Comparing the Accuracy of CCG and Penn Treebank Parsers

Stephen Clark
University of Cambridge
Computer Laboratory
15 JJ Thomson Avenue, Cambridge, UK
stephen.clark@cl.cam.ac.uk

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

Abstract

We compare the CCG parser of Clark and Curran (2007) with a state-of-the-art Penn Treebank (PTB) parser. An accuracy comparison is performed by converting the CCG derivations into PTB trees. We show that the conversion is extremely difficult to perform, but are able to fairly compare the parsers on a representative subset of the PTB test section, obtaining results for the CCG parser that are statistically no different to those for the Berkeley parser.

1 Introduction

There are a number of approaches emerging in statistical parsing. The first approach, which began in the mid-90s and now has an extensive literature, is based on the Penn Treebank (PTB) parsing task: inferring skeletal phrase-structure trees for unseen sentences of the WSJ, and evaluating accuracy according to the Parseval metrics. Collins (1999) is a seminal example. The second approach is to apply statistical methods to parsers based on linguistic formalisms, such as HPSG, LFG, TAG, and CCG, with the grammar being defined manually or extracted from a formalism-specific treebank. Evaluation is typically performed by comparing against predicate-argument structures extracted from the treebank, or against a test set of manually annotated grammatical relations (GRs). Examples of this approach include Riezler et al. (2002), Miyao and Tsujii (2005), Briscoe and Carroll (2006), and Clark and Curran (2007).[1]

Despite the many examples from both approaches, there has been little comparison across the two groups, which we refer to as PTB parsing and formalism-based parsing, respectively. The PTB parser we use for comparison is the publicly available Berkeley parser (Petrov and Klein, 2007). The formalism-based parser we use is the CCG parser of Clark and Curran (2007), which is based on CCGbank (Hockenmaier and Steedman, 2007), a CCG version of the Penn Treebank. We compare this parser with a PTB parser because both are derived from the same original source, and both produce phrase-structure in some form or another; the interesting question is whether anything is gained by converting the PTB into CCG.[2]

The comparison focuses on accuracy and is performed by converting CCG derivations into PTB phrase-structure trees. A contribution of this paper is to demonstrate the difficulty of mapping from a grammatical resource based on the PTB back to the PTB, and we also comment on the (non-)suitability of the PTB as a general formalism-independent evaluation resource. A second contribution is to provide the first accuracy comparison of the CCG parser with a PTB parser, obtaining competitive scores for the CCG parser on a representative subset of the PTB test sections. It is important to note that the purpose of this evaluation is *comparison* with a PTB parser, rather than evaluation of the CCG parser *per se*. The CCG parser has been extensively evaluated elsewhere (Clark and Curran, 2007), and arguably GRs or predicate-argument structures provide a more suitable test set for the CCG parser than PTB phrase-structure trees.

2 The CCG to PTB Conversion

There has been much recent work in attempting to convert native parser output into alternative representations for evaluation purposes, e.g. (Clark and Curran, 2007; Matsuzaki and Tsujii, 2008). The conclusion is that such conversions are surprisingly difficult. Clark and Curran (2007)

[1] A third approach is dependency parsing, but we restrict the comparison in this paper to phrase-structure parsers.

[2] Since this short paper reports a small, focused research contribution, we refer readers to Clark and Curran (2007) and Petrov and Klein (2007) for details of the two parsers.

Proceedings of the ACL-IJCNLP 2009 Conference Short Papers, pages 53–56,
Suntec, Singapore, 4 August 2009. ©2009 ACL and AFNLP

shows that converting *gold-standard* CCG derivations into the GRs in DepBank resulted in an F-score of only 85%; hence the upper bound on the performance of the CCG parser, using this evaluation scheme, was only 85%. Given that the current best scores for the PTB parsing task are over 90%, any loss from the conversion process needs to be considered carefully if a fair comparison with PTB parsers is to be achieved.

CCGbank was derived from the PTB, and so it might be considered that converting back to the PTB would be a relatively easy task, by essentially reversing the mapping Hockenmaier and Steedman (2007) used to create CCGbank. However, there are a number of differences between the two treebanks which make the conversion back far from trivial. First, the corresponding derivations in the treebanks are not isomorphic: a CCG derivation is not simply a relabelling of the nodes in the PTB tree; there are many constructions, such as coordination and control structures, where the trees are a different shape, as well as having different labels. It is important to realise that Hockenmaier and Steedman (2007) invested a significant amount of time and effort in creating the mapping. Second, some of the labels in the PTB do not appear in CCGbank, for example the QP label, and these must be added back in; however, developing rules to insert these labels in the right places is a far from trivial task.

There were two approaches we considered for the conversion. One possibility is to associate PTB tree structures with CCG lexical categories, and combine the trees together in step with the category combinations in a CCG derivation — in much the same way that an LTAG has elementary trees in the lexicon which are combined using the substitution and adjunction rules of TAG. The second approach is to associate conversion rules with each local tree – i.e. a parent and one or two child nodes – which appears in the CCGbank data.[3] In this paper we took the second approach.

2.1 Conversion Schemas

There are three types of conversion schema: schemas which introduce nodes for lexical items; schemas which insert or elide PTB nodes for unary rules and type-raising; and schemas which can perform arbitrary manipulation of generated PTB subtrees for binary CCG rule instances. Examples of these schemas are shown in Table 1. The primary operations in the binary schema are inserting and attaching. Inserting a new node, for example using the schema (S l r), creates a new S node dominating both the left and right children of a binary rule. The attaching schema can attach the left node under the right node (>); or the right node under the left node (<).

TYPE	RULE	SCHEMA
lexical	NP	NP
lexical	$NP[nb]/N$	–
lexical	$(S[dcl]\backslash NP)/NP$	VP
unary	$S[dcl] \longrightarrow NP\backslash NP$	(SBAR l)
type-raising	$PP \longrightarrow (S\backslash NP)\backslash((S\backslash NP)/PP)$	l
binary	$NP[nb]/N\ N \longrightarrow NP[nb]$	>
binary	$NP\ S[dcl]\backslash NP \longrightarrow S[dcl]$	(S l r)
binary	$NP/(S[dcl]\backslash NP)\ S[dcl]\backslash NP \longrightarrow NP$	(SBAR l (S r))

Table 1: Example conversion schemas

The lexical categories NP and $(S[dcl]\backslash NP)/NP$ (shown in Table 1) introduce the PTB nodes NP and VP, respectively, while other lexical categories such as $NP[nb]/N$ introduce no extra nodes. Some unary rules introduce nodes, such as SBAR for the reduced relative case, whilst others, such as the type-raised PP, do not. Finally, binary schemas may create no new nodes (e.g. when a determiner is attached to an existing NP), or one or more nodes (e.g. an extra S node is created when a verb phrase finds its subject).

A PTB tree is built from a CCG derivation by running over the derivation in a bottom-up fashion and applying these schemas to the local trees in the derivation.

2.2 Schema development

The schemas were developed by manual inspection using section 00 of CCGbank and the PTB as a development set, following the oracle methodology of Clark and Curran (2007), in which gold-standard derivations from CCGbank are converted to the new representation and compared with the gold standard for that representation. As well as giving an idea of the difficulty, and success, of the conversion, the resulting numbers provide an up-

[3] Another possible approach has been taken by Matsuzaki and Tsujii (2008), who convert HPSG analyses from a grammar automatically extracted from the PTB back into the PTB. They treat the problem as one of translation, learning a synchronous grammar to perform the mapping.

SECTION	P	R	F	COMP
00 (all)	93.37	95.15	94.25	39.68
00 (len $\leq$ 40)	94.11	95.65	94.88	42.11
23 (all)	93.68	95.13	94.40	39.93
23 (len $\leq$ 40)	93.75	95.23	94.48	42.15

Table 2: Oracle conversion evaluation

per bound on the performance of the CCG parser. The test set, section 23, was not inspected at any stage in the development of the schemas.

In total, we annotated 32 unary and 776 binary rule instances (of the possible 2853 instances) with conversion schemas, and 162 of the 425 lexical categories. We also implemented a small number of default catch-all cases for the general CCG combinatory rules and for the rules dealing with punctuation, which allowed most of the 2853 rule instances to be covered. Considerable time and effort was invested in the creation of these schemas.

The oracle conversion results from the gold standard CCGbank to the PTB for section 00 and 23 are shown in Table 2. The numbers are bracketing precision, recall, F-score and complete sentence matches, using the EVALB evaluation script. Note that these figures provide an upper bound on the performance of the CCG parser using EVALB, given the current conversion process.

The importance of this upper bound should not be underestimated, when the evaluation framework is such that incremental improvements of a few tenths of a percent are routinely presented as improving the state-of-the-art, as is the case with the Parseval metrics. The fact that the upper bound here is less than 95% shows that it is not possible to fairly evaluate the CCG parser on the complete test set. Even an upper bound of around 98%, which is achieved by Matsuzaki and Tsujii (2008), is not sufficient, since this guarantees a loss of at least 2%.[4]

3 Evaluation

The Berkeley parser (Petrov and Klein, 2007) provides performance close to the state-of-the-art for the PTB parsing task, with reported F-scores of around 90%. Since the oracle score for CCGbank is less than 95%, it would not be a fair comparison to use the complete test set. However, there are a number of sentences which are correct, or almost correct, according to EVALB after the conversion, and we are able to use those for a fair comparison.

Table 3 gives the EVALB results for the CCG parser on various subsets of section 00 of the PTB. The first row shows the results on only those sentences which the conversion process can convert sucessfully (as measured by converting gold-standard CCGbank derivations and comparing with PTB trees; although, to be clear, the scores are for the CCG parser on those sentences). As can be seen from the scores, these sentences form a slightly easier subset than the full section 00, but this is a subset which can be used for a fair comparison against the Berkeley parser, since the conversion process is not lossy for this subset.

The second row shows the scores on those sentences for which the conversion process was somewhat lossy, but when the gold-standard CCGbank derivations are converted, the oracle F-measure is greater than 95%. The third row is similar, but for sentences for which the oracle F-score is geater than 92%. The final row is for the whole of section 00. The UB column gives the upper bound on the accuracy of the CCG parser. Results are calculated using both gold standard and automatically assigned POS tags; # is the number of sentences in the sample, and the % column gives the sample size as a percentage of the whole section.

We compare the CCG parser to the Berkeley parser using the accurate mode of the Berkeley parser, together with the model supplied with the publicly available version. Table 3 gives the results for Section 23, comparing the CCG and Berkeley parsers. The projected columns give the projected scores for the CCG parser, if it performed at the same accuracy level for those sentences which could not be converted successfully. The purpose of this column is to obtain an approximation of the CCG parser score for a perfect conversion process.[5] The results in bold are those which we consider to be a fair comparison against the Berkeley parser. The difference in scores is not statistically significant at p=0.05 (using Dan Bikel's stratified shuffling test).

One possible objection to this comparison is that the subset for which we have a fair compar-

[4] The higher upper bound achieved by Matsuzaki and Tsujii (2008) could be due to the fact that their extracted HPSG grammars are closer to the PTB than CCGbank, or due to their conversion method. We leave the application of their method to the CCG parser for future work.

[5] This is likely to be an upper bound on the performance of the CCG parser, since the larger test sets contain sentences which were harder to convert, and hence are likely to be more difficult to parse.

SAMPLE	#	%	UB	actual F		projected F	
				gold	auto	gold	auto
00 (F=100)	759	39.7	100.00	94.19	93.41	–	–
00 (F≥95)	1164	60.8	98.49	91.08	89.93	92.46	91.29
00 (F≥92)	1430	74.6	97.41	89.73	88.47	92.05	90.76
00 (all)	1913	100.0	94.25	87.00	85.60	92.00	90.52

Table 3: Results on the development set (CCG parser only)

SAMPLE	#	%	UB	Berkeley F		actual F		projected F	
				gold	auto	gold	auto	gold	auto
23 (F=100)	961	39.9	100.0	**93.38**	**93.37**	**93.83**	**92.86**	–	–
23 (F≥95)	1401	58.2	98.61	91.66	91.63	90.82	89.84	92.08	91.09
23 (F≥92)	1733	72.0	97.44	91.01	90.88	89.53	88.54	91.82	90.81
23 (all)	2407	100.0	94.40	89.67	89.47	86.36	85.50	91.20	90.29

Table 4: Results on the test set (CCG parser and Berkeley)

ison is likely to be an easy subset consisting of shorter sentences, and so the most that can be said is that the CCG parser performs as well as the Berkeley parser on short sentences. In fact, the subset for which we perform a perfect conversion contains sentences with an average length of 18.1 words, compared to 21.4 for sentences with 40 words or less (a standard test set for reporting Parseval figures). Hence we do consider the comparison to be highly informative.

4 Conclusion

One question that is often asked of the CCG parsing work is "Why not convert back into the PTB representation and perform a Parseval evaluation?" By showing how difficult the conversion is, we believe that we have finally answered this question, as well as demonstrating comparable performance with the Berkeley parser. In addition, we have thrown further doubt on the possible use of the PTB for cross-framework parser evaluation, as recently suggested by Matsuzaki and Tsujii (2008). Even the smallest loss due to mapping across representations is significant when a few tenths of a percentage point matter. Whether PTB parsers could be competitive on alternative parser evaluations, such as those using GR schemes, for which the CCG parser performs very well, is an open question.

Acknowledgements

James Curran was funded under Australian Research Council Discovery grant DP0665973. Stephen Clark was funded under EPSRC grant EP/E035698/1.

References

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, Austrailia.

Stephen Clark and James R. Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552.

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

Julia Hockenmaier and Mark Steedman. 2007. CCGbank: a corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396.

Takuya Matsuzaki and Jun'ichi Tsujii. 2008. Comparative parser performance analysis across grammar frameworks through automatic tree conversion using synchronous grammars. In *Proceedings of COLING-08*, pages 545–552, Manchester, UK.

Yusuke Miyao and Jun'ichi Tsujii. 2005. Probabilistic disambiguation models for wide-coverage HPSG parsing. In *Proceedings of the 43rd meeting of the ACL*, pages 83–90, University of Michigan, Ann Arbor.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Proceedings of the HLT/NAACL conference*, Rochester, NY.

Stefan Riezler, Tracy H. King, Ronald M. Kaplan, Richard Crouch, John T. Maxwell III, and Mark Johnson. 2002. Parsing the Wall Street Journal using a Lexical-Functional Grammar and discriminative estimation techniques. In *Proceedings of the 40th Meeting of the ACL*, pages 271–278, Philadelphia, PA.