An HDP Model for Inducing 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 introduce a novel nonparametric Bayesian model for the induction of Combinatory Categorial Grammars from POS-tagged text. It achieves state of the art performance on a number of languages, and induces linguistically plausible lexicons.

1 Introduction

What grammatical representation is appropriate for unsupervised grammar induction? Initial attempts with context-free grammars (CFGs) were not very successful (Carroll and Charniak, 1992; Charniak, 1993). One reason may be that CFGs require the specification of a finite inventory of nonterminal categories and rewrite rules, but unless one adopts linguistic principles such as X-bar theory (Jackendoff, 1977), these nonterminals are essentially arbitrary labels that can be combined in arbitrary ways. While further CFG-based approaches have been proposed (Clark, 2001; Kurihara and Sato, 2004), most recent work has followed Klein and Manning (2004) in developing models for the induction of projective dependency grammars. It has been shown that more sophisticated probability models (Headden III et al., 2009; Gillenwater et al., 2011; Cohen and Smith, 2010) and learning regimes (Spitkovsky et al., 2010), as well as the incorporation of prior linguistic knowledge (Cohen and Smith, 2009; Berg-Kirkpatrick and Klein, 2010; Naseem et al., 2010) can lead to significant improvement over Klein and Manning's baseline model. The use of dependency grammars circumvents the question of how to obtain an appropriate inventory of categories, since dependency parses are simply defined by unlabeled edges between the lexical items in the sentence. But dependency grammars make it also difficult to capture non-local structures, and Blunsom and Cohn (2010) show that it may be advantageous to reformulate the underlying dependency grammar in terms of a tree-substitution grammar (TSG) which pairs words with treelets that specify the number of left and right dependents they have. In this paper, we explore yet another option: instead of dependency grammars, we use Combinatory Categorial Grammar (CCG, Steedman (1996; 2000)), a linguistically expressive formalism that pairs lexical items with rich categories that capture all language-specific information. This may seem a puzzling choice, since CCG requires a significantly larger inventory of categories than is commonly assumed for CFGs. However, unlike CFG nonterminals, CCG categories are not arbitrary symbols: they encode, and are determined by, the basic word order of the language and the number of arguments each word takes. CCG is very similar to TSG in that it also pairs lexical items with rich items that capture all language-specific information. Like TSG and projective dependency grammars, we restrict ourselves to a weakly context-free fragment of CCG. But while TSG does not distinguish between argument and modifier dependencies, CCG makes an explicit distinction between the two. And while the elementary trees of Blunsom and Cohn (2010)'s TSG and their internal nodel labels have no obvious linguistic interpretation, the syntactic behavior of any CCG constituent can be directly inferred from its category. To see whether

Transactions of the Association for Computational Linguistics, 1 (2013) 75–88. Action Editor: Sharon Goldwater.
Submitted 11/2012; Revised 1/2013; Published 3/2013.

the algorithm has identified the basic syntactic properties of the language, it is hence sufficient to inspect the induced lexicon. Conversely, Boonkwan and Steedman (2011) show that knowledge of these basic syntactic properties makes it very easy to create a language-specific lexicon for accurate unsupervised CCG parsing. We have recently proposed an algorithm for inducing CCGs (Bisk and Hockenmaier, 2012b) that has been shown to be competitive with other approaches even when paired with a very simple probability model (Gelling et al., 2012). In this paper, we pair this induction algorithm with a novel nonparametric Bayesian model that is based on a different factorization of CCG derivations, and show that it outperforms our original model and many other approaches on a large number of languages. Our results indicate that the use of CCG yields grammars that are significantly more robust when dealing with longer sentences than most dependency grammar-based approaches.

2 Combinatory Categorial Grammar

Combinatory Categorial Grammar (Steedman, 2000) is a linguistically expressive, lexicalized grammar formalism that associates rich syntactic types with words and constituents. For simplicity, we restrict ourselves to the standard two atomic types S (sentences) and N (encompassing both nouns and noun phrases) from which we recursively build categories. Complex categories are of the form $\mathsf{X/Y}$ or $\mathsf{X\backslash Y}$, and represent functions which return a result of type X when combined with an argument of type Y. The directionality of the slash indicates whether the argument precedes or follows the functor. We write $\mathsf{X|Y}$ when the direction of the slash does not matter.

The CCG lexicon encodes all language-specific information. It pairs every word with a set of categories that define both its specific syntactic behavior as well as the overall word order of the language:

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

To draw a simple contrast, in Spanish we would expect adjectives to take the category $\mathsf{N\backslash N}$ because Spanish word ordering dictates that the adjective follow the noun. The lexical categories also capture word-word dependencies: head-argument relations are captured by the lexical category of the head (e.g. $\mathsf{(S\backslash N)/N}$), whereas head-modifier relations are captured by the lexical category of the modifier, which is of the form $\mathsf{X\backslash X}$ or $\mathsf{X/X}$, and may take further arguments of its own. Our goal will be to automatically learn these types of lexicons for a language. In Figure 3, we juxtapose several such lexicons which were automatically discovered by our system.

The rules of CCG are defined by a small set of of combinatory rules, which are traditionally written as schemas that define how constituents can be combined in a bottom-up fashion (although generative probability models for CCG view them in a top-down manner, akin to CFG rules). The first, and most obvious, of these rules is function application:

$$\mathsf{X/Y \;\; Y} \Rightarrow \mathsf{X} \qquad (\mathsf{B}^0_{>})$$
$$\mathsf{Y \;\; X\backslash Y} \Rightarrow \mathsf{X} \qquad (\mathsf{B}^0_{<})$$

Here the functor $\mathsf{X/Y}$ or $\mathsf{X\backslash Y}$ is applied to an argument Y resulting in X. While standard CCG has a number of additional combinatory rules (type-raising, generalized variants of composition and substitution) that increase its generative capacity beyond context-free grammars and allow an elegant treatment of non-local dependencies arising in extraction, coordination and scrambling, we follow Bisk and Hockenmaier (2012b) and use a restricted fragment, without type-raising, that allows only basic composition and is context-free:

$$\mathsf{X/Y \;\; Y/Z} \Rightarrow \mathsf{X/Z} \qquad (\mathsf{B}^1_{>})$$
$$\mathsf{X/Y \;\; Y\backslash Z} \Rightarrow \mathsf{X\backslash Z} \qquad (\mathsf{B}^1_{\mathsf{X}>})$$
$$\mathsf{Y\backslash Z \;\; X\backslash Y} \Rightarrow \mathsf{X\backslash Z} \qquad (\mathsf{B}^1_{<})$$
$$\mathsf{Y/Z \;\; X\backslash Y} \Rightarrow \mathsf{X/Z} \qquad (\mathsf{B}^1_{\mathsf{X}<})$$

The superscript 1 denotes the arity of the composition which is too low to recover non-projective dependencies, and our grammar is thus weakly equivalent to the dependency grammar representations that are commonly used for grammar induction. The main role of composition in our fragment is that it allows sentential and verb modifiers to both take categories of the form $\mathsf{S\backslash S}$ and $\mathsf{S/S}$. Composition in-

troduces spurious ambiguities, which we eliminate by using Eisner (1996)'s normal form.[1]

Coordinating conjunctions have a special category conj, and we binarize coordination as follows (Hockenmaier and Steedman, 2007):

X	X[conj]	$\Rightarrow_{\&1}$	X	(&1)
conj	X	$\Rightarrow_{\&2}$	X[conj]	(&2)

3 Category induction

Unlike dependency grammars, CCG requires an inventory of lexical categories. Given a set of lexical categories, the combinatory rules define the set of parses for each sentence. We follow the algorithm proposed by Bisk and Hockenmaier (2012b) to automatically induce these categories. The lexicon is initialized by pairing all nominal tags (nouns, pronouns and determiners) with the category N, all verb tags with the category S, and coordinating conjunctions with the category conj:

CONJ	$\rightarrow$	conj
DET, NOUN, NUM, PRON	$\rightarrow$	N
VERB	$\rightarrow$	S

Although our lexicons are defined over corpus-specific POS tags, we use a slightly modified version of Petrov et al. (2012)'s Universal POS tagset to categorize them into these broad classes. The primary changes we make to their mappings are the addition of a distinction (where possible) between subordinating and coordinating conjunctions and between main and auxiliary verbs[2].

Since the initial lexicon consists only of atomic categories, it cannot parse any complex sentences:

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

Complex lexical categories are induced by considering the local context in which tokens appear. Given an input sentence, and a current lexicon which assigns categories to at least some of the tokens in the sentence, we apply the following two rules to add new categories to the lexicon: The argument rule allows any lexical tokens that have categories other than N and conj to take immediately adjacent Ns as arguments. The modifier rule allows any token (other than coordinating conjunctions that appear in the middle of sentences) to modify an immediate neighbor that has the category S or N or is a modifier (S|S or N|N) itself.

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

These rules can be applied iteratively to form more complex categories. We restrict lexical categories to a maximal arity of 2, and disallow the category (S/N)\N, since it is equivalent to (S\N)/N.

The	*man*	*ate*	*quickly*
DT	NNS	VBD	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, (**N\N**)\(**N\N**)

The resultant, overly general, lexicon is then used to parse the training data. Each complete parse has to be of category S or N, with the constraint that sentences that contain a main verb can only form parses of category S.

4 A new probability model for CCG

Generative models define the probability of a parse tree τ as the product of individual rule probabilities. Our previous work (Bisk and Hockenmaier, 2012b) uses the most basic model of Hockenmaier and Steedman (2002), which first generates the head direction (left, right, unary, or lexical), followed by the head category, and finally the sister category. [3]

This factorization does not take advantage of the unique functional nature of CCG. We therefore introduce a new factorization we call the Argument Model. It exploits the fact that CCG imposes strong constraints on a category's left and right children, since these must combine to create the parent type via one of the combinators. In practice this means that given the parent X/Z, the choice of combinator[4] c and an argument Y we can uniquely determine the categories of the left and right children:

[1]The normal-form of Hockenmaier and Bisk (2010) is not required for this fragment of CCG.

[2]This distinction was suggested by the authors (p.c.)

[3]Huang et al. (2012) present a (deficient) variant and Bayesian extension of the Bisk and Hockenmaier (2012b) model without k-best smoothing that both underperform our published results.

[4]If X is an atomic category, only application is possible.

Parent	c	$\Rightarrow$	Left	Right
$\mathsf{X/Z}$	$\mathsf{B}^0_>$		$\mathsf{(X/Z)/Y}$	Y
	$\mathsf{B}^0_<$		Y	$\mathsf{(X/Z)\backslash Y}$
	$\mathsf{B}^1_>$		$\mathsf{X/Y}$	$\mathsf{Y/Z}$
	$\mathsf{B}^1_<$		$\mathsf{Y/Z}$	$\mathsf{X\backslash Y}$

and correspondingly for $\mathsf{X\backslash Z}$:

Parent	c	$\Rightarrow$	Left	Right
$\mathsf{X\backslash Z}$	$\mathsf{B}^0_>$		$\mathsf{(X\backslash Z)/Y}$	Y
	$\mathsf{B}^0_<$		Y	$\mathsf{(X\backslash Z)\backslash Y}$
	$\mathsf{B}^1_>$		$\mathsf{X/Y}$	$\mathsf{Y\backslash Z}$
	$\mathsf{B}^1_<$		$\mathsf{Y\backslash Z}$	$\mathsf{X\backslash Y}$

While type-changing and raising are not used in this work the model's treatment of root productions extends easily to handle these other unary cases. We simply treat the argument Y as the unary outcome so that the parent, combinator and argument uniquely specify every detail of the unary rule:

Parent	c	$\Rightarrow$	Y
TOP	TOP		$\in \{\mathsf{S}, \mathsf{N}\}$
$\mathsf{S/(S\backslash N)}$	$\mathsf{T}_<$		N
$\mathsf{S\backslash(S/N)}$	$\mathsf{T}_>$		N

We still distinguish the same rule types as before (lexical, unary, binary with head left/right), leading us to the following model definition:

Given: $\mathsf{P := X/Z}$
where $\text{type}(t) \in \{\text{Left,Right,Unary,Lex}\}$

$$p(t|\mathsf{P}) \times \begin{cases} p(w|\mathsf{P}, t) & \text{Lex} \\ \underbrace{p(\mathsf{Y}|\mathsf{P}, t)}_{\text{Argument}} \times \underbrace{p(c|\mathsf{P}, t, \mathsf{Y})}_{\text{Combinator}} & \text{o.w.} \end{cases}$$

Note that this model generates only one CCG category but uniquely defines the two children of a parent node. We will see below that this greatly simplifies the development of non-parametric extensions.

5 HDP-CCG: a nonparametric model

Simple generative models such as PCFGs or Bisk and Hockenmaier (2012b)'s CCG model are not robust in the face of sparsity, since they assign zero probability to any unseen event. Sparsity is a particular problem for formalisms like CCG that have a rich inventory of object types. Nonparametric Bayesian models, e.g. Dirichlet Processes (Teh, 2010) or their hierarchical variants (Teh et al., 2006) and generalizations (Teh, 2006) overcome this problem in a very elegant manner, and are used by many state-of-the-art grammar induction systems (Naseem et al., 2010; Blunsom and Cohn, 2010; Boonkwan and Steedman, 2011). They also impose a rich-getting-richer behavior that seems to be advantageous in many modeling applications. By contrast, Bisk and Hockenmaier (2012b) propose a weighted top-k scheme to address these issues in an ad-hoc manner.

The argument model introduced above lends itself particularly well to nonparametric extensions such as the standard Hierarchical Dirichlet Processes (HDP). In this work the size of the grammar and the number of productions are fixed and small, but we present the formulation as infinite to allow for easy extension in the future. Specifically, this framework allows for extensions which grow the grammar during parsing/training or fully lexicalize the productions. Additionally, while our current work uses only a restricted fragment of CCG that has only a finite set of categories, the literature's generalized variants of composition make it possible to generate categories of unbounded arity. We therefore believe that this is a very natural probabilistic framework for CCG, since HDPs make it possible to consider a potentially infinite set of categories that can instantiate the Y slot, while allowing the model to capture language-specific preferences for the set of categories that can appear in this position.

The HDP-CCG model In Bayesian models, multinomials are drawn from a corresponding n-dimensional Dirichlet distribution. The Dirichlet Process (DP) generalizes the Dirichlet distribution to an infinite number of possible outcomes, allowing us to deal with a potentially infinite set of categories or words. DPs are defined in terms of a base distribution H that corresponds to the mean of the DP, and a concentration or shape parameter α. In a Hierarchical Dirichlet Process (Teh et al., 2006), there is a hierarchy of DPs, such that the base distribution of a DP at level n is a DP at level $n-1$.

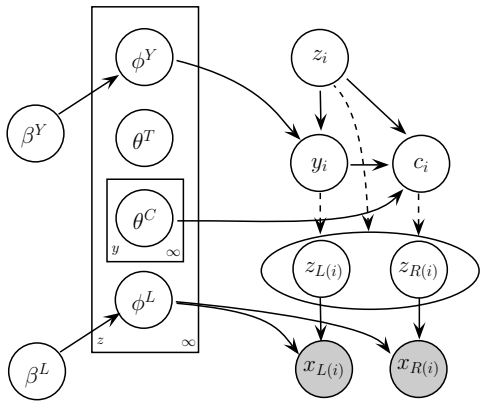
The HDP-CCG (Figure 1) is a reformulation of the Argument Model introduced above in terms of Hierarchical Dirichlet Processes.[5] At the heart of the model is a distribution over CCG categories. By combining a stick breaking process with a multinomial over categories we can define a DP over CCG

[5] An alternative HDP model for semantic parsing with CCG is proposed by Kwiatkowski et al. (2012).

HDP-CCG

1) Draw global parameters
- Define MLE root parameter θ_{TOP}
- Draw top-level symbol weights $\beta^Y \sim \text{GEM}(\alpha^Y)$
- Draw top-level lexical weights $\beta^L \sim \text{GEM}(\alpha^L)$
- For each grammar symbol $z \in \{1, 2, ...\}$:
  - Define MLE rule type parameters θ^T_z
  - Draw argument parameters $\phi^Y_z \sim \text{DP}(\alpha^Y, \beta^Y)$
  - Draw lexical emission parameters $\phi^L_z \sim \text{DP}(\alpha^L, \beta^L)$
  - For each grammar symbol $y \in \{1, 2, ...\}$:
    - Define MLE combinator parameters $\theta^C_{z,y}$

2) For each parse tree:
- Generate root node $z_{TOP} \sim \text{Binomial}(\theta_{TOP})$
- For each node i in the parse tree:
  - Choose rule type $t_i \sim \text{Multinomial}(\theta^T_{z_i})$
  - If t_i == Lex:
    - Emit terminal symbol $x_i \sim \text{Multinomial}(\phi^L_{z_i})$
  - If t_i == Left/Right/Unary:
    - Generate argument category $y_i \sim \text{Multinomial}(\phi^Y_{z_i})$
    - Generate combinator $c_i \sim \text{Multinomial}(\theta^C_{z_i,y_i})$
    - Deterministically create $z_{L(i)}$ (and $z_{R(i)}$ if binary)



Because we are working with CCG, the parent z_i, argument y_i and combinator c_i uniquely define the two children categories $(z_{L(i)}, z_{R(i)})$. The dashed arrows here represent the deterministic process used to generate these two categories.

Figure 1: The HDP-CCG has two base distributions, one over the space of categories and the other over words (or tags). For every grammar symbol, an argument distribution and emission distribution is drawn from the corresponding Dirichlet Processes. In addition, there are several MLE distributions tied to a given symbol for generating rule types, combinators and lexical tokens.

categories whose stick weights (β^Y) correspond to the frequency of the category in the corpus. Next we build the hierarchical component of our model by choosing an argument distribution (ϕ^Y), again over the space of categories, for every parent X/Z. This argument distribution is drawn from the previously defined base DP, allowing for an important level of parameter tying across all argument distributions.

While the base DP does define the mean around which all argument distributions are drawn, we also require a notion of variance or precision which determines how similar individual draws will be. This precision is determined by the magnitude of the hyperparameter α^Y. This hierarchy is paralleled for lexical productions which are drawn from a unigram base DP over terminal symbols controlled by α^L. For simplicity we use the same scheme for setting the values for α^L as α^Y. We present experimental results in which we vary the value of α^Y as a function of the number of outcomes allowed by the grammar for argument categories or the corpus in the case of terminal symbols. Specifically, we set $\alpha^Y = n^p$ for conditioning contexts with n outcomes. Since Liang et al. (2009) found that the ideal value for alpha appears to be superlinear but subquadratic in n, we present results where p takes the values 0, 1.0, 1.5, and 2.0 to explore the range from uniform to quadratic. This setting for α is the only free parameter in the model. By controlling precision we can tell the model to what extent global corpus statistics should be trusted. We believe this has a similar effect to Bisk and Hockenmaier (2012b)'s top-k upweighting and smoothing scheme.

One advantage of the argument model is that it only requires a single distribution over categories for each binary tree. In contrast to similar proposals for CFGs (Liang et al., 2007), which impose no formal restrictions on the nonterminals X, Y, Z that can appear in a rewrite rule $\mathsf{X} \rightarrow \mathsf{Y}\ \mathsf{Z}$, this greatly simplifies the modeling problem (yielding effectively a model that is more akin to nonparametric HMMs), since it avoids the need to capture correlations be-

tween different base distributions for Y and Z.

Variational Inference HDPs need to be estimated with approximate techniques. As an alternative to Gibbs sampling (Teh et al., 2006), which is exact, but typically very slow and has no clear convergence criteria, variational inference algorithms (Bishop, 2006; Blei and Jordan, 2004) estimate the parameters of a truncated model to maximize a lower bound of the likelihood of the actual model. This allows for factorization of the model and a training procedure analogous to the Inside-Outside algorithm (Lari and Young, 1991), allowing training to run very quickly and in a trivially parallelizable manner.

To initialize the base DP's stick weights, we follow the example of Kurihara et al. (2007) and use an MLE model initialized with uniform distributions to compute global counts for the categories in our grammar. When normalized these provide a better initialization than a uniform set of weights. Updates to the distributions are then performed in a coordinate descent manner which includes re-estimation of the base DPs.

In variational inference, multinomial weights W take the place of probabilities. The weights for an outcome Y with conditioning variable P are computed by summing pseudocounts with a scaled mean vector from the base DP. The computation involves moving in the direction of the gradient of the Dirichlet distribution which results in the following difference of Digammas (Ψ):

$$W_P(Y) = \Psi(C(P,Y) + \alpha^P \beta_Y) - \Psi(C(P,*) + \alpha^P)$$

Importantly, the Digamma and multinomial weights comprise a righ-get-richer scheme, biasing the model against rare outcomes. In addition, since variational inference is done by coordinate descent, it is trivially parallelizeable. In practice, training and testing our models on the corpora containing sentences up to length 15 used in this paper takes between one minute to at most three hours on a single 12-core machine depending on their size.

6 Evaluation

As is standard for this task, we evaluate our systems against a number of different dependency treebanks, and measure performance in terms of the accuracy of directed dependencies (i.e. the percentage of words in the test corpus that are correctly attached). We use the data from the PASCAL challenge for grammar induction (Gelling et al., 2012), the data from the CoNLL-X shared task (Buchholz and Marsi, 2006) and Goldberg (2011)'s Hebrew corpus.

Converting CCG derivations into dependencies is mostly straightforward, since the CCG derivation identifies the root word of each sentence, and head-argument and head-modifier dependencies are easily read off of CCG derivations, since the lexicon defines them explicitly. Unlike dependency grammar, CCG is designed to recover non-local dependencies that arise in control and binding constructions as well as in wh-extraction and non-standard coordination, but since this requires re-entrancies, or co-indexation of arguments (Hockenmaier and Steedman, 2007), within the lexical categories that trigger these constructions, our current system returns only local dependencies. But since dependency grammars also captures only local dependencies, this has no negative influence on our current evaluation.

However, a direct comparison between dependency treebanks and dependencies produced by CCG is more difficult (Clark and Curran, 2007), since dependency grammars allow considerable freedom in how to analyze specific constructions such as verb clusters (which verb is the head?) prepositional phrases and particles (is the head the noun or the preposition/particle?), subordinating conjunctions (is the conjunction a dependent of the head of the main clause and the head of the embedded clause a dependent of the conjunction, or vice versa?) and this is reflected in the fact that the treebanks we consider often apply different conventions for these cases. Although remedying this issue is beyond the scope of this work, these discrepancies very much hint at the need for a better mechanism to evaluate linguistically equivalent structures or treebank standardization.

The most problematic construction is coordination. In standard CCG-to-dependency schemes, both conjuncts are independent, and the conjunction itself is not attached to the dependency graph, whereas dependency grammars have to stipulate that either one of the conjuncts or the conjunction itself is the head, with multiple possibilities of where the remaining

constituents attach. In addition to the standard CCG scheme, we have identified five main styles of conjunction in our data (Figure 2), although several corpora distinguish multiple types of coordinating conjunctions which use different styles (not all shown here). Since our system has explicit rules for coordination, we transform its output into the desired target representation that is specific to each language.

7 Experiments

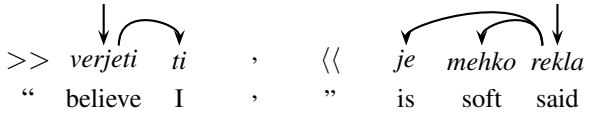
We evaluate our system on 13 different languages. In each case, we follow the test and training regimes that were used to obtain previously published results in order to allow a direct comparison. We compare our system to the results presented at the PASCAL Challenge on Grammar Induction (Gelling et al., 2012)[6], as well as to Gillenwater et al. (2011) and Naseem et al. (2012). We use Nivre (2006)'s Penn2Malt implementation of Collins (2003)'s head rules to translate the WSJ Penn Treebank (Marcus et al., 1993) into dependencies. Finally, when training the MLE version of our model we use a simple smoothing scheme which defines a small rule probability (e^{-15}) to prevent any rule used during training from going to zero.

7.1 PASCAL Challenge on Grammar Induction

In Table 1, we compare the performance of the basic Argument model (MLE), of our HDP model with four different settings of the hyperparameters (as explained above) and of the systems presented in the PASCAL Challenge on Grammar Induction (Gelling et al., 2012). The systems in this competition were instructed to train over the full dataset, including the unlabelled test data, and include Bisk and Hockenmaier (2012a)'s CCG-based system (BH) to Cohn et al. (2010)'s reimplementation of Klein and Manning (2004)'s DMV model in a tree-substitution grammar framework (BC), as well as three other dependency based systems which either incorporate Naseem et al. (2010)'s rules in a deterministic fashion (Søgaard, 2012), rely on extensive tuning on the development set (Tu, 2012) or incorporate millions of additional tokens from Wikipedia to estimate model parameters (Marecek and Zabokrtsky, 2012). We ignore punctuation for all experiments reported in this paper, but since the training data (but not the evaluation) includes punctuation marks, participants were free to choose whether to include punctuation or ignore it.

While BH is the only other system with directly interpretable linguistic output, we also include a direct comparison with BC, whose TSG representation is equally expressive to ours. Finally we present a row with the maximum performance among the other three models. As we have no knowledge of how much data was used in the training of other systems we simply present results for systems trained on length 15 (not including punctuation) sentences and then evaluated at lengths 10 and 15.

The MLE version of our model shows rather variable performance: although its results are particularly bad on Basque (Eu), it outperforms both BH and BC on some other settings. By contrast, the HDP system is always better than the MLE model. It outperforms all other systems on half of the corpora. On average, it outperforms BH and BC by 10.3% and 9.3% on length 10, or 9.7% and 7.8 % on length 15 respectively. The main reason why our system does not outperform BC by an even higher margin is the very obvious 11.4%/11.5% deficit on Slovene. However, the Slovene dependency treebank seems to follow a substantially different annotation scheme. In particular, the gold standard annotation of the 1,000 sentences in the Slovene development set treats many of them as consisting of independent sentences (often separated by punctuation marks that our system has no access to), so that the average number of roots per sentence is 2.7:



When our system is presented with these short components in isolation, it oftentimes analyzes them correctly, but since it has to return a tree with a single root, its performance degrades substantially.

We believe the HDP performs so well as compared to the MLE model because of the influence of the shared base distribution, which allows the

[6]Numbers are from personal correspondence with the organizers. The previously published numbers are not comparable to literature due to an error in the evaluation. http://wiki.cs.ox.ac.uk/InducingLinguisticStructure/ResultsDepComparable

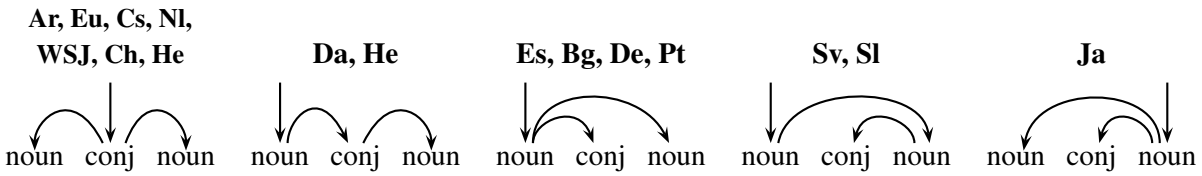


Figure 2: In the treebanks used for evaluation different standards exist for annotating coordination. While not exhaustive, this table demonstrates five of the most common schemes used in the literature. Syntactically these are identical and traditionally CCG draws arcs only to the arguments without attaching the conjunction. For the purposes of comparison with the literature we have implemented these five translation schemes.

		Arabic	**Danish**	**Slovene**	**Swedish**	**Dutch**	**Basque**	**Portuguese**	**WSJ**	**Childes**	**Czech**
	# Tokens	5,470	25,341	54,032	61,877	78,737	81,345	158,648	163,727	290,604	436,126
	# Tags	20	24	36	30	304	14	23	36	69	62
PASCAL	**BC**	60.8/58.4	44.7/39.4	62.6/57.9	63.2/56.6	**51.8/52.0**	53.0/**48.9**	52.4/50.2	68.6/63.3	47.4/46.1	47.9/43.1
PASCAL	**Max**	**67.2/66.8**	**60.1/56.0**	**65.6/61.8**	**72.8/63.4**	51.1/47.6	**53.7**/47.8	67.0/61.8	**71.2/64.8**	56.0/54.5	**58.3/54.4**
PASCAL	**BH**	41.6/43.7	46.4/43.8	49.6/43.9	63.7/57.0	49.7/43.6	45.1/39.6	**70.8/67.2**	68.2/59.6	**61.4/59.8**	45.0/38.9
This work	**MLE**	41.6/42.9	43.4/39.2	46.1/41.1	70.1/59.7	52.2/47.2	29.6/26.5	62.2/59.7	59.5/52.4	53.3/51.9	50.5/45.8
This work	**HDP**$_{0.0}$	48.0/50.0	**63.9/58.5**	44.8/39.8	67.6/62.1	45.0/33.9	41.6/39.1	71.0/66.0	59.8/52.9	56.3/55.2	54.0/49.0
This work	**HDP**$_{1.0}$	45.6/47.1	45.7/42.3	53.9/46.9	**74.5/66.9**	**58.5/54.4**	50.1/44.6	65.1/60.6	64.3/56.5	71.5/70.3	**55.8/50.7**
This work	**HDP**$_{1.5}$	49.6/50.4	58.7/54.4	53.2/48.2	74.3/67.1	57.4/54.5	**50.6/45.0**	70.0/64.7	65.5/57.2	69.6/68.6	55.6/50.3
This work	**HDP**$_{2.0}$	**66.4/65.1**	56.5/49.5	**54.2/46.4**	71.6/64.1	51.7/48.3	49.4/43.3	**76.3/70.5**	**70.7/62.9**	**74.1/73.3**	54.4/48.5
	+/−	-0.8/-1.7	+3.8/+2.5	-11.4/-15.4	+1.7/+3.5	+6.7/+2.4	-3.1/-3.9	+5.5/+3.3	-0.5/-1.9	+12.7/+13.5	-2.5/-3.7

Table 1: A comparison of the basic Argument model (MLE) and four hyper-parameter settings of the HDP-CCG against two syntactic formalisms that participated in the PASCAL Challenge (Gelling et al., 2012), BH (Bisk and Hockenmaier, 2012a) and BC (Blunsom and Cohn, 2010), in addition to a max over all other participants. We trained on length 15 data (punctuation removed), including the test data as recommended by the organizers. The last row indicates the difference between our best system and the competition.

global category distribution to influence each of the more specific distributions. Further, it provides a very simple knob in the choice of hyperparameters, which has a substantial effect on performance. A side effect of the hyperparameters is that their strength also determines the rate of convergence. This may be one of the reasons for the high variance seen in the four settings tested, although we note that since our initialization is always uniform, and not random, consecutive runs do not introduce variance in the model's performance.

7.2 Comparison with systems that capture linguistic constraints

Since our induction algorithm is based on the knowledge of which POS tags are nouns and verbs, we compare in Table 2 our system to Naseem et al. (2010), who present a nonparametric dependency model that incorporates thirteen universal linguistic constraints. Three of these constraints correspond to our rules that verbs are the roots of sentences and may take nouns as dependents, but the other ten constraints (e.g. that adjectives modify nouns, adverbs modify adjectives or verbs, etc.) have no equivalent in our system. Although our system has less prior knowledge, it still performs competitively.

On the WSJ, Naseem et al. demonstrate the importance and effect of the specific choice of syntactic rules by comparing the performance of their system with hand crafted universal rules (71.9), with English specific rules (73.8), and with rules proposed by Druck et al. (2009) (64.9). The performance of Naseem et al.'s system drops very significantly as sentence length (and presumable parse complexity)

	Sl	Es	Da	Pt	Sv
∼#Tokens	3.8K	4.2K	9.5K	15K	24K
N10	50.9	**67.2**	**51.9**	71.5	63.3
HDP	**56.6**	62.1	51.5	**74.7**	**69.8**

Table 2: A comparison of our system with Naseem et al. (2010), both trained and tested on the length 10 training data from the CoNLL-X Shared Task.

increases, whereas our system shows significantly less decline, and outperforms their universal system by a significant margin.[7]

	≤ 10	≤ 20
Naseem Universal Rules	71.9	50.4
Naseem English Rules	**73.8**	**66.1**
HDP-CCG	68.2	64.2
HDP-CCG (train ≤ 20)	71.9	

In contrast to Spitkovsky et al. (2010), who reported that performance of their dependency based system degrades when trained on longer sentences, our performance on length 10 sentences increases to **71.9** when we train on sentences up to length 20.

Another system that is also based on CCG, but captures significantly more linguistic knowledge than ours, was presented by Boonkwan and Steedman (2011), who achieve an accuracy of 74.5 on WSJ10 section 23 (trained on sections 02-22). Using the same settings, our system achieves an accuracy of 68.4. Unlike our approach, Boonkwan and Steedman do not automatically induce an appropriate inventory of lexical category, but use an extensive questionnaire that defines prototype categories for various syntactic constructions, and requires significant manual engineering of which POS tags are mapped to what categories to generate a language-specific lexicon. However, their performance degrades significantly when only a subset of the questions are considered. Using only the first 14 questions, covering facts about the ordering of subjects, verbs and objects, adjectives, adverbs, auxiliaries, adpositions, possessives and relative markers, they achieve an accuracy of 68.2, which is almost identical to ours, even though we use significantly less initial knowledge. However, the lexicons we present below indicate that we are in fact learning many of the very exact details that in their system are constructed by hand. The remaining 14 questions in Boonkwan and Steedman's questionnaire cover less frequent phenomena such as the order of negative markers, dative shift, and pro-drop. The obvious advantage of this approach is that this allows them to define a much more fine-grained inventory of lexical categories than our system can automatically induce. We also stipulate that for certain languages knowledge of pro-drop could play a significant role in the success of their approach: if complete sentences are allowed to be of the form $\mathsf{S}\backslash\mathsf{N}$ or S/N, the same lexical category can be used for the verb regardless of whether the subject is present or has been dropped.

[7] Our earlier generative model showed similar behavior, although the results in Bisk and Hockenmaier (2012b) are not directly comparable due to differences in the data.

	Sl	Es	Da	Pt	Sv
#Tokens	3,857	4,230	9,549	15,015	24,021
G10	51.2	62.4	47.2	54.3	48.6
HDP	**57.9**	**65.4**	**49.3**	**73.5**	**73.2**
	Bg	**WSJ**	**Nl**	**Ja**	**De**
#Tokens	38,220	42,442	43,405	43,501	77,705
G10	59.8	64.4	47.5	60.2	47.4
HDP	**66.1**	**70.3**	**56.2**	**64.1**	**68.4**

Table 3: A comparison of our system with Gillenwater et al. (2010), both trained on the length 10 training data, and tested on the length 10 test data, from the CoNLL-X Shared task.

7.3 Additional Languages

In order to provide results on additional languages, we present in Table 3 a comparison to the work of Gillenwater et al. (2010) (G10), using the ConLL-X Shared Task data (Buchholz and Marsi, 2006). Following Gillenwater et al., we train only on sentences of length 10 from the training set and evaluate on the test set. Since this is a different training regime, and these corpora differ for many languages from that of the PASCAL challenge, numbers from Table 1 cannot be compared directly with those in Table 3. We have also applied our model to Goldberg (2011)'s Hebrew corpus, where it achieves an accuracy of 62.1 (trained and tested on all sentences length 10; 7,253) and 59.6 (length 15; 21,422 tokens).

	Arabic	%	Swedish	%	WSJ	%	Childes	%	Japanese	%	Czech	%
VERB	(S\N)/N	56	S	45	S\N	52	S/N	44	S	84	S	26
	(S/N)/N	29	S\N	20	(S\N)/N	19	S	37			S\N	25
ADP	N\N	68	(S\S)/N	49	(S\S)/N	46	(S\S)/N	45	(S/S)\N	44	(S\S)/N	42
	N/N	21	(N\N)/N	25	(N\N)/N	20	N/N	25	N\N	23	(S/S)/N	26
NOUN	N\N	50	N	91	N	79	N	89	N	73	N	74
	N	35			N/N	12						
ADJ	N\N	82	N/N	50	N/N	70	N/N	46	S/S	64	N/N	55

Figure 3: Partial lexicons demonstrating language specific knowledge learned automatically for five languages. For ease of comparison between languages, we use the universal tag label (Verb, Adposition, Noun and Adjective). Shown are the most common categories and the fraction of occurrences of the tag that are assigned this category (according to the Viterbi parses).

8 The Induced Lexicons

Since our approach is based on a lexicalized formalism such as CCG, our system automatically induces lexicons that pair words (or, in our case, POS-tags) with language-specific categories that capture their syntactic behavior. If our approach is successful, it should learn the basic syntactic properties of each language, which will be reflected in the corresponding lexicon. In Figure 3 one sees how verbs subcategorize differently, how word ordering differs by language, and how the attachment structures of prepositions are automatically discovered and differ across languages. In Arabic, for example, the system learns that word order is variable and therefore the verb must allow for both SVO and VOS style constructions. We generally learn that adpositions (prepositions or postpositions) take nouns as arguments. In Czech, PPs can appear before and after the verb, leading to two different categories ((S\S)/N and (S/S)/N). Japanese has postpositions that appear in preverbal position ((S/S)\N), but when this category is assigned to nominal particles that correspond to case markers, it effectively absorbs the noun, leading to a preference for verbs that do not take any arguments (S), and to a misanalysis of adjectives as verb modifiers (S/S). Our lexicons also reflect differences in style: while Childes and the WSJ are both English, they represent very different registers. We learn that subjects are mostly absent in the informal speech and child-directed instructions contained in Childes, while effectively mandatory in the Wall Street Journal.

9 Conclusions

This paper has introduced a novel factorization for CCG models and showed how when combined with non-parametric Bayesian statistics it can compete with every other grammar induction system currently available, including those that capture a significant amount of prior linguistic knowledge. The use of a powerful syntactic formalism proves beneficial both in terms of requiring very limited universal knowledge and robustness at longer sentence lengths. Unlike standard grammar induction systems that are based on dependency grammar, our system returns linguistically interpretable lexicons for each language that demonstrate it has discovered their basic word order. Of particular note is the simplicity of the model both algorithmically and in terms of implementation. By not faltering on longer sentences or requiring extensive tuning, the system can be easily and quickly deployed on a new language and return state of the art performance and easily interpretable lexicons. In this paper, we have applied this model only to a restricted fragment of CCG, but future work will address the impact of lexicalization and the inclusion of richer combinators.

10 Acknowledgements

This work is supported by NSF CAREER award 1053856 (Bayesian Models for Lexicalized Grammars).

References

Taylor Berg-Kirkpatrick and Dan Klein. 2010. Phylogenetic Grammar Induction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1288–1297, Uppsala, Sweden, July.

Christopher Bishop. 2006. *Pattern Recognition and Machine Learning*. Springer-Verlag, August.

Yonatan Bisk and Julia Hockenmaier. 2012a. Induction of Linguistic Structure with Combinatory Categorial Grammars. In *NAACL HLT Workshop on Induction of Linguistic Structure*, pages 90–95, Montréal, Canada, June.

Yonatan Bisk and Julia Hockenmaier. 2012b. 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.

David M Blei and Michael I Jordan. 2004. Variational Methods for the Dirichlet Process. In *Proceedings of the Twenty-First International Conference on Machine Learning (ICML 2004)*, Banff, Alberta, Canada, July.

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

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X Shared Task on Multilingual Dependency Parsing. In *Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-X)*, pages 149–164, New York City, June.

Glenn Carroll and Eugene Charniak. 1992. Two Experiments on Learning Probabilistic Dependency Grammars from Corpora. *Working Notes of the Workshop Statistically-Based NLP Techniques*, pages 1–13.

Eugene Charniak. 1993. *Statistical Language Learning*. The MIT Press, Cambridge, Massachusetts.

Stephen Clark and James R Curran. 2007. Formalism-Independent Parser Evaluation with CCG and DepBank. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 248–255, Prague, Czech Republic, June.

Alex Clark. 2001. *Unsupervised Language Acquisition: Theory and Practice*. Ph.D. thesis, University of Sussex, September.

Shay B Cohen and Noah A Smith. 2009. Variational Inference for Grammar Induction with Prior Knowledge. *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 1–4.

Shay B Cohen and Noah A Smith. 2010. Covariance in Unsupervised Learning of Probabilistic Grammars. *The Journal of Machine Learning Research*, pages 3117–3151, November.

Trevor Cohn, Phil Blunsom, and Sharon Goldwater. 2010. Inducing Tree-Substitution Grammars. *The Journal of Machine Learning Research*, 11:3053–3096, November.

Michael Collins. 2003. Head-Driven Statistical Models for Natural Language Parsing. *Computational Linguistics*, 29(4):589–637, December.

Gregory Druck, Gideon Mann, and Andrew McCallum. 2009. Semi-supervised Learning of Dependency Parsers using Generalized Expectation Criteria. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 360–368, Suntec, Singapore, August.

Jason Eisner. 1996. Efficient Normal-Form Parsing for Combinatory Categorial Grammar. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 79–86, Santa Cruz, California, USA, June.

Douwe Gelling, Trevor Cohn, Phil Blunsom, and João V Graca. 2012. The PASCAL Challenge on Grammar Induction. In *NAACL HLT Workshop on Induction of Linguistic Structure*, pages 64–80, Montréal, Canada, June.

Jennifer Gillenwater, Kuzman Ganchev, João V Graca, Fernando Pereira, and Ben Taskar. 2010. Sparsity in Dependency Grammar Induction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 194–199, Uppsala, Sweden, July.

Jennifer Gillenwater, Kuzman Ganchev, João V Graca, Fernando Pereira, and Ben Taskar. 2011. Posterior Sparsity in Unsupervised Dependency Parsing. *The Journal of Machine Learning Research*, 12:455–490, February.

Yoav Goldberg. 2011. *Automatic Syntactic Processing of Modern Hebrew*. Ph.D. thesis, Ben-Gurion University of the Negev, November.

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, Colorado, June.

Julia Hockenmaier and Yonatan Bisk. 2010. Normal-form parsing for Combinatory Categorial Grammars with generalized composition and type-raising. In

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 465–473, Beijing, China, August. Coling 2010 Organizing Committee.

Julia Hockenmaier and Mark Steedman. 2002. Generative Models for Statistical Parsing with Combinatory Categorial Grammar. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 335–342, Philadelphia, Pennsylvania, USA, July.

Julia Hockenmaier and Mark Steedman. 2007. CCGbank: A Corpus of CCG Derivations and Dependency Structures Extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396, September.

Yun Huang, Min Zhang, and Chew Lim Tan. 2012. Improved Combinatory Categorial Grammar Induction with Boundary Words and Bayesian Inference. In *Proceedings of the 24rd International Conference on Computational Linguistics (Coling 2012)*, Mumbai, India, December.

Ray Jackendoff. 1977. *X-Bar Syntax: A Study of Phrase Structure*. The MIT Press.

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.

Kenichi Kurihara and Taisuke Sato. 2004. An Application of the Variational Bayesian Approach to Probabilistic Context-Free Grammars. *International Joint Conference on Natural Language Language Processing Workshop Beyond Shallow Analyses*, March.

Kenichi Kurihara, Max Welling, and Yee-Whye Teh. 2007. Collapsed Variational Dirichlet Process Mixture Models. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI07)*, pages 2796–2801, Hyderabad, India, January.

Tom Kwiatkowski, Sharon Goldwater, Luke Zettlemoyer, and Mark Steedman. 2012. A probabilistic model of syntactic and semantic acquisition from child-directed utterances and their meanings. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 234–244, Avignon, France, April. Association for Computational Linguistics.

Karim Lari and Steve J Young. 1991. Applications of stochastic context-free grammars using the Inside-Outside algorithm. *Computer speech & language*, 5(3):237–257, January.

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.

Percy Liang, Michael I Jordan, and Dan Klein. 2009. Probabilistic Grammars and Hierarchical Dirichlet Processes. In *The Oxford Handbook of Applied Bayesian Analysis*. Oxford University Press.

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

David Marecek and Zdenek Zabokrtsky. 2012. Unsupervised Dependency Parsing using Reducibility and Fertility features. In *NAACL HLT Workshop on Induction of Linguistic Structure*, pages 84–89, Montréal, Canada, June.

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, MA, October.

Tahira Naseem, Regina Barzilay, and Amir Globerson. 2012. Selective Sharing for Multilingual Dependency Parsing. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 629–637, Jeju, Republic of Korea, July.

Joakim Nivre. 2006. *Inductive Dependency Parsing*. Springer.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A Universal Part-of-Speech Tagset. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC-2012)*, pages 2089–2096, Istanbul, Turkey, May.

Anders Søgaard. 2012. Two baselines for unsupervised dependency parsing. In *NAACL HLT Workshop on Induction of Linguistic Structure*, pages 81–83, Montréal, Canada, June.

Valentin I Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2010. From Baby Steps to Leapfrog: How "Less is More" in Unsupervised Dependency Parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 751–759, Los Angeles, California, June.

Mark Steedman. 1996. *Surface Structure and Interpretation*. The MIT Press, January.

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

Yee-Whye Teh, Michael I Jordan, Matthew J Beal, and David M Blei. 2006. Hierarchical Dirichlet Pro-

cesses. *Journal of the American Statistical Association*, 101(476):1566–1581.

Yee-Whye Teh. 2006. A Hierarchical Bayesian Language Model based on Pitman-Yor Processes. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 985–992, Sydney, Australia, July.

Yee-Whye Teh. 2010. Dirichlet Process. In *Encyclopedia of Machine Learning*, pages 280–287. Springer.

Kewei Tu. 2012. Combining the Sparsity and Unambiguity Biases for Grammar Induction. In *NAACL HLT Workshop on Induction of Linguistic Structure*, pages 105–110, Montréal, Canada, June.