia
james@it.usyd.edu.au

Abstract

A key question facing the parsing community is how to compare parsers which use different grammar formalisms and produce different output. Evaluating a parser on the same resource used to create it can lead to non-comparable accuracy scores and an over-optimistic view of parser performance. In this paper we evaluate a CCG parser on DepBank, and demonstrate the difficulties in converting the parser output into DepBank grammatical relations. In addition we present a method for measuring the effectiveness of the conversion, which provides an upper bound on parsing accuracy. The CCG parser obtains an F-score of 81.9% on labelled dependencies, against an upper bound of 84.8%. We compare the CCG parser against the RASP parser, outperforming RASP by over 5% overall and on the majority of dependency types.

1 Introduction

Parsers have been developed for a variety of grammar formalisms, for example HPSG (Toutanova et al., 2002; Malouf and van Noord, 2004), LFG (Kaplan et al., 2004; Cahill et al., 2004), TAG (Sarkar and Joshi, 2003), CCG (Hockenmaier and Steedman, 2002; Clark and Curran, 2004b), and variants of phrase-structure grammar (Briscoe et al., 2006), including the phrase-structure grammar implicit in the Penn Treebank (Collins, 2003; Charniak, 2000). Different parsers produce different output, for example phrase structure trees (Collins, 2003), dependency trees (Nivre and Scholz, 2004), grammatical relations (Briscoe et al., 2006), and formalism-specific dependencies (Clark and Curran, 2004b). This variety of formalisms and output creates a challenge for parser evaluation.

The majority of parser evaluations have used test sets drawn from the same resource used to develop the parser. This allows the many parsers based on the Penn Treebank, for example, to be meaningfully compared. However, there are two drawbacks to this approach. First, parser evaluations using different resources cannot be compared; for example, the Parseval scores obtained by Penn Treebank parsers cannot be compared with the dependency F-scores obtained by evaluating on the Parc Dependency Bank. Second, using the same resource for development and testing can lead to an over-optimistic view of parser performance.

In this paper we evaluate a CCG parser (Clark and Curran, 2004b) on the Briscoe and Carroll version of DepBank (Briscoe and Carroll, 2006). The CCG parser produces head-dependency relations, so evaluating the parser should simply be a matter of converting the CCG dependencies into those in DepBank. Such conversions have been performed for other parsers, including parsers producing phrase structure output (Kaplan et al., 2004; Preiss, 2003). However, we found that performing such a conversion is a time-consuming and non-trivial task.

The contributions of this paper are as follows. First, we demonstrate the considerable difficulties associated with formalism-independent parser evaluation, highlighting the problems in converting the

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 248–255,
Prague, Czech Republic, June 2007.

output of a parser from one representation to another. Second, we develop a method for measuring how effective the conversion process is, which also provides an upper bound for the performance of the parser, given the conversion process being used; this method can be adapted by other researchers to strengthen their own parser comparisons. And third, we provide the first evaluation of a wide-coverage CCG parser outside of CCGbank, obtaining impressive results on DepBank and outperforming the RASP parser (Briscoe et al., 2006) by over 5% overall and on the majority of dependency types.

2 Previous Work

The most common form of parser evaluation is to apply the Parseval metrics to phrase-structure parsers based on the Penn Treebank, and the highest reported scores are now over 90% (Bod, 2003; Charniak and Johnson, 2005). However, it is unclear whether these high scores accurately reflect the performance of parsers in applications. It has been argued that the Parseval metrics are too forgiving and that phrase structure is not the ideal representation for a gold standard (Carroll et al., 1998). Also, using the same resource for training and testing may result in the parser learning systematic errors which are present in both the training and testing material. An example of this is from CCGbank (Hockenmaier, 2003), where all modifiers in noun-noun compound constructions modify the final noun (because the Penn Treebank, from which CCGbank is derived, does not contain the necessary information to obtain the correct bracketing). Thus there are non-negligible, systematic errors in both the training and testing material, and the CCG parsers are being rewarded for following particular mistakes.

There are parser evaluation suites which have been designed to be formalism-independent and which have been carefully and manually corrected. Carroll et al. (1998) describe such a suite, consisting of sentences taken from the Susanne corpus, annotated with Grammatical Relations (GRs) which specify the syntactic relation between a head and dependent. Thus all that is required to use such a scheme, in theory, is that the parser being evaluated is able to identify heads. A similar resource — the Parc Dependency Bank (DepBank) (King et al., 2003) — has been created using sentences from the Penn Treebank. Briscoe and Carroll (2006) reannotated this resource using their GRs scheme, and used it to evaluate the RASP parser.

Kaplan et al. (2004) compare the Collins (2003) parser with the Parc LFG parser by mapping LFG F-structures and Penn Treebank parses into DepBank dependencies, claiming that the LFG parser is considerably more accurate with only a slight reduction in speed. Preiss (2003) compares the parsers of Collins (2003) and Charniak (2000), the GR finder of Buchholz et al. (1999), and the RASP parser, using the Carroll et al. (1998) gold-standard. The Penn Treebank trees of the Collins and Charniak parsers, and the GRs of the Buchholz parser, are mapped into the required GRs, with the result that the GR finder of Buchholz is the most accurate.

The major weakness of these evaluations is that there is no measure of the difficultly of the conversion process for each of the parsers. Kaplan et al. (2004) clearly invested considerable time and expertise in mapping the output of the Collins parser into the DepBank dependencies, but they also note that "This conversion was relatively straightforward for LFG structures . . . However, a certain amount of skill and intuition was required to provide a fair conversion of the Collins trees". Without some measure of the difficulty — and effectiveness — of the conversion, there remains a suspicion that the Collins parser is being unfairly penalised.

One way of providing such a measure is to convert the original gold standard on which the parser is based and evaluate that against the new gold standard (assuming the two resources are based on the same corpus). In the case of Kaplan et al. (2004), the testing procedure would include running their conversion process on Section 23 of the Penn Treebank and evaluating the output against DepBank. As well as providing some measure of the effectiveness of the conversion, this method would also provide an upper bound for the Collins parser, giving the score that a perfect Penn Treebank parser would obtain on DepBank (given the conversion process).

We perform such an evaluation for the CCG parser, with the surprising result that the upper bound on DepBank is only 84.8%, despite the considerable effort invested in developing the conversion process.

3 The CCG Parser

Clark and Curran (2004b) describes the CCG parser used for the evaluation. The grammar used by the parser is extracted from CCGbank, a CCG version of the Penn Treebank (Hockenmaier, 2003). The grammar consists of 425 lexical categories — expressing subcategorisation information — plus a small number of combinatory rules which combine the categories (Steedman, 2000). A supertagger first assigns lexical categories to the words in a sentence, which are then combined by the parser using the combinatory rules and the CKY algorithm. A log-linear model scores the alternative parses. We use the normal-form model, which assigns probabilities to single derivations based on the normal-form derivations in CCGbank. The features in the model are defined over local parts of the derivation and include word-word dependencies. A packed chart representation allows efficient decoding, with the Viterbi algorithm finding the most probable derivation.

The parser outputs predicate-argument dependencies defined in terms of CCG lexical categories. More formally, a CCG predicate-argument dependency is a 5-tuple: $\langle h_f, f, s, h_a, l\rangle$, where h_f is the lexical item of the lexical category expressing the dependency relation; f is the lexical category; s is the argument slot; h_a is the head word of the argument; and l encodes whether the dependency is long-range. For example, the dependency encoding *company* as the object of *bought* (as in *IBM bought the company*) is represented as follows:

$$\langle \textit{bought},\ (S\backslash NP_1)/NP_2,\ 2,\ \textit{company},\ -\rangle \qquad (1)$$

The lexical category $(S\backslash NP_1)/NP_2$ is the category of a transitive verb, with the first argument slot corresponding to the subject, and the second argument slot corresponding to the direct object. The final field indicates the nature of any long-range dependency; in (1) the dependency is local.

The predicate-argument dependencies — including long-range dependencies — are encoded in the lexicon by adding head and dependency annotation to the lexical categories. For example, the expanded category for the control verb *persuade* is $(((S[dcl]_{persuade}\backslash NP_1)/(S[to]_2\backslash NP_X))/NP_{X,3})$. Numerical subscripts on the argument categories represent dependency relations; the head of the final declarative sentence is *persuade*; and the head of the infinitival complement's subject is identified with the head of the object, using the variable *X*, as in standard unification-based accounts of control.

Previous evaluations of CCG parsers have used the predicate-argument dependencies from CCGbank as a test set (Hockenmaier and Steedman, 2002; Clark and Curran, 2004b), with impressive results of over 84% F-score on labelled dependencies. In this paper we reinforce the earlier results with the first evaluation of a CCG parser outside of CCGbank.

4 Dependency Conversion to DepBank

For the gold standard we chose the version of DepBank reannotated by Briscoe and Carroll (2006), consisting of 700 sentences from Section 23 of the Penn Treebank. The B&C scheme is similar to the original DepBank scheme (King et al., 2003), but overall contains less grammatical detail; Briscoe and Carroll (2006) describes the differences. We chose this resource for the following reasons: it is publicly available, allowing other researchers to compare against our results; the GRs making up the annotation share some similarities with the predicate-argument dependencies output by the CCG parser; and we can directly compare our parser against a non-CCG parser, namely the RASP parser. We chose not to use the corpus based on the Susanne corpus (Carroll et al., 1998) because the GRs are less like the CCG dependencies; the corpus is not based on the Penn Treebank, making comparison more difficult because of tokenisation differences, for example; and the latest results for RASP are on DepBank.

The GRs are described in Briscoe and Carroll (2006) and Briscoe et al. (2006). Table 1 lists the GRs used in the evaluation. As an example, the sentence *The parent sold Imperial* produces three GRs: `(det parent The)`, `(ncsubj sold parent _)` and `(dobj sold Imperial)`. Note that some GRs — in this example `ncsubj` — have a *subtype slot*, giving extra information. The subtype slot for `ncsubj` is used to indicate passive subjects, with the null value "_" for active subjects and `obj` for passive subjects. Other subtype slots are discussed in Section 4.2.

The CCG dependencies were transformed into GRs in two stages. The first stage was to create a mapping between the CCG dependencies and the

GR	description
`conj`	coordinator
`aux`	auxiliary
`det`	determiner
`ncmod`	non-clausal modifier
`xmod`	unsaturated predicative modifier
`cmod`	saturated clausal modifier
`pmod`	PP modifier with a PP complement
`ncsubj`	non-clausal subject
`xsubj`	unsaturated predicative subject
`csubj`	saturated clausal subject
`dobj`	direct object
`obj2`	second object
`iobj`	indirect object
`pcomp`	PP which is a PP complement
`xcomp`	unsaturated VP complement
`ccomp`	saturated clausal complement
`ta`	textual adjunct delimited by punctuation

Table 1: GRs in B&C's annotation of DepBank

CCG lexical category	slot	GR
$(S[dcl]\backslash NP_1)/NP_2$	1	`(ncsubj %l %f _)`
$(S[dcl]\backslash NP_1)/NP_2$	2	`(dobj %l %f)`
$(S\backslash NP)/(S\backslash NP)_1$	1	`(ncmod _ %f %l)`
$(NP\backslash NP_1)/NP_2$	1	`(ncmod _ %f %l)`
$(NP\backslash NP_1)/NP_2$	2	`(dobj %l %f)`
$NP[nb]/N_1$	1	`(det %f %l)`
$(NP\backslash NP_1)/(S[pss]\backslash NP)_2$	1	`(xmod _ %f %l)`
$(NP\backslash NP_1)/(S[pss]\backslash NP)_2$	2	`(xcomp _ %l %f)`
$((S\backslash NP)\backslash(S\backslash NP)_1)/S[dcl]_2$	1	`(cmod _ %f %l)`
$((S\backslash NP)\backslash(S\backslash NP)_1)/S[dcl]_2$	2	`(ccomp _ %l %f)`
$((S[dcl]\backslash NP_1)/NP_2)/NP_3$	2	`(obj2 %l %f)`
$(S[dcl]\backslash NP_1)/(S[b]\backslash NP)_2$	2	`(aux %f %l)`

Table 2: Examples of the dependency mapping

GRs. This involved mapping each argument slot in the 425 lexical categories in the CCG lexicon onto a GR. In the second stage, the GRs created from the parser output were post-processed to correct some of the obvious remaining differences between the CCG and GR representations.

In the process of performing the transformation we encountered a methodological problem: without looking at examples it was difficult to create the mapping and impossible to know whether the two representations were converging. Briscoe et al. (2006) split the 700 sentences in DepBank into a test and development set, but the latter only consists of 140 sentences which was not enough to reliably create the transformation. There are some development files in the RASP release which provide examples of the GRs, which were used when possible, but these only cover a subset of the CCG lexical categories.

Our solution to this problem was to convert the gold standard dependencies from CCGbank into GRs and use these to develop the transformation. So we did inspect the annotation in DepBank, and compared it to the transformed CCG dependencies, but only the *gold-standard* CCG dependencies. Thus the parser output was never used during this process. We also ensured that the dependency mapping and the post processing are general to the GRs scheme and not specific to the test set or parser.

4.1 Mapping the CCG dependencies to GRs

Table 2 gives some examples of the mapping; `%l` indicates the word associated with the lexical category and `%f` is the head of the constituent filling the argument slot. Note that the order of `%l` and `%f` varies according to whether the GR represents a complement or modifier, in line with the Briscoe and Carroll annotation. For many of the CCG dependencies, the mapping into GRs is straightforward. For example, the first two rows of Table 2 show the mapping for the transitive verb category $(S[dcl]\backslash NP_1)/NP_2$: argument slot 1 is a non-clausal subject and argument slot 2 is a direct object.

Creating the dependency transformation is more difficult than these examples suggest. The first problem is that the mapping from CCG dependencies to GRs is many-to-many. For example, the transitive verb category $(S[dcl]\backslash NP)/NP$ applies to the copula in sentences like *Imperial Corp. is the parent of Imperial Savings & Loan*. With the default annotation, the relation between *is* and *parent* would be `dobj`, whereas in DepBank the argument of the copula is analysed as an `xcomp`. Table 3 gives some examples of how we attempt to deal with this problem. The constraint in the first example means that, whenever the word associated with the transitive verb category is a form of *be*, the second argument is `xcomp`, otherwise the default case applies (in this case `dobj`). There are a number of categories with similar constraints, checking whether the word associated with the category is a form of *be*.

The second type of constraint, shown in the third line of the table, checks the lexical category of the word filling the argument slot. In this example, if the lexical category of the preposition is PP/NP, then the second argument of $(S[dcl]\backslash NP)/PP$ maps to `iobj`; thus in *The loss stems from several factors* the relation between the verb and preposition is `(iobj stems from)`. If the lexical category of

CCG lexical category	slot	GR	constraint	example
$(S[dcl]\backslash NP_1)/NP_2$	2	`(xcomp _ %l %f)`	word=be	The parent *is Imperial*
		`(dobj %l %f)`		The parent *sold Imperial*
$(S[dcl]\backslash NP_1)/PP_2$	2	`(iobj %l %f)`	cat=PP/NP	The loss *stems from* several factors
		`(xcomp _ %l %f)`	cat=$PP/(S[ng]\backslash NP)$	The future *depends on* building ties
$(S[dcl]\backslash NP_1)/(S[to]\backslash NP)_2$	2	`(xcomp %f %l %k)`	cat=$(S[to]\backslash NP)/(S[b]\backslash NP)$	*wants to wean* itself away from

Table 3: Examples of the many-to-many nature of the CCG dependency to GRs mapping, and a ternary GR

the preposition is $PP/(S[ng]\backslash NP)$, then the GR is `xcomp`; thus in *The future depends on building ties* the relation between the verb and preposition is `(xcomp _ depends on)`. There are a number of CCG dependencies with similar constraints, many of them covering the `iobj`/`xcomp` distinction.

The second difficulty is that not all the GRs are binary relations, whereas the CCG dependencies are all binary. The primary example of this is to-infinitival constructions. For example, in the sentence *The company wants to wean itself away from expensive gimmicks*, the CCG parser produces two dependencies relating *wants*, *to* and *wean*, whereas there is only one GR: `(xcomp to wants wean)`. The final row of Table 3 gives an example. We implement this constraint by introducing a `%k` variable into the GR template which denotes the argument of the category in the constraint column (which, as before, is the lexical category of the word filling the argument slot). In the example, the current category is $(S[dcl]\backslash NP_1)/(S[to]\backslash NP)_2$, which is associated with *wants*; this combines with $(S[to]\backslash NP)/(S[b]\backslash NP)$, associated with *to*; and the argument of $(S[to]\backslash NP)/(S[b]\backslash NP)$ is *wean*. The `%k` variable allows us to look beyond the arguments of the current category when creating the GRs.

A further difficulty is that the head passing conventions differ between DepBank and CCGbank. By *head passing* we mean the mechanism which determines the heads of constituents and the mechanism by which words become arguments of long-range dependencies. For example, in the sentence *The group said it would consider withholding royalty payments*, the DepBank and CCGbank annotations create a dependency between *said* and the following clause. However, in DepBank the relation is between *said* and *consider*, whereas in CCGbank the relation is between *said* and *would*. We fixed this problem by defining the head of *would consider* to be *consider* rather than *would*, by changing the annotation of all the relevant lexical categories in the CCG lexicon (mainly those creating `aux` relations).

There are more subject relations in CCGbank than DepBank. In the previous example, CCGbank has a subject relation between *it* and *consider*, and also *it* and *would*, whereas DepBank only has the relation between *it* and *consider*. In practice this means ignoring a number of the subject dependencies output by the CCG parser.

Another example where the dependencies differ is the treatment of relative pronouns. For example, in *Sen. Mitchell, who had proposed the streamlining*, the subject of *proposed* is *Mitchell* in CCGbank but *who* in DepBank. Again, we implemented this change by fixing the head annotation in the lexical categories which apply to relative pronouns.

4.2 Post processing of the GR output

To obtain some idea of whether the schemes were converging, we performed the following oracle experiment. We took the CCG derivations from CCGbank corresponding to the sentences in DepBank, and forced the parser to produce gold-standard derivations, outputting the newly created GRs. Treating the DepBank GRs as a gold-standard, and comparing these with the CCGbank GRs, gave precision and recall scores of only 76.23% and 79.56% respectively (using the RASP evaluation tool). Thus given the current mapping, the perfect CCGbank parser would achieve an F-score of only 77.86% when evaluated against DepBank.

On inspecting the output, it was clear that a number of general rules could be applied to bring the schemes closer together, which was implemented as a post-processing script. The first set of changes deals with coordination. One significant difference between DepBank and CCGbank is the treatment of coordinations as arguments. Consider the example *The president and chief executive officer said the loss stems from several factors.* For both DepBank and the transformed CCGbank there are two `conj` GRs arising from the coordination: `(conj and president)` and `(conj and officer)`. The difference arises in the subject of *said*: in DepBank the subject is *and*: `(ncsubj said and _)`, whereas in CCGbank there are two subjects: `(ncsubj said president _)` and `(ncsubj said officer _)`. We deal with this difference by replacing any pairs of GRs which differ only in their arguments, and where the arguments are coordinated items, with a single GR containing the coordination term as the argument.

Ampersands are a frequently occurring problem in WSJ text. For example, the CCGbank analysis of *Standard & Poor's index* assigns the lexical category N/N to both *Standard* and *&*, treating them as modifiers of *Poor*, whereas DepBank treats *&* as a coordinating term. We fixed this by creating `conj` GRs between any *&* and the two words either side; removing the modifier GR between the two words; and replacing any GRs in which the words either side of the *&* are arguments with a single GR in which *&* is the argument.

The `ta` relation, which identifies text adjuncts delimited by punctuation, is difficult to assign correctly to the parser output. The simple punctuation rules used by the parser do not contain enough information to distinguish between the various cases of `ta`. Thus the only rule we have implemented, which is somewhat specific to the newspaper genre, is to replace GRs of the form `(cmod _ say arg)` with `(ta quote arg say)`, where `say` can be any of *say*, *said* or *says*. This rule applies to only a small subset of the `ta` cases but has high enough precision to be worthy of inclusion.

A common source of error is the distinction between `iobj` and `ncmod`, which is not surprising given the difficulty that human annotators have in distinguishing arguments and adjuncts. There are many cases where an argument in DepBank is an adjunct in CCGbank, and vice versa. The only change we have made is to turn all `ncmod` GRs with *of* as the modifier into `iobj` GRs (unless the `ncmod` is a partitive predeterminer). This was found to have high precision and applies to a large number of cases.

There are some dependencies in CCGbank which do not appear in DepBank. Examples include any dependencies in which a punctuation mark is one of the arguments; these were removed from the output.

We attempt to fill the subtype slot for some GRs. The subtype slot specifies additional information about the GR; examples include the value `obj` in a passive `ncsubj`, indicating that the subject is an underlying object; the value `num` in `ncmod`, indicating a numerical quantity; and `prt` in `ncmod` to indicate a verb particle. The passive case is identified as follows: any lexical category which starts $S[pss]\backslash NP$ indicates a passive verb, and we also mark any verbs POS tagged VBN and assigned the lexical category N/N as passive. Both these rules have high precision, but still leave many of the cases in DepBank unidentified. The numerical case is identified using two rules: the `num` subtype is added if any argument in a GR is assigned the lexical category $N/N[num]$, and if any of the arguments in an `ncmod` is POS tagged CD. `prt` is added to an `ncmod` if the modifiee has any of the verb POS tags and if the modifier has POS tag RP.

The final columns of Table 4 show the accuracy of the transformed gold-standard CCGbank dependencies when compared with DepBank; the simple post-processing rules have increased the F-score from 77.86% to 84.76%. This F-score is an *upper bound* on the performance of the CCG parser.

5 Results

The results in Table 4 were obtained by parsing the sentences from CCGbank corresponding to those in the 560-sentence test set used by Briscoe et al. (2006). We used the CCGbank sentences because these differ in some ways to the original Penn Treebank sentences (there are no quotation marks in CCGbank, for example) and the parser has been trained on CCGbank. Even here we experienced some unexpected difficulties, since some of the tokenisation is different between DepBank and CCGbank and there are some sentences in DepBank which have been significantly shortened compared to the original Penn Treebank sentences. We modified the CCGbank sentences — and the CCGbank analyses since these were used for the oracle experiments — to be as close to the DepBank sentences as possible. All the results were obtained using the RASP evaluation scripts, with the results for the RASP parser taken from Briscoe et al. (2006). The results for CCGbank were obtained using the oracle method described above.

Relation	RASP Prec	RASP Rec	RASP F	CCG parser Prec	CCG parser Rec	CCG parser F	CCGbank Prec	CCGbank Rec	CCGbank F	# GRs
aux	93.33	91.00	**92.15**	94.20	89.25	91.66	96.47	90.33	93.30	400
conj	72.39	72.27	72.33	79.73	77.98	**78.84**	83.07	80.27	81.65	595
ta	42.61	51.37	**46.58**	52.31	11.64	19.05	62.07	12.59	20.93	292
det	87.73	90.48	89.09	95.25	95.42	**95.34**	97.27	94.09	95.66	1 114
ncmod	75.72	69.94	72.72	75.75	79.27	**77.47**	78.88	80.64	79.75	3 550
xmod	53.21	46.63	**49.70**	43.46	52.25	47.45	56.54	60.67	58.54	178
cmod	45.95	30.36	36.56	51.50	61.31	**55.98**	64.77	69.09	66.86	168
pmod	30.77	33.33	**32.00**	0.00	0.00	0.00	0.00	0.00	0.00	12
ncsubj	79.16	67.06	72.61	83.92	75.92	**79.72**	88.86	78.51	83.37	1 354
xsubj	33.33	28.57	**30.77**	0.00	0.00	0.00	50.00	28.57	36.36	7
csubj	12.50	50.00	**20.00**	0.00	0.00	0.00	0.00	0.00	0.00	2
dobj	83.63	79.08	81.29	87.03	89.40	**88.20**	92.11	90.32	91.21	1 764
obj2	23.08	30.00	26.09	65.00	65.00	**65.00**	66.67	60.00	63.16	20
iobj	70.77	76.10	73.34	77.60	70.04	**73.62**	83.59	69.81	76.08	544
xcomp	76.88	77.69	**77.28**	76.68	77.69	77.18	80.00	78.49	79.24	381
ccomp	46.44	69.42	55.55	79.55	72.16	**75.68**	80.81	76.31	78.49	291
pcomp	72.73	66.67	**69.57**	0.00	0.00	0.00	0.00	0.00	0.00	24
macroaverage	62.12	63.77	62.94	65.61	63.28	**64.43**	71.73	65.85	68.67	
microaverage	77.66	74.98	76.29	82.44	81.28	**81.86**	86.86	82.75	84.76	

Table 4: Accuracy on DepBank. F-score is the balanced harmonic mean of precision (P) and recall (R): $2PR/(P+R)$. # GRs is the number of GRs in DepBank.

The CCG parser results are based on automatically assigned POS tags, using the Curran and Clark (2003) tagger. The coverage of the parser on DepBank is 100%. For a GR in the parser output to be correct, it has to match the gold-standard GR exactly, including any subtype slots; however, it is possible for a GR to be incorrect at one level but correct at a *subsuming level*.[1] For example, if an ncmod GR is incorrectly labelled with xmod, but is otherwise correct, it will be correct for all levels which subsume both ncmod and xmod, for example mod. The micro-averaged scores are calculated by aggregating the counts for all the relations in the hierarchy, including the subsuming relations; the macro-averaged scores are the mean of the individual scores for each relation (Briscoe et al., 2006).

The results show that the performance of the CCG parser is higher than RASP overall, and also higher on the majority of GR types (especially the more frequent types). RASP uses an unlexicalised parsing model and has not been tuned to newspaper text. On the other hand it has had many years of development; thus it provides a strong baseline for this test set. The overall F-score for the CCG parser, 81.86%, is only 3 points below that for CCGbank, which provides an upper bound for the CCG parser (given the conversion process being used).

[1]The GRs are arranged in a hierarchy, with those in Table 1 at the leaves; a small number of more general GRs subsume these (Briscoe and Carroll, 2006).

6 Conclusion

A contribution of this paper has been to highlight the difficulties associated with cross-formalism parser comparison. Note that the difficulties are not unique to CCG, and many would apply to any cross-formalism comparison, especially with parsers using automatically extracted grammars. Parser evaluation has improved on the original Parseval measures (Carroll et al., 1998), but the challenge remains to develop a representation and evaluation suite which can be easily applied to a wide variety of parsers and formalisms. Despite the difficulties, we have given the first evaluation of a CCG parser outside of CCGbank, outperforming the RASP parser by over 5% overall and on the majority of dependency types.

Can the CCG parser be compared with parsers other than RASP? Briscoe and Carroll (2006) give a rough comparison of RASP with the Parc LFG parser on the different versions of DepBank, obtaining similar results overall, but they acknowledge that the results are not strictly comparable because of the different annotation schemes used. Comparison with Penn Treebank parsers would be difficult because, for many constructions, the Penn Treebank trees and

CCG derivations are different shapes, and reversing the mapping Hockenmaier used to create CCGbank would be very difficult. Hence we challenge other parser developers to map their own parse output into the version of DepBank used here.

One aspect of parser evaluation not covered in this paper is efficiency. The CCG parser took only 22.6 seconds to parse the 560 sentences in DepBank, with the accuracy given earlier. Using a cluster of 18 machines we have also parsed the entire Gigaword corpus in less than five days. Hence, we conclude that accurate, large-scale, linguistically-motivated NLP is now practical with CCG.

Acknowledgements

We would like to thanks the anonymous reviewers for their helpful comments. James Curran was funded under ARC Discovery grants DP0453131 and DP0665973.

References

Rens Bod. 2003. An efficient implementation of a new DOP model. In *Proceedings of the 10th Meeting of the EACL*, pages 19–26, Budapest, Hungary.

Ted Briscoe and John Carroll. 2006. Evaluating the accuracy of an unlexicalized statistical parser on the PARC DepBank. In *Proceedings of the Poster Session of COLING/ACL-06*, Sydney, Australia.

Ted Briscoe, John Carroll, and Rebecca Watson. 2006. The second release of the RASP system. In *Proceedings of the Interactive Demo Session of COLING/ACL-06*, Sydney, Australia.

Sabine Buchholz, Jorn Veenstra, and Walter Daelemans. 1999. Cascaded grammatical relation assignment. In *Proceedings of EMNLP/VLC-99*, pages 239–246, University of Maryland, June 21-22.

A. Cahill, 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 Meeting of the ACL*, pages 320–327, Barcelona, Spain.

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 and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting of the ACL*, University of Michigan, Ann Arbor.

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

Stephen Clark and James R. Curran. 2004a. The importance of supertagging for wide-coverage CCG parsing. In *Proceedings of COLING-04*, pages 282–288, Geneva, Switzerland.

Stephen Clark and James R. Curran. 2004b. Parsing the WSJ using CCG and log-linear models. In *Proceedings of the 42nd Meeting of the ACL*, pages 104–111, Barcelona, Spain.

Michael Collins. 2003. Head-driven statistical models for natural language parsing. *Computational Linguistics*, 29(4):589–637.

James R. Curran and Stephen Clark. 2003. Investigating GIS and smoothing for maximum entropy taggers. In *Proceedings of the 10th Meeting of the EACL*, pages 91–98, Budapest, Hungary.

Julia Hockenmaier and Mark Steedman. 2002. Generative models for statistical parsing with Combinatory Categorial Grammar. In *Proceedings of the 40th Meeting of the ACL*, pages 335–342, Philadelphia, PA.

Julia Hockenmaier. 2003. *Data and Models for Statistical Parsing with Combinatory Categorial Grammar*. Ph.D. thesis, University of Edinburgh.

Ron Kaplan, Stefan Riezler, Tracy H. King, John T. Maxwell III, Alexander Vasserman, and Richard Crouch. 2004. Speed and accuracy in shallow and deep stochastic parsing. In *Proceedings of the HLT Conference and the 4th NAACL Meeting (HLT-NAACL'04)*, Boston, MA.

Tracy H. King, Richard Crouch, Stefan Riezler, Mary Dalrymple, and Ronald M. Kaplan. 2003. The PARC 700 Dependency Bank. In *Proceedings of the LINC-03 Workshop*, Budapest, Hungary.

Robert Malouf and Gertjan van Noord. 2004. Wide coverage parsing with stochastic attribute value grammars. In *Proceedings of the IJCNLP-04 Workshop: Beyond shallow analyses - Formalisms and statistical modeling for deep analyses*, Hainan Island, China.

Joakim Nivre and Mario Scholz. 2004. Deterministic dependency parsing of English text. In *Proceedings of COLING-2004*, pages 64–70, Geneva, Switzerland.

Judita Preiss. 2003. Using grammatical relations to compare parsers. In *Proceedings of the 10th Meeting of the EACL*, pages 291–298, Budapest, Hungary.

Anoop Sarkar and Aravind Joshi. 2003. Tree-adjoining grammars and its application to statistical parsing. In Rens Bod, Remko Scha, and Khalil Sima'an, editors, *Data-oriented parsing*. CSLI.

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

Kristina Toutanova, Christopher Manning, Stuart Shieber, Dan Flickinger, and Stephan Oepen. 2002. Parse disambiguation for a rich HPSG grammar. In *Proceedings of the First Workshop on Treebanks and Linguistic Theories*, pages 253–263, Sozopol, Bulgaria.