Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence

Simple Robust Grammar Induction with Combinatory Categorial Grammars

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

We present a simple EM-based grammar induction algorithm for Combinatory Categorial Grammar (CCG) that achieves state-of-the-art performance by relying on a minimal number of very general linguistic principles. Unlike previous work on unsupervised parsing with CCGs, our approach has no prior language-specific knowledge, and discovers all categories automatically. Additionally, unlike other approaches, our grammar remains robust when parsing longer sentences, performing as well as or better than other systems. We believe this is a natural result of using an expressive grammar formalism with an extended domain of locality.

Introduction

What kind of inductive bias and supervision signal are necessary to learn natural language syntax? Chomsky (1965)'s argument of the poverty of the stimulus and his resulting proposal of an innate 'universal grammar' were crucial to the development of much of modern linguistic theory, even though there is now strong evidence that even very young children are very good at identifying patterns in the speech stream (Saffran, Aslin, and Newport 1996; Lany and Saffran 2010). This question is also of considerable practical interest: Accurate statistical parsing, which is a core component of many natural language processing systems, relies largely on so-called treebanks (Marcus, Santorini, and Marcinkiewicz 1993), i.e. corpora that have been manually annotated with syntactic analyses, and do not exist for all languages or domains. In recent years, a number of approaches to automatically induce grammars from text alone have been introduced (Klein and Manning 2002; 2004; 2005; Headden III, Johnson, and McClosky 2009; Spitkovsky, Alshawi, and Jurafsky 2010; Tu and Honavar 2011; Cohn, Blunsom, and Goldwater 2011; Naseem et al. 2010). This literature has shown that improvements over a simple EM-based system (Klein and Manning 2004) can be achieved through complex smoothing (Headden III, Johnson, and McClosky 2009), more expressive, hierarchical Bayesian models (Cohn, Blunsom, and Goldwater 2011; Naseem et al. 2010), and richer representations (Cohn, Blunsom, and Goldwater 2011; Boonkwan and Steedman 2011). While it is standard to assume that the words in the training data have been part-of-speech tagged (words are in fact typically replaced by their POS tags), most systems assume no further linguistic knowledge. Recently, Naseem et al. (2010) and Boonkwan and Steedman (2011) have shown that the incorporation of universal or language-specific prior knowledge can significantly improve performance, but it is still unclear what amount of prior linguistic knowledge is really necessary for this task. Naseem et al. assume that the main syntactic roles of major parts of speech classes (e.g. adverbs tend to modify verbs whereas adjectives tend to modify nouns), are known. Boonkwan and Steedman use expert knowledge to predefine a grammar which produces a set of candidate parses over which the model is defined and show that the performance of their system degrades significantly as the amount of prior knowledge is reduced.

In this paper, we show that a simple EM-based algorithm that has enough information about the POS tag set to distinguish between nouns, verbs and other word classes, and has enough universal linguistic knowledge to know that sentences are headed by verbs and that verbs can take nouns as arguments, achieves state-of-the-art performance on the standard WSJ10 (Klein and Manning 2002) task, and outperforms most other approaches, which are based on more complex models, on longer sentences. Our algorithm uses Combinatory Categorial Grammar (CCG) (Steedman 2000), an expressive lexicalized grammar formalism that provides an explicit encoding of head-argument and head-modifier dependencies by associating rich syntactic types with the tokens in the language. These types differ from phrase-structure categories in that they are not arbitrary atomic symbols, but capture information about the context in which a word or constituent tends to appear. Like dependency grammar (assumed by the bulk of the approaches we compare ourselves against), we capture the fact that words (e.g. verbs) may take other words or constituents (e.g. nouns or adverbs) as dependents. Unlike dependency grammar, CCG makes an explicit distinction between (obligatory) arguments and (optional) modifiers, and associates words with lexical types which determine which arguments they take. Syntactic types are defined recursively from two primitives, sentences and nouns (i.e. propositions and entities). Unlike Boonkwan and Steedman (2011), we automatically induce the inventory of language-specific types from the training data.

Combinatory Categorial Grammar (CCG)

Combinatory Categorial Grammar (Steedman 2000) is a linguistically expressive, lexicalized grammar formalism, which associates rich syntactic types with words and constituents. Typically, one assumes two atomic types: S (sentences) and N (nouns). Complex types are of the form $\mathsf{X/Y}$ or $\mathsf{X\backslash Y}$, and represent functions which combine with an immediately adjacent argument of type Y to yield a constituent of type X as result. The slash indicates whether the Y precedes ($\backslash$) or follows ($/$) the functor. The lexicon pairs words with categories, and is of crucial importance, since it captures the only language-specific information in the grammar. An English lexicon may contain entries such as:

$$\begin{array}{rlrl}
\mathsf{N}: & \{he, girl, lunch, ...\} & \mathsf{N/N}: & \{good, the, eating, ...\}\\
\mathsf{S\backslash N}: & \{sleeps, ate, eating, ...\} & \mathsf{(S\backslash N)/N}: & \{sees, ate, ...\}\\
\mathsf{S\backslash S}: & \{quickly, today ...\} & \mathsf{(S\backslash N)/(S\backslash N)}: & \{good, the, ...\}
\end{array}$$

While the set of categories is theoretically unbounded, the inventory of lexical category types is assumed to be finite and of a bounded maximal arity (typically 3 or 4). Categorial grammar rules are defined as schemas over categories (where X, Y, Z etc. are category variables and $| \in \{\backslash, /\}$ is a slash variable), and are usually given in a bottom-up manner. All variants of categorial grammar (Ajdukiewicz 1935; Bar-Hillel 1953) use the basic rule of forward ($>$) and backward ($<$) application, which specifies that a functor $\mathsf{X|Y}$ can combine with an adjacent argument Y to form a new X:

$$\begin{array}{rlll}
\mathsf{X/Y} & \mathsf{Y} & \rightarrow \mathsf{X} & (>)\\
\mathsf{Y} & \mathsf{X\backslash Y} & \rightarrow \mathsf{X} & (<)
\end{array}$$

CCG includes additional rules: in function composition (the B combinator of Curry and Feys (1958)), the arity of the secondary functor can vary from 1 to a fixed upper limit n. We examine the effect of limiting the grammar to application and simple composition B^1 on our induction algorithm below, but generally restrict ourselves to $n = 2$:

$$\begin{array}{rlll}
\mathsf{X/Y} & \mathsf{Y|_iZ} & \Rightarrow \mathsf{X|_iZ} & (\mathsf{B}^1_>)\\
\mathsf{Y|_iZ} & \mathsf{X\backslash Y} & \Rightarrow \mathsf{X|_iZ} & (\mathsf{B}^1_<)\\
\mathsf{X/Y} & \mathsf{(Y|_iZ_1)|_jZ_2} & \Rightarrow \mathsf{(X|_iZ_1)|_jZ_2} & (\mathsf{B}^2_>)\\
\mathsf{(Y|_iZ_1)|_jZ_2} & \mathsf{X\backslash Y} & \Rightarrow \mathsf{(X|_iZ_1)|_jZ_2} & (\mathsf{B}^2_<)
\end{array}$$

(C)CG parses are typically written as logical derivations:

$$\begin{array}{cccc}
\textit{The} & \textit{man} & \textit{ate} & \textit{quickly}\\ \hline
\mathsf{N/N} & \mathsf{N} & \mathsf{S\backslash N} & \mathsf{S\backslash S}\\ \hline
\multicolumn{2}{c}{\mathsf{N}\quad >} & \multicolumn{2}{c}{\mathsf{S\backslash N}\quad <\mathbf{B}}\\ \hline
\multicolumn{4}{c}{\mathsf{S}\quad <}
\end{array}$$

Here, the intransitive verb *ate* only takes a subject as argument, whereas the determiner *the* modifies the noun *man*, and the adverb *quickly* can combine with the verb via composition. This example illustrates the two main kinds of dependency relations between words or constituents (Tesnière 1959), which CCG distinguishes explicitly: in a head-argument relation, the head $\mathsf{X|Y}$ (e.g. $\mathsf{S\backslash N}$) takes its dependent Y (N) as argument, whereas in a head-modifier relation, the modifier $\mathsf{X|X}$ ($\mathsf{N/N}$) takes the head X (N) as argument. One of the roles of CCG's composition is that it allows modifiers such as adverbs to have generic categories such as $\mathsf{S\backslash S}$, regardless of the verb they modify:

$$\begin{array}{cccc}
\textit{he} & \textit{ate} & \textit{quickly} & \textit{the lunch he bought}\\ \hline
\mathsf{N} & \mathsf{(S\backslash N)/N} & \mathsf{S\backslash S} & \mathsf{N}\\ \hline
 & \multicolumn{2}{c}{\mathsf{(S\backslash N)/N}\quad <\mathbf{B}_\times} &
\end{array}$$

CCG also includes a unary type-raising rule, which reverses the relationship between functors and arguments, and allows Y (which may be the argument of $\mathsf{X\backslash Y}$) to turn into a functor that takes $\mathsf{X\backslash Y}$ as argument and returns X:

$$\begin{array}{lll}
\mathsf{Y} \Rightarrow & \mathsf{X/(X\backslash Y)} & (>\mathsf{T})\\
\mathsf{Y} \Rightarrow & \mathsf{X\backslash(X/Y)} & (<\mathsf{T})
\end{array}$$

The category $\mathsf{X\backslash Y}$ or $\mathsf{X/Y}$ is generally restricted to be of a type that also occurs in the lexicon of the language (Steedman 2000). Although type-raising Y followed by application of the type-raised argument to the original functor $\mathsf{X\backslash Y}$ is equivalent to applying the functor itself (and we therefore disallow type-raised categories to apply to other categories to reduce the number of spurious ambiguities), type-raising and composition act together to capture non-local dependencies which arise through extraction or coordination, e.g.:

$$\begin{array}{cccc}
\textit{the man} & \textit{that} & \textit{I} & \textit{saw}\\ \hline
\mathsf{N} & \mathsf{(N\backslash N)/(S/N)} & \mathsf{N} & \mathsf{(S\backslash N)/N}\\ \hline
 & & \mathsf{S/(S\backslash N)}\quad >\mathbf{T} & \\ \hline
 & & \multicolumn{2}{c}{\mathsf{S/N}\quad >\mathbf{B}}\\ \hline
 & \multicolumn{3}{c}{\mathsf{N\backslash N}\quad >}\\ \hline
\multicolumn{4}{c}{\mathsf{N}\quad <}
\end{array}$$

Although type-raising and composition introduce additional derivations, our system does not try to eliminate the spurious ambiguities they introduce, such as:

$$\begin{array}{cccc}
\textit{The} & \textit{man} & \textit{ate} & \textit{quickly}\\ \hline
\mathsf{N/N} & \mathsf{N} & \mathsf{S\backslash N} & \mathsf{S\backslash S}\\ \hline
\multicolumn{2}{c}{\mathsf{N}\quad >} & \multicolumn{2}{c}{\mathsf{S\backslash N}\quad <\mathbf{B}}\\ \hline
\multicolumn{2}{c}{\mathsf{S/(S\backslash N)}\quad >\mathbf{T}} & & \\ \hline
\multicolumn{4}{c}{\mathsf{S}\quad >}
\end{array}$$

For coordination we assume a special ternary rule (Hockenmaier and Steedman 2007) that is binarized as follows:

$$\begin{array}{rllll}
\mathsf{X} & \mathsf{X[conj]} & \Rightarrow_{\&1} & \mathsf{X} & (\&1)\\
\mathrm{conj} & X & \Rightarrow_{\&2} & \mathsf{X[conj]} & (\&2)
\end{array}$$

Since CCG contains only unary and binary rules, it can be parsed with the standard CKY algorithm.

An algorithm for unsupervised CCG induction

We now describe our induction algorithm, which consists of two stages: category induction (creation of the grammar), followed by parameter estimation for the probability model.

The category induction stage

We assume there are two atomic categories, N (nouns or noun phrases) and S (sentences), a special conjunction category conj, and a special start symbol TOP. We assume that all strings we encounter are either nouns or sentences:

$$\mathsf{N} \Rightarrow \mathsf{TOP} \qquad \mathsf{S} \Rightarrow \mathsf{TOP}$$

We also assume that we can group POS-tags into four groups: nominal, verbal, coordinations, and others. This allows us to create an initial lexicon $\mathcal{L}^{(0)}$, which only contains entries for atomic categories, e.g. for the English Penn Treebank tag set (Marcus, Santorini, and Marcinkiewicz 1993):

$$\begin{aligned} \mathsf{N} &: \{\texttt{NN}, \texttt{NNS}, \texttt{NNP}, \texttt{PRP}, \texttt{DT}\} \\ \mathsf{S} &: \{\texttt{MD}, \texttt{VB}, \texttt{VBZ}, \texttt{VBG}, \texttt{VBN}, \texttt{VBD}\} \\ \mathsf{conj} &: \{\texttt{CC}\} \end{aligned}$$

We force any string that contains one or more verbs (besides VBG), to be parsed with the $\mathsf{S} \Rightarrow \mathsf{TOP}$ rule.

Since the initial lexicon would only allow us to parse single word utterances (or conjunctions thereof), we need to induce complex functor categories. The lexicon for atomic categories remains fixed, but all POS-tags will be able to acquire complex, categories during induction. We impose the following constraints on the lexical categories we induce:

1. Nouns (N) do not take any arguments.
2. The heads of sentences (S|...) and modifiers (X|X, (X|X)|(X|X)) may take N or S as arguments.
3. Sentences (S) may only take nouns (N) as arguments. (We assume S\S and S/S are modifiers).
4. Modifiers (X/X or X\X) can be modified by categories of the form (X/X)|(X/X) or (X\X)|(X\X).
5. The maximal arity of any lexical category is 3.
6. Since (S\N)/N is completely equivalent to (S/N)\N, we only allow the former category.

Induction is an iterative process. At each stage, we aim to parse all sentences S_i in our training corpus $\mathcal{D} = \{S_1, ..., S_D\}$ with the current lexicon $\mathcal{L}^{(t)}$. In order to parse a sentence $S = w_0...w_n$, all words $w_i \in S$ need to have lexical categories that allow a complete parse (resulting in a constituent TOP that spans the entire sentence). Initially, only some words will have lexical categories:

The	*man*	*ate*	*quickly*
DT	NNS	VBD	RB
-	N	S	-

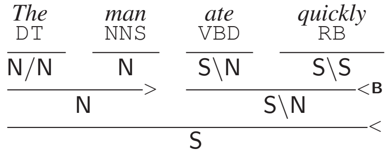
We assume that any word may modify adjacent constituents:

The	*man*	*ate*	*quickly*
DT	NNS	VBD	RB
N/N	N, S/S	S, N\N	S\S

We also assume that any word that previously had a category other than N (which we postulate does not take any arguments) can take any adjacent non-modifier category as argument, leading us here to introduce S\N for the verb:

The	*man*	*ate*	*quickly*
DT	NNS	VBD	RB
N/N	N, S/S	S, N\N, S\N	S\S

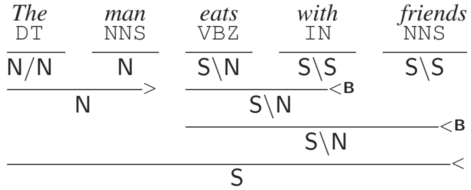
With these categories, we obtain the correct parse:



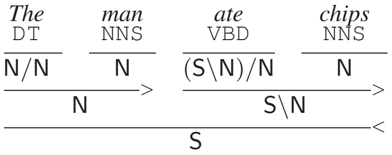
We then update the lexicon with all new tag-category pairs that have been found, excluding those that did not lead to a successful parse:

$$\mathsf{N/N} : \{\texttt{DT}\} \quad \mathsf{S\backslash N} : \{\texttt{VBD}, \texttt{VBZ}\} \quad \mathsf{S\backslash S} : \{\texttt{RB}, \texttt{NNS}, \texttt{IN}\}$$

The first stage of induction can only introduce functors of arity 1, but many words, such as prepositions or transitive verbs, require more complex categories, leading us to complete, but incorrect parses such as



During the second iteration, we can discover additional simple, as well as more complex, categories. We now discover transitive verb categories:



The second stage also introduces a large number of complex modifiers of the form (X/X)|(X/X) or (X\X)|(X\X), e.g.:

The	*man*	*ate*	*very*	*quickly*
DT	NNS	VBD	RB	RB
N/N, (S/S)/(S/S)	N, S/S (N\N)/(N\N) (N/N)\(N/N)	S, N\N, S\N (S/S)\(S/S) (S\S)/(S\S)	S\S, (S\S)/(S\S) (N\N)\(N\N)	S\S, (S\S)\(S\S)

Finally, we include a final induction step which takes adjacent constituents that can be derived from the existing lexicon into account. Since we can combine e.g. *a friend* to N, we can now also induce (S\S)/N for IN.

After constructing the lexicon, we parse the training corpus, and use the Inside-Outside algorithm (Lari and Young 1991), a variant of the Expectation-Maximization algorithm for probabilistic context-free grammars, to estimate model parameters. We use the baseline model of Hockenmaier and Steedman (2002), which is a simple generative model that is

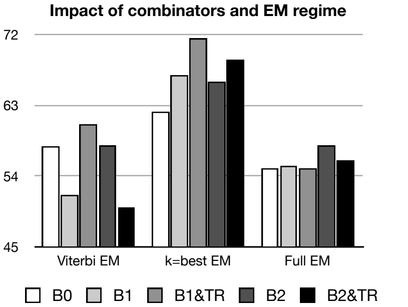


Figure 1: Impact of the beam width k and the expressiveness of the grammar on performance

equivalent to an unlexicalized PCFG. In a CFG, the set of terminals and non-terminals is disjoint, but in CCG, not every category can be lexical. Since this model is also the basis of a lexicalized model that captures dependencies, it distinguishes between lexical expansions (which produce words), unary expansions (which are the result of type-raising or the TOP rules), binary expansions where the head is the left child, and binary expansions whose head is the right child. Each tree is generated top-down from the start category TOP. For each (parent) node, first its expansion type $exp \in \{\mathrm{Lex}, \mathrm{Unary}, \mathrm{Left}, \mathrm{Right}\}$ is generated. Based on the expansion type, the model then produces either the word w, or the category of the head child (H), and, possibly the category of the non-head sister category (S):

$$\begin{array}{ll} \text{Lexical} & p_e(\mathrm{exp}{=}\mathrm{Lex} \mid \mathrm{P}) \times p_w(w \mid \mathrm{P}, \mathrm{exp}{=}\mathrm{Lex}) \\ \text{Unary} & p_e(\mathrm{exp}{=}\mathrm{Unary} \mid \mathrm{P}) \times p_{\mathrm{H}}(\mathrm{H} \mid \mathrm{P}, \mathrm{exp}{=}\mathrm{Unary}) \\ \text{Left} & p_e(\mathrm{exp}{=}\mathrm{Left} \mid \mathrm{P}) \times p_{\mathrm{H}}(\mathrm{H} \mid \mathrm{P}, \mathrm{exp}{=}\mathrm{Left}) \\ & \quad \times\, p_S(\mathrm{S} \mid \mathrm{P}, \mathrm{H}, \mathrm{exp}{=}\mathrm{Left}) \\ \text{Right} & p_e(\mathrm{exp}{=}\mathrm{Right} \mid \mathrm{P}) \times p_{\mathrm{H}}(\mathrm{H} \mid \mathrm{P}, \mathrm{exp}{=}\mathrm{Right}) \\ & \quad \times\, p_S(\mathrm{S} \mid \mathrm{P}, \mathrm{H}, \mathrm{exp}{=}\mathrm{Right}) \end{array}$$

Training regime Spitkovsky et al. (2010) demonstrated the utility of using hard EM (i.e. restricting the updates to only the single best, Viterbi, parse according to the current model) for unsupervised grammar induction. We compare standard full (soft) EM, where we use the entire parse forest during estimation with Viterbi EM, as well as with a smoothed variant of k-best EM which interpolates the probabilities from the top-k parses with those of the full forest,[1] using the algorithm of Huang and Chiang (2005) to compute the k-best parses according to the current model.

Experimental setup

Data As is standard, our induction uses sentences of length 10 or less (not counting punctuation) from the WSJ part of the Penn Treebank (Marcus, Santorini, and Marcinkiewicz 1993), and relies on gold part-of-speech tags. We use all sentences of length 10 or less (not counting punctuation) from sections 02-21 for training, section 23 for testing, and 00 for development.

Training procedure We run the induction routine N times over the entire data set, initialize the probability model uniformly based on the events observed in the training data, and run EM until convergence. We do not allow any probabilities to go below $e^{-6}(\simeq 0.0025)$. During k-best estimation, the top k parses for each sentence are recomputed according to the current model.

Evaluation metrics As is standard, we evaluate against word-word dependencies obtained from the original Penn Treebank. We use Johansson and Nugues (2007)'s code to obtain these dependencies and the CoNLL 2008 shared task script to evaluate unlabeled directed attachment. In order to allow a direct comparison with approaches based on other formalisms (and in contrast to the CCG dependencies for the Penn Treebank provided by Hockenmaier and Steedman (2007)), we treat modifiers as dependents of their heads (even though in CCG terms, the constituent X is an argument of X|X), assume that the head of the sentence is a dependent of a root node at position 0 in the sentence, and analyze conjunction (X_1 conj X_2) as creating a dependency from X_1 (the head) to conj, and from conj to X_2.

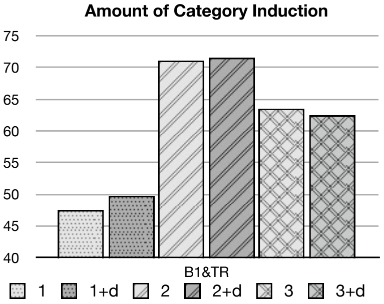


Figure 2: Impact of the number of induction stages on performance ("d": derived constituents are considered).

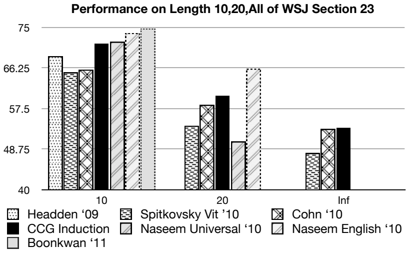
Tag	Cat	P(c | t)	Tag	Cat	P(c |t)
NN	N	0.839	RB	S/S	0.527
	N/N	0.133		(S\S)/(S\S)	0.275
	(N/N)/N	0.021		S\S	0.119
DT	N/N	0.925	VBD	(S\N)/N	0.419
	N	0.034		S\N	0.339
	(N/N)/N	0.011		(S\N)\S	0.339
JJ	N/N	0.861	TO	(S\S)/S	0.498
	S\S	0.114		(S\S)/N	0.437
	(S/S)/N	0.012		N/N	0.012
IN	(S\S)/N	0.678	VB	S/N	0.743
	(N\N)/N	0.148		S	0.151
	(N/N)/N	0.069		N/N	0.031

Figure 3: The most likely induced lexical categories for common parts of speech (probabilities based on Viterbi parses of section 00)

[1] If c_k and c_{full} are the frequencies of y in the k-best and the full parse forests, we define $\lambda_k = c_k/(c_k + c_{\mathrm{full}})$, and compute $\tilde{p}(x \mid y) = \lambda_k \hat{p}_k(x \mid y) + (1 - \lambda_k)\hat{p}_{\mathrm{full}}(x \mid y)$.

	10	20	Inf
Klein & Manning ’04	47.5		
Headden ’09	68.8		
Spitkovsky Vit ’10	65.3*	53.8*	47.9*
Cohn ’10	65.9	58.3	53.1
CCG Induction	71.5	**60.3**	**53.3**
Naseem Universal ’10	**71.9**	50.4*	
Naseem English ’10	73.8	66.1*	
Boonkwan ’11	74.8		

(a) Full table of comparison results for section 23



(b) Bar graph of the top models’ performance

Figure 4: Results on section 23 when trained on length 10 data. Starred results were obtained with additional training data: up to length 20 (Naseem 20) or 45 (Spitkovsky)

Experimental results

Before providing a comparison with related work, we first evaluate the effect that the size of the induced grammar (as determined by the inventory of combinatory rules and the number of stages of induction before model estimation) and the width of the beam have during k-best estimation.

Impact of combinators and beam width k Which combinatory rules are allowed during parsing determines the expressiveness of the grammar, and hence the complexity of linguistic phenomena which can be captured. It also has a significant effect on the size of the grammar and number of parses per sentence. The training regime impacts the effective size of the grammar as well: Viterbi training (i.e. only using the highest scoring parse to update the model) prunes the grammar, while our smoothed k-best algorithm assigns significantly more mass to frequent constructions and categories. We therefore found a strong interaction between these two parameters. Figure 1 provides results on the test set for each grammar setting with Viterbi, $k = \text{best}_G$ (a grammar-specific setting of k that was found to optimize performance on the development set) and full EM. Unlike Spitkovsky et al. (2010), Viterbi parsing does not in general outperform full EM, but with the right choice of k, k-best parsing can yield substantial improvements in performance. We also found that type-raising proved beneficial in both the B^1 and B^2 cases. These results are based on the use of two induction steps at initialization in addition to a final induction step with derived constituents.

Number of induction stages The second dimension we examine is the impact of the number of induction iterations on performance (again using a grammar-specific optimal k). Figure 2 shows that, for grammars that use B^1 with type raising, two iterations of induction perform the best, while induction from derived constituents has a minimal (or slightly detrimental) effect.

The induced lexicons We generally find that the lexical categories our system induces match very well the commonly assumed CCG categories for English. Figure 3 lists common POS tags and their most likely categories (probabilities of category given tag were computed based on the Viterbi parses of our best performing model on the development set). Besides actual performance (which depends not just on the lexicon, but also on the kinds of attachment decisions the model makes), this is a very good indicator of how much of the language the model as captured, because CCG encodes all language specific information in its lexicon. We find that most of the mass is centered on exactly those categories that a linguist would include in a basic (C)CG lexicon for English, and that generally little mass is assigned to non-standard categories such as $(\mathsf{N}/\mathsf{N})/\mathsf{N}$ for `NN` (common noun) or `IN` (prepositions and subordinating conjunctions). The only possible exceptions are $(\mathsf{S}\backslash\mathsf{S})/\mathsf{S}$ for infinitival `TO` (*to*) and S/N for `VB` (infinitival verbs), which can both be explained by the fact that infinitives are rarely preceded by subjects, whereas $(\mathsf{S}\backslash\mathsf{N})\backslash\mathsf{S}$ for `VBD` (past tense verbs) is actually required for inversions that are frequently used with direct speech in our domain (*‘This was obvious”, he said.*).

Why is our model able to induce these linguistically correct categories? Since our induction scheme allows all categories to be modifiers or modifiers of modifiers, one obvious grammar that it permits is one where verbs are S, and everything else is either S/S or $\mathsf{S}\backslash\mathsf{S}$. The reason that this does not plague us is subtle yet important. Because we do not differentiate between lexical and non-lexical non-terminals but rather have a distribution over expansions (discussed in the model), the frequent use of S throughout the tree in various binary productions leaves little mass for a lexical S. In contrast, a category like $(\mathsf{S}\backslash\mathsf{N})/\mathsf{N}$ will nearly never appear anywhere but at the lexical level, resulting in a very high probability of being a lexical category. An additional question is why do nouns learn the English ordering N/N when they have the equally valid opportunity to be $\mathsf{N}\backslash\mathsf{N}$. Although we allow the tag `DT` (e.g. *this*) to act as a noun, many noun phrases do not contain a determiner, increasing the relative frequency with which N generates a nominal tag, and decreasing the probability of it generating `DT`. Furthermore, our current system assumes that adjectives, which tend to precede nouns, cannot carry the category N.

Comparison with related work

Since we evaluate on dependencies, we compare ourselves against the following other approaches: Klein and Manning (2004)'s DMV model is a very simple EM-based generative dependency model that is the foundation for most of the subsequent proposals (The performance quoted here is achieved by their slightly more complex DMV-CCM model). Spitkovsky et al. (2010) build on previous work which demonstrated improved DMV performance when trained with a curriculum, and find that Viterbi parsing helps as it enables learning from longer sentences which would otherwise lead the model astray, resulting in a multi-stage training process. Headden III, Johnson, and McClosky (2009)'s Lexicalized Extended Valence Grammar both lexicalizes the DMV models, and includes a valence term which captures subcategorization information and models the proximity of a word to the head. Cohn, Blunsom, and Goldwater (2011) learn a non-parametric Bayesian model of tree-substitution grammar biased towards a sparse grammar with shallow productions. Underlying the model is a base distribution computed over CFG trees derived from the DMV model. Naseem et al. (2010) demonstrate the effectiveness of universal linguistic knowledge. Their model has access to 13 universal dependency constraints for all major parts of speech, including rules such as sentences tend to be headed by verbs or auxiliaries, auxiliaries tend to take verbs as dependents, verbs tend to take nouns and adverbs as dependents, adjectives tend to modify nouns, prepositions tend to take nouns as arguments, etc. In contrast to this rich information, we only capture two basic principles: sentences are headed by verbs and auxiliaries, and verbs and auxiliaries can take nouns as arguments. All other information regarding the relationship between verbs and adverbs/auxiliaries or between nouns and adjectives/articles/numerals/prepositions is learned automatically by our system with signals from the data and our induction procedure. Naseem et al. also evaluate a variant of their system that uses a number of highly effective English-specific heuristics at test time. Boonkwan and Steedman (2011) also use a categorial grammar, but use a 30 question survey to discover basic facts about word order. The results of this survey determine the inventory of lexical types for the language, which are then mapped to specific part-of-speech tags by the experimenter to create a custom language specific lexicon. Without this knowledge, their English WSJ10 results drop from 74.8 to 40.2. By comparison, we do not encode any language specific information and discover the underlying word order automatically.

Comparing experimental results Figure 4 compares our performance on section 23 against state-of-the-art systems given in the literature, as well as the DMV model of Klein and Manning (2004) that many of the more recent approaches are based on. We evaluate our performance on sentences of length 10 (WSJ10), length 20 (WSJ20) as well as all sentences (∞) in section 23, and compare against results in the literature where available. Our WSJ20 and WSJ∞ results are trained on WSJ10 data. On WSJ20, our performance is significantly higher than Naseem et al. (Universal), and at ∞, our performance is indistinguishable from Cohn et al.'s, although we outperform them at shorter lengths. It is important to note that our system's simplicity leads to very efficient EM training, unlike other systems which require sampling, complex smoothing or training curricula.

We draw two main conclusions from this comparison: First, it is clear that performance on this task depends strongly on the amount of prior linguistic knowledge that is provided to the system. On the one hand, the DMV baseline system (Klein and Manning 2004) as well as the related approaches of Spitkovsky et al. (2010), Headden III, Johnson, and McClosky (2009), Cohn, Blunsom, and Goldwater and (2011) assume no prior linguistic knowledge (other than that implicit in the gold POS tags). Although the words themselves clearly carry important information, it is unclear how much of that information can be captured based on the standard WSJ10 corpus without sophisticated smoothing, and Headden III, Johnson, and McClosky's is the only approach that incorporates actual words into the model. On the other hand, both Naseem et al.'s set of universal constraints and their English heuristics, as well as Boonkwan and Steedman's expert linguistic knowledge clearly outperform the linguistically uninformed systems.

Second, the capacity of the model to represent linguistic phenomena such as valence (subcategorization), either explicitly through richer representations (Cohn, Blunsom, and Goldwater 2011), or implicitly through an expressive model (Headden III, Johnson, and McClosky 2009) has a clear impact on performance. Like categorial grammar, Cohn, Blunsom, and Goldwater's tree-substitution grammar is a lexicalized formalism with an extended domain of locality (Joshi 1988), which associates words (or POS-tags, for the current task) with complex symbols that capture the context in which they appear. While the generative capacity of TSG (and CCG without crossing composition) is weakly equivalent to that of CFGs and to projective dependency grammars that disallow crossing dependencies (which is the case for all dependency-based systems discussed here), the choice of representation (in the grammar or the model) determines the independence assumptions, and hence generalizations, that are made during induction. Cohn et al.'s and our system are remarkably robust at longer sentence lengths.

Conclusions

This paper has demonstrated that unsupervised CCG induction can rival other methods; although our approach cannot be compared directly to approaches without prior linguistic knowledge, we perform as well as or better (on longer sentences) than the system of Naseem et al. (2010), which arguably captures much more prior knowledge, and uses a richer statistical model. Future work will examine the effect of relaxing our assumptions (e.g. about the ability to identify nouns and verbs), and the impact of richer models on CCG.

Acknowledgments

This research is supported by the National Science Foundation through CAREER award 1053856 and award 0803603.

References

Ajdukiewicz, K. 1935. Die syntaktische Konnexität. In McCall, S., ed., *Polish Logic 1920-1939*. Oxford University Press. 207–231. Translated from *Studia Philosophica*, 1, 1-27.

Bar-Hillel, Y. 1953. A quasi-arithmetical notation for syntactic description. *Language* 29:47–58.

Boonkwan, P., and Steedman, M. 2011. Grammar Induction from Text Using Small Syntactic Prototypes. In *International Joint Conference on Natural Language Processing*, 438–446.

Chomsky, N. 1965. *Aspects of the Theory of Syntax*. The M.I.T. Press.

Cohn, T.; Blunsom, P.; and Goldwater, S. 2011. Inducing tree-substitution grammars. *Journal of Machine Learning Research* 3053–3096.

Curry, H. B., and Feys, R. 1958. *Combinatory Logic*, volume I. Amsterdam: North-Holland.

Headden III, W. P.; Johnson, M.; and McClosky, D. 2009. Improving Unsupervised Dependency Parsing with Richer Contexts and Smoothing. *North American chapter of the Association for Computational Linguistics conference* 101–109.

Hockenmaier, J., and Steedman, M. 2002. Generative models for statistical parsing with Combinatory Categorial Grammar. In *Association for Computational Linguistics*, 335–342.

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

Huang, L., and Chiang, D. 2005. Better k-best parsing. In *International Workshop on Parsing Technology*, 53–64.

Johansson, R., and Nugues, P. 2007. Extended constituent-to-dependency conversion for english. In *Proceedings of NODALIDA 2007*.

Joshi, A. 1988. Tree Adjoining Grammars. In Dowty, D.; Karttunen, L.; and Zwicky, A., eds., *Natural Language Parsing*. Cambridge: Cambridge University Press. 206–250.

Klein, D., and Manning, C. D. 2002. A generative constituent-context model for improved grammar induction. In *Association for Computational Linguistics*, 128–135.

Klein, D., and Manning, C. D. 2004. Corpus-Based Induction of Syntactic Structure: Models of Dependency and Constituency. In *Association for Computational Linguistics*, 478–485.

Klein, D., and Manning, C. D. 2005. Natural language grammar induction with a generative constituent-context model. *Pattern recognition* 38(9):1407–1419.

Lany, J., and Saffran, J. R. 2010. From statistics to meaning. *Psychological Science* 21(2):284–291.

Lari, K., and Young, S. 1991. Applications of stochastic context-free grammars using the inside-outside algorithm. *Computer speech & language* 5(3):237–257.

Marcus, M. P.; Santorini, B.; and Marcinkiewicz, M. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics* 19(2):313–330.

Naseem, T.; Chen, H.; Barzilay, R.; and Johnson, M. 2010. Using universal linguistic knowledge to guide grammar induction. In *Empirical Methods in Natural Language Processing*, 1234–1244.

Saffran, J. R.; Aslin, R. N.; and Newport, E. L. 1996. Statistical learning by 8-month-old infants. *Science* 274(5294):1926–1928.

Spitkovsky, V. I.; Alshawi, H.; and Jurafsky, D. 2010. From Baby Steps to Leapfrog: How “Less is More” in Unsupervised Dependency Parsing. In *North American chapter of the Association for Computational Linguistics conference*, 751–759.

Spitkovsky, V. I.; Alshawi, H.; Jurafsky, D.; and Manning, C. D. 2010. Viterbi training improves unsupervised dependency parsing. In *Conference on Computational Natural Language Learning*, 9–17.

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

Tesnière, L. 1959. *Éléments de Syntaxe Structurale*. Paris, France: Klinsieck.

Tu, K., and Honavar, V. 2011. On the Utility of Curricula in Unsupervised Learning of Probabilistic Grammars. In *International Joint Conferences on Artificial Intelligence*, 1523–1528.