Reranking a wide-coverage CCG parser

Dominick Ng and **Matthew Honnibal** and **James R. Curran**
School of Information Technologies
University of Sydney
NSW 2006, Australia
{dong7223,mhonn,james}@it.usyd.edu.au

Abstract

n-best parse reranking is an important technique for improving the accuracy of statistical parsers. Reranking is not constrained by the dynamic programming required for tractable parsing, so arbitrary features of each parse may be considered.

We adapt the reranking features and methodology used by Charniak and Johnson (2005) for the C&C Combinatory Categorial Grammar parser, and develop new features based on the richer formalism. The reranker achieves a labeled dependency F-score of 87.59%, which is a significant improvement over prior results.

1 Introduction

Accurate syntactic parsing has proven to be critical for many tasks in natural language processing (NLP), including semantic role labeling (Gildea and Jurafsky, 2002), question answering (Echihabi and Marcu, 2003), and machine translation (DeNeefe and Knight, 2009). Improved parser accuracy benefits many downstream tasks in the field.

One method of improving parsing accuracy is *reranking* – the process of reordering the top n analyses as determined by a base parser (Collins, 2000). The statistical models used in phrase-structure and dependency parsers rely on dynamic programming algorithms that restrict possible features to a local context. This is necessary for efficient decoding of the potential parse forest, ensuring tractability at the cost of excluding any non-local features from consideration. Reranking operates over complete trees that are the most probable derivations under the dynamic programming model, allowing arbitrary complex features of the parse to be incorporated without sacrificing efficiency. Poor local decisions made by parsers are easier to model and capture in the reranking phase.

Collins (2000) reports a 1.55% accuracy improvement with reranking for the Collins parser, and Charniak and Johnson (2005) reports a 1.3% improvement for a reranked Charniak parser. An open question is how well reranking applies to parsers of different design to the Charniak and Collins parsers. An attempt to port the Charniak and Johnson reranker for the Berkeley parser (Petrov et al., 2006) produced only minimal accuracy improvements (Johnson and Ural, 2010), suggesting that careful feature engineering is necessary for good performance.

In this paper we describe the implementation of a discriminative maximum entropy reranker for the C&C parser (Clark and Curran, 2007), a state-of-the-art system based on Combinatory Categorial Grammar (CCG). We reimplement the features described in Charniak and Johnson (2005) to suit the CCG parser and replicate the Charniak reranker setup. Our experiments show that the PCFG-style features are less effective at reranking CCG than Penn Treebank-style trees. We hypothesise that the binary-branching structure of CCG is the cause, as CCG trees are deeper and create different constituent structures compared to Penn Treebank trees. To address this, we develop a number of new features to take advantage of the more detailed formalism and the evaluation over recovered dependencies. We also experiment with regression and classification approaches, variations in feature pruning, and differing numbers of n-best parses for the reranker to consider.

The reranker achieves a best labeled dependency F-score of 87.13% on Section 00 of CCGbank and 87.59% on Section 23. The performance gains are statistically significant, but small in real terms, indicating that crafting reranking features is not a trivial process. However, the continued improvements in parsing accuracy will benefit downstream applications utilising the parser through more accurate syntactic analysis.

Dominick Ng, Matthew Honnibal and James R. Curran. 2010. Reranking a wide-coverage CCG parser. In *Proceedings of Australasian Language Technology Association Workshop*, pages 90−98

2 Parser Reranking

Reranking has been successfully applied to dependency parsing (Sangati et al., 2009), machine translation (Shen et al., 2004), and natural language generation with CCG (White and Rajkumar, 2009). Collins (2000) describes reranking for the Collins (Model 2) parser (Collins, 1999). 36,000 sentences from Sections 02-21 of the Penn Treebank WSJ data are parsed with a modified version of the base parser, producing an average of 27 parses per sentence. Features are extracted from the parses to create reranker training data, including lexical heads and the distances between them, context-free rules in the tree, n-grams and their ancestors, and parent-grandparent relationships. Collins reports a final PARSEVAL F-score of 89.75% using a boosting-based reranker, a 1.55% improvement compared to the baseline parser.

The potential benefits from reranking are dependent on the quality of the candidate n-best parses. Huang and Chiang (2005) describe efficient and accurate algorithms for this task based on a directed hypergraph analysis framework (Klein and Manning, 2001). By improving the quality of the candidate parses, Huang and Chiang demonstrate how oracle reranking scores (using a perfect reranker that always choses the best parse from an n-best list) can be dramatically improved compared to the parses used in Collins (2000).

Charniak and Johnson (2005) describe discriminative reranking for the Charniak parser. A coarse-to-fine parsing approach allows high-quality n-best parses to be tractably computed while retaining dynamic programming in the parser. When run in 50-best mode the Charniak n-best parser has an oracle F-score of 96.8% in the standard PARSEVAL metric – much higher than the 89.7% parser baseline. The reranker produces a final F-score of 91.0% in 50-best mode. This is an improvement of 1.3% over the baseline model. Self-training over the reranked parses further improves performance to 92.1% F-score, which remains the state-of-the-art (McClosky et al., 2006). Self-training provides the additional benefit of improving the Charniak parser's performance on out-of-domain data – a known weakness of supervised parsing.

More recently, the Charniak reranking system has been adapted for the Berkeley parser (Petrov et al., 2006). Unlike the Collins and Charniak parsers, which are broadly similar and heavily based on lexicalised models, the Berkeley parser uses a split-merge technique to acquire a much smaller, unlexicalised grammar from its training data. Johnson and Ural (2010) report that reranking leads to negligible performance improvements for the Berkeley parser, and acknowledge that the reranker's feature set, adapted from Charniak and Johnson (2005), may be implicitly tailored to the Charniak parser over the Berkeley parser. In particular, the feature pruning process for reranking was conducted over output from the Charniak parser, which may have prevented useful features for the Berkeley parser from being chosen.

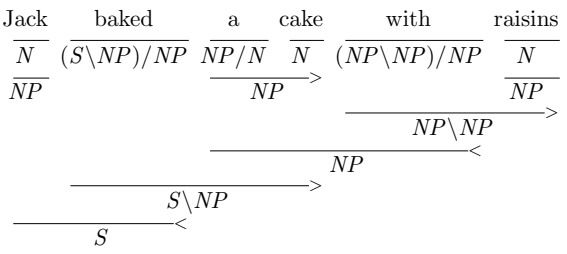


Figure 1: A simple CCG derivation.

3 Combinatory Categorial Grammar

Combinatory Categorial Grammar (CCG, Steedman (2000)) is a lexicalised grammar formalism based on combinatory logic. The grammar is directly encoded in the lexicon in the form of categories that govern the syntactic behaviour of each word. A small number of generic rules combine categories together to form a spanning analysis.

Categories may be atomic or complex. Atomic categories represent words and constituents that are syntactically complete, such as nouns (N), noun phrases (NP), prepositional phrases (PP), and sentences (S). Complex categories are binary structures of the form X/Y or $X\backslash Y$, and represent structures which combine with an argument of category Y to produce a result of category X. The forward and backward slashes indicate that Y is expected to the right and left respectively.

Complex categories can be thought of as functors that require particular arguments to produce a grammatical construction. Subcategorization information is encoded using nested categories. For example, transitive verbs have the category $(S\backslash NP)/NP$, which indicates that one object NP is expected to the right to form a verb phrase $S\backslash NP$, which in turn expects one subject NP to the left to form a sentence.

In addition to forward and backward application, CCG has a number of other binary combinators based on function composition. There are also unary type-changing combinators that take a single category and transform it into another category. Figure 1 gives a simple CCG derivation, showing how categories are successively combined together to yield an analysis.

4 The C&C parser

The C&C parser (Clark and Curran, 2007) is a fast, highly accurate parser based on the CCG formalism. The parser is used in question answering systems (Bos et al., 2007), computational semantics tools (Bos et al., 2004), and has been shown to perform well in recovering unbounded dependencies (Rimell et al., 2009).

The parser divides the parsing process into two main phases: *supertagging* and *parsing*. First, the supertagger assigns a small set of initial categories to each word in the sentence. Then, the parser attempts to find a spanning analysis using the proposed categories using the modified CKY algorithm described in Steedman (2000). If the parser cannot find an analysis (i.e. there is no sequence of combinators that can combine the proposed categories) the supertagger is run again at a higher ambiguity level, giving each word a larger set of possible categories, and the process is repeated. The supertagging phase dramatically reduces the number of derivations for the parser to consider, making the system highly efficient.

An n-best version of the C&C parser has recently been developed (Brennan, 2008), incorporating the algorithms described in Huang and Chiang (2005). The n-best parser is almost as efficient as the baseline 1-best version, and we use it as the basis for all experiments presented in this paper.

CCGbank is the standard corpus for English parsing with CCG. It is a transformation of the Penn Treebank WSJ data into CCG derivations and dependencies (Hockenmaier and Steedman, 2007). Sections 02-21 are the standard training data for the C&C parser, with Section 00 used for development and Section 23 for evaluation. The supertagger requires part-of-speech information for each word as part of its feature set, so a POS tagger is also included with the C&C parser. Both the supertagger and the POS tagger are trained over tags extracted from Sections 02-21 of CCGbank.

5 Methodology

We frame the reranking task for CCG parsing as follows: given an n-best list of parses, ranked by the parser, choose the parse that is as close as possible to the gold standard. We use the standard CCG labeled dependency metric as described in Hockenmaier (2003) to define closeness to the gold standard, allowing us to explore both classification and regression as frameworks for the task. In classification, the closest sentence(s) to the gold standard with respect to F-score are labeled as positive, while all other sentences are labeled as negative. If there are multiple parses with the highest F-score, they are all labeled as positive. In regression, the F-score of each parse is used as the target value. Both classification and regression approaches were implemented using MEGAM[1].

n-best lists of parses were generated using the n-best C&C parser using Algorithm 3 of Huang and Chiang (2005). We used the normal-form model for the C&C parser as described in Clark and Curran (2007) for all experiments in this paper. Reranker training data was created using n-best parses of each sentence in Sections 02-21 of CCGbank. As this is also the parser's training data, care must be taken to avoid generating training data where the parser's confidence level is different to that at run-time (caused by parsing the training data). We constructed ten folds of Sections 02-21, training the POS tagger, supertagger, and parser on nine of the folds and producing n-best parses over the remaining fold.

Features were generated over the n-best parses of the folded training data and the appropriate label assigned based on the F-score. This data was used to train the reranker. Similarly, Section 24 of CCGbank was parsed using a model trained over Sections 02-21 for use as tuning data. At run-time, features were generated over the n-best parses of the test data, and the most probable parse (classification) or the parse with the highest predicted F-score (regression) was returned.

We experimented with values of 10 and 50 for n to balance between the potential accuracy improvement and the efficiency of the reranker. n was kept constant between the training data and the final test data (i.e. a reranker trained on 50-best parses was then tested over 50-best parses).

Following Charniak and Johnson (2005) we implemented feature pruning for the reranker train-

[1] http://www.umiacs.umd.edu/~hal/megam

ing data as follows. For each sentence, define a feature as being *pseudo-constant* if it does not differ in value over all the parses for that sentence. We keep all features that are non pseudo-constant in at least t sentences in the training data. We experimented with values of 0, 2, and 5 for t to investigate the effect of feature pruning.

6 Reranking Features

The features described in this section are calculated over CCG derivation trees produced by the C&C parser. We began by implementing the features described by Charniak and Johnson (2005), before developing features specifically for CCG derivations. We also implemented the CCG parsing features described by Clark and Curran (2007), so that our reranking model would have access to the information used by the parser. These features include various combinations of word-category, word-POS, CCG rule, distance, and dependency information. Finally, the log score and rank assigned to each derivation by the parser were encoded as core features for the reranker.

CCG derivation trees have some important structural differences from the trees that the Charniak and Johnson features were designed for. The most important difference is that CCG trees are at most binary branching[2]. As the longest non-terminal in the Penn Treebank has 51 children, features designed to generalise long production rules are useful in the Charniak and Johnson reranker but are less relevant to CCG trees.

Another important difference is that CCG production rules are constrained by the combinatory rules, whereas Penn Treebank productions combine unrelated atomic symbols. For instance, a Penn Treebank production NP→ NP PP would be translated into CCG as $NP\rightarrow\ NP\ NP\backslash NP$. Much of the information in the production is already present in the structure of the $NP\backslash NP$ category. We speculate that this will make the features that capture the vertical context of a production rule less useful for CCG.

Finally, each ccg tree corresponds to exactly one dependency analysis, and this is produced as output by the C&C parser. This gives the reranker access to the full dependency analysis of each sentence, making the dependency-approximation heuristics used by Charniak and Johnson (2005) unnecessary for our purposes.

The features adapted from Charniak and Johnson (2005) are described in Sections 6.1 and 6.2 below. The novel CCG features we develop are described in Section 6.3. Most features were implemented as simple boolean indicator functions. Maximum entropy modelling exponentiates feature values, so real-valued features are more influential than boolean features. We mitigated this effect by taking the log of real-valued features.

6.1 Tree Topology Features

These features attempt to describe the overall shape of the parse tree, to capture the fact that English generally favours right-branching parse trees, with phonologically heavy constituents generally occurring in sentence-final position. Tree topology can also be useful in capturing the balance found in coordination attachment. These guidelines distinguish the correct parse tree in Figure 2a from the incorrect parse tree in Figure 2b – the incorrect tree is more left-branching than the correct tree, with a shallower depth of balance in the coordination.

CoPar: records coordination parallelism at various depths. Indicates whether both sides of a coordination are identical in structure and category labels at depths of 1 to 4 from the coordinator.

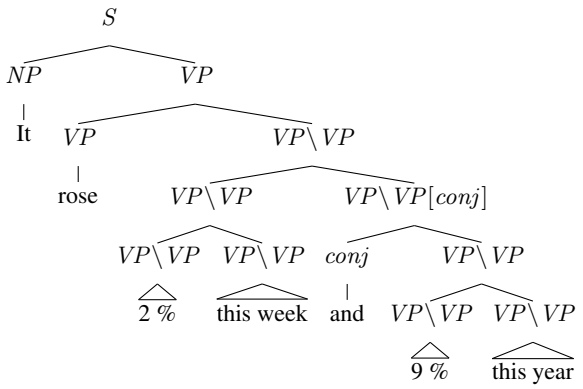
CoLenPar: indicates the difference in size between two halves of a conjunction (where size is the number of nodes in the yield) as well as whether the latter half is the final element.

Heavy: encodes the category and size of the subtree rooted at each non-terminal, whether the non-terminal is at the end of the sentence, and whether it is followed by punctuation. This crudely captures the tendency for larger constituents to lie further to the right in a tree.

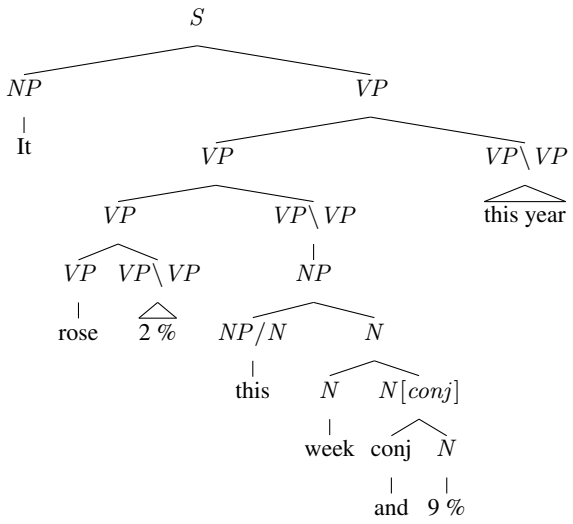
RightBranch: encodes the number of non-terminals on the longest path from the root of the tree to the right-most non-punctuation node in the tree, and the number of non-terminals in the tree that are not on this path.

SubjVerbAgr: captures the conjoined POS tags of the subject noun and verb in a sentence to distinguish cases where the pluralisation does not agree. The subject is assumed to be the final NP before the verb phrase ($S\backslash NP$) in a sentence.

[2] Steedman (2000) describes a ternary conjunction rule, but this is broken into two binary productions in CCGbank, using the marker $[conj]$.



(a) The correct parse.



(b) Parse featuring a conjunction error.

Figure 2: Two CCG derivations for the sentence, *It rose 2% this week and 9% this year.*

6.2 Local Context Features

These features, adapted from Charniak and Johnson (2005), attempt to represent various fragments of the tree as well as incorporate layers of vertical and horizontal context that are difficult to encode in the parser model.

Edge: captures the words and POS tags immediately preceding and following the subtree rooted at each non-terminal in the tree. This crudely captures poor attachment decisions in local trees.

Heads: represents pairs of constituent heads as indicated by the parser at various levels in the tree. Heads are encoded as lexical items and POS tags.

HeadTree: records the entire tree fragment (in a bracketed string format) projected upwards from the head word of the sentence.

Neighbours: encodes the category of each non-terminal, its binned size, and the POS tags of the ℓ_1 preceding words and the ℓ_2 following words, where $\ell_1 = 1$ or 2 and $\ell_2 = 1$. Binned size is the number of words in the yield of the non-terminal, bucketed into 0, 1, 2, 4, or 5+.

NGramTree: records tree fragments rooted at the lowest common ancestor node of $\ell = 2$ or 3 contiguous terminals in the tree. This represents the subtree encompassing each sequence of ℓ words in the sentence.

Rule: captures the equivalent CCG rule application represented at each non-terminal node; equivalent to a context-free production rule.

SynSemHeads: yield pairs of semantic heads (e.g. the rightmost noun in a noun phrase) and functional heads (e.g. the determiner in a noun phrase) at each non-terminal in the tree. Heads are encoded as lexical items and POS tags.

Word: yields each word in a sentence along with the categories of $\ell = 2$ or 3 of its immediate ancestor nodes in the tree.

WProj: for each terminal in the tree, encode the word combined with the category of its maximal projection parent, which is the first node found by climbing the tree until the child node is no longer the head of its parent.

6.3 CCG Features

We devised a number of new features for CCG aimed at uncovering various combinator sequences or combinations that may indicate an overly complicated or undesirable derivation. Additionally, these features attempt to encode more information about the dependencies licensed by the derivation as it is these dependencies which will be evaluated.

Balance: encodes the overall balance of the tree in terms of the ratio of leaves and the ratio of nodes in the left and right subtrees from the root. This feature reflects the decision to make all nominal compounds in CCGbank right branching (Hock-

enmaier and Steedman, 2007).

CoHeads: records the heads of both halves of a coordination as indicated by the parser, along with the depth at which the head is found. This attempts to encode the conjunction dependencies in the tree as incorrect conjunction dependencies propagate through to other dependencies in the tree. Heads are encoded as lexical items and POS tags.

LexDep: CCG dependencies can be partially captured via the children of non-terminals in the tree. This feature is active for non-terminals with two children and encodes the heads of the children in terms of lexical items, POS tags, categories, and depth from the non-terminal. Dependencies involving punctuation are ignored as they are not assessed in the evaluation.

NumDeps: distinguishes between parses based on the log number of dependencies that they yield ignoring punctuation. Dependencies are located using the same heuristic as the LexDep feature.

TypeRaising: indicates the presence of unary type-raising in the tree. While type-raising is necessary to analyse some constructions in CCG, it has tightly restricted in the parser due to its power, and is expected to appear only rarely.

UnaryRule, BiUnaryRule: indicates the unary rules present in the tree and the bigram combinations of these rules. The unary rules do not include type-raising and are non-standard in CCG; they were added by Hockenmaier and Steedman (2007) to CCGbank for constructions such as clausal adjuncts, which are poorly handled by the formalism.

C&C Features: Finally, we also incorporate the dependency and normal-form features used by the C&C parser as described in Clark and Curran (2007). These features encode various combinations of word-category, word-POS, root-word, CCG rule, distance, and dependency information.

7 Evaluation Measures

We follow the CCG dependency evaluation methodology established by Hockenmaier (2003), using the EVALUATE scorer distributed with the C&C parser. It evaluates a CCG parse as a set of labeled dependencies consisting of the head, its lexical category, the child, and the argument slot that it fills. A dependency is considered correct only if all four elements match the gold standard.

	LP	LR	LF	AF
Baseline	87.19	86.32	86.75	84.80
Oracle 10	91.98	90.89	91.43	89.47
Oracle 50	93.43	92.26	92.84	90.96

Table 1: Baseline and oracle n-best parser performance over Section 00 of CCGbank.

Statistical significance was calculated using the test described in Chinchor (1992), which measures the probability that the two sets of responses are drawn from the same distribution. A score below 0.05 is considered significant.

We report labeled precision (LP), labeled recall (LR), and labeled F-score (LF) results over gold standard POS tags and labeled F-score over automatically assigned POS tags (AF).

8 Results

8.1 Oracle Performance

Reranking is dependent on high-quality parses from the n-best parser. As seen in Table 1, the oracle labeled dependency F-score of the n-best C&C parser is 92.48% given a perfect reranker over 50-best parses. This is a significant improvement over the baseline result of 86.75% and provides a solid basis for a reranker.

Our oracle score falls notably short of the 50-best oracle of 96.8% reported by Charniak and Johnson (2005), over a baseline of 89.7%. However, these numbers refer to the PARSEVAL score for constituency parses, so they are not directly comparable to our dependency recovery metric.

We present results in Tables 2 and 3 comparing the 1-best C&C parser using the normal-form model (Clark and Curran, 2007), randomized baselines (choosing a parse at random from the n-best list), and the reranking C&C parser in labeled dependency recovery over Section 00 of CCGbank. Our best result for 10-best reranking is an F-score of 87.13% with gold POS tags and 85.22% with automatically assigned POS tags. This is achieved using the regression setup and all features without pruning. The best result for 50-best reranking is F-scores of 87.08% and 85.23% respectively, using the classification setup with all features and a pruning value of 2. These two results are both statistically significant improvements over the baseline parser.

Randomly choosing a parse from the n-best list

	t	LP	LR	LF	AF
Baseline	-	87.19	86.32	86.75	84.80
Random	-	85.40	84.46	84.93	83.00
Class+CJ	0	**87.21**	**86.06**	**86.63**	84.81
	2	87.16	85.98	86.57	**84.82**
	5	87.12	85.95	86.53	84.74
Class+CCG	0	87.17	86.18	86.67	**84.75**
	2	**87.19**	**86.19**	**86.69**	84.74
	5	87.11	86.09	86.59	84.68
Class+ALL	0	**87.32**	**86.32**	**86.82**	**84.85**
	2	87.29	86.29	86.78	84.82
	5	87.23	86.25	86.74	84.79
Regress+CJ	0	**86.96**	**85.99**	**86.47**	**84.58**
	2	86.75	85.76	86.26	84.34
	5	86.69	85.72	86.20	84.30
Regress+CCG	0	**87.27**	**86.41**	**86.83**	**85.08**
	2	87.05	86.12	86.58	84.70
	5	86.96	86.08	86.52	84.73
Regress+ALL	0	**87.60**	**86.67**	**87.13**	**85.22**
	2	87.42	86.47	86.94	85.00
	5	87.41	86.50	86.95	84.96

.

Table 2: 10-best reranking performance on Section 00 of CCGbank for various combinations of features, pruning values t, and classification and regression experiments. Bolded scores are the highest for the feature set and approach.

	t	LP	LR	LF	AF
Baseline	-	87.19	86.32	86.75	84.80
Random	-	83.90	82.58	83.24	81.50
Class+CJ	0	**86.93**	**85.87**	**86.40**	**84.69**
	2	86.72	85.54	86.12	84.48
	5	86.77	85.61	86.19	84.56
Class+CCG	0	87.17	86.10	86.63	84.62
	2	87.14	86.07	86.60	**84.66**
	5	**87.29**	**86.17**	**86.72**	**84.66**
Class+ALL	0	87.38	86.29	86.83	84.91
	2	**87.61**	**86.56**	**87.08**	**85.23**
	5	87.30	86.22	86.76	84.74
Regress+CJ	0	**86.49**	**85.64**	**86.07**	84.22
	2	86.44	85.46	85.95	**84.32**
	5	86.32	85.28	85.80	84.12
Regress+CCG	0	**87.08**	**86.15**	**86.61**	84.65
	2	87.00	86.06	86.53	84.66
	5	87.07	86.08	86.57	**84.72**
Regress+ALL	0	**87.28**	**86.30**	**86.79**	**84.89**
	2	86.73	85.77	86.25	84.43
	5	87.04	86.06	86.55	84.66

Table 3: 50-best reranking performance on Section 00 of CCGbank for various combinations of features, pruning values t, and classification and regression experiments.

	LP	LR	LF	AF
Best	87.60	86.67	87.13	85.22
-CoPar	87.47	86.57	87.02	85.11
-CoLenPar	87.53	86.59	87.06	85.17
-Heavy	87.44	86.55	86.99	85.09
-RightBranch	87.59	86.67	87.13	85.17
-SubjVerbAgr	**87.26**	**86.26**	**86.76**	**84.88**
-Edges	**87.11**	**86.22**	**86.67**	**84.87**
-Heads	87.55	86.65	87.10	85.26
-HeadTree	87.61	86.64	87.12	85.22
-Neighbours	87.50	86.59	87.05	85.16
-NGramTree	87.51	86.55	87.03	85.08
-Rule	87.54	86.58	87.05	85.14
-SynSemHeads	87.42	86.47	86.94	85.07
-Word	87.44	86.51	86.97	85.11
-WProj	87.44	86.55	86.99	85.09
-Balance	87.44	86.53	86.98	85.17
-CoHeads	87.38	86.47	86.93	84.90
-LexDep	87.52	86.58	87.04	85.15
-NumDeps	87.40	86.54	86.97	85.04
-TypeRaising	87.41	86.48	86.95	85.05
-UnaryRule	87.58	86.68	87.13	85.27
-BiUnaryRule	87.55	86.64	87.09	85.20
-C&C	87.44	86.48	86.96	84.97

Table 4: Subtractive analysis on the top performing 10-best model on Section 00. Bold indicates a statistically significant change from the baseline.

results in much poorer performance than the 1-best baseline. All our experiments produced results that were significantly higher than the randomized result, indicating that our approaches were learning useful features from the training data. Even though the oracle scores increase with n (as shown in Table 1), the overall parse quality deteriorates.

Regression was generally more successful for 10-best reranking, while classification was better for 50-best reranking. However, there were very few cases where a statistically significant difference in performance was observed between regression and classification approaches.

8.2 Features

We investigated the performance of three sets of features: those adapted from Charniak and Johnson (2005) (CJ), our new features (CCG), and the union of the two sets (ALL). The log score and rank of each parse was included as core features in every experiment. In general, more features improved performance. The best results were produced using all of the possible features in the reranker model. In terms of the top F-score for

	LP	LR	LF	AF
Baseline	87.19	86.32	86.75	84.80
SubjVerbAgr	86.76	85.87	86.31	84.33
Edges	86.29	85.45	85.87	83.95
Both	86.30	85.49	85.89	83.95

Table 5: 10-best isolation experiments for the SubjVerbAgr and Edges features on Section 00 using regression and no pruning.

	LP	LR	LF	AF
Baseline	87.75	86.98	87.36	85.07
Reranker	**87.98**	**87.21**	**87.59**	**85.36**

Table 6: Baseline and final reranker performance over Section 23 of CCGbank with the normal-form model.

each set of features, the CCG-specific features were better than the Charniak and Johnson (2005) features by a statistically significant margin. This held in all experiments except one (10-best classification), indicating that features tailored to CCG trees and dependency evaluation were more discriminative between good and bad CCG parses. This also implies that for reranking to improve the accuracy of a parser, the features must target that parser and the nature of its evaluation. Features producing state-of-the-art performance for the Charniak reranker had no positive impact on CCG parsing in isolation.

We conducted subtractive feature analysis on our best performing model (10-best regression with all features and no pruning) to investigate the contribution of individual features. Features were individually removed and the reranker was retrained and retested on Section 00. The removal of the SubjVerbAgr and Edges features are statistically significant, while the removal of any other single feature results in a non-significant decrease in F-score. We then performed an isolation experiment, training and testing the reranker using just the SubjVerbAgr and Edge features with the log score and rank from the parser. Table 5 shows that these features do significantly worse than the baseline in isolation, indicating that it is the combination of features together which produces the improved performance.

8.3 Pruning

We found that increased feature pruning had a negative impact on parsing accuracy. None of our experiments showed a significant improvement with higher pruning values, as opposed to Charniak and Johnson (2005) who found the count-based pruning to be useful. The best performing systems overall used pruning values of 0 or 2, implying that the pruning strategy is ineffective with respect to performance over such a varied set of features. One area where pruning does help is in the training times for the reranker: some experiments are nearly twice as fast with a pruning value $t = 5$ compared to $t = 0$. However, as this cost must only be paid once, the benefit of pruning with respect to actual parsing time is negligible.

8.4 Final Results

Table 6 summarises the performance of our best reranker model against the baseline normal-form model on Section 23 of CCGbank. We achieve statistical significant improvement in F-score over the baseline. However, in real terms the change in F-score is small, indicating that reranking may not guarantee performance improvements even if it is carefully targeted to the parser.

9 Conclusion

We have implemented a maximum entropy reranker for the C&C CCG parser, building on the methodology and features of Charniak and Johnson (2005) and extending the approach with new features. We have found that performance improvements from reranking stem from targeting the reranker features at the parser and its evaluation: features tailored to CCG perform better than PCFG-style features in isolation. Our best system achieves an of 87.59%, which is a statistically significant improvement over the baseline parser.

The reranker scales with the efficiency of calculating features on parse trees. The features described in this paper require time linear in the number of nodes in the tree. However, the reranker is currently implemented as an external post-processing step. This leads to an order of magnitude speed decrease; future work will include integrating the reranker into the parser itself to mitigate this speed impact.

The improvement in accuracy that we achieve is small in absolute terms, showing that reranking is a considerably difficult task. However, continued

improvements such as this one in parsing accuracy will benefit the variety of downstream applications that utilise parsing for practical NLP tasks.

Acknowledgments

This work was supported by Australian Research Council Discovery grants DP0665973 and DP1097291, the Capital Markets CRC and a University of Sydney Merit Scholarship.

References

Johan Bos, Stephen Clark, Mark Steedman, James R. Curran, and Julia Hockenmaier. 2004. Wide-Coverage Semantic Representations from a CCG Parser. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING-04)*, pages 1240–1246, Geneva, Switzerland, August.

Johan Bos, James R. Curran, and Edoardo Guzzetti. 2007. The Pronto QA system at TREC-2007: Harvesting Hyponyms, Using Nominalisation Patterns, and Computing Answer Cardinality. In *Proceedings of the Sixteenth Text REtrieval Conference (TREC 2007)*, pages 726–732, Gaitersburg, Maryland, USA.

Forrest Brennan. 2008. k-best Parsing Algorithms for a Natural Language Parser. Master's thesis, University of Oxford.

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 Association for Computational Linguistics (ACL-05)*, pages 173–180, Ann Arbor, Michigan, USA, June.

Nancy Chinchor. 1992. The statistical significance of the MUC-4 results. In *Proceedings of the Fourth Message Understanding Conference (MUC-4)*, pages 30–50, McLean, Virginia, June.

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, Philadelphia, Pennsylvania, USA.

Michael Collins. 2000. Discriminative Reranking for Natural Language Parsing. In *Proceedings of the 17th International Conference on Machine Learning (ICML-00)*, pages 175–182, Palo Alto, California, USA, June.

Steve DeNeefe and Kevin Knight. 2009. Synchronous Tree Adjoining Machine Translation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP-09)*, pages 727–736, Singapore, August.

Abdessamad Echihabi and Daniel Marcu. 2003. A Noisy-Channel Approach to Question Answering. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL-03)*, pages 16–23, Sapporo, Japan, July.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic Labeling of Semantic Roles. *Computational Linguistics*, 28(3):245–288.

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.

Julia Hockenmaier. 2003. Parsing with Generative Models of Predicate-Argument Structure. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL-03)*, pages 359–366, Sapporo, Japan, July.

Liang Huang and David Chiang. 2005. Better k-best Parsing. In *Proceedings of the Ninth International Workshop on Parsing Technology (IWPT-05)*, pages 53–64, Vancouver, British Columbia, Canada, October.

Liang Huang. 2008. Forest Reranking: Discriminative Parsing with Non-Local Features. In *Proceedings of the Human Language Technology Conference at the 45th Annual Meeting of the Association for Computational Linguistics (HLT/ACL-08)*, pages 586–594, Columbus, Ohio, June.

Mark Johnson and Ahmet Engin Ural. 2010. Reranking the Berkeley and Brown Parsers. In *Proceedings of Human Language Technologies: the 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics (HLT/NAACL-10)*, pages 665–668, Los Angeles, California, USA, June.

Dan Klein and Christopher D. Manning. 2001. Parsing and Hypergraphs. In *Proceedings of the 7th International Workshop on Parsing Technologies (IWPT-01)*, Beijing, China, October.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective Self-Training for Parsing. In *Proceedings of the 2006 Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics (HLT/NAACL-06)*, pages 152–159, New York City, New York, USA, June.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning Accurate, Compact, and Interpretable Tree Annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING/ACL-06)*, pages 433–440, Sydney, Australia, July.

Laura Rimell, Stephen Clark, and Mark Steedman. 2009. Unbounded Dependency Recovery for Parser Evaluation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP-09)*, pages 813–821, Singapore, August.

Federico Sangati, Willem Zuidema, and Rens Bod. 2009. A generative re-ranking model for dependency parsing. In *Proceedings of the 11th International Conference on Parsing Technologies (IWPT-09)*, pages 238–241, Paris, France, October.

Libin Shen, Anoop Sarkar, and Franz Josef Och. 2004. Discriminative Reranking for Machine Translation. In *Proceedings of the 2004 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT/NAACL-04)*, pages 177–184, Boston, Massachusetts, USA, May.

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

Michael White and Rajakrishnan Rajkumar. 2009. Perceptron Reranking for CCG Realization. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP-09)*, pages 410–419, Singapore, August.