A Bayesian Sequence Model for Grammar Induction using Human-like Memory Constraints

Anonymous ACL submission

Abstract

This paper describes an application of a depth-bounded left-corner parsing strategy to a grammar induction task. The proposed model is severely constrained to a single memory element, allowing no center embedding but unlimited left and right embedding, which may resemble memory constraints of early language learners. Despite this severe constraint, the model described in this paper still manages to perform competitively with unconstrained models on an existing task of acquiring grammar from short (ten-word or fewer) sentences.

1 Introduction

Grammar induction is often approached using chart parsing techniques (Klein and Manning, 2002), which allow any pair of adjacent spans to be hypothesized as a constituent. As a result, trees with any amount of center-embedding recursion can be induced by these models. However, center embedding is known to be difficult for human sentence processing (Chomsky and Miller, 1963; Karlsson, 2007), leading to famously difficult sentences like '[$_{\text{NP}}$ The cart [$_{\text{NP}}$ the horse [$_{\text{NP}}$ the man] bought] pulled] broke.' Sentence processing models proposed in the cognitive modeling community therefore often use variants of a left-corner parsing strategy (Aho and Ullman, 1972; Johnson-Laird, 1983; Abney and Johnson, 1991; Gibson, 1991; Henderson, 2004; Lewis and Vasishth, 2005; Schuler et al., 2010), which isolate and apply memory constraints to such embeddings.

This paper describes an application of a depth-bounded left-corner parsing strategy to a grammar induction task. The proposed model is severely constrained to a single memory element, allowing no center embedding but unlimited left and right embedding. This severe constraint may resemble memory constraints of early language learners. This constrained model may also function as a base case for a more complex model, able to hypothesize multiple center embeddings using hierarchical priors which depend on the learnability of a depth-one model as a necessary precondition. Despite the severe constraint of only a single depth level in processing, the model described in this paper still manages to perform competitively with unconstrained models on an existing task of acquiring grammar from short (ten-word or fewer) sentences.

The remainder of this paper is organized as follows. Section 2 describes some related work on grammar induction and sequence modeling. Section 3 describes the proposed memory-bounded left-corner parsing grammar induction model. Section 4 describes experiments showing competitive performance of this proposed model to existing grammar induction models which are not similarly constrained. Section 5 provides a summary and conclusion.

2 Background and Related Work

This work is primarily related to three different strains in the computational linguistics and machine learning literature – grammar induction, Bayesian part-of-speech tag induction, and sequence models for syntactic parsing. We will briefly cover the most relevant work from each area.

Grammar induction models learn the syntactic structure of a language from a sample of unlabeled text, rather than a gold-standard treebank. The constituent context model (Klein and Manning, 2002) uses expectation-maximization (EM) to learn differences between observed and unobserved bracketings, and the dependency model with valence (Klein and Manning, 2004) uses EM to learn distributions that generate child dependencies, conditioned on valence (left or right direction) in addition to the lexical head. One approach that shares the sequential nature of our work uses cascaded hidden Markov models (HMMs) (Ponvert et al., 2011), building up structure by repeated applications of the HMM to previously discovered chunks. However, the above models operate over a standard Treebank-style syntactic space and therefore do not take advantage of cognitively-motivated depth limitations that can be introduced by using a left-corner parsing strategy.

Bayesian models have been widely used in part-of-speech (POS) tag induction for their ability to flexibly adapt to data size and complexity as well as their ability to inject domain knowledge through the use of priors. The POS tag induction task is relevant to our work because it also is done in the context of sequence models, typically variants of HMMs. Johnson (2007) found that Bayesian inference for POS tag induction can improve over EM, especially for small amounts of data where the priors are important. Nonparametric Bayesian models, specifically the infinite HMM, have also been applied to POS induction (van Gael et al., 2009). Van Gael et al. (2009) takes advantage of efficient inferencing algorithms for sequences (van Gael et al., 2008), which our work also uses and extends.

Finally, this work builds on sequential generative models for parsing, specifically a cognitively motivated hierarchical sequence model (van Schijndel et al., 2013). This method transforms a grammar into a set of operations over a hierarchical hidden Markov model; Schuler et al. (2010) demonstrate that a fixed four-level hierarchy can parse nearly all human-generated sentences. While this model has been applied to parsing with state of the art results (van Schijndel et al., 2013), it has thus far only been used in a supervised setting.

3 Model

We use a generative sequence model representing syntactic structures inspired by left-corner parsing (Aho and Ullman, 1972) and hierarchical hidden Markov model (HHMM) parsing (Schuler et al., 2010). Our core innovation is the adaptation of this model to unsupervised induction using constrained priors.

A left-corner parser maintains a sequence of incomplete categories $a/b, a'/b', \ldots$, each consisting of an active category a lacking an awaited category b yet to come (van Schijndel et al., 2013). These incomplete categories can be assembled into any possible tree structure for a given sequence of words using four operations: 'fork,' 'no fork,' 'join,' and 'no join,' as defined by the following natural deduction rules. Fork (+F) and no-fork (–F) operations deduce a complete category c from observed word w or from a/b and w, respectively:

$$\frac{a/b \quad w}{a/b \quad c}\, b \xrightarrow{+} c \,...\,; \;\; c \rightarrow w \qquad \text{(+F)}$$

$$\frac{a/b \quad w}{c}\, a = c;\; b \rightarrow w \qquad \text{(–F)}$$

where $b \xrightarrow{+} c$ ... constrains c to be a leftmost descendant of b at some depth. Join (+J) and no-join (–J) operations deduce an incomplete category a/b' or a'/b' from a/b and c, or just from c, respectively:

$$\frac{a/b \quad c}{a/b'}\, b \rightarrow c\, b' \qquad \text{(+J)}$$

$$\frac{a/b \quad c}{a/b \quad a'/b'}\, b \xrightarrow{+} a' \,...\,; \;\; a' \rightarrow c\, b' \qquad \text{(–J)}$$

Human-like memory constraints may then be defined on the number of such incomplete categories that can be maintained and the length of time they can be kept. By limiting the model to a single level of recursive depth (as opposed to the four levels in supervised HHMM parsers), we greatly improve inference speeds, while still allowing for learning and parsing of most of the sentences with less than ten tokens in the Wall Street Journal section of the Penn Treebank, a standard grammar induction task. The syntax of each token is represented with three grammatical random variables; an *Active* category A, representing the constituent type currently being built; an *Awaited* category B, representing the constituent type required to complete the active category; and a part of speech (POS) tag P. The model also makes use of two binary switching variables, F (for *Fork*) and J (for *Join*) that guide the transitions of the other states. These two binary switching variables yield four cases: $+/+$, $+/-$, $-/+$ and $-/-$, but in the depth-one model only two of these are used within sentences: $+/+$ (fork and join) and $-/-$ (no fork and no join).

In the $+/+$ case, the active category remains the same ($a_t = a_{t\text{–}1}$) and the awaited category merges with the POS tag to create a new awaited category. An example of this case is where the state at t–1 is VP/NP (verb phrase awaiting a noun phrase) and $p_{t\text{–}1}$ is a determiner (DT); in this case the NP is likely not complete, and so we need a fork operation to generate the next item, and an immediate join operation to indicate that the next element can be reduced immediately (i.e., the system does not need to store an extra element).

In the $-/-$ (no fork and no join) case, the previous active category ($a_{t\text{–}1}$) is reduced (completed), but a new active category (a_t) is generated to continue the sentence. An example of this case is after encountering the subject, where the state at $t - 1$ is NP/NN (noun phrase missing a common noun) and $p_{t\text{–}1}$ is a common noun; no new element is needed to complete the active constituent. Given no fork, join must not occur, unless the sentence is ready to terminate.

The other two cases ($+/-$ and $-/+$), which add and remove memory elements, are used deterministically at sentence start and end, respectively, and are therefore not learned.

It is important to note that this constrained process still allows more parses than purely left-branching trees (using only $-/-$ operations) and purely right-branching trees (using only $+/+$ operations) because it can switch between these two options within the same sentence as long as this does not lead to center embedding ($+/-$ operations followed by $-/+$ operations).

We follow the approach of van Gael et al. (2009) and apply nonparametric priors over the active, awaited, and part-of-speech variables. This approach allows us to learn not only the parameters of the model—such as what parts of speech are likely to be created from what awaited categories—but also the cardinality of how many active, awaited, and part of speech categories are present in a fully unsupervised fashion. No labels are needed for inference, which alternates between inferring these unseen categories and the associated model parameters.

3.1 Parser Model Definition

Let F represent the fork variable, J the join variable, A the active variable, B the awaited variable, P the POS tag variable, and W the observed word token. Let the state s_t be the collection of the hidden active, awaited, part of speech, fork, and join variables $\{f_t, j_t, a_t, b_t, p_t\}$ at position t in the sequence. The joint probability of the hidden state s_t and observed word w_t, given their previous context, are defined using Markov independence assumptions and the fork-join variable decomposition of van Schijndel et al. (2013), which preserves PCFG probabilities in incremental sentence processing:

$$\begin{aligned}
\mathsf{P}(s_t, w_t | s_1^{t\text{–}1}, w_1^{t\text{–}1}) &\stackrel{\text{def}}{=} \mathsf{P}(s_t, w_t | s_{t\text{–}1}) \\
&\stackrel{\text{def}}{=} \mathsf{P}(s_t | s_{t\text{–}1}) \cdot \mathsf{P}(w_t | s_t) \\
&\stackrel{\text{def}}{=} \mathsf{P}(f_t, j_t, a_t, b_t, p_t | s_{t\text{–}1}) \cdot \mathsf{P}(w_t | s_t) \\
&= \mathsf{P}_F(f_t | s_{t\text{–}1}) \cdot \\
&\quad \mathsf{P}_J(j_t | f_t, s_{t\text{–}1}) \cdot \\
&\quad \mathsf{P}_A(a_t | f_t, j_t, s_{t\text{–}1}) \cdot \\
&\quad \mathsf{P}_B(b_t | f_t, j_t, a_t, s_{t\text{–}1}) \cdot \\
&\quad \mathsf{P}_P(p_t | f_t, j_t, a_t, b_t, s_{t\text{–}1}) \cdot \\
&\quad \mathsf{P}_W(w_t | s_t)
\end{aligned} \tag{1}$$

We now describe the models for each of the distributions P_F, P_J, P_A, P_B, P_P, and P_W. In the depth-one model, we only need to consider situations in which the fork/join variables take values $+/+$ or $-/-$. The dependencies for these two cases are shown in a graphical model in Figure 1.

First the fork model P_F is assumed to be independent of previous state s_{t-1} variables except for the previous awaited category b_{t-1} and POS tag p_{t-1}:

$$\mathsf{P}_F(f_t | s_{t\text{–}1}) \stackrel{\text{def}}{=} \mathsf{P}_{F'}(f_t | b_{t\text{–}1}, p_{t\text{–}1}) \tag{2}$$

This models whether the POS tag $p_{t\text{–}1}$ just seen can end the awaited variable $b_{t\text{–}1}$ (f_t=$-$) or whether it will require another fork (f_t=$+$).

Then the join model P_J is decomposed into P_{JF+} and P_{JF-} depending on the outcomes of the F variable:

$$\mathsf{P}_J(j_t | f_t, s_{t\text{–}1}) \stackrel{\text{def}}{=} \begin{cases} \mathsf{P}_{JF+}(j_t | b_{t\text{–}1}, p_{t\text{–}1}), & \text{if } f_t{=}+ \\ \mathsf{P}_{JF-}(j_t | a_{t\text{–}1}), & \text{if } f_t{=}- \end{cases} \tag{3}$$

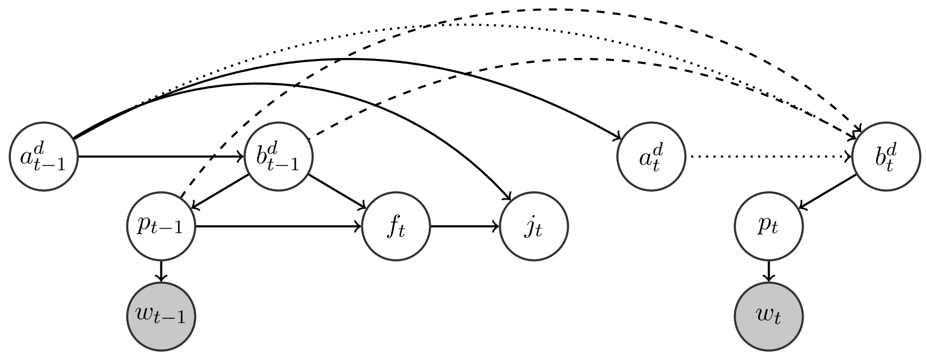

Figure 1: Two time steps of the HHMM model. s_t=All hidden variables at time t, a = Active category, b = Awaited category, f = Fork node, j=Join node, p = Part of speech tag, w = word. Dashed lines indicate +/+ dependencies, dotted lines indicate -/- dependencies. Outward links from f and j omitted for clarity.

When f_t=+, that is, a fork has been created, the decision of j is whether to transition the awaited category (j=+) or create a new stack level (j=−). When f_t=−, that is, the fork has not been created, the decision of j is whether to reduce a stack level (j_t=+) or to transition both the active and awaited categories (j_t=−). In a depth-one learner, both cases are deterministic given f_t since we will know whether we are at the first or last word in the sentence.

The active model P_A is decomposed into $\mathsf{P}_{A^{F-J-}}$ and $\mathsf{P}_{A^{F+J-}}$ depending on both the previous state $s_{t\text{-}1}$ and the current fork and join variables f_t and j_t:[1]

$$\mathsf{P}_A(a_t|f_t, j_t, s_{t\text{-}1}) \stackrel{\text{def}}{=} \begin{cases} [\![a_t = a_{t\text{-}1}]\!], & \text{if } f_t{=}{+}, j_t{=}{+} \\ \mathsf{P}_{A^{F-J-}}(a_t|a_{t\text{-}1}), & \text{if } f_t{=}{-}, j_t{=}{-} \\ \mathsf{P}_{A^{F+J-}}(a_t|a_{t\text{-}1}, p_{t\text{-}1}), & \text{if } f_t{=}{+}, j_t{=}{-} \end{cases} \quad (4)$$

With a $+/+$ transition, the active variable is not allowed to change (first case). For example, if the previous syntactic state is S/VP, and a transitive verb POS tag is hypothesized, the active variable will remain the same (S) as the VP is not yet able to completely reduce without seeing its object argument. When there is no fork and no reduce ($-/-$), a new active variable is selected with the A^{F-J-} model. For example, when an NP/NN generates a noun POS, which can end the NP ($-/-$), the next time step might be an S/VP (having completed a sentence-starting noun phrase the state is with some probability a sentence lacking only a verb phrase). In the fork/no join case ($+/-$), at the start of the sentence, we create a new active variable with the A^{F+J-} model. In the depth one version of the model the dependency on the $a_{t\text{-}1}$ is unnecessary and in fact the only dependency is the POS tag of the first word ($p_{t\text{-}1}$).

The awaited model P_B depends on the outcome of the join variable j_t:

$$\mathsf{P}_B(b_t|f_t, j_t, a_t, s_{t\text{-}1}) \stackrel{\text{def}}{=} \begin{cases} \mathsf{P}_{B^{J+}}(b_t|b_{t\text{-}1}, p_{t\text{-}1}), & j_t{=}{+} \\ \mathsf{P}_{B^{J-}}(b_t|a_{t\text{-}1}, a_t) & j_t{=}{-} \end{cases} \quad (5)$$

The B^{J+} model is used when a join occurs (usually $+/+$), meaning that the active variable has not changed. To continue the example from above, if the previous state is S/VP and a transitive verb POS is hypothesized, the VP $+/+$ transition will occur, and the B^{J+} model will merge the VP and transitive verb to create a new state of S/NP (a sentence lacking an object noun phrase).

The B^{J-} model is used when a new awaited variable must be generated from the new active value. For example, if the previous state was NP/NN and a noun is encountered, it can complete the noun phrase, but the $-/-$ transition followed by the A^{F+J-} model application generates an active value of S. In this case, the B^{J-} model generates likely completions of a sentence given the current active value S and the recently completed active constituent NP.

The part-of-speech p_t only depends on the

[1] Here $[\![\phi]\!]$ is an indicator function, equal to one when ϕ is true and zero otherwise.

awaited (b_t) category at the same time step:

$$\mathsf{P}_P(p_t|f_t, j_t, a_t, b_t, s_{t-1}) \stackrel{\text{def}}{=} \mathsf{P}_{P'}(p_t|b_t) \quad (6)$$

Finally, the lexical item (w_t) only depends on the part of speech tag (p_t) at the same time step:

$$\mathsf{P}_W(w_t|s_t) = \mathsf{P}_{W'}(w_t|p_t) \quad (7)$$

3.2 Model priors

To define priors over the syntactic models we build on the infinite hidden Markov model (iHMM) used for part of speech tagging (van Gael et al., 2009). In that model, a hierarchical Dirichlet process HMM (Teh et al., 2006) is used to allow the observed number of states—corresponding to parts of speech—in the HMM to grow as the data requires. The hierarchical structure of the iHMM ensures that transition distributions share the same set of states, which would not be possible if we used a flat infinite mixture model.

In our model, we use nonparametric priors on *each* of the active, awaited, and part-of-speech variables, allowing the cardinality of each of these variables to grow as the data requires. In each case, we first draw a base distribution from a root Dirichlet process; we then use that base distribution as a parameter to an infinite set of Dirichlet processes, one each for each applicable combination of the conditioning variables a_{t-1}, b_{t-1}, p_{t-1}, j_t, f_t, a_t, and b_t:

$$\begin{aligned} \beta_A &\sim GEM(\gamma_A) \\ \mathsf{P}_{A^{F-J-}}(a_t|a_{t-1}) &\sim DP(\alpha_A, \beta_A) \\ \mathsf{P}_{A^{F+J-}}(a_t|a_{t-1}, p_{t-1}) &\sim DP(\alpha_A, \beta_A) \end{aligned}$$

$$\begin{aligned} \beta_B &\sim GEM(\gamma_B) \\ \mathsf{P}_{B^{J+}}(b_t|b_{t-1}, p_{t-1}) &\sim DP(\alpha_B, \beta_B) \\ \mathsf{P}_{B^{J-}}(b_t|a_{t-1}, a_t) &\sim DP(\alpha_B, \beta_B) \end{aligned}$$

$$\begin{aligned} \beta_P &\sim GEM(\gamma_P) \\ \mathsf{P}_{P'}(p_t|b_t) &\sim DP(\alpha_P, \beta_P) \end{aligned}$$

Where DP is Dirichlet process and GEM is the stick-breaking construction for DPs (Sethuraman, 1994).

3.3 Inference

We base our inference process on the beam sampling approach employed in van Gael et al. (2009) for part-of-speech induction. This inference approach alternates between two phases in each iteration. First, given the distributions P_F, P_J, P_A, P_B, P_P, and P_W, we resample values for all the hidden states $\{s_t\}$. Next, given the state values $\{s_t\}$, we resample each set of multinomial distributions P_F, P_J, P_A, P_B, P_P, and P_W.

We initialize the sampler by conservatively setting the cardinalities of the number of active, awaited, and part-of-speech states we expect to see in the data set, randomly initializing the state space, and then sampling the parameters for each distribution P_F, P_J, P_A, P_B, P_P, and P_W given the randomly initialized states and fixed hyperparameters (specified in Section 4).

As noted by van Gael et al. (2008), token-level Gibbs sampling in a sequence model can be slow to mix. In our preliminary work, we found that mixing with token-level Gibbs sampling is even slower in our model due to the tight constraints imposed by the switching variables—it is technically ergodic but exploring the state space requires many low probability moves. Therefore, we use sentence-level sampling instead of token-level sampling, first computing forward probabilities for the sequence and then doing sampling in a backwards pass; resampling the parameters for the probability distributions only requires computing the counts from the sampled sequence and combining with the hyperparameters. To account for the infinite size of the state spaces, we employ the beam sampler (van Gael et al., 2008), with some modifications for computational speed.

The standard beam sampler introduces an auxiliary variable u at each time step, which acts as a threshold below which transition probabilities are ignored. This auxiliary variable u is drawn from *Uniform*$(0, p(s_t|s_{t-1}))$, so it will be between 0 and the probability of the previously sampled transition. The joint distribution over transitions, emissions, and auxiliary variables can be reduced so that the transition matrix is transformed into a boolean matrix with a 1 indicating an allowed transition. Depending on the cut-off value u, the size of the instantiated transition matrix will be different for every time-step.

In our model, we must sample values of u for active, awaited, and POS variables at every time step, rather than a single u for the transition matrix. It is possible to compile all the operations at each time step into a single large transition matrix, but computing this matrix is prohibitively slow for an operation that must be done at each time step in the data.

To address this issue, we interleave several iterations holding the cardinality of the instantiated space fixed and with full beam-sampling steps in which the cardinality of the state space can change. When the cardinality of the state space is fixed, we can multiply out the states into one large, structured transition matrix that is valid for all time steps. Our forward pass is thus reduced to an HMM forward pass (albeit one over a much larger set of states), vastly improving the speed of inference. Alternating between sampling the parameters of this matrix and the state values themselves corresponds to updating a finite portion of the infinite possible state space; by interleaving these finite steps with occasional full beam-sampling iterations, we are still properly exploring the posterior over models.

3.4 Parsing

There are multiple ways to extract parses from an unsupervised grammar induction system such as this. The optimal Bayesian approach would involve averaging over the values sampled for each model across many iterations, and then use those models in a Viterbi decoding parser to find the best parse for each sentence. Alternatively, if the model parameters have ceased to change much between iterations, we can assume that we have found a local optimum. We can then use a single sample from the end of the run as our model and the analyses of each sentence in that run as the parses to be evaluated.

4 Evaluation

Following Klein (2005), Seginer (2007) and Ponvert et al. (2011), we evaluate our induced grammars on standard induction tasks in three languages; the WSJ-10 unlabeled bracketing task (English), the NEGRA-10 task (German), and the CTB-10 task (Mandarin). These tasks examine the extent to which a parser run using an induced grammar correctly identifies constituent spans (disregarding span labels) in the subset of Wall Street Journal Penn Treebank (Marcus et al., 1994), NEGRA Treebank (Skut et al., 1997), and Chinese Treebank (Xia et al., 2000) sentences containing no more than ten words. In order to approximate human-like language learning, following Klein and others, we evaluate on text without punctuation and without part-of-speech annotations.

We also observe that many grammar induction models (in particluar, Seginer and Ponvert et al, mentioned above) exceed a right branching baseline only on the basis of precision, essentially only by predicting annotators' decisions not to include certain binary projections as corpus annotations. Since we feel these annotator decisions were primarily driven by considerations of annotation speed and are of questionable linguistic value for downstream applications, and since a purely binary-branching tree structure with no unary branches naturally constrains precision to be no higher than recall, we focus our evaluation exclusively on recall.

We train our unsupervised hierarchical hidden Markov model (UHHMM) on the 7422 unlabeled sentences of WSJ-10, 7536 sentences of NEGRA-10 and 4624 sentences of CTB-10. We ran the model on these corpora for 4000 iterations with 10 nodes with Intel Xeon x5650 CPUs, 120 cores in total. Each training process took 4 days to complete.

We conservatively initialize the number of active categories $|A| = 10$, the number of awaited categories $|B| = 10$, and the number of POS categories $|P| = 15$. The values for α_A, α_B, and α_P were each set to 0.5, while the values for α_F and α_J were each set to 0.1. The value of γ was set to 0.75.

4.1 Model Performance and Convergence

During training we used the joint log likelihood over the entire model as the metric to test for convergence. Figure 2 shows the fluctuation of the log probabilities for the WSJ-10 dataset, averaged over a window of 100 iterations. The log probabilities converge at around 2000 iterations.

The recall curve in Figure 3 shows how recall of UHHMM on gold WSJ-10 brackets changes through iterations. For comparison with other models, we compute recall using the model at iteration 3610, which scored closest to peak in averaged log probability after the log probability appeared to converge. Like the log probability plot, the recall curve also steadily improves over several iterations and converges at about the same place. Having established this method on English, we similarly tested for convergence on NEGRA and

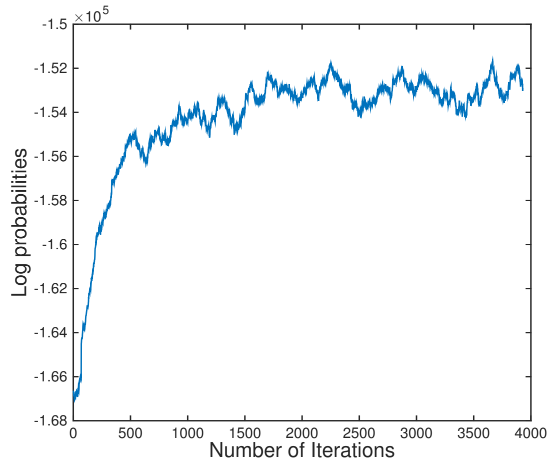

Figure 2: Log probabilities of the WSJ-10 dataset from UHHMM at each iteration, averaged over a window of 100 iterations.

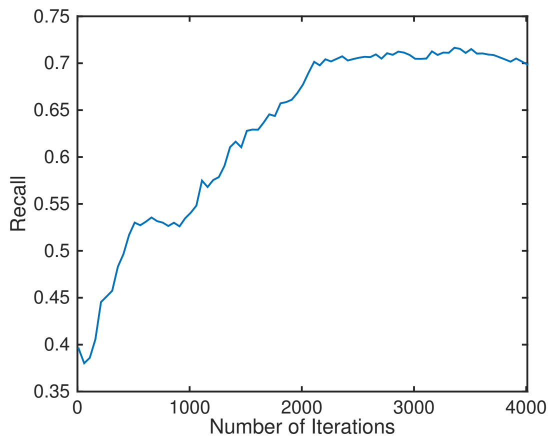

Figure 3: Recall curve of UHHMM on WSJ-10.

CTB by evaluating parser performance at the same iteration at which peak log probability occurred.

4.2 Baseline approaches

While much of the work on grammar induction uses gold POS tags as the basic units in the sequences to induce grammar, the technique proposed in this work induces syntactic structures directly from raw text. We compare our proposed approach with four existing grammar induction techniques that also operate directly on the raw text, as well as two more competitive baselines make use of a priori POS tags. We report results on three datasets in three different languages: Penn Treebank for English, NEGRA for German and Chinese Treebank for Chinese.

We compare against four existing techniques that similarly train on raw text, but which use more training data than our system uses. The common cover link (CCL No Punc) model of Seginer (2007) was trained on the entire WSJ Penn Treebank (with punctuation removed), NEGRA corpus and Chinese Treebank respectively, including those sentences with more than ten words. The probabilistic right linear grammar model (PRLG) and hidden Markov model (HMM) of Ponvert et al. (2011) were trained on WSJ sections 00-22, first 18602 sentences of NEGRA which is almost 90% of the whole corpus and 85% of CTB respectively, but with punctuation retained. The right-branching model is a deterministic baseline where all sentences in all three corpora are bracketed as if they were all purely right-branching.

For completeness, we also compare against two more competitive baselines that do make use of a priori POS tags. CCM is the chart-based grammar induction model from Klein and Manning (2002), trained and tested on the WSJ-10 with induced POS tags. DMV+CCM (DISTR.) is a model proposed by Klein and Manning (2004) and Klein (2005), where a joint model of the constituent context model and the dependency model with valence is used to train and induce structures, also using WSJ-10 and NEGRA-10 sentences with automatically induced POS tags.

WSJ-10 Model	**Training set size**	**Unlabeled precision**	**Unlabeled recall**
HMM	45.4k	64.4	64.7
CCL No Punc	49.2k	68.7	65.5
PRLG	45.4k	74.6	66.7
Right-branching	-	55.2	70.0
UHHMM (this work)	7.4k	56.2	**71.0**
CCM (Induced)	*7.4k*	*56.8*	*71.1*
DMV+CCM (DISTR.)	*7.4k*	*65.2*	*82.8*

Table 1: Unlabeled bracketing evaluation results of different unsupervised algorithms for the WSJ-10 dataset. The results of CCM and DMV+CCM are italicized as a reminder that they each use induced POS tags instead of raw text for grammar induction. Our model (UHHMM) has the highest recall of all the models that trained only on raw text, some of which use substantially larger training sets.

4.3 Results

In Table 1, we report unlabeled bracketing recall on the WSJ-10 dataset. The results for German

NEGRA-10 Model	Unlabeled precision	Unlabeled recall
HMM	47.7	72.0
CCL No Punc	39.8	61.2
PRLG	56.3	72.1
Right-branching	33.9	60.1
UHHMM (this work)	41.8	**72.4**
DMV+CCM (DISTR.)	*49.6*	*89.7*

Table 2: Unlabeled bracketing evaluation results of different unsupervised algorithms for the NEGRA-10 dataset. The results of DMV+CCM are italicized as a reminder that they each use induced POS tags. Our model (UHHMM) has the highest recall of all the models that trained only on raw text, all of which use the whole or a large part of the NEGRA corpus as the training set.

CTB-10 Model	Unlabeled precision	Unlabeled recall
HMM	55.8	53.1
CCL No Punc	48.5	47.8
PRLG	62.7	56.9
Right-branching	43.3	60.4
UHHMM (this work)	24.2	**33.0**

Table 3: Unlabeled bracketing evaluation results of different unsupervised algorithms for the CTB-10 dataset.

and Chinese are in Table 2 and Table 3. We observe state-of-the-art recall performance by our system on English and German compared with other recent induction systems which also do not require a priori POS tags, some of which are trained on substantially larger training sets. For English and German we also observe higher recall results by our system than the pure right-branching baseline, indicating that our system is not simply relying on sequences of $+/+$ operations. For Chinese, our system performs worse than the comparable systems and the right-branching baseline.

4.4 Analysis

The UHHMM did well on both the WSJ-10 and NEGRA-10 datasets, correctly learning mostly right-branching structures with no supervision, and obtaining the highest recall among comparable systems and the right-branching baselines. However, the UHHMM performed surprising poorly on the CTB-10 dataset. This poor performance on CTB-10 may be attributable to the relative lack of common function words in Chinese. Inspection of the UHHMM model output on English and German indicates that it quickly assigns characteristic part-of-speech categories to function words like determiners and copulas, which then presumably constrain the remaining categories. With no such function words in Chinese, and a relatively small training set size, the model may be facing severe sparse data problems. We therefore anticipate that a larger training set would substantially boost performance on highly analytical languages like Chinese.

Another source of error in all three datasets is that our model is constrained to posit tree structures that require no more than a single level of left-corner recursive depth (recall §3). By examining the WSJ-10 data set, we observe that all but 26 sentences in this 7422 sentence corpus comply with this depth restriction. In order to be directly comparable to previous results, we evaluate on the full WSJ-10 data set, even though our system is guaranteed to mislabel portions of these 26 more deeply recursive sentences. The NEGRA-10 and CTB-10 datasets are similarly predominantly depth-one.

Future work will look to extend the model described here to greater depths by using the learned models we have described here as priors to distributions at greater depths. The results we have obtained with a depth one system on shorter sentences are good evidence that the problem is learnable in this model, and we are therefore encouraged that this approach will be likely to succeed.

5 Conclusion

This paper has presented a grammar induction model based on a highly constrained version of a memory-bounded left-corner parsing strategy, which is able to achieve parsing performance for an induced grammar that is comparable to existing models that are not similarly cognitively constrained. The fact that an induction model can achieve competitive results on an existing grammar induction task despite very restrictive memory constraints is reassuring and suggests that these kinds of memory constraints may be exploited in human language acquisition.

The system and instructions for replicating our evaluation setup are available on github.[2]

[2] `http://anonymous.url`

References

Steven P. Abney and Mark Johnson. 1991. Memory requirements and local ambiguities of parsing strategies. *Journal of Psycholinguistic Research*, 20(3):233–250.

Alfred V Aho and Jeffery D Ullman. 1972. *The Theory of Parsing, Translation and Compiling; Volume. I: Parsing*. Prentice-Hall.

Noam Chomsky and George A Miller. 1963. Introduction to the formal analysis of natural languages. In *Handbook of Mathematical Psychology*, pages 269–321. Wiley.

Edward Gibson. 1991. *A computational theory of human linguistic processing: Memory limitations and processing breakdown*. Ph.D. thesis.

James Henderson. 2004. Lookahead in deterministic left-corner parsing. In *Proceedings of the workshop on incremental parsing: Bringing engineering and cognition together*, pages 26–33. Association for Computational Linguistics.

Philip N Johnson-Laird. 1983. *Mental models: towards a cognitive science of language, inference, and consciousness*. Harvard University Press.

Mark Johnson. 2007. Why doesn't EM find good HMM POS-taggers. *Proceedings of the 2007 Joint Conference on* , (June):296–305.

Fred Karlsson. 2007. Constraints on multiple center-embedding of clauses. *Journal of Linguistics*, 43:365–392.

Dan Klein and Christopher D Manning. 2002. A generative constituent-context model for improved grammar induction. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*.

Dan Klein and Christopher D Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*.

Dan Klein. 2005. *The unsupervised learning of natural language structure*. Ph.D. thesis, Stanford University.

Richard L Lewis and Shravan Vasishth. 2005. An activation-based model of sentence processing as skilled memory retrieval. *Cognitive Science*, 29(3):375–419.

Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyreand Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The Penn TreeBank: Annotating predicate argument structure. In *Proceedings of the ARPA Human Language Technology Workshop*.

Elias Ponvert, Jason Baldridge, and Katrin Erk. 2011. Simple unsupervised grammar induction from raw text with cascaded finite state models. *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, (1999):1077–1086.

William Schuler, Samir AbdelRahman, Tim Miller, and Lane Schwartz. 2010. Broad-coverage incremental parsing using human-like memory constraints. *Computational Linguistics*, 36(1):1–30.

Yoav Seginer. 2007. *Learning Syntactic Structure*. Ph.D. thesis, University of Amsterdam.

Jayaram Sethuraman. 1994. A constructive definition of dirichlet priors. *Statistica Sinica*, 4:639–650.

Wojciech Skut, Brigitte Krenn, Thorsten Brants, and Hans Uszkoreit. 1997. An annotation scheme for free word order languages. In *Proceedings of the Fifth Conference on Applied Natural Language Processing {ANLP}-97*.

Y W Teh, M I Jordan, M J Beal, and D M Blei. 2006. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581.

Jurgen van Gael, Yunus Saatci, Yee Whye Teh, and Zoubin Ghahramani. 2008. Beam sampling for the infinite hidden Markov model. pages 1–8.

Jurgen van Gael, Andreas Vlachos, and Zoubin Ghahramani. 2009. The infinite HMM for unsupervised PoS tagging. (August):678–687.

Marten van Schijndel, Andy Exley, and William Schuler. 2013. A model of language processing as hierarchic sequential prediction. *Topics in Cognitive Science*, 5(3):522–540.

F Xia, M Palmer, N Xue, and ME Okurowski. 2000. Developing Guidelines and Ensuring Consistency for Chinese Text Annotation. *LREC*.