Weakly-Supervised Grammar-Informed Bayesian CCG Parser Learning

Dan Garrette[*] **Chris Dyer**[†] **Jason Baldridge**[‡] **Noah A. Smith**[†]

[*]Department of Computer Science, University of Texas at Austin, dhg@cs.utexas.edu
[†]School of Computer Science, Carnegie Mellon University, {cdyer,nasmith}@cs.cmu.edu
[‡]Department of Linguistics, University of Texas at Austin, jbaldrid@utexas.edu

Abstract

Combinatory Categorial Grammar (CCG) is a lexicalized grammar formalism in which words are associated with categories that, in combination with a small universal set of rules, specify the syntactic configurations in which they may occur. Previous work has shown that learning sequence models for CCG tagging can be improved by using priors that are sensitive to the formal properties of CCG as well as cross-linguistic universals. We extend this approach to the task of learning a full CCG parser from weak supervision. We present a Bayesian formulation for CCG parser induction that assumes only supervision in the form of an incomplete tag dictionary mapping some word types to sets of potential categories. Our approach outperforms a baseline model trained with uniform priors by exploiting universal, intrinsic properties of the CCG formalism to bias the model toward simpler, more cross-linguistically common categories.

Introduction

Supervised learning of natural language parsers of various types (context-free grammars, dependency grammars, categorial grammars, and the like) is by now a well-understood task with plenty of high-performing models—when training data is abundant. Learning from sparse, incomplete information is, naturally, a greater challenge. To build parsers for domains and languages where resources are scarce, we need techniques that take advantage of very limited kinds and amounts of supervision. The strategy we pursue in this paper is to approach the problem in a Bayesian framework, using priors built from linguistic knowledge such as grammar universals, linguistic typology, and cheaply obtained annotations from a linguist.

We focus on the task of learning parsers for Combinatory Categorial Grammar (CCG) (Steedman 2000; Steedman and Baldridge 2011) without having access to annotated parse trees. CCG is a lexicalized grammar formalism in which every constituent in a parse is assigned a *category* that describes its grammatical role in the sentence. CCG categories, in contrast to the labels used in standard phrase structure grammars, are not atomic labels like ADJECTIVE or VERB PHRASE, but instead have a detailed recursive structure. Instead of VERB, a CCG category might encode information such as "this category can combine with a noun phrase to the right (an object) and then a noun phrase to the left (a subject) to produce a sentence."

Practical interest in CCG has grown in the last few years within an array of NLP applications—of particular note are semantic parsing (Zettlemoyer and Collins 2005) and machine translation (Weese, Callison-Burch, and Lopez 2012). As these tasks mature and move into new domains and languages, the ability to learn CCG parsers from scarce data will be increasingly important. In this regard, CCG has particular appeal—both theoretical and practical—in the setting of weakly-supervised learning because the structure of its categories and its small set of universal rules provide a foundation for constructing linguistically-informative priors that go beyond general preferences for sparseness that are commonly expressed with generic priors. In this paper, we show that the intrinsic structure of CCG categories can be exploited cross-linguistically to learn better parsers when supervision is scarce.

Our starting point is the method we presented in Garrette et al. (2014) for specifying a probability distribution over the space of potential CCG categories. This distribution served as a prior for the training of a hidden Markov model (HMM) *supertagger*, which assigns a category to each word in a sentence (lexical categories are often called "supertags"). The prior is designed to bias the HMM toward the use of cross-linguistically common categories: those that are less complex or those that are modifiers of other categories. This method improves supertagging performance in limited data situations; however, supertags are just the start of a full syntactic analysis of a sentence, and—for all but very short sentences—an HMM is very unlikely to produce a sequence of supertags that can actually be combined into a complete tree. Here we model the entire tree, which additionally requires supertags for different words to be compatible with each other.

Bisk and Hockenmaier (2013) present a model of CCG parser induction from only basic properties of the CCG formalism. However, their model produces trees that use only a simplified form of CCG consisting of just two atomic categories, and they require gold-standard part-of-speech tags for each token. We wish to model the same kinds of complex

categories encoded in resources like CCGBank or which are used in hand-crafted grammars such as those used with OpenCCG (Baldridge et al. 2007), and to do so by starting with words rather than gold tags.

Our inputs are unannotated sentences and an incomplete tag dictionary mapping some words to their potential categories, and we model CCG trees with a probabilistic context-free grammar (PCFG). The parameters of the PCFG are estimated using a blocked sampling algorithm based on the Markov chain Monte Carlo approach of Johnson, Griffiths, and Goldwater (2007). This allows us to efficiently sample parse trees for sentences in an unlabeled training corpus according to their posterior probabilities as informed by the linguistically-informed priors. This approach yields improvements over a baseline uniform-prior PCFG. Further, as a demonstration of the universality of our approach in capturing valuable grammatical biases, we evaluate on three diverse languages: English, Italian, and Chinese.

Combinatory Categorial Grammar

In the CCG formalism, every constituent, including those at the lexical level, is associated with a structured CCG *category* that provides information about that constituent's relationships in the overall grammar of the sentence. Categories are defined by a simple recursive structure, where a category is either *atomic*, which may potentially have *features* that restrict the categories with which they can unify, or a function from one category to another, as indicated by one of two slash operators:

$$C \rightarrow \{\mathsf{s}, \mathsf{s_{dcl}}, \mathsf{s_{adj}}, \mathsf{s_b}, \mathsf{n}, \mathsf{np}, \mathsf{np_{nb}}, \mathsf{pp}, ...\}$$
$$C \rightarrow \{(C/C), (C\backslash C)\}$$

Categories of adjacent constituents can be combined using one of a set of combination rules to form categories of higher-level constituents, as seen in Figure 1. The direction of the slash operator gives the behavior of the function. A category $(\mathsf{s}\backslash\mathsf{np})/\mathsf{pp}$ might describe an intransitive verb with a prepositional phrase complement; it combines on the right ($/$) with a constituent with category pp, and then on the left ($\backslash$) with a noun phrase (np) that serves as its subject.

We follow Lewis and Steedman (2014) in allowing only a small set of generic, linguistically-plausible grammar rules. More details can be found there; in addition to the standard binary combination rules, we use their set of 13 unary category-rewriting rules, as well as rules for combining with punctuation to the left and right. We further allow for a *merge* rule "X X $\rightarrow$ X" since this is seen frequently in the corpora (Clark and Curran 2007).

Because of their structured nature, CCG categories (unlike the part-of-speech tags and non-terminals of a standard PCFG) contain intrinsic information that gives evidence of their frequencies. First, simple categories are *a priori* more likely than complex categories. For example, the transitive verb "buy" appears with supertag $(\mathsf{s_b}\backslash\mathsf{np})/\mathsf{np}$ 342 times in CCGbank, but just once with $(((\mathsf{s_b}\backslash\mathsf{np})/\mathsf{pp})/\mathsf{pp})/\mathsf{np}$. Second, modifier categories, those of the form X/X or $X\backslash X$, are more likely than non-modifiers of similar complexity. For example, a category containing six atoms may, in general, be very unlikely, but a six-atom category that is merely a modifier of a three-atom category (like an adverb modifying a transitive verb) would be fairly common.

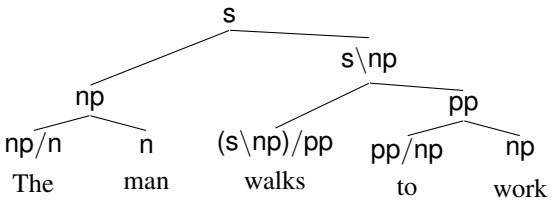

Figure 1: CCG parse for "The man walks to work."

Baldridge (2008) used this information to bias supertagger learning via an ad hoc initialization of Expectation-Maximization for an HMM. Garrette et al. (2014) built upon these concepts to introduce a probabilistic grammar over CCG categories that provides a well-founded notion of category complexity. The ideas are described in detail in previous work, but we restate some of them here briefly. The category grammar captures important aspects of what makes a category complex: smaller categories are more likely (defined by $p_{term} > \frac{1}{2}$, the probability of generating a terminal, atomic, category), some atomic categories are more likely than others (p_{atom}), modifier categories are more likely than non-modifiers (p_{mod}), and slash operators may occur with different likelihoods (p_{fwd}). This category grammar defines the probability distribution P_{G} via the following recursive definition (let $\overline{p}$ denote $1 - p$):

$C \rightarrow a$	$p_{term} \cdot p_{atom}(a)$
$C \rightarrow A/A$	$\overline{p}_{term} \cdot p_{fwd} \cdot p_{mod} \cdot P_{\text{G}}(A)$
$C \rightarrow A/B,\ A \neq B$	$\overline{p}_{term} \cdot p_{fwd} \cdot \overline{p}_{mod} \cdot P_{\text{G}}(A) \cdot P_{\text{G}}(B)$
$C \rightarrow A\backslash A$	$\overline{p}_{term} \cdot \overline{p}_{fwd} \cdot p_{mod} \cdot P_{\text{G}}(A)$
$C \rightarrow A\backslash B,\ A \neq B$	$\overline{p}_{term} \cdot \overline{p}_{fwd} \cdot \overline{p}_{mod} \cdot P_{\text{G}}(A) \cdot P_{\text{G}}(B)$

where A, B, C are recursively defined categories and a is an atomic category: $a \in \{\mathsf{s}, \mathsf{s_{dcl}}, \mathsf{s_{adj}}, \mathsf{s_b}, \mathsf{n}, \mathsf{np}, \mathsf{pp}, \dots\}$.

Supertagging accuracy was improved further by complementing the language-universal knowledge from CCG with corpus-specific information extracted automatically. Counts were estimated using a tag dictionary and unannotated data, then used to empirically set the parameters of the prior.

Generative Model

Our CCG parsing model assumes the following generative process. First, the parameters that define our PCFG are drawn. We generate a distribution σ over root categories, a conditional distribution $\theta_{\mathbf{t}}$ over binary branching non-terminal productions given each category $\mathbf{t}$, a conditional distribution $\pi_{\mathbf{t}}$ over unary non-terminal productions given each category $\mathbf{t}$, and a conditional distribution $\mu_{\mathbf{t}}$ over terminal (word) productions given each category $\mathbf{t}$. Each of these parameters is drawn from a Dirichlet distribution parameterized by a concentration parameter ($\alpha_\sigma, \alpha_\theta, \alpha_\pi, \alpha_\mu$) and a prior mean distribution ($\sigma^0, \theta^0, \pi^0, \mu^0_{\mathbf{t}}$). By setting each α close to zero, we can bias learning toward relatively peaked distributions. The prior means, explained in detail below, are used to encode both universal linguistic knowledge as well as information automatically extracted from the weak supervision.

Note that unlike a standard phrase-structure grammar where the sets of terminal and non-terminal labels are non-overlapping (part-of-speech tags vs. internal nodes), a CCG category may appear at any level the tree and, thus, may yield binary, unary, or terminal word productions. Therefore, we also generate a distribution $\lambda_{\mathbf{t}}$ for every category $\mathbf{t}$ that defines the mixture over production types (binary, unary, terminal) yielded by $\mathbf{t}$. For simplicity, these parameters are generated by draws from an unbiased Dirichlet.

Next, the process generates each sentence in the corpus. This begins by generating a root category $\mathbf{s}$ and then recursively generating subtrees. For each subtree rooted by a category $\mathbf{t}$, with probability determined by $\lambda_{\mathbf{t}}$, we generate either a binary ($\langle \mathbf{u}, \mathbf{v} \rangle$), unary ($\langle \mathbf{u} \rangle$), or terminal ($w$) production from $\mathbf{t}$; for binary and unary productions, we generate child categories and recursively generate subtrees. A tree is complete when all branches end in terminal words.

Borrowing from the recursive generative function notation of Johnson, Griffiths, and Goldwater (2007), our process can be summarized as:

Parameters:

$\sigma \sim \text{Dirichlet}(\alpha_\sigma, \sigma^0)$		root categories
$\theta_{\mathbf{t}} \sim \text{Dirichlet}(\alpha_\theta, \theta^0)$	$\forall \mathbf{t} \in \mathcal{T}$	binary productions
$\pi_{\mathbf{t}} \sim \text{Dirichlet}(\alpha_\pi, \pi^0)$	$\forall \mathbf{t} \in \mathcal{T}$	unary productions
$\mu_{\mathbf{t}} \sim \text{Dirichlet}(\alpha_\mu, \mu^0_{\mathbf{t}})$	$\forall \mathbf{t} \in \mathcal{T}$	terminal productions
$\lambda_{\mathbf{t}} \sim \text{Dir}(\langle 1, 1, 1 \rangle)$	$\forall \mathbf{t} \in \mathcal{T}$	production mixture

Sentence:

$\mathbf{s} \sim \text{Categorical}(\sigma)$

generate$(\mathbf{s})$

where

function *generate*$(\mathbf{t})$:

$z \sim \text{Categorical}(\lambda_{\mathbf{t}})$

if $z = 1 : \langle \mathbf{u}, \mathbf{v} \rangle \mid \mathbf{t} \sim \text{Categorical}(\theta_{\mathbf{t}})$

Tree$(\mathbf{t},$ *generate*$(\mathbf{u}),$ *generate*$(\mathbf{v}))$

if $z = 2 : \langle \mathbf{u} \rangle \mid \mathbf{t} \sim \text{Categorical}(\pi_{\mathbf{t}})$

Tree$(\mathbf{t},$ *generate*$(\mathbf{u}))$

if $z = 3 : w \mid \mathbf{t} \sim \text{Categorical}(\mu_{\mathbf{t}})$

Leaf$(\mathbf{t}, w)$

Root prior mean (σ^0)

Since σ is a distribution over root categories, we can use P_{G}, the probability of a category as defined above in terms of the category grammar, as its prior mean, biasing our model toward simpler root categories. Thus, $\sigma^0(\mathbf{t}) = P_{\text{G}}(\mathbf{t})$.

Non-terminal production prior means (θ^0 and π^0)

Our model includes two types of non-terminal productions: binary productions of the form "A $\rightarrow$ B C", and unary productions of the form "A $\rightarrow$ B". As with the root distribution prior, we would like our model to prefer productions that yield high-likelihood categories. To provide this bias, we again use P_{G}:

$$\theta^0(\langle \mathbf{u}, \mathbf{v} \rangle) = P_{\text{G}}(\mathbf{u}) \cdot P_{\text{G}}(\mathbf{v})$$
$$\pi^0(\langle \mathbf{u} \rangle) = P_{\text{G}}(\mathbf{u})$$

Terminal production prior means ($\mu^0_{\mathbf{t}}$)

Because we model terminal productions separately, we are able to borrow directly from Garrette et al. (2014) to define the terminal production prior mean $\mu^0_{\mathbf{t}}$ in a way that exploits the dictionary and unlabeled corpus to estimate the distribution over words for each supertag. Terminal productions in our grammar are defined as "word given supertag," which is exactly the relationship of the emission distribution in an HMM supertagger. Thus, we simply use the supertagger's emission prior mean, as defined in the previous work, for our terminal productions:

$$\mu^0_{\mathbf{t}}(w) = P_{em}(w \mid \mathbf{t})$$

If $C(w)$ is the number of times of word w appears in the raw corpus, $\text{TD}(w)$ is the set of supertags associated with w in the tag dictionary, and $\text{TD}(\mathbf{t})$ is the set of *known* words (words appearing in the tag dictionary) for which supertag $\mathbf{t} \in \text{TD}(w)$, the count of a word/tag pair for a known word is estimated by uniformly distributing the word's (δ-smoothed) raw counts over its tag dictionary entries:

$$C_{known}(\mathbf{t}, w) = \begin{cases} \frac{C(w)+\delta}{|\text{TD}(w)|} & \text{if } \mathbf{t} \in \text{TD}(w) \\ 0 & \text{otherwise} \end{cases}$$

To address *unknown* words, we employ the concept of tag "openness", estimating the probability of a tag $\mathbf{t}$ applying to some unknown word: if a tag is known to apply to many word types, it is likely to also apply to some new word type.

$$P(\text{unk} \mid \mathbf{t}) \propto |\text{known words } w \text{ s.t. } \mathbf{t} \in \text{TD}(w)|$$

We can calculate $P(\mathbf{t} \mid \text{unk})$ using Bayes' rule, which allows us to estimate word/tag counts for unknown words:

$$P(\mathbf{t} \mid \text{unk}) \propto P(\text{unk} \mid \mathbf{t}) \cdot P_{\text{G}}(\mathbf{t})$$

$$C_{\text{unk}}(\mathbf{t}, w) = C(w) \cdot P(\mathbf{t} \mid \text{unk})$$

Finally, we can calculate a probability estimate considering the relationship between $\mathbf{t}$ and all known and unknown words:

$$P_{em}(w \mid \mathbf{t}) = \frac{C_{known}(\mathbf{t}, w) + C_{unk}(\mathbf{t}, w)}{\sum_{w'} C_{known}(\mathbf{t}, w') + C_{unk}(\mathbf{t}, w')}$$

Decoding

In order to parse with our model, we seek the highest-probability parse tree for a given sentence $\mathbf{w}$:

$$\hat{\mathbf{y}} = \text{argmax}_{\mathbf{y}} P(\mathbf{y} \mid \mathbf{w}).$$

This can be computed efficiently using the well-known probabilistic CKY algorithm.

Posterior Inference

Since inference about the parameters of our model using a corpus of unlabeled training data is intractable, we resort to Gibbs sampling to find an approximate solution. Our strategy is based on that of Johnson, Griffiths, and Goldwater (2007), using a block sampling approach. We initialize our parameters by setting each distribution to its prior mean ($\sigma = \sigma^0$, $\theta_{\mathbf{t}} = \theta^0$, etc.) and $\lambda_{\mathbf{t}} = \langle \frac{1}{3}, \frac{1}{3}, \frac{1}{3} \rangle$. We then alternate between sampling trees given the current model parameters and observed word sequences, and sampling model parameters $(\sigma, \theta, \pi, \mu, \lambda)$ given the current set of parse trees. To efficiently sample new model parameters, we exploit Dirichlet-multinomial conjugacy. We accumulate all parse trees sampled across all sampling iterations and use them to approximate the posterior quantities.

Our inference procedure takes as input each of the distribution prior means $(\sigma^0, \theta^0, \pi^0, \mu^0)$, along with the raw corpus and tag dictionary. During sampling, we always restrict the possible supertag choices for a word w to the categories found in the tag dictionary entry for that w: $\text{TD}(w)$. Since real-world learning scenarios will always lack complete knowledge of the lexicon, we, too, want to allow for unknown words. Thus, we use *incomplete* tag dictionaries in our experiments, meaning that for a word w not present in the dictionary, we assign $\text{TD}(w)$ to be the full set of known categories, indicating maximal ambiguity. It is also possible that the "correct" supertag for a given word is not present in the tag dictionary, though in these scenarios we hope that the parse will succeed through a different route.

Our Gibbs sampler, based on the one proposed by Goodman (1998) and used by Johnson, Griffiths, and Goldwater (2007), uses a block sampling approach to sample an entire parse tree at once. The procedure is similar in principle to the Forward-Filter Backward-Sampler algorithm used by Garrette et al. (2014) for the HMM supertagger, but sampling trees instead of sequences (Carter and Kohn 1996). To sample a tree for a sentence $\mathbf{w}$, the strategy is to use the Inside algorithm (Lari and Young 1990) to inductively compute, for each potential non-terminal position (i, j) (spanning words w_i through w_{j-1}) and category $\mathbf{t}$, going "up" the tree, the probability of generating $w_i, \ldots, w_{j-1}$ via any arrangement of productions that is rooted by $y_{ij} = \mathbf{t}$:

$$
\begin{aligned}
p(y_{i,i+1} = \mathbf{t} \mid w_i) &= \lambda_{\mathbf{t}}(3) \cdot \mu_{\mathbf{t}}(w_i) \\
&\quad + \sum_{\mathbf{t} \to \mathbf{u}} \lambda_{\mathbf{t}}(2) \cdot \pi_{\mathbf{t}}(\langle \mathbf{u} \rangle) \cdot p(y_{ij} = \mathbf{u} \mid w_{i:j-1}) \\
p(y_{ij} = \mathbf{t} \mid w_{i:j-1}) &= \\
&\quad \sum_{\mathbf{t} \to \mathbf{u}} \lambda_{\mathbf{t}}(2) \cdot \pi_{\mathbf{t}}(\langle \mathbf{u} \rangle) \cdot p(y_{ij} = \mathbf{u} \mid w_{i:j-1}) \\
&\quad + \sum_{\mathbf{t} \to \mathbf{u}\ \mathbf{v}} \sum_{i<k<j} \lambda_{\mathbf{t}}(1) \cdot \theta_{\mathbf{t}}(\langle \mathbf{u}, \mathbf{v} \rangle) \\
&\qquad\qquad \cdot p(y_{ik} = \mathbf{u} \mid w_{i:k-1}) \\
&\qquad\qquad \cdot p(y_{kj} = \mathbf{v} \mid w_{k:j-1})
\end{aligned}
$$

We then pass through the chart again, this time "downward" starting at the root and sampling productions until we reach a terminal word on all branches:

$$
\begin{aligned}
y_{0n} &\sim \sigma_{\mathbf{t}} \cdot p(y_{0n} = \mathbf{t} \mid w_{0:n-1}) \\
x \mid y_{ij} &\sim \big\langle \theta_{y_{ij}}(\langle \mathbf{u}, \mathbf{v} \rangle) \cdot p(y_{ik} = \mathbf{u} \mid w_{i:k-1}) \\
&\qquad\qquad \cdot p(y_{kj} = \mathbf{v} \mid w_{k:j-1}) \quad \forall\, y_{ik}, y_{kj}, \\
&\qquad \pi_{y_{ij}}(\langle \mathbf{u} \rangle) \cdot p(y'_{ij} = \mathbf{u} \mid w_{i:j-1}) \quad \forall\, y'_{ij}, \\
&\qquad \mu_{y_{ij}}(w_i) \ \big\rangle
\end{aligned}
$$

where x is either a split point k and pair of categories y_{ik}, y_{kj} resulting from a binary rewrite rule, a single category y'_{ij} resulting from a unary rule, or a word w resulting from a terminal rule.

Resampling the parameters uses the just-sampled parse trees $\mathbf{y}$ to compute $C_{root}(\mathbf{t})$, the count of trees in which the root category is $\mathbf{t}$, $C(\mathbf{t} \to \langle \mathbf{u}, \mathbf{v} \rangle)$, the count of binary non-terminal productions whose category is $\mathbf{t}$ that are producing the pair of categories $\langle \mathbf{u}, \mathbf{v} \rangle$, the count of unary non-terminal productions $C(\mathbf{t} \to \langle \mathbf{u} \rangle)$, and the count of terminal productions $C(\mathbf{t} \to w)$. We then sample, for each $\mathbf{t} \in \mathcal{T}$ where $\mathcal{T}$ is the full set of valid CCG categories (and V is the full vocabulary of known words):

$$
\begin{aligned}
\sigma &\sim \text{Dir}\left(\langle \alpha_\sigma \cdot \sigma^0(\mathbf{t}) + C_{root}(\mathbf{t}) \rangle_{\mathbf{t} \in \mathcal{T}}\right) \\
\theta_{\mathbf{t}} &\sim \text{Dir}\left(\langle \alpha_\theta \cdot \theta^0(\langle \mathbf{u}, \mathbf{v} \rangle) + C(\mathbf{t} \to \langle \mathbf{u}, \mathbf{v} \rangle) \rangle_{\mathbf{u}, \mathbf{v} \in \mathcal{T}}\right) \\
\pi_{\mathbf{t}} &\sim \text{Dir}\left(\langle \alpha_\pi \cdot \pi^0(\langle \mathbf{u} \rangle) + C(\mathbf{t} \to \langle \mathbf{u} \rangle) \rangle_{\mathbf{u} \in \mathcal{T}}\right) \\
\mu_{\mathbf{t}} &\sim \text{Dir}\left(\langle \alpha_\mu \cdot \mu^0_{\mathbf{t}}(w) + C(\mathbf{t} \to w) \rangle_{w \in V}\right) \\
\lambda_{\mathbf{t}} &\sim \text{Dir}\,(\langle\ 1 + \textstyle\sum C(\mathbf{t} \to \langle \mathbf{u}, \mathbf{v} \rangle), \\
&\qquad\qquad 1 + \textstyle\sum C(\mathbf{t} \to \langle \mathbf{u} \rangle), \\
&\qquad\qquad 1 + \textstyle\sum C(\mathbf{t} \to w) \qquad \rangle)
\end{aligned}
$$

These distributions are derived from the conjugacy of the Dirichlet prior to the multinomial; note that the result selects parameters based on both the data (counts) and the bias encoded in the prior.

After 50 sampling iterations have completed, the parameters are estimated as the maximum likelihood estimate of the pool of trees resulting from all sampling iterations. Dramatic time savings can be obtained by generating and reusing a chart that compactly stores all possible parses for all possible sentences. This allows avoiding calculation for subtrees and productions that never participate in a complete parse.

As a further optimization, we enforce the use of punctuation as phrasal-boundary indicators, a technique used previously by Ponvert, Baldridge, and Erk (2011) and Spitkovsky, Alshawi, and Jurafsky (2011). This means that when we attempt to parse a sentence (or sample a parse tree for a sentence), we do not allow constituents that cross punctuation without covering the whole inter-punctuation phrase. For example, the sentence "On Sunday, he walked.", the constituent "Sunday , he" would be disallowed. Since punctuation does regularly mark a phrasal boundary, this choice has negligible effect on accuracy while reducing runtime and memory use. In cases where a punctuation-as-boundary requirement (along with the tag dictionary) renders a sentence unparseable according to the CCG rules, we lift the punctuation requirement for that sentence.

	English	Chinese	Italian
1. uniform	53.38	35.94	58.16
2. P_{G}	54.75	40.08	59.21
3. P_{G}, P_{em}	**55.69**	**42.00**	**60.04**

(a) Main results.

English			Chinese			Italian		
0.001	0.01	0.1	0.001	0.01	0.1	0.001	0.01	0.1
56.62	61.30	56.46	34.88	41.24	47.42	**60.29**	59.91	**54.72**
57.79	61.93	56.69	41.59	42.86	47.74	58.58	**60.76**	53.31
60.12	**62.11**	**57.20**	**43.42**	**43.84**	**49.40**	59.60	60.02	53.39

(b) Increasing degrees of artificial tag dictionary pruning (see text). The pruning cutoff is given at the top of each column.

Table 1: Experimental results: test-set dependency accuracies. (1) uses uniform priors on all distributions. (2) uses the category prior P_{G}. (3) uses the tag dictionary and raw corpus to automatically estimate category prior and word production information. Table (a) gives the results obtained when no artificial tag dictionary pruning was performed; table (b) shows performance as the artificial pruning is increased.

Experiments

We evaluated our approach on the three available CCG corpora: English CCGBank (Hockenmaier and Steedman 2007), Chinese Treebank CCG (Tse and Curran 2010), and the Italian CCG-TUT corpus (Bos, Bosco, and Mazzei 2009). Each corpus was split into four non-overlapping datasets: a portion for constructing the tag dictionary, sentences for the unlabeled training data, development trees (used for tuning α, p_{term}, p_{mod}, and p_{fwd} hyperparameters), and test trees. We used the same splits as Garrette et al. (2014). Since these treebanks use special representations for conjunctions, we chose to rewrite the trees to use conjunction categories of the form $(X\backslash X)/X$ so that additional special rules would not need to be introduced.

We ran our sampler for 50 iterations.[1] For the category grammar, we used p_{term}=0.7, p_{mod}=0.1, p_{fwd}=0.5. For the priors, we use α_σ=1, α_θ=100, α_π=10,000, α_μ=10,000.[2] We trained on 1,000 sentences for English and 750 for Chinese, but only 150 for Italian since it is a much smaller corpus.

To limit the amount of spurious ambiguity, we limit combinatory rules to forward and backward application, punctuation, and merge:

$X/Y \;\; Y \rightarrow X$	forward application
$Y \;\; X\backslash Y \rightarrow X$	backward application
$X \;\; . \rightarrow X$	right punctuation
$. \;\; X \rightarrow X$	left punctuation
$X \;\; X \rightarrow X$	merge

We also allow the 13 unary rules proposed by Lewis and Steedman (2014). CCG composition rules are rarely necessary to parse a sentence, but do increase the overall number of parses, many of which represent the same underlying grammatical structures. This choice drastically reduces the time and space requirements for learning, without sacrifices in accuracy. Allowing the *backward crossed composition* rule—the third most-frequent rule in CCGBank—not only dramatically increases the time and memory requirements, but also tends to lower the accuracy of the resulting parser by 1% or more, likely because it increases ambiguity.

CCG parsers are typically evaluated on the *dependencies* they produce instead of their CCG derivations directly. There can be many different CCG parse trees that all represent the same dependency relationships (spurious ambiguity), and CCG-to-dependency conversion can collapse those differences. To convert a CCG tree into a dependency tree, we traverse the parse tree, dictating at every branching node which words will be the dependents of which. For binary branching nodes of *forward* rules, the right side—the argument side—is the dependent, unless the left side is a modifier (X/X) of the right, in which case the left is the dependent. The opposite is true for *backward* rules. For *punctuation* rules, the punctuation is always the dependent. For *merge* rules, the right side is always made the parent. The results presented in this paper are dependency accuracy scores: the proportion of words that were assigned the correct parent (or "root" for the root of a tree).

During training, we only use sentences for which we are able to find at least one parse since we cannot sample a parse tree for a sentence that has no available parses. However, for testing, we must make extra efforts to find valid parses. To that end, if we encounter a test sentence that cannot be parsed given the tag dictionary and CCG rules (either with or without enforcement of a punctuation-as-boundary requirement), then we fall back to a plan in which additional supertag options are added for each token. For a word w_i, we add the set of tags $X\backslash X$ for all $X \in \text{TD}(w_{i-1})$, and Y/Y for all $Y \in \text{TD}(w_{i+1})$. Since $X\backslash X$ and Y/Y represent *modifier* categories, the result of these categories is to provide the parser the option of simply making w_i a modifier of one of its immediate neighbors, resulting in w_i simply being assigned as a dependent of that neighboring word. This effectively allows the parser to ignore inconvenient tokens as it searches for the optimal tree. This is similar to the "deletion" strategy employed by Zettlemoyer and Collins (2007) in which words can be skipped.

Baseline

As a baseline, we trained our model with uniform prior mean distributions $(\sigma^0, \theta^0, \pi^0, \mu^0_{\mathbf{t}})$. The uniform priors do not make any distinction among the relative likelihoods of different CCG categories or words, and thus do not take ad-

[1] We experimented with higher numbers of iterations but found that accuracy was not improved past 50 iterations.

[2] In order to ensure that these concentration parameters, while high, were not dominating the posterior distributions, we ran experiments in which they were set much higher (including using the prior alone), and found that accuracies plummeted in those cases, demonstrating that there is a good balance with the prior.

vantage of either the universal properties of the CCG formalism (P_{G}), or the initialization information that can be automatically estimated from the type-supervised data (P_{em}).

Results

The results of our experiments are given in Table 1a. We find that the use of a well-designed category prior (P_{G}) achieves performance gains over the baseline across all three languages. Our results also show that still further gains can be achieved by using the available weak supervision—the tag dictionary and unlabeled text—to estimate corpus counts that can be used to influence the priors on terminal productions (P_{em}).

The largest gains are in the Chinese data, though the accuracies are lower on Chinese overall, indicating the difficulty of the Chinese parsing task.

Error analysis

Supertag accuracy degrades roughly 2% for each language from the uniform prior to the full prior. Inspection of the errors shows us that this is due in part to the category prior encouraging simpler categories, e.g., categories like $((\mathsf{s}\backslash\mathsf{np})/(\mathsf{s}\backslash\mathsf{np}))/\mathsf{np}$ being learned as pp/np. It is counter-intuitive that supertagging accuracy decreases while parsing performance improves, but note that it may be easier for the parser to recover correct dependencies, using the *merge* rule, when an incorrect supertag is simpler.

The most frequent errors under uniform priors involve very complex categories, like $((\mathsf{s}_{\mathsf{dcl}}\backslash\mathsf{np})/(\mathsf{s}_{\mathsf{dcl}}\backslash\mathsf{np}))/\mathsf{np}$ in Chinese. When the category prior is introduced, these complex categories vanish from the errors; the most complex common category error with the category prior in Chinese is $(\mathsf{s}_{\mathsf{dcl}}\backslash\mathsf{np})/\mathsf{np}$. After bringing the data-based prior in, we again see more complex categories, plus others that have high arity, like modifiers of modifiers $(\mathsf{np}/\mathsf{np})/(\mathsf{np}/\mathsf{np})$. This suggests that good performance relies on priors that blend theoretical constraints with empirical guidance.

We also trained the parser on gold-standard trees for an upper-bound on performance for the given training sentences, obtaining 66%, 48%, and 65% for English, Chinese, and Italian, respectively.

Supertag dictionary pruning

Tag dictionaries used in experimental setups are typically extracted from labeled corpora by finding all word/tag pairs in some set of annotated sentences. As dictionaries, however, the distinctions between high- and low-frequency tags are lost, and all tags in the dictionary entry appear equally valid for a given word. Unfortunately, the inclusion of low-probability tags in this way tends to cause problems during training by over-representing the likelihoods of tags that are only rarely applicable, or even tags that are the result of annotation errors and should not have been included in the dictionary at all.

Traditionally, researchers have avoided this problem by using tag frequency information to automatically *prune* the tag dictionary of its low-frequency tags (Merialdo 1994; Kupiec 1992), leading Banko and Moore (2004), among others, to argue that early successes in type-supervised learning were due, in large part, to the use of that frequency information that is not available from unlabeled data alone, undercutting the promises of weakly-supervised learning.

Since it is the goal of this research is to develop techniques that can be applied without artificial data cleaning, we desire models that are robust to noise in the training data. To see how this noise affects our model, we executed a series of experiments in which varing degrees of noise were artificially removed. For different cutoff levels (0.001, 0.01, 0.1), we computed the tag dictionary entry for word w, as the supertags $\mathbf{t}$ where:

$$\text{TD}(w) = \left\{ \mathbf{t} \;\middle|\; \frac{\mathit{freq}(w,\mathbf{t})}{\sum_{\mathbf{t}' \in \mathcal{T}} \mathit{freq}(w,\mathbf{t}')} \geq \mathit{cutoff} \right\}.$$

Results under pruned conditions are given in Table 1b. Table 1a can be interpreted as results when $\mathit{cutoff} = 0$.

From the results, we can see that in most scenarios, grammar-informed priors still provide benefits to the model. More notable, however, is that these priors provide *more* value in cases where there is less artificial pruning. This tells us that our constructed priors are most helpful in the noisier, more difficult, and *more realistic* learning scenarios.

When only uniform priors are used, the model is not able to differentiate *a priori* between probable and improbable categories. This results in poor performance when artificial assistance is not given. However, our category prior, with its knowledge of the intrinsic properties of the CCG formalism, is able to overcome this problem, allowing the model to differentiate between likely and unlikely categories and biasing the model toward better categories even though category frequency information is not available. Importantly, it also does this without eliminating tags that (though infrequent) are useful for parsing.

These results support our hypothesis that when supervised data is scarce, it becomes more important to take advantage of linguistic knowledge.

Conclusion and Future Work

We have presented a Bayesian approach to CCG parser learning that can be trained given only a lexicon and raw text. It flexibly incorporates linguistically-informed prior distributions and naturally accommodates the deviations from pure CCG grammars that have been employed in annotations for existing CCG corpora, especially CCGBank. The model enhances a standard PCFG by factoring in prior distributions over categories into both non-terminal and terminal productions; those priors can be derived from a universal prior distribution, from a distribution built by combining a tag dictionary with raw text, or both. Our results show that using both sources for defining these priors leads to better performing CCG parsers in low-resource scenarios.

The idea of incorporating linguistic knowledge into a model via priors is very appealing when supervised data is scarce. We have shown how knowledge about the structure of CCG categories can be used, but a more sophisticated model may be able to additionally make use of knowledge about the relative likelihoods of various CCG rules. For example, we know that *application* rules are always preferred,

and that *composition* rules should only be used when necessary, and in specific scenarios (Baldridge 2002). Further, rules like *merge* should only be used as a last resort.

Finally, while a relatively large tag dictionary was used for the experiments presented here, it may be possible to learn from even less supervision by generalizing a small dictionary into a large one, as has been done successfully for type-supervised part-of-speech tagging (Das and Petrov 2011; Garrette and Baldridge 2013; Garrette, Mielens, and Baldridge 2013).

Acknowledgements

This work was supported by the U.S. Department of Defense through the U.S. Army Research Office (grant number W911NF-10-1-0533). Experiments were run on the UTCS Mastodon Cluster, provided by NSF grant EIA-0303609.

References

Baldridge, J.; Chatterjee, S.; Palmer, A.; and Wing, B. 2007. DotCCG and VisCCG: Wiki and programming paradigms for improved grammar engineering with OpenCCG. In *Proceedings of the Workshop on Grammar Engineering Across Frameworks*.

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

Baldridge, J. 2008. Weakly supervised supertagging with grammar-informed initialization. In *Proceedings of COLING*.

Banko, M., and Moore, R. C. 2004. Part-of-speech tagging in context. In *Proceedings of COLING*.

Bisk, Y., and Hockenmaier, J. 2013. An HDP model for inducing combinatory categorial grammars. *Transactions of the Association for Computational Linguistics* 1.

Bos, J.; Bosco, C.; and Mazzei, A. 2009. Converting a dependency treebank to a categorial grammar treebank for Italian. In Passarotti, M.; Przepiórkowski, A.; Raynaud, S.; and Van Eynde, F., eds., *Proceedings of the Eighth International Workshop on Treebanks and Linguistic Theories (TLT8)*.

Carter, C. K., and Kohn, R. 1996. On Gibbs sampling for state space models. *Biometrika* 81(3):341–553.

Clark, S., and Curran, J. R. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics* 33.

Das, D., and Petrov, S. 2011. Unsupervised part-of-speech tagging with bilingual graph-based projections. In *Proceedings of ACL-HLT*.

Garrette, D., and Baldridge, J. 2013. Learning a part-of-speech tagger from two hours of annotation. In *Proceedings of NAACL*.

Garrette, D.; Dyer, C.; Baldridge, J.; and Smith, N. A. 2014. Weakly-supervised Bayesian learning of a CCG supertagger. In *Proceedings of CoNLL*.

Garrette, D.; Mielens, J.; and Baldridge, J. 2013. Real-world semi-supervised learning of POS-taggers for low-resource languages. In *Proceedings of ACL*.

Goodman, J. 1998. *Parsing inside-out*. Ph.D. Dissertation, Harvard University. Available from http://research.microsoft.com/∼joshuago/.

Hockenmaier, J., and Steedman, M. 2007. CCGbank: A corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics* 33(3).

Johnson, M.; Griffiths, T.; and Goldwater, S. 2007. Bayesian inference for PCFGs via Markov chain Monte Carlo. In *Proceedings of NAACL*.

Kupiec, J. 1992. Robust part-of-speech tagging using a hidden Markov model. *Computer Speech & Language* 6(3).

Lari, K., and Young, S. J. 1990. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language* 4:35–56.

Lewis, M., and Steedman, M. 2014. A* CCG parsing with a supertag-factored model. In *Proceedings of EMNLP*.

Merialdo, B. 1994. Tagging English text with a probabilistic model. *Computational Linguistics* 20(2).

Ponvert, E.; Baldridge, J.; and Erk, K. 2011. Simple unsupervised grammar induction from raw text with cascaded finite state models. In *Proceedings of ACL-HLT*.

Spitkovsky, V. I.; Alshawi, H.; and Jurafsky, D. 2011. Punctuation: Making a point in unsupervised dependency parsing. In *Proceedings of CoNLL*.

Steedman, M., and Baldridge, J. 2011. Combinatory categorial grammar. In Borsley, R., and Borjars, K., eds., *Non-Transformational Syntax: Formal and Explicit Models of Grammar*. Wiley-Blackwell.

Steedman, M. 2000. *The Syntactic Process*. MIT Press.

Tse, D., and Curran, J. R. 2010. Chinese CCGbank: Extracting CCG derivations from the Penn Chinese Treebank. In *Proceedings of COLING*.

Weese, J.; Callison-Burch, C.; and Lopez, A. 2012. Using categorial grammar to label translation rules. In *Proceedings of WMT*.

Zettlemoyer, L. S., and Collins, M. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of UAI*.

Zettlemoyer, L. S., and Collins, M. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *Proceedings of EMNLP*.