Accurate Context-Free Parsing with Combinatory Categorial Grammar

Timothy A. D. Fowler and **Gerald Penn**
Department of Computer Science, University of Toronto
Toronto, ON, M5S 3G4, Canada
{tfowler, gpenn}@cs.toronto.edu

Abstract

The definition of combinatory categorial grammar (CCG) in the literature varies quite a bit from author to author. However, the differences between the definitions are important in terms of the language classes of each CCG. We prove that a wide range of CCGs are strongly context-free, including the CCG of CCGbank and of the parser of Clark and Curran (2007). In light of these new results, we train the PCFG parser of Petrov and Klein (2007) on CCGbank and achieve state of the art results in supertagging accuracy, PARSEVAL measures and dependency accuracy.

1 Introduction

Combinatory categorial grammar (CCG) is a variant of categorial grammar which has attracted interest for both theoretical and practical reasons. On the theoretical side, we know that it is mildly context-sensitive (Vijay-Shanker and Weir, 1994) and that it can elegantly analyze a wide range of linguistic phenomena (Steedman, 2000). On the practical side, we have corpora with CCG derivations for each sentence (Hockenmaier and Steedman, 2007), a wide-coverage parser trained on that corpus (Clark and Curran, 2007) and a system for converting CCG derivations into semantic representations (Bos et al., 2004).

However, despite being treated as a single unified grammar formalism, each of these authors use variations of CCG which differ primarily on which combinators are included in the grammar and the restrictions that are put on them. These differences are important because they affect whether the mild context-sensitivity proof of Vijay-Shanker and Weir (1994) applies. We will provide a generalized framework for CCG within which the full variation of CCG seen in the literature can be defined. Then, we prove that for a wide range of CCGs there is a context-free grammar (CFG) that has exactly the same derivations. Included in this class of strongly context-free CCGs are a grammar including all the derivations in CCGbank and the grammar used in the Clark and Curran parser.

Due to this insight, we investigate the potential of using tools from the probabilistic CFG community to improve CCG parsing results. The Petrov parser (Petrov and Klein, 2007) uses latent variables to refine the grammar extracted from a corpus to improve accuracy, originally used to improve parsing results on the Penn treebank (PTB). We train the Petrov parser on CCGbank and achieve the best results to date on sentences from section 23 in terms of supertagging accuracy, PARSEVAL measures and dependency accuracy.

These results should not be interpreted as proof that grammars extracted from the Penn treebank and from CCGbank are equivalent. Bos's system for building semantic representations from CCG derivations is only possible due to the categorial nature of CCG. Furthermore, the long distance dependencies involved in extraction and coordination phenomena have a more natural representation in CCG.

2 The Language Classes of Combinatory Categorial Grammars

A categorial grammar is a grammatical system consisting of a finite set of words, a set of categories, a finite set of sentential categories, a finite lexicon mapping words to categories and a rule system dictating how the categories can be combined. The set of categories are constructed from a finite set of atoms A (e.g. $A = \{S, NP, N, PP\}$) and a finite set of binary connectives B (e.g. $B = \{/, \backslash\}$) to build an infinite set of categories $\mathcal{C}(A, B)$ (e.g. $\mathcal{C}(A, B) = \{S, S\backslash NP, (S\backslash NP)/NP, \ldots\}$). For a category C, its size $|C|$ is the

number of atom occurrences it contains. When not specified, connectives are left associative.

According to the literature, combinatory categorial grammar has been defined to have a variety of rule systems. These rule systems vary from a small rule set, motivated theoretically (Vijay-Shanker and Weir, 1994), to a larger rule set, motivated linguistically, (Steedman, 2000) to a very large rule set, motivated by practical coverage (Hockenmaier and Steedman, 2007; Clark and Curran, 2007). We provide a definition general enough to incorporate these four main variants of CCG, as well as others.

A combinatory categorial grammar (CCG) is a categorial grammar whose rule system consists of rule schemata where the left side is a sequence of categories and the right side is a single category where the categories may include variables over both categories and connectives. In addition, rule schemata may specify a sequence of categories and connectives using the ... convention[1]. When ... appears in a rule, it matches any sequence of categories and connectives according to the connectives adjacent to the ... . For example, the rule schema for forward composition is:

$$X/Y, Y/Z \rightarrow X/Z$$

and the rule schema for generalized forward crossed composition is:

$$X/Y, Y|_1Z_1|_2 \ldots |_nZ_n \rightarrow X|_1Z_1|_2 \ldots |_nZ_n$$

where X, Y and Z_i for $1 \leq i \leq n$ are variables over categories and $|_i$ for $1 \leq i \leq n$ are variables over connectives. Figure 1 shows a CCG derivation from CCGbank.

A well-known categorial grammar which is not a CCG is Lambek categorial grammar (Lambek, 1958) whose introduction rules cannot be characterized as combinatory rules (Zielonka, 1981).

2.1 Classes for defining CCG

We define a number of *schema classes* general enough that the important variants of CCG can be defined by selecting some subset of the classes. In addition to the schema classes, we also define two *restriction classes* which define ways in which the rule schemata from the schema classes can be restricted. We define the following schema classes:

(1) Application

- $X/Y, Y \rightarrow X$
- $Y, X\backslash Y \rightarrow X$

(2) Composition

- $X/Y, Y/Z \rightarrow X/Z$
- $Y\backslash Z, X\backslash Y \rightarrow X\backslash Z$

(3) Crossed Composition

- $X/Y, Y\backslash Z \rightarrow X\backslash Z$
- $Y/Z, X\backslash Y \rightarrow X/Z$

(4) Generalized Composition

- $X/Y, Y/Z_1/\ldots/Z_n \rightarrow X/Z_1/\ldots/Z_n$
- $Y\backslash Z_1\backslash\ldots\backslash Z_n, X\backslash Y \rightarrow X\backslash Z_1\backslash\ldots\backslash Z_n$

(5) Generalized Crossed Composition

- $X/Y, Y|_1Z_1|_2\ldots|_nZ_n \rightarrow X|_1Z_1|_2\ldots|_nZ_n$
- $Y|_1Z_1|_2\ldots|_nZ_n, X\backslash Y \rightarrow X|_1Z_1|_2\ldots|_nZ_n$

(6) Reducing Generalized Crossed Composition

Generalized Composition or Generalized Crossed Composition where $|X| \leq |Y|$.

(7) Substitution

- $(X/Y)|_1Z, Y|_1Z \rightarrow X|_1Z$
- $Y|_1Z, (X\backslash Y)|_1Z \rightarrow X|_1Z$

(8) D Combinator[2]

- $X/(Y|_1Z), Y|_2W \rightarrow X|_2(W|_1Z)$
- $Y|_2W, X\backslash(Y|_1Z) \rightarrow X|_2(W|_1Z)$

(9) Type-Raising

- $X \rightarrow T/(T\backslash X)$
- $X \rightarrow T\backslash(T/X)$

(10) Finitely Restricted Type-Raising

- $X \rightarrow T/(T\backslash X)$ where $\langle X, T\rangle \in S$ for finite S
- $X \rightarrow T\backslash(T/X)$ where $\langle X, T\rangle \in S$ for finite S

(11) Finite Unrestricted Variable-Free Rules

- $\vec{X} \rightarrow Y$ where $\langle \vec{X}, Y\rangle \in S$ for finite S

[1]The ... convention (Vijay-Shanker and Weir, 1994) is essentially identical to the $ convention of Steedman (2000).

[2]Hoyt and Baldridge (2008) argue for the inclusion of the D Combinator in CCG.

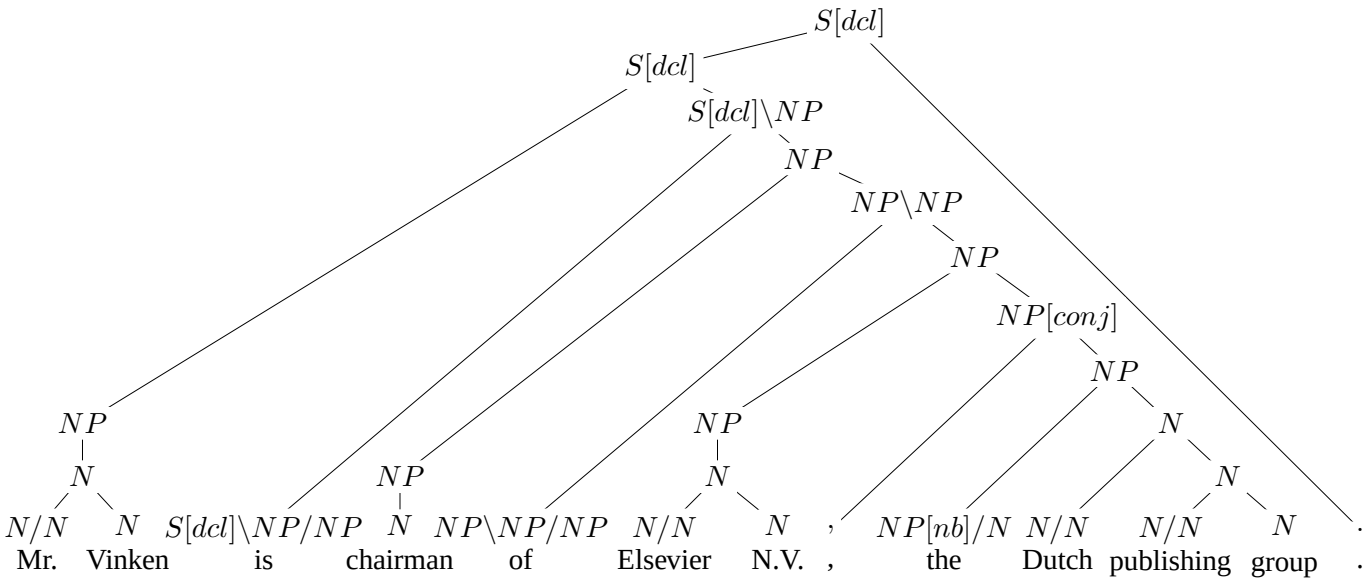

Figure 1: A CCG derivation from section 00 of CCGbank.

We define the following restriction classes:

(A) Rule Restriction to a Finite Set

The rule schemata in the schema classes of a CCG are limited to a finite number of instantiations.

(B) Rule Restrictions to Certain Categories [3]

The rule schemata in the schema classes of a CCG are limited to a finite number of instantiations although variables are allowed in the instantiations.

Vijay-Shanker and Weir (1994) define CCG to be schema class (4) with restriction class (B). Steedman (2000) defines CCG to be schema classes (1-5), (6), (10) with restriction class (B).

2.2 Strongly Context-Free CCGs

Proposition 1. *The set of atoms in any derivation of any CCG consisting of a subset of the schema classes (1-8) and (10-11) is finite.*

Proof. A finite lexicon can introduce only a finite number of atoms in lexical categories.

Any rule corresponding to a schema in the schema classes (1-8) has only those atoms on the right that occur somewhere on the left. Rules in classes (10-11) can each introduce a finite number of atoms, but there can be only a finite number of such rules, limiting the new atoms to a finite number. □

Definition 1. The *subcategories* for a category c are c_1 and c_2 if $c = c_1 \bullet c_2$ for $\bullet \in B$ and c if c is atomic. Its *second subcategories* are the subcategories of its subcategories.

Proposition 2. *Any CCG consisting of a subset of the rule schemata (1-3), (6-8) and (10-11) has derivations consisting of only a finite number of categories.*

Proof. We first prove the proposition excluding schema class (8). We will use structural induction on the derivations to prove that there is a bound on the size of the subcategories of any category in the derivation. The base case is the assignment of a lexical category to a word and the inductive step is the use of a rule from schema classes (1-4), (6-7) and (10-11).

Given that the lexicon is finite, there is a bound k on the size of the subcategories of lexical categories. Furthermore, there is a bound l on the size of the subcategories of categories on the right side of any rule in (10) and (11). Let $m = max(k, l)$.

For rules from schema class (1), the category on the right is a subcategory of the first category on the left, so the subcategories on the right are bound by m. For rules from schema classes (2-3), the category on the right has subcategories X and Z each of which is bound in size by m since they occur as subcategories of categories on the left.

For rules from schema class (6), since reducing generalized composition is a special case of re-

[3] Baldridge (2002) introduced a variant of CCG where modalities are added to the connectives / and \ along with variants of the combinatory rules based on these modalities. Our proofs about restriction class (B) are essentially identical to proofs regarding the multi-modal variant.

ducing generalized crossing composition, we need only consider the latter. The category on the right has subcategories $X|_1Z_1|_2 \ldots |_{n-1}|Z_{n-1}$ and Z_n. Z_n is bound in size by m because it occurs as a subcategory of the second category on the left. Then, the size of $Y|_1Z_1|_2 \ldots |_{n-1}|Z_{n-1}$ must be bound by m and since $|X| \leq |Y|$, the size of $X|_1Z_1|_2 \ldots |_{n-1}|Z_{n-1}$ must also be bound by m.

For rules from schema class (7), the category on the right has subcategories X and Z. The size of Z is bound by m because it is a subcategory of a category on the left. The size of X is bound by m because it is a second subcategory of a category on the left.

Finally, the use of rules in schema classes (10-11) have categories on the right that are bounded by l, which is, in turn, bounded by m. Then, by proposition 1, there must only be a finite number of categories in any derivation in a CCG consisting of a subset of rule schemata (1-3), (6-7) and (10-11).

The proof including schema class (8) is essentially identical except that k must be defined in terms of the size of the second subcategories. □

Definition 2. A grammar is *strongly context-free* if there exists a CFG such that the derivations of the two grammars are identical.

Proposition 3. *Any CCG consisting of a subset of the schema classes (1-3), (6-8) and (10-11) is strongly context-free.*

Proof. Since the CCG generates derivations whose categories are finite in number let C be that set of categories. Let $S(C, X)$ be the subset of C matching category X (which may have variables). Then, for each rule schema $C_1, C_2 \rightarrow C_3$ in (1-3) and (6-8), we construct a context-free rule $C'_3 \rightarrow C'_1, C'_2$ for each C'_i in $S(C, C_i)$ for $1 \leq i \leq 3$. Similarly, for each rule schema $C_1 \rightarrow C_2$ in (10), we construct a context-free rule $C'_2 \rightarrow C'_1$ which results in a finite number of such rules. Finally, for each rule schema $\vec{X} \rightarrow Z$ in (11) we construct a context-free rule $Z \rightarrow \vec{X}$. Then, for each entry in the lexicon $w \rightarrow C$, we construct a context-free rule $C \rightarrow w$.

The constructed CFG has precisely the same rules as the CCG restricted to the categories in C except that the left and right sides have been reversed. Thus, by proposition 2, the CFG has exactly the same derivations as the CCG. □

Proposition 4. *Any CCG consisting of a subset of the schema classes (1-3), (6-8) and (10-11) along with restriction class (B) is strongly context-free.*

Proof. If a CCG is allowed to restrict the use of its rules to certain categories as in schema class (B), then when we construct the context-free rules by enumerating only those categories in the set C allowed by the restriction. □

Proposition 5. *Any CCG that includes restriction class (A) is strongly context-free.*

Proof. We construct a context-free grammar with exactly those rules in the finite set of instantiations of the CCG rule schemata along with context-free rules corresponding to the lexicon. This CFG generates exactly the same derivations as the CCG. □

We have thus proved that of a wide range of the rule schemata used to define CCGs are context-free.

2.3 Combinatory Categorial Grammars in Practice

CCGbank (Hockenmaier and Steedman, 2007) is a corpus of CCG derivations that was semi-automatically converted from the Wall Street Journal section of the Penn treebank. Figure 2 shows a categorization of the rules used in CCGbank according to the schema classes defined in the preceding section where a rule is placed into the least general class to which it belongs. In addition to having no generalized composition other than the reducing variant, it should also be noted that in all generalized composition rules, $X = Y$ implying that the reducing class of generalized composition is a very natural schema class for CCGbank.

If we assume that type-raising is restricted to those instances occurring in CCGbank[4], then a CCG consisting of schema classes (1-3), (6-7) and (10-11) can generate all the derivations in CCGbank. By proposition 3, such a CCG is strongly context-free. One could also observe that since CCGbank is finite, its grammar is not only a context-free grammar but can produce only a finite number of derivations. However, our statement is much stronger because this CCG can generate all of the derivations in CCGbank given only the lexicon, the finite set of unrestricted rules and the finite number of type-raising rules.

[4]Without such an assumption, parsing is intractable.

Schema Class	*Rules*	*Instances*
Application	519	902176
Composition	102	7189
Crossed Composition	64	14114
Reducing Generalized Crossed Composition	50	612
Generalized Composition	0	0
Generalized Crossed Composition	0	0
Substitution	3	4
Type-Raising	27	3996
Unrestricted Rules	642	335011
Total	1407	1263102

Figure 2: The rules of CCGbank by schema class.

The Clark and Curran CCG Parser (Clark and Curran, 2007) is a CCG parser which uses CCGbank as a training corpus. Despite the fact that there is a strongly context-free CCG which generates all of the derivations in CCGbank, it is still possible that the grammar learned by the Clark and Curran parser is not a context-free grammar. However, in addition to rule schemata (1-6) and (10-11) they also include restriction class (A) by restricting rules to only those found in the training data[5]. Thus, by proposition 5, the Clark and Curran parser is a context-free parser.

3 A Latent Variable CCG Parser

The context-freeness of a number of CCGs should not be considered evidence that there is no advantage to CCG as a grammar formalism. Unlike the context-free grammars extracted from the Penn treebank, these allow for the categorial semantics that accompanies any categorial parse and for a more elegant analysis of linguistic structures such as extraction and coordination. However, because we now know that the CCG defined by CCGbank is strongly context-free, we can use tools from the CFG parsing community to improve CCG parsing.

To illustrate this point, we train the Petrov parser (Petrov and Klein, 2007) on CCGbank. The Petrov parser uses latent variables to refine a coarse-grained grammar extracted from a training corpus to a grammar which makes much more fine-grained syntactic distinctions. For example, in Petrov's experiments on the Penn treebank, the syntactic category NP was refined to the more fine-grained NP^1 and NP^2 roughly corresponding to NPs in subject and object positions. Rather than requiring such distinctions to be made in the corpus, the Petrov parser hypothesizes these splits automatically.

The Petrov parser operates by performing a fixed number of iterations of splitting, merging and smoothing. The splitting process is done by performing Expectation-Maximization to determine a likely potential split for each syntactic category. Then, during the merging process some of the splits are undone to reduce grammar size and avoid overfitting according to the likelihood of the split against the training data.

The Petrov parser was chosen for our experiments because it refines the grammar in a mathematically principled way without altering the nature of the derivations that are output. This is important because the input to the semantic backend and the system that converts CCG derivations to dependencies requires CCG derivations as they appear in CCGbank.

3.1 Experiments

Our experiments use CCGbank as the corpus and we use sections 02-21 for training (39603 sentences), 00 for development (1913 sentences) and 23 for testing (2407 sentences).

CCGbank, in addition to the basic atoms S, N, NP and PP, also differentiates both the S and NP atoms with *features* allowing more subtle distinctions. For example, declarative sentences are $S[dcl]$, wh-questions are $S[wq]$ and sentence fragments are $S[frg]$ (Hockenmaier and Steedman, 2007). These features allow finer control of the use of combinatory rules in the resulting grammars. However, this fine-grained control is exactly what the Petrov parser does automatically. Therefore, we trained the Petrov parser twice, once on the original version of CCGbank (denoted "Petrov") and once on a version of CCGbank without these features (denoted "Petrov no feats"). Furthermore, we will evaluate the parsers obtained after 0, 4, 5 and 6 training iterations (denoted I-0, I-4, I-5 and I-6). When we evaluate on sets of sentences for which not all parsers return an analysis, we report the coverage (denoted "Cover").

We use the `evalb` package for PARSEVAL evaluation and a modified version of Clark and

[5]The Clark and Curran parser has an option, which is disabled by default, for not restricting the rules to those that appear in the training data. However, they find that this restriction is "detrimental to neither parser accuracy or coverage" (Clark and Curran, 2007).

Parser	*Accuracy %*	*No feats %*
C&C Normal Form	92.92	93.38
C&C Hybrid	93.06	93.52
Petrov I-5	**93.18**	93.73
Petrov no feats I-6	-	**93.74**

Figure 3: Supertagging accuracy on the sentences in section 00 that receive derivations from the four parsers shown.

Parser	*Accuracy %*	*No feats %*
C&C Hybrid	92.98	93.43
Petrov I-5	**93.10**	93.59
Petrov no feats I-6	-	**93.62**

Figure 4: Supertagging accuracy on the sentences in section 23 that receive derivations from the three parsers shown.

Curran's `evaluate` script for dependency evaluation. To determine statistical significance, we obtain p-values from Bikel's randomized parsing evaluation comparator[6], modified for use with tagging accuracy, F-score and dependency accuracy.

3.2 Supertag Evaluation

Before evaluating the parse trees as a whole, we evaluate the categories assigned to words. In the supertagging literature, POS tagging and supertagging are distinguished – POS tags are the traditional Penn treebank tags (e.g. NN, VBZ and DT) and supertags are CCG categories. However, because the Petrov parser trained on CCGbank has no notion of Penn treebank POS tags, we can only evaluate the accuracy of the supertags.

The results are shown in figures 3 and 4 where the "Accuracy" column shows accuracy of the supertags against the CCGbank categories and the "No feats" column shows accuracy when features are ignored. Despite the lack of POS tags in the Petrov parser, we can see that it performs slightly better than the Clark and Curran parser. The difference in accuracy is only statistically significant between Clark and Curran's Normal Form model ignoring features and the Petrov parser trained on CCGbank without features (p-value = 0.013).

3.3 Constituent Evaluation

In this section we evaluate the parsers using the traditional PARSEVAL measures which measure recall, precision and F-score on constituents in both labeled and unlabeled versions. In addition, we report a variant of the labeled PARSEVAL measures where we ignore the features on the categories. For reasons of brevity, we report the PARSEVAL measures for all sentences in sections 00 and 23, rather than for sentences of length is less than 40 or less than 100. The results are essentially identical for those two sets of sentences.

Figure 5 gives the PARSEVAL measures on section 00 for Clark and Curran's two best models and the Petrov parser trained on the original CCGbank and the version without features after various numbers of training iterations. Figure 7 gives the accuracies on section 23.

In the case of Clark and Curran's hybrid model, the poor accuracy relative to the Petrov parsers can be attributed to the fact that this model chooses derivations based on the associated dependencies at the expense of constituent accuracy (see section 3.4). In the case of Clark and Curran's normal form model, the large difference between labeled and unlabeled accuracy is primarily due to the mislabeling of a small number of features (specifically, NP[nb] and NP[num]). The labeled accuracies without features gives the results when features are disregarded.

Due to the similarity of the accuracies and the difference in the coverage between I-5 of the Petrov parser on CCGbank and I-6 of the Petrov parser on CCGbank without features, we reevaluate their results on only those sentences for which they both return derivations in figures 6 and 8. These results show that the features in CCGbank actually inhibit accuracy (to a statistically significant degree in the case of unlabeled accuracy on section 00) when used as training data for the Petrov parser.

Figure 9 gives a comparison between the Petrov parser trained on the Penn treebank and on CCGbank. These numbers should not be directly compared, but the similarity of the unlabeled measures indicates that the difference between the structure of the Penn treebank and CCGbank is not large.[7]

3.4 Dependency Evaluation

The constituent-based PARSEVAL measures are simple to calculate from the output of the Petrov parser but the relationship of the PARSEVAL

[6] http://www.cis.upenn.edu/ dbikel/software.html

[7] Because punctuation in CCG can have grammatical function, we include it in our accuracy calculations resulting in lower scores for the Petrov parser trained on the Penn treebank than those reported in Petrov and Klein (2007).

Parser	Labeled %			Labeled no feats %			Unlabeled %			Cover
	R	*P*	*F*	*R*	*P*	*F*	*R*	*P*	*F*	
C&C Normal Form	71.14	70.76	70.95	80.66	80.24	80.45	86.16	85.71	85.94	98.95
C&C Hybrid	50.08	49.47	49.77	58.13	57.43	57.78	61.27	60.53	60.90	98.95
Petrov I-0	74.19	74.27	74.23	74.66	74.74	74.70	78.65	78.73	78.69	99.95
Petrov I-4	85.86	85.78	85.82	86.36	86.29	86.32	89.96	89.88	89.92	99.90
Petrov I-5	**86.30**	**86.16**	**86.23**	86.84	86.70	86.77	90.28	90.13	90.21	99.90
Petrov I-6	85.95	85.68	85.81	86.51	86.23	86.37	90.22	89.93	90.08	99.22
Petrov no feats I-0	-	-	-	72.16	72.59	72.37	76.52	76.97	76.74	99.95
Petrov no feats I-5	-	-	-	86.67	86.57	86.62	90.30	90.20	90.25	99.90
Petrov no feats I-6	-	-	-	**87.45**	**87.37**	**87.41**	**90.99**	**90.91**	**90.95**	99.84

Figure 5: Constituent accuracy on all sentences from section 00.

Parser	Labeled %			Labeled no feats %			Unlabeled %		
	R	*P*	*F*	*R*	*P*	*F*	*R*	*P*	*F*
Petrov I-5	86.56	86.46	86.51	87.10	87.01	87.05	90.43	90.33	90.38
Petrov no feats I-6	-	-	-	**87.45**	**87.37**	**87.41**	**90.99**	**90.91**	**90.95**
p-value	-	-	-	0.089	0.090	0.088	0.006	0.008	0.007

Figure 6: Constituent accuracy on the sentences in section 00 that receive a derivation from both parsers.

Parser	Labeled %			Labeled no feats %			Unlabeled %			Cover
	R	*P*	*F*	*R*	*P*	*F*	*R*	*P*	*F*	
C&C Normal Form	71.15	70.79	70.97	80.73	80.32	80.53	86.31	85.88	86.10	99.58
Petrov I-5	**86.94**	**86.80**	**86.87**	87.47	87.32	87.39	90.75	90.59	90.67	99.83
Petrov no feats I-6	-	-	-	**87.49**	**87.49**	**87.49**	**90.81**	**90.82**	**90.81**	99.96

Figure 7: Constituent accuracy on all sentences from section 23.

Parser	Labeled %			Labeled no feats %			Unlabeled %		
	R	*P*	*F*	*R*	*P*	*F*	*R*	*P*	*F*
Petrov I-5	86.94	86.80	86.87	87.47	87.32	87.39	90.75	90.59	90.67
Petrov no feats I-6	-	-	-	**87.48**	**87.49**	**87.49**	**90.81**	**90.82**	**90.81**
p-value	-	-	-	0.463	0.215	0.327	0.364	0.122	0.222

Figure 8: Constituent accuracy on the sentences in section 23 that receive a derivation from both parsers.

Parser	Labeled %			Unlabeled %			Cover
	R	*P*	F	*R*	*P*	*F*	
Petrov on PTB I-6	89.65	89.97	89.81	90.80	91.13	90.96	100.00
Petrov on CCGbank I-5	86.94	86.80	86.87	90.75	90.59	90.67	99.83
Petrov on CCGbank no feats I-6	87.49	87.49	87.49	90.81	90.82	90.81	99.96

Figure 9: Constituent accuracy for the Petrov parser on the corpora on all sentences from Section 23.

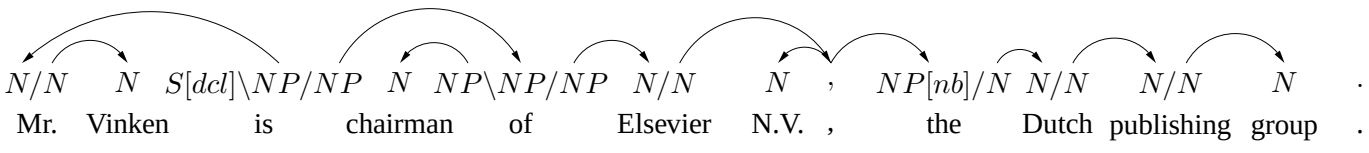

Figure 10: The argument-functor relations for the CCG derivation in figure 1.

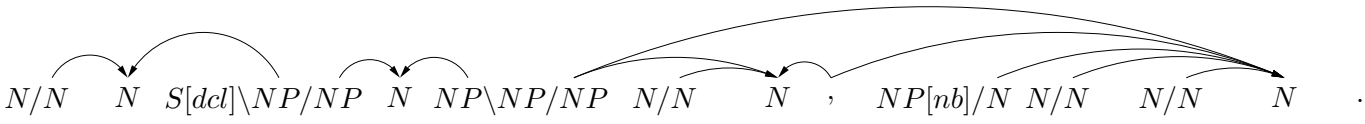

Figure 11: The set of dependencies obtained by reorienting the argument-functor edges in figure 10.

	Labeled %			*Unlabeled %*			
Parser	*R*	*P*	F	*R*	*P*	*F*	*Cover*
C&C Normal Form	84.39	85.28	84.83	90.93	91.89	91.41	98.95
C&C Hybrid	84.53	**86.20**	85.36	90.84	**92.63**	91.73	98.95
Petrov I-0	79.87	78.81	79.34	87.68	86.53	87.10	96.45
Petrov I-4	84.76	85.27	85.02	91.69	92.25	91.97	96.81
Petrov I-5	**85.30**	85.87	**85.58**	**92.00**	92.61	**92.31**	96.65
Petrov I-6	84.86	85.46	85.16	91.79	92.44	92.11	96.65

Figure 12: Dependency accuracy on CCGbank dependencies on all sentences from section 00.

	Labeled %			*Unlabeled %*		
Parser	*R*	*P*	F	*R*	*P*	*F*
C&C Hybrid	84.71	**86.35**	85.52	90.96	92.72	91.83
Petrov I-5	**85.50**	86.08	**85.79**	**92.12**	**92.75**	**92.44**
p-value	0.005	0.189	0.187	< 0.001	0.437	0.001

Figure 13: Dependency accuracy on the section 00 sentences that receive an analysis from both parsers.

	Labeled %			*Unlabeled %*		
Parser	*R*	*P*	F	*R*	*P*	*F*
C&C Hybrid	85.11	**86.46**	85.78	91.15	92.60	91.87
Petrov I-5	**85.73**	86.29	**86.01**	**92.04**	**92.64**	**92.34**
p-value	0.013	0.278	0.197	< 0.001	0.404	0.005

Figure 14: Dependency accuracy on the section 23 sentences that receive an analysis from both parsers.

Parser	*Training Time in CPU minutes*	*Parsing Time in CPU minutes*	*Training RAM in gigabytes*
Clark and Curran Normal Form Model	1152	2	28
Clark and Curran Hybrid Model	2672	4	37
Petrov on PTB I-0	1	5	2
Petrov on PTB I-5	180	20	8
Petrov on PTB I-6	660	21	16
Petrov on CCGbank I-0	1	5	2
Petrov on CCGbank I-4	103	70	8
Petrov on CCGbank I-5	410	600	14
Petrov on CCGbank I-6	2760	2880	24
Petrov on CCGbank no feats I-0	1	5	2
Petrov on CCGbank no feats I-5	360	240	7
Petrov on CCGbank no feats I-6	1980	390	13

Figure 15: Time and space usage when training on sections 02-21 and parsing on section 00.

scores to the quality of a parse is not entirely clear. For this reason, the word to word dependencies of categorial grammar parsers are often evaluated. This evaluation is aided by the fact that in addition to the CCG derivation for each sentence, CCGbank also includes a set of dependencies. Furthermore, extracting dependencies from a CCG derivation is well-established (Clark et al., 2002).

A CCG derivation can be converted into dependencies by, first, determining which arguments go with which functors as specified by the CCG derivation. This can be represented as in figure 10. Although this is not difficult, some care must be taken with respect to punctuation and the conjunction rules. Next, we reorient some of the edges according to information in the lexical categories. A language for specifying these instructions using variables and indices is given in Clark et al. (2002). This process is shown in figures 1, 10 and 11 with the directions of the dependencies reversed from Clark et al. (2002).

We used the CCG derivation to dependency converter `generate` included in the C&C tools package to convert the output of the Petrov parser to dependencies. Other than a CCG derivation, their system requires only the lexicon of edge reorientation instructions and methods for converting the unrestricted rules of CCGbank into the argument-functor relations. Important for the purpose of comparison, this system does not depend on their parser.

An unlabeled dependency is correct if the ordered pair of words is correct. A labeled dependency is correct if the ordered pair of words is correct, the head word has the correct category and the position of the category that is the source of that edge is correct. Figure 12 shows accuracies from the Petrov parser trained on CCGbank along with accuracies for the Clark and Curran parser. We only show accuracies for the Petrov parser trained on the original version of CCGbank because the dependency converter cannot currently generate dependencies for featureless derivations.

The relatively poor coverage of the Petrov parser is due to the failure of the dependency converter to output dependencies from valid CCG derivations. However, the coverage of the dependency converter is actually lower when run on the gold standard derivations indicating that this coverage problem is not indicative of inaccuracies in the Petrov parser. Due to the difference in coverage, we again evaluate the top two parsers on only those sentences that they both generate dependencies for and report those results in figures 13 and 14. The Petrov parser has better results by a statistically significant margin for both labeled and unlabeled recall and unlabeled F-score.

3.5 Time and Space Evaluation

As a final evaluation, we compare the resources that are required to both train and parse with the Petrov parser on the Penn Treebank, the Petrov parser on the original version of CCGbank, the Petrov parser on CCGbank without features and the Clark and Curran parser using the two models. All training and parsing was done on a 64-bit machine with 8 dual core 2.8 Ghz Opteron 8220 CPUs and 64GB of RAM. Our training times are much larger than those reported in Clark and Curran (2007) because we report the cumulative time spent on all CPUs rather than the maximum time spent on a CPU. Figure 15 shows the results.

As can be seen, the Clark and Curran parser has similar training times, although significantly greater RAM requirements than the Petrov parsers. In contrast, the Clark and Curran parser is significantly faster than the Petrov parsers, which we hypothesize to be attributed to the degree to which Clark and Curran have optimized their code, their use of `C++` as opposed to `Java` and their use of a supertagger to prune the lexicon.

4 Conclusion

We have provided a number of theoretical results proving that CCGbank contains no non-context-free structure and that the Clark and Curran parser is actually a context-free parser. Based on these results, we trained the Petrov parser on CCGbank and achieved state of the art results in terms of supertagging accuracy, PARSEVAL measures and dependency accuracy.

This demonstrates the following. First, the ability to extract semantic representations from CCG derivations is not dependent on the language class of a CCG. Second, using a dedicated supertagger, as opposed to simply using a general purpose tagger, is not necessary to accurately parse with CCG.

Acknowledgments

We would like to thank Stephen Clark, James Curran, Jackie C. K. Cheung and our three anonymous reviewers for their insightful comments.

References

J. Baldridge. 2002. *Lexically Specified Derivational Control in Combinatory Categorial Grammar*. Ph.D. thesis, University of Edinburgh.

J. Bos, S. Clark, M. Steedman, J. R Curran, and J. Hockenmaier. 2004. Wide-coverage semantic representations from a CCG parser. In *Proceedings of COLING*, volume 4, page 1240–1246.

S. Clark and J. R. Curran. 2007. Wide-Coverage efficient statistical parsing with CCG and Log-Linear models. *Computational Linguistics*, 33(4):493–552.

S. Clark, J. Hockenmaier, and M. Steedman. 2002. Building deep dependency structures with a wide-coverage CCG parser. In *Proceedings of the 40th Meeting of the ACL*, page 327–334.

J. Hockenmaier and M. Steedman. 2007. CCGbank: a corpus of CCG derivations and dependency structures extracted from the penn treebank. *Computational Linguistics*, 33(3):355–396.

F. Hoyt and J. Baldridge. 2008. A logical basis for the d combinator and normal form in CCG. In *Proceedings of ACL-08: HLT*, page 326–334, Columbus, Ohio. Association for Computational Linguistics.

J. Lambek. 1958. The mathematics of sentence structure. *American Mathematical Monthly*, 65(3):154–170.

S. Petrov and D. Klein. 2007. Improved inference for unlexicalized parsing. In *Proceedings of NAACL HLT 2007*, page 404–411.

M. Steedman. 2000. *The syntactic process*. MIT Press.

K. Vijay-Shanker and D. Weir. 1994. The equivalence of four extensions of context-free grammars. *Mathematical Systems Theory*, 27(6):511–546.

W. Zielonka. 1981. Axiomatizability of Ajdukiewicz-Lambek calculus by means of cancellation schemes. *Zeitschrift fur Mathematische Logik und Grundlagen der Mathematik*, 27:215–224.