Weakly-Supervised Bayesian Learning of a CCG Supertagger

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

[*]Department of Computer Science, The University of Texas at Austin
[†]School of Computer Science, Carnegie Mellon University
[‡]Department of Linguistics, The University of Texas at Austin
[*]Corresponding author: dhg@cs.utexas.edu

Abstract

We present a Bayesian formulation for weakly-supervised learning of a Combinatory Categorial Grammar (CCG) supertagger with an HMM. We assume supervision in the form of a tag dictionary, and our prior encourages the use of cross-linguistically common category structures as well as transitions between tags that can combine locally according to CCG's combinators. Our prior is theoretically appealing since it is motivated by language-independent, universal properties of the CCG formalism. Empirically, we show that it yields substantial improvements over previous work that used similar biases to initialize an EM-based learner. Additional gains are obtained by further shaping the prior with corpus-specific information that is extracted automatically from raw text and a tag dictionary.

1 Introduction

Unsupervised part-of-speech (POS) induction is a classic problem in NLP. Many proposed solutions are based on Hidden Markov models (HMMs), with various improvements obtainable through: inductive bias in the form of tag dictionaries (Kupiec, 1992; Merialdo, 1994), sparsity constraints (Lee et al., 2010), careful initialization of parameters (Goldberg et al., 2008), feature based representations (Berg-Kirkpatrick et al., 2010; Smith and Eisner, 2005), and priors on model parameters (Johnson, 2007; Goldwater and Griffiths, 2007; Blunsom and Cohn, 2011, *inter alia*).

When tag dictionaries are available, a situation we will call **type-supervision**, POS induction from unlabeled corpora can be relatively successful; however, as the number of possible tags increases, performance drops (Ravi and Knight, 2009). In such cases, there are a large number of possible labels for each token, so picking the right one simply by chance is unlikely; the parameter space tends to be large; and devising good initial parameters is difficult. Therefore, it is unsurprising that the unsupervised (or even weakly-supervised) learning of a Combinatory Categorial Grammar (CCG) supertagger, which labels each word with one of a large (possibly unbounded) number of structured categories called **supertags**, is a considerable challenge.

Despite the apparent complexity of the task, supertag sequences have regularities due to universal properties of the CCG formalism (§2) that can be used to reduce the complexity of the problem; previous work showed promising results by using these regularities to initialize an HMM that is then refined with EM (Baldridge, 2008). Here, we exploit CCG's category structure to motivate a novel prior over HMM parameters for use in Bayesian learning (§3). This prior encourages (i) cross-linguistically common tag types, (ii) tag bigrams that can combine using CCG's combinators, and (iii) sparse transition distributions. We also go beyond the use of these universals to show how additional, *corpus-specific* information can be automatically extracted from a combination of the tag dictionary and raw data, and how that information can be combined with the universal knowledge for integration into the model to improve the prior.

We use a blocked sampling algorithm to sample supertag sequences for the sentences in the training data, proportional to their posterior probability (§4). We experimentally verify that our Bayesian formulation is effective and substantially outperforms the state-of-the-art baseline initialization/EM strategy in several languages (§5). We also evaluate using tag dictionaries that are unpruned and have only partial word coverage, finding even greater improvements in these more realistic scenarios.

2 CCG and Supertagging

CCG (Steedman, 2000; Steedman and Baldridge, 2011) is a grammar formalism in which each lexical token is associated with a structured category, often referred to as a *supertag*. CCG categories are defined by the following recursive definition:

$$C \rightarrow \{\text{S, N, NP, PP, ...}\}$$
$$C \rightarrow \{C/C, C\backslash C\}$$

A CCG category can either be an *atomic* category indicating a particular type of basic grammatical phrase (S for a sentence, N for a noun, NP for a noun phrase, etc), or a *complex* category formed from the combination of two categories by one of two *slash* operators. In CCG, complex categories indicate a grammatical relationship between the two operands. For example, the category (S\NP)/NP might describe a transitive verb, looking first to its right (indicated by /) for an object, then to its left (\) for a subject, to produce a sentence. Further, atomic categories may be augmented with *features*, such as S_{dcl}, to restrict the set of atoms with which they may unify. The task of assigning a category to each word in a text is called *supertagging* (Bangalore and Joshi, 1999).

Because they are recursively defined, there is an infinite number of potential CCG categories (though in practice it is limited by the number of actual grammatical contexts). As a result, the number of supertags appearing in a corpus far exceeds the number of POS tags (see Table 1). Since supertags specify the grammatical context of a token, and high frequency words appear in many contexts, CCG grammars tend to have very high lexical ambiguity, with frequent word types associating with a large number of categories. This ambiguity has made type-supervised supertagger learning very difficult because the typical approaches to initializing parameters for EM become much less effective.

Grammar-informed supertagger learning. Baldridge (2008) was successful in extending the standard type-supervised tagger learning to the task of CCG supertagging by setting the initial parameters for EM training of an HMM using two intrinsic properties of the CCG formalism: the tendency for adjacent tags to combine, and the tendency to use less complex tags. These properties are explained in detail in the original work, but we restate the ideas briefly throughout this paper for completeness.

X/Y	Y	$\Rightarrow$	X	$(>)$
Y	$X\backslash Y$	$\Rightarrow$	X	$(<)$
X/Y	Y/Z	$\Rightarrow$	X/Z	$(>\mathbf{B})$
$Y\backslash Z$	$X\backslash Y$	$\Rightarrow$	$X\backslash Z$	$(<\mathbf{B})$
Y/Z	$X\backslash Y$	$\Rightarrow$	X/Z	$(<\mathbf{B}_{\times})$

Figure 1: Combination rules used by CCGBank.

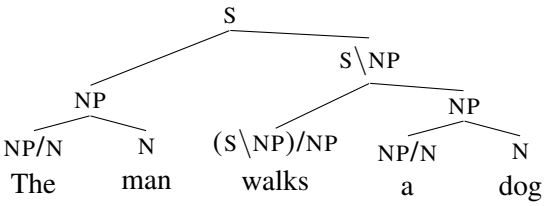

Figure 2: CCG parse for "The man walks a dog."

Tag combinability. A CCG parse of a sentence is derived by recursively combining the categories of sub-phrases. Category combination is performed using only a small set of generic rules (see Figure 1). In the tree in Figure 2, we can see that *a* and *dog* can combine via Forward Application ($>$), with NP/N and N combining to produce NP.

The associativity engendered by CCG's composition rules means that most adjacent lexical categories may be combined. In the Figure 2 tree, we can see that instead of combining (walks·(a·dog)), we could have combined ((walks·a)·dog) since (S\NP)/NP and NP/N can combine using $>\mathbf{B}$.

3 Model

In this section we define the generative process we use to model a corpus of sentences. We begin by generating the model parameters: for each supertag type $\mathbf{t}$ in the tag set T, the transition probabilities to the next state ($\pi_{\mathbf{t}}$) and the emission probabilities ($\phi_{\mathbf{t}}$) are generated by draws from Dirichlet distributions parameterized with per-tag mean distributions ($\pi^0_{\mathbf{t}}$ and $\phi^0_{\mathbf{t}}$, respectively) and concentration parameters (α_π and α_ϕ). By setting α_π close to zero, we can encode our prior expectation that transition distributions should be relatively peaked (i.e., that each tag type should be followed by relatively few tag types). The prior means, discussed below, encode both linguistic intuitions about expected tag-tag transition behavior and automatically-extracted corpus information. Given these parameters, we next generate the sentences of the corpus. This process is summarized as follows:

Parameters:

$$\phi_{\mathbf{t}} \sim \text{Dirichlet}(\alpha_\phi, \phi^0_{\mathbf{t}}) \quad \forall \mathbf{t} \in T$$
$$\pi_{\mathbf{t}} \sim \text{Dirichlet}(\alpha_\pi, \pi^0_{\mathbf{t}}) \quad \forall \mathbf{t} \in T$$

Sentence:

$$y_1 \sim \text{Categorical}(\pi_{\langle \text{S} \rangle})$$
$$\text{for } i \in \{1, 2, \ldots\}, \text{until } y_i = \langle \text{E} \rangle$$
$$x_i \mid y_i \sim \text{Categorical}(\phi_{y_i})$$
$$y_{i+1} \mid y_i \sim \text{Categorical}(\pi_{y_i})$$

This model can be understood as a Bayesian HMM (Goldwater and Griffiths, 2007). We next discuss how the prior distributions are constructed to build in additional inductive bias.

3.1 Transition Prior Means ($\pi^0_{\mathbf{t}}$)

We use the prior mean for each tag's transition distribution to build in two kinds of bias. First, we want to favor linguistically probable tags. Second, we want to favor transitions that result in a tag pair that combines according to CCG's combinators. For simplicity, we will define $\pi^0_{\mathbf{t}}$ as a mixture of two components, the first, $P_\pi(\mathbf{u})$ is an (unconditional) distribution over category types $\mathbf{u}$ that favors cross-linguistically probable categories. The second component, $P_\pi(\mathbf{u} \mid \mathbf{t})$, conditions on the previous tag type, $\mathbf{t}$, and assigns higher probability to pairs of tags that can be combined. That is, the probability of transitioning from $\mathbf{t}$ to $\mathbf{u}$ in the Dirichlet mean distribution is given by[1]

$$\pi^0_{\mathbf{t}}(\mathbf{u}) = \lambda \cdot P_\pi(\mathbf{u}) + (1 - \lambda) \cdot P_\pi(\mathbf{u} \mid \mathbf{t}).$$

We discuss the two mixture components in turn.

3.1.1 Unigram Category Generator ($P_\pi(\mathbf{u})$)

In this section, we define a CCG category generator that generates cross-linguistically likely category types. Baldridge's approach estimated the likelihood of a category using the inverse number of sub-categories: $P_{\text{CPLX}}(\mathbf{u}) \propto 1/\textit{complexity}(\mathbf{u})$. We propose an improvement, P_{G}, expressed as a probabilistic grammar:[2]

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

[1] Following Baldridge (2008), we fix $\lambda = 0.5$ for our experiments.

[2] For readability, we use the notation $\overline{p} = (1 - p)$.

where A, B, C are categories and a is an atomic category (and terminal): $a \in \{\text{S}, \text{N}, \text{NP}, \ldots\}$.[3]

We have designed this grammar to capture several important CCG characteristics. In particular we encode four main ideas, each captured through a different parameter of the grammar and discussed in greater detail below:

1. Simpler categories are more likely: e.g. N/N is *a priori* more likely than (N/N)/(N/N).
2. Some atoms are more likely than others: e.g. NP is more likely than S, much more than NP_{nb}.
3. Modifiers are more likely: e.g. $(\text{S}\backslash\text{NP})/(\text{S}\backslash\text{NP})$ is more likely than $(\text{S}\backslash\text{NP})/(\text{NP}\backslash\text{NP})$.
4. Operators occur with different frequencies.

The first idea subsumes the complexity measure used by Baldridge, but accomplishes the goal naturally by letting the probabilities decrease as the category grows. The rate of decay is governed by the p_{term} parameter: the marginal probability of generating a terminal (atomic) category in each expansion. A higher p_{term} means a stronger emphasis on simplicity. The probability distribution over categories is guaranteed to be proper so long as $p_{term} > \frac{1}{2}$ since the probability of the depth of a tree will decrease geometrically (Chi, 1999).

The second idea is a natural extension of the complexity concept and is particularly relevant when features are used. The original complexity measure treated all atoms uniformly, but e.g. we would expect NP_{expl}/N to be less likely than NP/N since it contains the more specialized, and thus rarer, atom NP_{expl}. We define the distribution $p_{atom}(a)$ as the prior over atomic categories.

Due to our weak, type-only supervision, we have to estimate p_{atom} from just the tag dictionary and raw corpus, without frequency data. Our goal is to estimate the number of each atom in the supertags that should appear on the raw corpus tokens. Since we don't know what the correct supertags are, we first estimate counts of supertags, from which we can extract estimated atom counts. Our strategy is to uniformly distribute each raw corpus token's counts over all of its possible supertags, as specified in the tag dictionary. Word types not appearing in the tag dictionary are ig-

[3] While very similar to standard probabilistic context-free grammars seen in NLP work, this grammar is not context-free because modifier categories must have matching operands. However, this is not a problem for our approach since the grammar is unambiguous, defines a proper probability distribution, and is only used for modeling the relative likelihoods of categories (not parsing categories).

nored for the purposes of these estimates. Assuming that $C(w)$ is the number of times that word type w is seen in the raw corpus, $\mathit{atoms}(a, t)$ is the number of times atom a appears in $\mathbf{t}$, $\text{TD}(w)$ is the set of tags associated with w, and $\text{TD}(\mathbf{t})$ is the set of word types associated with $\mathbf{t}$:

$$C_{\mathit{supertag}}(\mathbf{t}) = \sum_{w \in \text{TD}(\mathbf{t})} (C(w)+\delta)/|\text{TD}(w)|$$
$$C_{\mathit{atom}}(a) = \sum_{\mathbf{t} \in T} \mathit{atoms}(a, \mathbf{t}) \cdot C_{\mathit{supertag}}(\mathbf{t})$$
$$p_{\mathit{atom}}(a) \propto C_{\mathit{atom}}(a) + \delta$$

Adding δ smooths the estimates.

Using the raw corpus and tag dictionary data to set p_{atom} allows us to move beyond Baldridge's work in another direction: it provides us with a natural way to combine CCG's universal assumptions with corpus-specific data.

The third and fourth ideas pertain only to complex categories. If the category is complex, then we consider two additional parameters. The parameter p_{fw} is the marginal probability that the complex category's operator specifies a forward argument. The parameter p_{mod} gives the amount of marginal probability mass that is allocated for modifier categories. Note that it is not necessary for p_{mod} to be greater than $\frac{1}{2}$ to achieve the desired result of making modifier categories more likely than non-modifier categories: the number of potential modifiers make up only a tiny fraction of the space of possible categories, so allocating more than that mass as p_{mod} will result in a category grammar that gives disproportionate weight to modifiers, increasing the likelihood of any particular modifier from what it would otherwise be.

3.1.2 Bigram Category Generator ($P_\pi(\mathbf{u} \mid \mathbf{t})$)

While the above processes encode important properties of the distribution over categories, the internal structure of categories is not the full story: cross-linguistically, the categories of adjacent tokens are much more likely to be combinable via some CCG rule. This is the second component of our mixture model.

Baldridge derives this bias by allocating the majority of the transition probability mass from each tag $\mathbf{t}$ to tags that can follow $\mathbf{t}$ according to some combination rule. Let $\kappa(\mathbf{t}, \mathbf{u})$ be an indicator of whether $\mathbf{t}$ connects to $\mathbf{u}$; for $\sigma \in [0, 1]$:[4]

$$P_\kappa(\mathbf{u} \mid \mathbf{t}) = \begin{cases} \sigma \cdot \text{uniform}(\mathbf{u}) & \text{if } \kappa(\mathbf{t}, \mathbf{u}) \\ (1-\sigma) \cdot \text{uniform}(\mathbf{u}) & \text{otherwise} \end{cases}$$

[4] Again, following Baldridge (2008), we fix $\sigma = 0.95$ for our experiments.

There are a few additional considerations that must be made in defining κ, however. In assuming the special tags $\langle \text{S} \rangle$ and $\langle \text{E} \rangle$ for the start and end of the sentence, respectively, we can define $\kappa(\langle \text{S} \rangle, \mathbf{u}) = 1$ when $\mathbf{u}$ seeks no left-side arguments (since there are no tags to the left with which to combine) and $\kappa(\mathbf{t}, \langle \text{E} \rangle) = 1$ when $\mathbf{t}$ seeks no right-side arguments. So $\kappa(\langle \text{S} \rangle, \text{NP/N}) = 1$, but $\kappa(\langle \text{S} \rangle, \text{S}\backslash\text{NP}) = 0$. If atoms have *features* associated, then the atoms are allowed to unify if the features match, or if at least one of them does not have a feature. So $\kappa(\text{NP}_{nb}, \text{S}\backslash\text{NP}) = 1$, but $\kappa(\text{NP}_{nb}, \text{S}\backslash\text{NP}_{conj}) = 0$. In defining κ, it is also important to ignore possible arguments on the wrong side of the combination since they can be consumed without affecting the connection between the two. To achieve this for $\kappa(\mathbf{t}, \mathbf{u})$, it is assumed that it is possible to consume all preceding arguments of $\mathbf{t}$ and all following arguments of $\mathbf{u}$. So $\kappa(\text{NP}, (\text{S}\backslash\text{NP})/\text{NP}) = 1$. This helps to ensure the associativity discussed earlier. Finally, the atom NP is allowed to unify with N if N is the argument. So $\kappa(\text{N}, \text{S}\backslash\text{NP}) = 1$, but $\kappa(\text{NP/N}, \text{NP}) = 0$. This is due to the fact that CCGBank assumes that N can be rewritten as NP.

Type-supervised initialization. As above, we want to improve upon Baldridge's ideas by encoding not just universal CCG knowledge, but also automatically-induced corpus-specific information where possible. To that end, we can define a conditional distribution $P_{\mathit{tr}}(\mathbf{u} \mid \mathbf{t})$ based on statistics from the raw corpus and tag dictionary. We use the same approach as we did above for setting p_{atom} (and the definition of $\phi^0_{\mathbf{t}}$ below): we estimate by evenly distributing raw corpus counts over the tag dictionary entries. Assume that $C(w_1, w_2)$ is the (δ-smoothed) count of times word type w_1 was directly followed by w_2 in the raw corpus, and ignoring any words not found in the tag dictionary:

$$C(\mathbf{t}, \mathbf{u}) = \delta + \sum_{w_1 \in \text{TD}(\mathbf{t}),\ w_2 \in \text{TD}(\mathbf{u})} \frac{C(w_1, w_2)}{|\text{TD}(w_1)| \cdot |\text{TD}(w_2)|}$$

$$P_{\mathit{tr}}(\mathbf{u} \mid \mathbf{t}) = C(\mathbf{t}, \mathbf{u}) / \textstyle\sum_{\mathbf{u}'} C(\mathbf{t}, \mathbf{u}')$$

Then the alternative definition of the compatibility distribution is as follows:

$$P^{\mathit{tr}}_\kappa(\mathbf{u} \mid \mathbf{t}) = \begin{cases} \sigma \cdot P_{\mathit{tr}}(\mathbf{u} \mid \mathbf{t}) & \text{if } \kappa(\mathbf{t}, \mathbf{u}) \\ (1\text{–}\sigma) \cdot P_{\mathit{tr}}(\mathbf{u} \mid \mathbf{t}) & \text{otherwise} \end{cases}$$

Our experiments compare performance when $\pi^0_{\mathbf{t}}$ is set using $P_\pi(\mathbf{u}) = P_{\text{CPLX}}$ (experiment 3) versus our category grammar P_{G} (4–6), and using $P_\pi(\mathbf{u} \mid \mathbf{t}) = P_\kappa$ as the compatibility distribution (3–4) versus P^{tr}_κ (5–6).

3.2 Emission Prior Means ($\phi^0_{\mathbf{t}}$)

For each supertag type $\mathbf{t}$, $\phi^0_{\mathbf{t}}$ is the mean distribution over words it emits. While Baldridge's approach used a uniform emission initialization, treating all words as equally likely, we can, again, induce token-level corpus-specific information:[5] To set $\phi^0_{\mathbf{t}}$, we use a variant and simplification of the procedure introduced by Garrette and Baldridge (2012) that takes advantage of our prior over categories P_{G}.

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

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

For *unknown* words, we first use the idea of tag "openness" to estimate the likelihood of a particular tag t applying to an unknown word: if a tag applies to many word types, it is likely to apply to some new word type.

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

Then, we apply Bayes' rule to get $P(\mathbf{t} \mid \text{unk})$, and use that to estimate word/tag counts for unknown words:

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

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

Thus, with the estimated counts for all words:

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

We perform experiments comparing performance when $\phi^0_{\mathbf{t}}$ is uniform (3–5) and when $\phi^0_{\mathbf{t}}(w) = P_{em}(w \mid \mathbf{t})$ (6).

[5] Again, without gold tag frequencies.

4 Posterior Inference

We wish to find the most likely supertag of each word, given the model we just described and a corpus of training data. Since there is exact inference with these models is intractable, we resort to Gibbs sampling to find an approximate solution. At a high level, we alternate between resampling model parameters $(\phi_{\mathbf{t}}, \pi_{\mathbf{t}})$ given the current tag sequence and resampling tag sequences given the current model parameters and observed word sequences. It is possible to sample a new tagging from the posterior distribution over tag sequences for a sentence, given the sentence and the HMM parameters using the forward-filter backward-sample (FFBS) algorithm (Carter and Kohn, 1996). To efficiently sample new HMM parameters, we exploit Dirichlet-multinomial conjugacy. By repeating these alternating steps and accumulating the number of times each supertag is used in each position, we obtain an approximation of the required posterior quantities.

Our inference procedure takes as input the transition prior means $\pi^0_{\mathbf{t}}$, the emission prior means $\phi^0_{\mathbf{t}}$, and concentration parameters α_π and α_ϕ, along with the raw corpus and tag dictionary. The set of supertags associated with a word w will be known as $\text{TD}(w)$. We will refer to the set of word types included in the tag dictionary as "known" words and others as "unknown" words. For simplicity, we will assume that $\text{TD}(w)$, for any unknown word w, is the full set of CCG categories. During sampling, we always restrict the possible tag choices for a word w to the categories found in $\text{TD}(w)$. We refer to the sequence of word tokens as $\mathbf{x}$ and tags as $\mathbf{y}$.

We initialize the sampler by setting $\pi_{\mathbf{t}} = \pi^0_{\mathbf{t}}$ and $\phi_{\mathbf{t}} = \phi^0_{\mathbf{t}}$ and then sampling tagging sequences using FFBS.

To sample a tagging for a sentence $\mathbf{x}$, the strategy is to inductively compute, for each token x_i starting with $i = 0$ and going "forward", the probability of generating $x_0, x_1, \ldots, x_i$ via any tag sequence that ends with $y_i = \mathbf{u}$:

$$p(y_i = \mathbf{u} \mid x_{0:i}) = \phi_{\mathbf{u}}(x_i) \cdot \sum_{\mathbf{t} \in T} \pi_{\mathbf{t}}(\mathbf{u}) \cdot p(y_{i-1} = \mathbf{t} \mid x_{0:i-1})$$

We then pass through the sequence again, this time "backward" starting at $i = |\mathbf{x}| - 1$ and sampling

$$y_i \mid y_{i+1} \sim p(y_i = \mathbf{t} \mid x_{0:i}) \cdot \pi_{\mathbf{t}}(y_{i+1}).$$

Corpus		num. tags	raw tokens	TD tokens	TD entries	ambiguity type	ambiguity token	dev tokens	test tokens
English	CCGBank POS	50	158k	735k	45k	3.75	13.11	—	—
	CCGBank	1,171			65k	56.98	296.18	128k	127k
Chinese	CTB-CCG	829	99k	439k	60k	96.58	323.37	59k	85k
Italian	CCG-TUT	955	6k	27k	9k	178.88	426.13	5k	5k

Table 1: Statistics for the various corpora used. CCGBank is English, CCG-CTB is Chinese, and TUT is Italian. The number of tags includes only those tags found in the tag dictionary (TD). Ambiguity rates are the average number of entries in the unpruned tag dictionary for each word in the raw corpus. English POS statistics are shown only for comparison; only CCG experiments were run.

The block-sampling approach of choosing new tags for a sentence all at once is particularly beneficial given the sequential nature of the model of the HMM. In an HMM, a token's adjacent tags tend to hold onto its current tag due to the relationships between the three. Resampling all tags at once allows for more drastic changes at each iteration, providing better opportunities for mixing during inference. The FFBS approach has the additional advantage that, by resampling the distributions only once per iteration, we are able to resample all sentences in parallel. This is not strictly true of all HMM problems with FFBS, but because our data is divided by sentence, and each sentence has a known start and end tag, the tags chosen during the sampling of one sentence cannot affect the sampling of another sentence in the same iteration.

Once we have sampled tags for the entire corpus, we resample π and ϕ. The newly-sampled tags $\mathbf{y}$ are used to compute $C(w, \mathbf{t})$, the count of tokens with word type w and tag $\mathbf{t}$, and $C(\mathbf{t}, \mathbf{u})$, the number of times tag $\mathbf{t}$ is directly followed by tag $\mathbf{u}$. We then sample, for each $\mathbf{t} \in T$ where T is the full set of valid CCG categories:

$$\pi_{\mathbf{t}} \sim \text{Dir}\left(\langle \alpha_\pi \cdot \pi^0_{\mathbf{t}}(\mathbf{u}) + C(\mathbf{t}, \mathbf{u})\rangle_{\mathbf{u} \in T}\right)$$
$$\phi_{\mathbf{t}} \sim \text{Dir}\left(\langle \alpha_\phi \cdot \phi^0_{\mathbf{t}}(w) + C(w, \mathbf{t})\rangle_{w \in V}\right)$$

It is important to note that this method of resampling allows the draws to incorporate both the data, in the form of counts, and the prior mean, which includes all of our carefully-constructed biases derived from both the intrinsic, universal CCG properties as well as the information we induced from the raw corpus and tag dictionary.

With the distributions resampled, we can continue the procedure by resampling tags as above, and then resampling distributions again, until a maximum number of iterations is reached.

5 Experiments[6]

To evaluate our approach, we used CCGBank (Hockenmaier and Steedman, 2007), which is a transformation of the English Penn Treebank (Marcus et al., 1993); the CTB-CCG (Tse and Curran, 2010) transformation of the Penn Chinese Treebank (Xue et al., 2005); and the CCG-TUT corpus (Bos et al., 2009), built from the TUT corpus of Italian text (Bosco et al., 2000). Statistics on the size and ambiguity of these datasets are shown in Table 1.

For CCGBank, sections 00–15 were used for extracting the tag dictionary, 16–18 for the raw corpus, 19–21 for development data, and 22–24 for test data. For TUT, the first 150 sentences of each of the CIVIL_LAW and NEWSPAPER sections were used for raw data, the next sentences 150–249 of each was used for development, and the sentences 250–349 were used for test; the remaining data, 457 sentences from CIVIL_LAW and 548 from NEWSPAPER, plus the much smaller 132-sentence JRC_ACQUIS data, was used for the tag dictionary. For CTB-CCG, sections 00–11 were used for the tag dictionary, 20–24 for raw, 25–27 for dev, and 28–31 for test.

Because we are interested in showing the relative gains that our ideas provide over Baldridge (2008), we reimplemented the initialization procedure from that paper, allowing us to evaluate all approaches consistently. For each dataset, we ran a series of experiments in which we made further changes from the original work. We first ran a baseline experiment with uniform transition and emission initialization of EM (indicated as "1." in Table 2) followed by our reimplementation of the initialization procedure by Baldridge (2). We then

[6] All code and experimental scripts are available at http://www.github.com/dhgarrette/2014-ccg-supertagging

Corpus		English				Chinese				Italian			
TD cutoff		0.1	0.01	0.001	no	0.1	0.01	0.001	no	0.1	0.01	0.001	no
1. uniform	EM	77	62	47	38	64	39	30	26	51	32	30	30
2. init (Baldridge)	EM	78	67	55	41	66	43	33	28	**54**	36	33	32
3. init	Bayes	74	68	56	42	65	56	47	37	52	46	40	40
4. P_{G}	Bayes	74	70	59	42	64	57	47	36	52	40	39	40
5. P_{G}, P_{κ}^{tr}	Bayes	75	72	61	50	66	58	49	44	52	44	41	43
6. P_{G}, P_{κ}^{tr}, P_{em}	Bayes	**80**	**80**	**73**	**51**	**69**	**62**	**56**	**49**	53	**47**	**45**	**46**

Table 2: Experimental results: test-set per-token supertag accuracies. "TD cutoff" indicates the level of tag dictionary pruning; see text. (1) is uniform EM initialization. (2) is a reimplementation of (Baldridge, 2008). (3) is Bayesian formulation using only the ideas from Baldridge: P_{CPLX}, P_{κ}, and uniform emissions. (4–6) are our enhancements to the prior: using our category grammar in P_{G} instead of P_{CPLX}, using P_{κ}^{tr} instead of P_{κ}, and using P_{em} instead of uniform.

experimented with the Bayesian formulation, first using the same information used by Baldridge, and then adding our enhancements: using our category grammar in P_{G}, using P_{κ}^{tr} as the transition compatability distribution, and using P_{em} as $\phi_{\mathbf{t}}^{0}(w)$.

For each dataset, we ran experiments using four different levels of tag dictionary pruning. Pruning is the process of artificially removing noise from the tag dictionary by using token-level annotation counts to discard low-probability tags; for each word, for cutoff x, any tag with probability less than x is excluded. Tag dictionary pruning is a standard procedure in type-supervised training, but because it requires information that does not truly conform to the type-supervised scenario, we felt that it was critical to demonstrate the performance of our approach under situations of less pruning, *including* no artificial pruning at all.

We emphasize that unlike in most previous work, we use *incomplete* tag dictionaries. Most previous work makes the unrealistic assumption that the tag dictionary contains an entry for every word that appears in either the training *or testing* data. This is a poor approximation of a real tagging system, which will never have complete lexical knowledge about the test data. Even work that only assumes complete knowledge of the tagging possibilities for the lexical items in the training corpus is problematic (Baldridge, 2008; Ravi et al., 2010). This still makes learning unrealistically easy since it dramatically reduces the ambiguity of words that would have been unseen, and, in the case of CCG, introduces additional tags that would not have otherwise been known. To ensure that our experiments are more realistic, we draw our tag dictionary entries from data that is totally disjoint from both the raw and test corpora. During learning, any unknown words (words not appearing in the tag dictionary) are unconstrained so that they may take *any* tag, and are, thus, maximally ambiguous.

We only performed minimal parameter tuning, choosing instead to stay consistent with Baldridge (2008) and simply pick reasonable-seeming values for any additional parameters. Any tuning that was performed was done with simple hill-climbing on the development data of English CCGBank. All parameters were held consistent across experiments, including across languages. For EM, we used 50 iterations; for FFBS we used 100 burn-in iterations and 200 sampling iterations.[7] For all experiments, we used $\sigma = 0.95$ for $P_{\kappa}^{(tr)}$ and $\lambda = 0.5$ for $\pi_{\mathbf{t}}^{0}$ to be consistent with previous work, $\alpha_{\pi} = 3000$, $\alpha_{\phi} = 7000$, $p_{term} = 0.6$, $p_{fw} = 0.5$, $p_{mod} = 0.8$, and $\delta = 1000$ for p_{atom}. Test data was run only once, for the final figures.

The final results reported were achieved by using the following training sequence: initialize parameters according to the scenario, train an HMM using EM or FFBS starting with that set of parameters, tag the raw corpus with the trained HMM, add-0.1 smooth counts from the now-tagged raw corpus, and train a maximum entropy Markov model (MEMM) from this "auto-supervised" data.[8]

Results are shown in Table 2. Most notably, the contributions described in this paper improve results in nearly every experimental scenario. We can see immediate, often sizable, gains in most

[7] Final counts are averaged across the sampling iterations.

[8] Auto-supervised training of an MEMM increases accuracy by 1–3% on average (Garrette and Baldridge, 2013). We use the OpenNLP MEMM implementation with its standard set of features: `http://opennlp.apache.org`

cases simply by using the Bayesian formulation. Further gains are seen from adding each of the other various contributions of this paper. Perhaps most interestingly, the gains are only minimal with maximum pruning, but the gains increase as the pruning becomes less aggressive — as the scenarios become more realistic. This indicates that our improvements make the overall procedure more robust.

Error Analysis Like POS-taggers, the learned supertagger frequently confuses nouns (N) and their modifiers (N/N), but the most frequent error made by the English (6) experiment was $(((S\backslash NP)\backslash(S\backslash NP))/N)$ instead of (NP_{nb}/N). However, these are both determiner types, indicating an interesting problem for the supertagger: it often predicts an object type-raised determiner instead of the vanilla NP/N, but in many contexts, both categories are equally valid. (In fact, for parsers that use type-raising as a rule, this distinction in lexical categories does not exist.)

6 Related Work

Ravi et al. (2010) also improved upon the work by Baldridge (2008) by using integer linear programming to find a minimal model of supertag transitions, thereby generating a better starting point for EM than the grammatical constraints alone could provide. This approach is complementary to the work presented here, and because we have shown that our work yields gains under tag dictionaries of various levels of cleanliness, it is probable that employing minimization to set the base distribution for sampling could lead to still higher gains.

On the Bayesian side, Van Gael et al. (2009) used a non-parametric, *infinite* HMM for truly unsupervised POS-tagger learning (Van Gael et al., 2008; Beal et al., 2001). While their model is not restricted to the standard set of POS tags, and may learn a more fine-grained set of labels, the induced labels are arbitrary and not grounded in any grammatical formalism.

Bisk and Hockenmaier (2013) developed an approach to CCG grammar induction that does not use a tag dictionary. Like ours, their procedure learns from general properties of the CCG formalism. However, while our work is intended to produce categories that match those used in a particular training corpus, however complex they might be, their work produces categories in a simplified form of CCG in which N and S are the only atoms and no atoms have features. Additionally, they assume that their training corpus is annotated with POS tags, whereas we assume truly raw text.

Finally, we find the task of weakly-supervised supertagger learning to be particularly relevant given the recent surge in popularity of CCG. An array of NLP applications have begun using CCG, including semantic parsing (Zettlemoyer and Collins, 2005) and machine translation (Weese et al., 2012). As CCG finds more applications, and as these applications move to lower-resource domains and languages, there will be increased need for the ability to learn without full supervision.

7 Conclusion and Future Work

Standard strategies for type-supervised HMM estimation are less effective as the number of categories increases. In contrast to POS tag sets, CCG supertags, while quite numerous, have structural clues that can simplify the learning problem. Baldridge (2008) used this formalism-specific structure to inform an initialization procedure for EM. In this work, we have shown that CCG structure can instead be used to motivate an effective *prior distribution* over the parameters of an HMM supertagging model, allowing our work to outperform Baldridge's previously state-of-the-art approach, and to do so in a principled manner that lends itself better to future extensions such as incorporation in more complex models.

This work also improves on Baldridge's simple "complexity" measure, developing instead a probabilistic category grammar over supertags that allows our prior to capture a wider variety of interesting and useful properties of the CCG formalism.

Finally, we were able to achieve further gains by augmenting the universal CCG knowledge with corpus-specific information that could be automatically extracted from the weak supervision that is available: the raw corpus and the tag dictionary. This allows us to combine the cross-linguistic properties of the CCG formalism with corpus- or language-specific information in the data into a single, unified Bayesian prior.

Our model uses a relatively large number of parameters, e.g., p_{term}, p_{fw}, p_{mod}, p_{atom}, in the prior. Here, we fixed each to a single value (i.e., a "fully Bayesian" approach). Future work might explore sensitivity to these choices, or empirical Bayesian or maximum *a posteriori* inference for their values (Johnson and Goldwater, 2009).

In this work, as in most type-supervised work, the tag dictionary was automatically extracted from an existing tagged corpus. However, a tag dictionary could instead be automatically induced via multi-lingual transfer (Das and Petrov, 2011) or generalized from human-provided information (Garrette and Baldridge, 2013; Garrette et al., 2013). Again, since the approach presented here has been shown to be somewhat robust to tag dictionary noise, it is likely that the model would perform well even when using an automatically-induced tag dictionary.

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

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

Srinivas Bangalore and Aravind K. Joshi. 1999. Supertagging: An approach to almost parsing. *Computational Linguistics*, 25(2).

Matthew J. Beal, Zoubin Ghahramani, and Carl Edward Rasmussen. 2001. The innite hidden Markov model. In *NIPS*.

Taylor Berg-Kirkpatrick, Alexandre Bouchard-Côté, John DeNero, and Dan Klein. 2010. Painless unsupervised learning with features. In *Proceedings of NAACL*.

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

Phil Blunsom and Trevor Cohn. 2011. A hierarchical Pitman-Yor process HMM for unsupervised part of speech induction. In *Proceedings of ACL*.

Johan Bos, Cristina Bosco, and Alessandro Mazzei. 2009. Converting a dependency treebank to a categorial grammar treebank for Italian. In M. Passarotti, Adam Przepiórkowski, S. Raynaud, and Frank Van Eynde, editors, *Proceedings of the Eighth International Workshop on Treebanks and Linguistic Theories (TLT8)*.

Cristina Bosco, Vincenzo Lombardo, Daniela Vassallo, and Leonardo Lesmo. 2000. Building a treebank for Italian: a data-driven annotation schema. In *Proceedings of LREC*.

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

Zhiyi Chi. 1999. Statistical properties of probabilistic context-free grammars. *Computational Linguistics*, 25(1).

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

Dan Garrette and Jason Baldridge. 2012. Type-supervised hidden Markov models for part-of-speech tagging with incomplete tag dictionaries. In *Proceedings of EMNLP*.

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

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

Yoav Goldberg, Meni Adler, and Michael Elhadad. 2008. EM can find pretty good HMM POS-taggers (when given a good start). In *Proceedings of ACL*.

Sharon Goldwater and Thomas L. Griffiths. 2007. A fully Bayesian approach to unsupervised part-of-speech tagging. In *Proceedings of ACL*.

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

Mark Johnson and Sharon Goldwater. 2009. Improving nonparameteric Bayesian inference: Experiments on unsupervised word segmentation with adaptor grammars. In *Proceedings of NAACL*.

Mark Johnson. 2007. Why doesn't EM find good HMM POS-taggers? In *Proceedings of EMNLP-CoNLL*.

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

Yoong Keok Lee, Aria Haghighi, and Regina Barzilay. 2010. Simple type-level unsupervised pos tagging. In *Proceedings of EMNLP*.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2).

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

Sujith Ravi and Kevin Knight. 2009. Minimized models for unsupervised part-of-speech tagging. In *Proceedings of ACL-AFNLP*.

Sujith Ravi, Jason Baldridge, and Kevin Knight. 2010. Minimized models and grammar-informed initialization for supertagging with highly ambiguous lexicons. In *Proceedings of ACL*, pages 495–503.

Noah A. Smith and Jason Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of ACL*.

Mark Steedman and Jason Baldridge. 2011. Combinatory categorial grammar. In Robert Borsley and Kersti Borjars, editors, *Non-Transformational Syntax: Formal and Explicit Models of Grammar*. Wiley-Blackwell.

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

Daniel Tse and James R. Curran. 2010. Chinese CCGbank: Extracting CCG derivations from the Penn Chinese treebank. In *Proceedings of COLING*.

Jurgen Van Gael, Yunus Saatci, Yee Whye Teh, and Zoubin Ghahramani. 2008. Beam sampling for the infinite hidden Markov model. In *Proceedings of ICML*.

Jurgen Van Gael, Andreas Vlachos, and Zoubin Ghahramani. 2009. The infinite HMM for unsupervised PoS tagging. In *Proceedings of EMNLP*.

Jonathan Weese, Chris Callison-Burch, and Adam Lopez. 2012. Using categorial grammar to label translation rules. In *Proceedings of WMT*.

Nianwen Xue, Fei Xia, Fu-Dong Chiou, and Martha Palmer. 2005. The Penn Chinese TreeBank: Phrase structure annotation of a large corpus. *Natural Language Engineering*, 11(2):207–238.

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