Probing the Linguistic Strengths and Limitations of Unsupervised Grammar Induction

Yonatan Bisk and **Julia Hockenmaier**
Department of Computer Science
The University of Illinois at Urbana-Champaign
201 N Goodwin Ave Urbana, IL 61801
{bisk1,juliahmr@illinois.edu}

Abstract

Work in grammar induction should help shed light on the amount of syntactic structure that is discoverable from raw word or tag sequences. But since most current grammar induction algorithms produce unlabeled dependencies, it is difficult to analyze what types of constructions these algorithms can or cannot capture, and, therefore, to identify where additional supervision may be necessary. This paper provides an in-depth analysis of the errors made by unsupervised CCG parsers by evaluating them against the labeled dependencies in CCGbank, hinting at new research directions necessary for progress in grammar induction.

1 Introduction

Grammar induction aims to develop algorithms that can automatically discover the latent syntactic structure of language from raw or part-of-speech tagged text. While such algorithms would have the greatest utility for low-resource languages for which no treebank is available to train supervised parsers, most work in this area has focused on languages where existing treebanks can be used to measure and compare the performance of the resultant parsers. Despite significant progress in the last decade (Klein and Manning, 2004; Headden III et al., 2009; Blunsom and Cohn, 2010; Spitkovsky et al., 2013; Mareček and Straka, 2013), there has been little analysis performed on the types of errors these induction systems make, and our understanding of what kinds of constructions these parsers can or cannot recover is still rather limited. One likely reason for this lack of analysis is the fact that most of the work in this domain has focused on parsers that return unlabeled dependencies, which cannot easily be assigned a linguistic interpretation.

This paper shows that approaches that are based on categorial grammar (Steedman, 2000) are amenable to more stringent evaluation metrics, which enable detailed analyses of the constructions they capture, while the commonly used unlabeled directed attachment scores hide linguistically important errors. Any categorial grammar based system, whether deriving its grammar from seed knowledge distinguishing nouns and verbs (Bisk and Hockenmaier, 2013), from a lexicon constructed from a simple questionnaire for linguists (Boonkwan and Steedman, 2011), or from sections of a treebank (Garrette et al., 2015), will attach linguistically expressive categories to individual words, and can therefore produce labeled dependencies. We provide a simple proof of concept for how these labeled dependencies can be used to isolate problem areas in CCG induction algorithms. We illustrate how they make the linguistic assumptions and mistakes of the model transparent, and are easily comparable to a treebank where available. They also allow us to identify linguistic phenomena that require additional supervision or training signal to master. Our analysis will be based on extensions of our earlier system (Bisk and Hockenmaier, 2013), since it requires less supervision than the CCG-based approaches of Boonkwan and Steedman (2011) or Garrette et al. (2015). Our aim in presenting this analysis is to initiate a broader conversation and classification of the impact of various types of supervision provided to these approaches. We will see that most of the constructions that our system cannot capture, even when they are included in the model's search space, involve precisely the kinds of non-local dependencies that elude even supervised dependency parsers (since they require dependency graphs, instead of trees), and that have motivated the use of categorial grammar-based approaches for supervised parsing.

Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, pages 1395–1404,
Beijing, China, July 26-31, 2015.

First, we provide a brief introduction to CCG. Next, we define a labeled evaluation metric that allows us to compare the labeled dependencies produced by Bisk and Hockenmaier (2013)'s unsupervised parser with those in CCGbank (Hockenmaier and Steedman, 2007). Third, we extend their induction algorithm to allow it to induce more complex categories, and refine their probability model to handle punctuation and lexicalization, which we show to be necessary when handling the larger grammars induced by our variant of their algorithm. While we also perform a traditional dependency evaluation for comparison to the non-CCG based literature, we focus on our CCG-based labeled evaluation metrics to perform a comparative analysis of Bisk and Hockenmaier (2013)'s parser and our extensions.

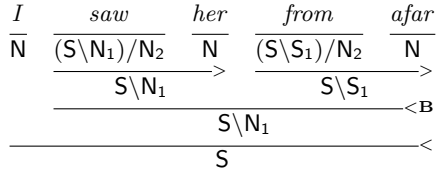
2 Combinatory Categorial Grammar

CCG categories CCG (Steedman, 2000) is a lexicalized grammar formalism which associates each word with a set of lexical categories that fully specify its syntactic behavior. Lexical categories indicate the expected number, type and relative location of arguments a word should take, or what constituents it may modify. Even without explicit evaluation against a treebank, the CCG lexicon that an unsupervised parser produces provides an easily interpretable snapshot of the assumptions the model has made about a language (Bisk and Hockenmaier, 2013). The set of CCG categories is defined recursively over a small set of atomic categories (e.g. $\mathsf{S}, \mathsf{N}, \mathsf{NP}, \mathsf{PP}$). Complex categories take the form $\mathsf{X}\backslash\mathsf{Y}$ or X/Y and represent functions which create a result of category X when combined with an argument Y. The slash indicates whether the argument precedes ($\backslash$) or follows ($/$) the functor (descriptions of CCG commonly use the vertical slash $|$ to range over both $/$ and $\backslash$). Modifiers are categories of the form $\mathsf{X}|\mathsf{X}$, and may take arguments of their own.

CCG rules CCG rules are defined schematically as function application ($>, <$), unary ($>\mathsf{B}_1, <\mathsf{B}_1$) and generalized composition ($>\mathsf{B}_n, <\mathsf{B}_n$), type-raising ($>\mathsf{T}, <\mathsf{T}$) and conjunction:

$\mathsf{X/Y}$	Y	$\Rightarrow_{>}$	X
$\mathsf{X/Y}$	$\mathsf{Y\|Z}$	$\Rightarrow_{>\mathsf{B}_1}$	$\mathsf{X\|Z}$
$\mathsf{X/Y}$	$\mathsf{Y\|Z_1\|...\|Z_n}$	$\Rightarrow_{>\mathsf{B}_n}$	$\mathsf{X\|Z_1\|...\|Z_n}$
X		$\Rightarrow_{>\mathsf{T}}$	$\mathsf{T/(T\backslash X)}$
Y	$\mathsf{X\backslash Y}$	$\Rightarrow_{<}$	X
$\mathsf{Y\|Z}$	$\mathsf{X\backslash Y}$	$\Rightarrow_{<\mathsf{B}_1}$	$\mathsf{X\|Z}$
$\mathsf{Y\|Z_1\|...\|Z_n}$	$\mathsf{X\backslash Y}$	$\Rightarrow_{<\mathsf{B}_n}$	$\mathsf{X\|Z_1\|...\|Z_n}$
	X	$\Rightarrow_{<\mathsf{T}}$	$\mathsf{T\backslash(T/X)}$

CCG derivations In the following derivation, forward application is used in line 1) as both the verb and the preposition take their NP arguments. In line 2), the prepositional phrase modifies the verb via backwards composition. Finally, in line 3), the derivation completes by producing a sentence (S) via backwards application:



CCG dependencies CCG has two standard evaluation metrics. Supertagging accuracy simply computes how often a model chooses the correct lexical category for a given word. The correct category is a prerequisite for recovering the correct labeled dependency. By tracing through which word fills which argument of which category, a set of dependency arcs, labeled by lexical category and slot, can be extracted:

In this example, *I* fills the first argument of *saw*. This is represented by an edge from *saw* to *I*, labeled as a transitive verb ($(\mathsf{S\backslash N})/\mathsf{N}$). This procedure is followed for every argument of every predicate, leading to a labeled directed graph.

Evaluation metrics for supervised CCG parsers (Clark et al., 2002) measure labeled f-score (LF1) precision of these dependencies (requiring the functor, argument, lexical category of the functor and slot of the argument to all match). A second, looser, evaluation is often also performed which measures unlabeled, undirected dependency scores (UF1).

Non-local dependencies and complex arguments One advantage of CCG is its ability to recover the non-local dependencies involved in control, raising, or *wh*-extraction. Since these constructions introduce additional dependencies, CCG parsers return dependency graphs (DAGs), not trees. To obtain these additional dependencies, relative pronouns and control verbs require lexical categories that take complex arguments of the form $\mathsf{S\backslash NP}$ or $\mathsf{S/NP}$, and a mechanism for co-indexation of the NP inside this argument with another NP argument (e.g. $(\mathsf{NP\backslash NP_i})/(\mathsf{S|NP_i})$ for relative pronouns). These co-indexed subjects can be seen in Figure 1.

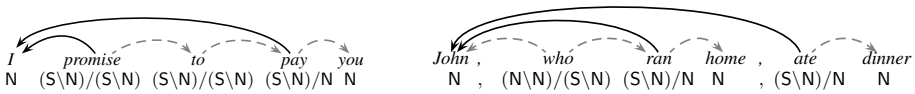
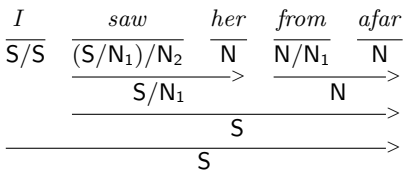
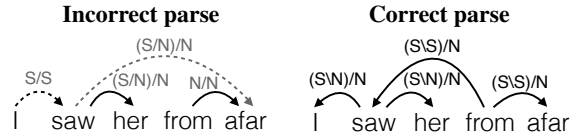


Figure 1: Unlabeled predicate-argument dependency graphs for two sentences with co-indexed subjects.

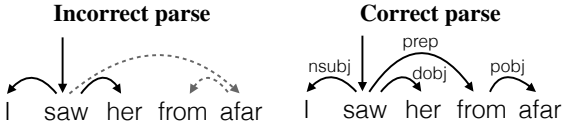
Errors exposed by labeled evaluation We now illustrate how the lexical categories and labeled dependencies produced by CCG parsers expose linguistic mistakes. First, we consider a wildly incorrect analysis of the first example sentence, in which the subject is treated as an adverb, and the PP as an NP object of the verb:



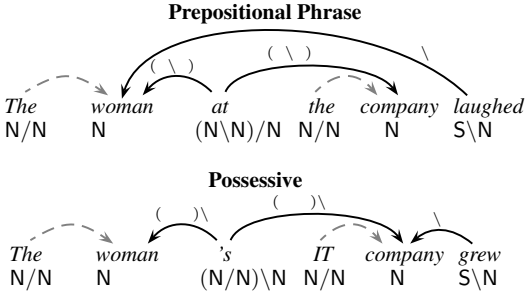
None of the labeled directed CCG dependencies are correct. But under the more lenient unlabeled directed evaluation of Garrette et al. (2015), and the even more lenient unlabeled undirected metric of Clark et al. (2002), two (or three) of the four dependencies would be deemed correct:



When we translate the CCG analysis to an unlabeled dependency tree (and hence flip the direction of modifier dependencies and add a root edge), a similar picture emerges, and three out of five attachments are deemed correct:



We now turn to a subtle distinction that corresponds to a systematic mistake made by all models we evaluate. The categories of noun-modifying prepositions (*at*) and possessive markers (*'*) differ only in the directionality of their slashes:



The unlabeled dependencies inside the noun phrases are identical, but the heads differ. The first sentence turns the prepositional phrase (*at the company*) into a modifier of *woman*. In contrast, in the possessive case, *woman 's* modifies *company*. According to an unlabeled (directed) score, confusing these analyses would be 80% correct, whereas LF1 would only be 20%. But without a semantic bias for companies growing and women laughing, there is no signal for the learner.

3 Labeled Evaluation for CCG Induction

We have just seen that labeled evaluation can expose many linguistically important mistakes. In order to enable a fair and informative comparison of unsupervised CCG parsers against the lexical categories and labeled dependencies in CCGbank, we define a simplification of CCGbank's lexical categories that does not alter the number or direction of dependencies, but makes the categories and dependency labels directly comparable to those produced by an unsupervised parser. We also do not alter the derivations themselves, although these may contain type-changing rules (which allow e.g. participial verb phrases S[ng]\NP to be used as NP modifiers NP\NP) that are beyond the scope of our induction algorithm.

Although the CCG derivations and dependencies that CCG-based parsers return should in principle be amenable to a quantitative labeled evaluation when a gold-standard CCG corpus is available, there may be minor systematic differences between the sets of categories assumed by the induced parser and those in the treebank. In particular, the lexical categories in the English CCGbank are augmented with morphosyntactic features that indicate e.g. whether sentences are declarative (S[dcl]), or verb phrases are infinitival (S[to]\NP). Prior work on supervised parsing with CCG found that many of these features can be recovered with proper modeling of latent state splitting (Fowler and Penn, 2010). Since we wish to evaluate a system that does not aim to induce such features, we remove them. We also remove the distinction between noun phrases (NP) and nouns (N), which is predicated on knowledge of

	Our simplification of CCGbank's lexical categories					
	Congress	*has*	*n't*	*lifted*	*the*	*ceiling*
Original	NP	(S[dcl]\NP)/(S[pt]\NP)	(S\NP)\(S\NP)	(S[pt]\NP)/NP	NP[nb]/N	N
Simplified	N	(S\N)/(S\N)	S\S	(S\N)/N	N/N	N

Figure 2: We remove morphosyntactic features, simplify verb phrase modifiers, and change NP to N.

	CCGbank	w/out Feats	Simplified
All	1640	458	444
Lexical	1286	393	384

Table 1: Category types in CCGbank 02-21

determiners and other structural elements of a language. Finally, CCGbank distinguishes between sentential modifiers (which have categories of the form S|S, without features) and verb phrase modifiers ((S\NP)|(S\NP), again without features). But since the NP argument slot of a VP modifier is never filled, we can maintain the same number of gold standard dependencies by removing this distinction and changing all VP modifiers to be of the form S|S. However, categories of the form $(S[\cdot]\backslash NP_i)/(S[\cdot]\backslash NP_i)$, which are used e.g. for modals and auxiliaries, are changed to $(S\backslash N_i)/(S\backslash N_i)$, allowing us to maintain the dependency on the subject. With these three simplifications we eliminate much of the detailed knowledge required to construct the precise CCGbank-style categories, and dramatically reduce the set of categories without losing expressive power. One distinction that we do not conflate, even though it is currently beyond the scope of the induction algorithm, is the distinction between PP arguments (requiring prepositions to have the category PP/NP) and adjuncts (requiring prepositions to be (NP\NP)/NP or ((S\NP)\(S\NP))/NP).

This simplification is consistent with the most basic components of CCG and can therefore be easily used for the evaluation and analysis of any weakly or fully supervised CCG system, not just that of Bisk and Hockenmaier (2012). An example simplification is present in Figure 2, and the reduction in the set of categories can be seen in Table 1. Similar simplifications should also be possible for CCGbanks in other languages.

4 Our approach

There are two parts to our approach: 1) inducing a CCG grammar from seed knowledge and 2) learning a probability model over parses. The induction algorithm (Bisk and Hockenmaier, 2012) uses the seed knowledge that nouns can take the CCG category N, that verbs can take the category S and may take N arguments, and that any word may modify a constituent it is adjacent to, to iteratively induce a CCG lexicon to parse the training data. In Bisk and Hockenmaier (2013), we introduced a model that is based on Hierarchical Dirichlet Processes (Teh et al., 2006). This HDP-CCG model gave state-of-the-art performance on a number languages, and qualitative analysis of the resultant lexicons indicated that the system was learning the word order and many of the correct attachments of the tested languages. But this system also had a number of shortcomings: the induction algorithm was restricted to a small fragment of CCG, the model emitted only POS tags rather than words, and punctuation was ignored. Here, we use our previous HDP-CCG system as a baseline, and introduce three novel extensions that attempt to address these concerns.

5 Experimental Setup

For our experiments we will follow the standard practice in supervised parsing of using WSJ Sections 02-21 for training, Section 22 for development and error analysis, and a final evaluation of the best models on Section 23. Because the induced lexicons are overly general, the memory footprint grows rapidly as the complexity of the grammar increases. For this reason, we only train on sentences that contain up to 20 words (as well as an arbitrary number of punctuation marks). All analyses and evaluation are performed with sentences of all lengths unless otherwise indicated. Finally, Bisk and Hockenmaier (2013) followed Liang et al. (2007) in setting the values of the hyperparameters α to powers (eg. the square) of the number of observed outcomes in the distribution. But when the output consists of words rather than POS tags, the concentration parameter $\alpha = V^2$ is too large to allow the model to learn. For this reason, experiments will be reported with all hyperparameters set to a constant of 2500.[1]

[1] We tested three values (1000, 2500, 5000) and found that the basic model at 2500 performed closest to the previously

		Base	+ Lexicalization	+ Punctuation	+ Punc & Lex	+ Allow (X|X)|X
Only Atomic Arguments	B_1	**34.2**	35.2	36.3	36.9	36.8
(S, N)	B_3	34.4	35.1	33.8	38.9	**38.8**
Allow Complex Arguments	B_1	33.0	34.9	33.2	35.7	**35.8**
(S, N, S|N)	B_3	29.4	29.5	31.2	31.2	31.2

Table 2: The impact of our changes to Bisk and Hockenmaier's (2013) model (henceforth: B_1, top left) on CCGbank dependencies (LF1, Section 22, all sentences). The best overall model ($B_3^{P\&L}$) uses B_3, punctuation and lexicalization. The best model with complex arguments (B_1^C) uses only B_1.

6 Extending the HDP-CCG system

We now examine how extending the HDP-CCG baseline model to capture lexicalization and punctuation, and how increasing the complexity of the induced grammars affect performance (Table 2).

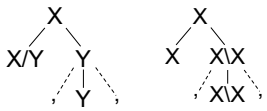
6.1 Modeling Lexicalization

In keeping with most work in grammar induction from part-of-speech tagged text, Bisk and Hockenmaier's (2013) HDP-CCG treats POS tags t rather than words w as the terminals it generates based on their lexical categories c. The advantage of this approach is that tag-based emissions $p(t|c)$ are a lot less sparse than word-based emissions $p(w|c)$. It is therefore beneficial to first train a model that emits tags rather than words (Carroll and Rooth, 1998), and then to use this simpler model to initialize a lexicalized model that generates words instead of tags. To perform the switch we simply estimate counts for the parse forests using the unlexicalized model during the E-Step and then apply those counts to the lexicalized model during the M-Step. Inside-Outside then continues as before. Many words, like prepositions, differ systematically in their preferred syntactic role from that of their part-of-speech tags. This change benefits all settings of the model (Column 2 of Table 2).

6.2 Modeling Punctuation

Spitkovsky et al. (2011) performed a detailed analysis of punctuation for dependency-based grammar induction, and proposed a number of constraints that aimed to capture the different ways in which dependencies might cross constituent boundaries implied by punctuation marks.

A constituency-based formalism like CCG allows us instead to define a very simple, but effective Dirichlet Process (DP) based Markov grammar that emits punctuation marks at the maximal projections of constituents. We note that CCG derivations are binary branching, and that virtually every instance of a binary rule in a normal-form derivation combines a head X or X|Y with an argument Y or modifier X|X. Without reducing the set of strings generated by the grammar, we can therefore assume that punctuation marks can only be attached to the argument Y or the adjunct X|X:



To model this, for each maximal projection (i.e. whenever we generate a non-head child) with category C, we first decide whether punctuation marks should be emitted ($M = \{true, false\}$) to the *left* or *right* side (*Dir*) of C. Since there may be multiple adjacent punctuation marks (... ."), we treat this as a Markov process in which the *history* variable captures whether previous punctuation marks have been generated or not. Finally, we generate an actual punctuation mark w_m:

$$
\begin{array}{ll}
p(M \mid Dir, Hist, \mathsf{C}) & \sim DP(\alpha, p(M \mid dir)) \\
p(M \mid Dir) & \sim DP(\alpha, p(M)) \\
p(w_m \mid Dir, Hist, \mathsf{C}) & \sim DP(\alpha, p(w_m \mid dir, hist)) \\
p(w_m \mid Dir, Hist) & \sim DP(\alpha, p(w_m))
\end{array}
$$

We treat # and $ symbols as ordinary lexical items for which CCG categories will be induced by the regular induction algorithm, but treat all other punctuation marks, including quotes and brackets. Commas and semicolons (,, ;) can act both as punctuation marks generated by this Markov grammar, and as conjunctions with lexical category conj. This model leads to further performance gains (Columns 3 and 4 of Table 2).

6.3 Increasing Grammatical Complexity

The existing grammar induction scheme is very simplistic. It assumes that adjacent words either modify one another or can be taken as arguments. Left unconstrained this space of grammatical cat-

reported dependency evaluation comparison with the work of Naseem et al. (2010). We fixed this hyperparameter setting for experimental simplicity but a more rigorous grid search might find better parameters for the complex models.

Model	Supertagging	LF1	UF1
$\mathbf{B_1}$	59.2	34.5	60.6
$\mathbf{B_1^C}$	59.9	34.9	63.6
$\mathbf{B_3^{P\&L}}$	62.3	37.1	64.9

Table 3: Test set performance of the final systems discussed in this paper (Section 23)

egories introduced grows very rapidly, introducing a tremendous number of incorrect categories (analyzed later in Table 9). For this reason Bisk and Hockenmaier (2013) applied the HDP-CCG model to a context-free fragment of CCG, limiting the arity of lexical categories (number of arguments they can take) to two and the arity of composition (how many arguments can be passed through composition) to one. We know the space of grammatical constructions is larger than this, so we will allow the model to induce categories with three arguments and use generalized composition (B_3). Bisk and Hockenmaier (2013) allow lexical categories to only take atomic arguments, but, as explained above, non-local dependencies require complex arguments of the form $S|N$. We therefore allow lexical categories to take up to one complex argument of the form $S|N$. Atomic lexical categories are not allowed to take complex arguments, eliminating $S|(S|N)$ and $N|(S|N)$. Increasing the search space (Rows 3 and 4 of Table 2) shows corresponding decreases in performance.

Finally, Bisk and Hockenmaier (2013) eliminated the possessive-preposition ambiguity explained above by disallowing categories of the form $(X\backslash X)/X$ and $(X/X)\backslash X$ to be used simultaneously. Removing this restriction does not harm performance (Column 5 of Table 2).

6.4 Summary and test set performance

Table 2 shows the performance of 20 different model settings on Section 22 under the simplified labeled CCG-based dependency evaluation proposed above, starting with Bisk and Hockenmaier's (2013) original model (henceforth: $\mathbf{B_1}$, top left). We see that modeling punctuation and lexicalization both increase performance. We also show that allowing categories of the form $(X\backslash X)/X$ and $(X/X)\backslash X$ on top of the lexicalized models with punctuation does not lead to a noticeable decrease in performance. We also see that an increase in grammatical and lexical complexity is only beneficial for the grammars that allow only atomic arguments, and only if both lexicalization and punctuation are modeled. Allowing complex arguments is generally not beneficial, and performance drops further if the grammatical complexity is increased to B_3. Our further analysis will focus on the three bolded models, $\mathbf{B_1}$, $\mathbf{B_1^C}$ (the best model with complex arguments) and $\mathbf{B_3^{P\&L}}$ (the best overall model), whose supertag accuracy, labeled (LF1) and unlabeled undirected CCG dependency recovery on Section 23 are shown in Table 3. We see that $\mathbf{B_1^C}$ and $\mathbf{B_3^{P\&L}}$ both outperform $\mathbf{B_1}$ on all metrics, although the unlabeled metric (UF1) perhaps misleadingly suggests that $\mathbf{B_1^C}$ leads to a greater improvement than the supertagging and LF1 metrics indicate.

	CCGbank 02-21		WSJ2-21 DA		
Model	LF1	UF1	@10	@20	@∞
Naseem (Universal)			71.9	50.4	
Naseem (English)			73.8	66.1	
$\mathbf{B_1}$	33.8	60.3	70.7	63.1	58.4
$\mathbf{B_1^C}$	34.4	62.0	70.5	65.4	61.9
$\mathbf{B_3^{P\&L}}$	38.3	66.2	71.3	65.9	62.3

Table 4: Performance on CCGbank and CoNLL-style dependencies (Sections 02-21) for a comparison with Naseem et al. (2010).

6.5 CCGbank vs. dependency trees

Finally, to compare our models directly to a comparable unsupervised dependency parser (Naseem et al., 2010), we evaluate them against the unlabeled dependencies produced by Yamada and Matsumoto's (2003) head rules for Sections 02-21 of the Penn Treebank (Table 4)[2]. Naseem et al. (2010) only report performance on sentences of up to length 20 (without punctuation marks). Their approach incorporates prior linguistic knowledge either in the form of "universal" constraints (e.g. that adjectives may modify nouns) or "English-specific" constraints (e.g. that adjectives tend to modify and precede nouns). These universal constraints are akin to, but more explicit and detailed than the information given to the induction algorithm (see Bisk and Hockenmaier (2013) for a discussion). Comparing these numbers to labeled and unlabeled CCG dependencies on the same corpus (all sentences, hence, @∞), we see that performance increases on CCGbank do not translate to similar gains on these unlabeled dependencies. While we have done our best to convert the predicate argument structure of CCG into dependencies

[2]BH13 use hyperparameter schemes and report 64.2@20.

Correct Category	B_1 LR	B_1 Used instead (%)		$B_3^{P\&L}$ LR	$B_3^{P\&L}$ Used instead (%)		B_1^C LR	B_1^C Used instead (%)	
N	**82.6**	N/N	7.5	74.5	N/N	8.3	77.4	N/N	9.8
N/N	78.5	(S\S)\(S\S)	9.8	71.9	(S\S)\(S\S)	8.7	**80.6**	N	7.7
S\N	17.3	S\S	43.5	**22.1**	S\S	27.6	18.3	S\S	39.5
S\S	38.1	N	24.3	34.9	N	16.0	**39.4**	N	22.7
S/S	37.8	N\N	20.8	41.1	N/N	16.3	**57.2**	(S\S)/S	13.8
(N\N)/N	**64.3**	(S\S)/N	20.8	60.5	(S\S)/N	13.8	53.1	(S\S)/N	23.8
(S\N)/N	25.6	S/N	27.0	26.0	(S/N)/N	23.5	**29.4**	S/N	22.3
(S\S)/N	51.0	(N\N)/N	23.1	48.0	(N\N)/N	18.2	**62.6**	N/N	10.1
(S\N)/S	**60.7**	S\N	12.1	55.7	S\N	12.4	57.9	S\N	11.0
(S\S)/S	38.0	(N\N)/N	35.2	50.8	S/S	14.4	**61.5**	N	7.5

Table 5: Detailed supertagging analysis: Recall scores of B_1, B_1^C, and $B_3^{P\&L}$ on the most common recoverable (simplified) lexical categories in Section 22 along with the most commonly produced error.

Category	Example usage	Used instead by B_1^C (%)					
(N/N)\N	The woman **'s** company ...	(N\N)/N	89.9	N/N	3.7	N	2.9
(S/S)/N	**Before** Monday, ...	S/S	69.9	N/N	14.8	(N\N)/N	8.2
(N/N)/(N/N)	The **very** tall man ...	N/N	38.0	(S\S)\(S\S)	33.9	(S\S)/N	10.1
(N\N)/(S\N)	John, **who** ran home, ...	(S\S)/(S/N)	26.5	N\N	23.3	S/S	14.9
(S\N)/(S\N)	I **promise to** pay ...	S\N	32.6	(S\S)/(S/N)	21.5	(S\N)/(S/N)	12.4
((S\N)/N)/N	I **gave** her a gift.	(S\N)/N	34.6	(S/N)/N	34.6	N/N	7.7
((S\N)/(S\N))/N	I **persuaded** her to pay ...	(S\N)/N	24.8	(S/N)/N	22.0	N/N	11.0

Table 6: Categories that are in the search space of the induction algorithm, but do not occur in any Viterbi parse, and what B_1^C uses instead.

there are many constructions which have vastly different analysis, making a proper conversion too difficult for the scope of this paper.[3]

7 Error analysis

Supertagging error analysis We first consider the lexical categories that are induced by the models. Table 5 shows the accuracy with which they recover the most common gold lexical categories, together with the category that they most often produced instead. We see that the simplest model (B_1) performs best on N, and perhaps over generates (N\N)/N (noun-modifying prepositions), while the overall best model ($B_3^{P\&L}$) outperforms both other models only on intransitive verbs.

The most interesting component of our analysis is the long tail of constructions that must be captured in order to produce semantically appropriate representations. We can inspect the confusion matrix of the lexical categories that the model fails to use to obtain insight into how its predictions disagree with the ground truth, and why these constructions may require special attention. Table 6 shows the most common CCGbank categories that were in the search space of some of the more complex models (e.g. B_3^C), but were never used by any of the parsers in a Viterbi parse. These include possessives, relative pronouns, modals/auxiliaries, control verbs and ditransitives. We show the categories that the B_1^C model uses instead. The gold categories shown correspond to the bold words in Table 6. While the reason many of these cases are difficult is intuitive (e.g. *very* modifying *tall* instead of *man*), a more difficult type of error than previously discussed is that of recovering non-local dependencies. The recovery of non-local dependencies is beyond the scope of both standard dependency-based approaches and Bisk and Hockenmaier (2013)'s original induction algorithm. But the parser does not learn to use lexical categories with complex arguments correctly even when the algorithm is extended, to induce them. For example, B_1^C prefers to treat auxiliaries or equi verbs like *promise* as intransitives rather than as an auxiliary that shares its subject with *pay*. The surface string supports this decision, as it can be parsed without having to capture the non-local dependencies (top row) present in the correct (bottom row) analysis:

I	promise	to	pay	you
N	S\N	(S\S)/S	S/N	N
N	(S\N)/(S\N)	(S\N)/(S\N)	(S\N)/N	N

[3] The overlap (F-score of unlabeled undirected attachment scores) between CCGbank dependencies and those obtained via Matsumoto's head finding rules is only 81.9%.

	1st Argument			2nd Argument		
	B_1	B_1^C	$B_3^{P\&L}$	B_1	B_1^C	$B_3^{P\&L}$
N/N	68.4	69.7	**71.6**			
S\N	12.2	**24.9**	14.6			
S\S	17.0	16.2	**18.7**			
S/S	24.0	27.1	**33.8**			
(N\N)/N	49.7	**54.4**	51.2	41.0	**46.2**	42.4
(S\N)/N	26.6	32.9	**34.4**	30.6	33.2	**33.8**
(S\S)/N	21.6	19.2	**24.7**	24.0	24.9	**29.3**
(S\N)/S	23.9	**50.3**	32.5	25.2	**59.1**	35.0
(S\S)/S	6.1	**22.7**	14.1	9.5	**34.6**	19.5

Table 7: LF1 scores of B_1, B_1^C and $B_3^{P\&L}$ on the most common dependency types in Section 22.

We also see that this model uses seemingly non-English verb categories of the form (S/N)/N, both for ditransitives, and object control verbs, perhaps because the possibly spurious /N argument could be swallowed by other categories that take arguments of the form S/N, like the (incorrect) treatment of subject relative pronouns. One possible lesson we can extract from this is that practical approaches for building parsers for new languages might need to focus on injecting semantic information that is outside the scope of the learner.

Dependency error analysis Table 7 shows the labeled recall of the most common dependencies. We see that both new models typically outperform the baseline, although they yield different improvements on different dependency types. $\mathbf{B_1^C}$ is better at recovering the subjects of intransitive verbs (S\N) and verbs that take sentential complements ((S\N)/S), while $\mathbf{B_3}$ is better for simple adjuncts (N/N, S/S, S\S) and transitive verbs.

Wh-words and the long tail To dig slightly deeper into the set of missing constructions, we tried to identify the most common categories that are beyond the search space of the current induction algorithm. We first computed the set of categories used by each part of speech tag in CCGbank, and thresholded the lexicon at 95% token coverage for each tag. Removing the categories that contain PP and those that can be induced by the algorithm in its most general setting, we are left with the categories shown in Table 8. The tags that are missing categories are predominantly wh-words required for wh-questions, relative clauses or free relative clauses. Some of these categories violate the assumptions made by the induction algorithm: question words return a sentence (S) but are not themselves verbs. Free relative pronouns return a noun, but take arguments. However, this is a surprisingly small set of special function words and therefore perhaps a strategic place for supervision. Questions in particular pose an interesting learning question – how does one learn that these constructions indicate missing information which only becomes available later in the discourse?

Additional Category	$p(cat \mid tag)$	
((N\N)/(S\N))/N	.93	WP$
N/(S/N)	.14	WP
N/(S\N)	.08	WP
((N\N)/S)\((N\N)/N)	.07	WDT
((S\S)\(S\S))\N	.04	RBR
S/(S\N)	.04	WP
S/(S/N)	.02	WP

Table 8: Common categories that the algorithm cannot induce

Size, ambiguity, coverage and precision of the induced lexicons				
Arguments:	**Atomic**		**Complex**	
# Lexical Arity:	**2**	**3**	**2**	**3**
# Lexical Categories	37	53	61	133
Avg. #Cats / Tag	26.4	29.5	42.3	56.3
Token-based Coverage	84.3	84.4	89.8	90.2
Type-based Coverage	20.3	21.6	27.0	32.4
Type-based Precision	81.1	60.4	65.6	36.1

Table 9: Size, ambiguity, coverage and precision (evaluated on Section 22) of the induced lexicons.

Grammatical complexity and size of the search space As lexical categories are a good proxy for the set of constructions the grammar will entertain, we can measure the size and ambiguity of the search space as a function of the number of lexical category types it induces as compared to the percentage that are actually valid categories for the language. In Table 9, we compare the lexicons induced by variants of the induction algorithm by their token-based coverage (the percent of tokens in Sections 22 for which the induced tag lexicon contains the correct category), type-based coverage (the percent of category types that the induced lexicon contains), as well as type-based precision (the percent of induced category types that occur in Section 22). This analysis is independent of the learned models, as their probabilities are not taken into account. We see that as the number of lexical categories induced (subject to the constraints of Bisk and Hockenmaier (2012)) increases, the percent that are valid English categories decreases rapidly (type-based precision falls from 81.1% to 36.1%). Despite this, and despite a high token

coverage of up to 90%, we still miss almost 70% of the required category types. This helps explain why performance degrades so much for $\mathbf{B_3^C}$, the arity three lexicon with complex arguments.

8 Dealing with Non-Local Dependencies

While the methodology used here is restricted to CCG based algorithms, we believe the lessons to be very general. The aforementioned constructions involve optional arguments, non-local dependencies, and multiple potential heads. Even though CCG is theoretically expressive enough to handle these constructions, they present the unsupervised learner with additional ambiguity that will pose difficulties independently of the underlying grammatical representation.

For example, although our approach learns that subject NPs are taken as arguments by verbs, the task of deciding which verb to attach the subject to is frequently ambiguous. This most commonly occurs in verb chains, and is compounded in the presence of subject-modifying relative clauses (in CCGbank, both constructions are in fact treated as several verbs sharing a single subject). To illustrate this, we ran the $\mathsf{B}_1^{\mathsf{C}}$ and $\mathsf{B}_3^{\mathsf{P\&L}}$ systems on the following three sentences:

1. The woman **won** an award
2. The woman **has won** an award
3. The woman **being promoted has won** an award

The single-verb sentence is correctly parsed by both models, but they flounder as distractors are added. Both treat *has* as an intransitive verb, *won* as an adverb and *an* as a preposition:

	The	woman	**won**	an	award	
$\mathsf{B}_3^{\mathsf{P\&L}}/\mathsf{B}_1^{\mathsf{C}}$:	N/N	N	(S\N)/N	N/N	N	
	The	woman	**has**	**won**	an	award
$\mathsf{B}_3^{\mathsf{P\&L}}/\mathsf{B}_1^{\mathsf{C}}$:	N/N	N	S\N	S\S	(S\S)/N	N

To accommodate the presence of two additional verbs, both models analyze *being* as a noun modifier that takes *promoted* as an argument. $\mathsf{B}_1^{\mathsf{C}}$ (correctly) stipulates a non-local dependency involving *promoted*, but treats it (arguably incorrectly) as a case of object extraction:

...	**being**	**promoted**	**has**	**won**	an	award
$\mathsf{B}_3^{\mathsf{P\&L}}$	(N\N)/S	S	S\N	S\S	(S\S)/N	N
$\mathsf{B}_1^{\mathsf{C}}$	(N\N)/(S/N)	S/N	S\N	S\S	(S\S)/N	N

Discovering these, and many of the other systematic errors describe here, may be less obvious when analyzing unlabeled dependency trees. But we would expect similar difficulties for any unsupervised approach when sentence complexity grows without a specific bias for a given analysis.

9 Conclusions

In this paper, we have introduced labeled evaluation metrics for unsupervised CCG parsers, and have shown that these expose many common syntactic phenomena that are currently out of scope for any unsupervised grammar induction systems. While we do not wish claim that CCGbank's analyses are free of arbitrary decisions, we hope to have demonstrated that these labeled metrics enable linguistically informed error analyses, and hence allow us to at least in part address the question of where and why the performance of these approaches might plateau. We focused our analysis on English for simplicity, but many of the same types of problems exist in other languages and can be easily identified as stemming from the same lack of supervision. For example, in Japanese we would expect problems with post-positions, in German with verb clusters, in Chinese with measure words, or in Arabic with morphology and variable word order.

We believe that one way to overcome the issues we have identified is to incorporate a semantic signal. Lexical semantics, if sparsity can be avoided, might suffice; otherwise learning with grounding or an extrinsic task could be used to bias the choice of predicates, their arity and in turn the function words that connect them. Alternatively, a simpler solution might be to follow the lead of Boonkwan and Steedman (2011) or Garrette et al. (2015) where gold categories are assigned by a linguist or treebank to tags and words. It is possible that more limited syntactic supervision might be sufficient if focused on the semantically ambiguous cases we have isolated.

More generally, we hope to initiate a conversation about grammar induction which includes a discussion of how these non-trivial constructions can be discovered, learned, and modeled. Relatedly, in future extensions to semi-supervised or projection based approaches, these types of constructions are probably the most useful to get right despite comprising the tail, as analyses without them may not be semantically appropriate. In summary, we hope to begin to pull back the veil on the types of information that a truly unsupervised system, if one should ever exist, would need to learn, and we pose a challenge to the community to find ways that a learner might discover this knowledge without hand-engineering it.

10 Acknowledgments

This material is based upon work supported by the National Science Foundation under Grants No. 1053856, 1205627 and 1405883. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

Yonatan Bisk and Julia Hockenmaier. 2012. Simple Robust Grammar Induction with Combinatory Categorial Grammars. In *Proceedings of the Twenty-Sixth Conference on Artificial Intelligence (AAAI-12)*, pages 1643–1649, Toronto, Canada, July.

Yonatan Bisk and Julia Hockenmaier. 2013. An HDP Model for Inducing Combinatory Categorial Grammars. *Transactions of the Association for Computational Linguistics*, 1:75–88.

Phil Blunsom and Trevor Cohn. 2010. Unsupervised Induction of Tree Substitution Grammars for Dependency Parsing. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1204–1213, Cambridge, USA, October.

Prachya Boonkwan and Mark Steedman. 2011. Grammar Induction from Text Using Small Syntactic Prototypes. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 438–446, Chiang Mai, Thailand, November.

Glenn Carroll and M Rooth. 1998. Valence induction with a head-lexicalized PCFG. In *Proceedings of the 3rd Conference on Empirical Methods in Natural Language Processing*, page 36–45, Granada, Spain.

Stephen Clark, Julia Hockenmaier, and Mark Steedman. 2002. Building Deep Dependency Structures using a Wide-Coverage CCG Parser. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 327–334, Philadelphia, USA, July.

Timothy A D Fowler and Gerald Penn. 2010. Accurate Context-Free Parsing with Combinatory Categorial Grammar. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 335–344, Uppsala, Sweden, July.

Dan Garrette, Chris Dyer, Jason Baldridge, and Noah A Smith. 2015. Weakly-Supervised Grammar-Informed Bayesian CCG Parser Learning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI-15)*, Austin, USA.

William P Headden III, Mark Johnson, and David McClosky. 2009. Improving Unsupervised Dependency Parsing with Richer Contexts and Smoothing. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 101–109, Boulder, USA, June.

Julia Hockenmaier and Mark Steedman. 2007. CCGbank: A Corpus of CCG Derivations and Dependency Structures Extracted from the Penn Treebank. *Computational Linguistics*, 33:355–396, September.

Dan Klein and Christopher D Manning. 2004. Corpus-Based Induction of Syntactic Structure: Models of Dependency and Constituency. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 478–485, Barcelona, Spain, July.

Percy Liang, Slav Petrov, Michael I Jordan, and Dan Klein. 2007. The Infinite PCFG Using Hierarchical Dirichlet Processes. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 688–697, Prague, Czech Republic, June.

David Mareček and Milan Straka. 2013. Stop-probability estimates computed on a large corpus improve Unsupervised Dependency Parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 281–290, Sofia, Bulgaria, August.

Tahira Naseem, Harr Chen, Regina Barzilay, and Mark Johnson. 2010. Using Universal Linguistic Knowledge to Guide Grammar Induction. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1234–1244, Cambridge, USA, October.

Valentin I Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2011. Punctuation: Making a Point in Unsupervised Dependency Parsing. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, pages 19–28, Portland, USA, June.

Valentin I Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2013. Breaking Out of Local Optima with Count Transforms and Model Recombination: A Study in Grammar Induction. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1983–1995, Seattle, USA, October.

Mark Steedman. 2000. *The Syntactic Process*. The MIT Press.

Yee-Whye Teh, Michael I Jordan, Matthew J Beal, and David M Blei. 2006. Hierarchical Dirichlet Processes. *Journal of the American Statistical Association*, 101(476):1566–1581.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical Dependency Analysis With Support Vector Machines. In *Proceedings of 8th International Workshop on Parsing Technologies (IWPT)*, pages 195–206, Nancy, France.