Unsupervised Recurrent Neural Network Grammars

Yoon Kim† Alexander M. Rush† Lei Yu
Adhiguna Kuncoro‡ Chris Dyer‡ Gábor Melis‡

†Harvard University ‡University of Oxford ‡DeepMind
{yoonkim, srush}@seas.harvard.edu
{leiyu, akuncoro, cdyer, melisgl}@google.com

Abstract

Recurrent neural network grammars (RNNGs) are generative models of language which jointly model syntax and surface structure by incrementally generating a syntax tree and sentence in a top-down, left-to-right order. Supervised RNNGs achieve strong language modeling and parsing performance, but require an annotated corpus of parse trees. In this work, we experiment with unsupervised learning of RNNGs. Since directly marginalizing over the space of latent trees is intractable, we instead apply amortized variational inference. To maximize the evidence lower bound, we develop an inference network parameterized as a neural CRF constituency parser. On language modeling, unsupervised RNNGs perform as well their supervised counterparts on benchmarks in English and Chinese. On constituency grammar induction, they are competitive with recent neural language models that induce tree structures from words through attention mechanisms.

1 Introduction

Recurrent neural network grammars (RNNGs) (Dyer et al., 2016) model sentences by first generating a nested, hierarchical syntactic structure which is used to construct a context representation to be conditioned upon for upcoming words. Supervised RNNGs have been shown to outperform standard sequential language models, achieve excellent results on parsing (Dyer et al., 2016; Kuncoro et al., 2017), better encode syntactic properties of language (Kuncoro et al., 2018), and correlate with electrophysiological responses in the human brain (Hale et al., 2018). However, these all require annotated syntactic trees for training. In this work, we explore unsupervised learning of recurrent neural network grammars for language modeling and grammar induction.

The standard setup for unsupervised structure learning is to define a generative model \( p_\theta(x, z) \) over observed data \( x \) (e.g. sentence) and unobserved structure \( z \) (e.g. parse tree, part-of-speech sequence), and maximize the log marginal likelihood \( \log p_\theta(x) = \log \sum_z p_\theta(x, z) \). Successful approaches to unsupervised parsing have made strong conditional independence assumptions (e.g. context-freeness) and employed auxiliary objectives (Klein and Manning, 2002) or priors (Johnson et al., 2007). These strategies imbue the learning process with inductive biases that guide the model to discover meaningful structures while allowing tractable algorithms for marginalization; however, they come at the expense of language modeling performance, particularly compared to sequential neural models that make no independence assumptions.

Like RNN language models, RNNGs make no independence assumptions. Instead they encode structural bias through operations that compose linguistic constituents. The lack of independence assumptions contributes to the strong language modeling performance of RNNGs, but make unsupervised learning challenging. First, marginalization is intractable. Second, the biases imposed by the RNNG are relatively weak compared to those imposed by models like PCFGs. There is little pressure for non-trivial tree structure to emerge during unsupervised RNNG (URNNG) learning.

In this work, we explore a technique for handling intractable marginalization while also injecting inductive bias. Specifically we employ amortized variational inference (Kingma and Welling, 2014; Rezende et al., 2014; Mnih and Gregor, 2014) with a \textit{structured} inference network. Variational inference lets us tractably optimize a lower bound on the log marginal likelihood, while employing a structured inference network encourages non-trivial structure. In particular, a con-
ditional random field (CRF) constituency parser (Finkel et al., 2008; Durrett and Klein, 2015), which makes significant independence assumptions, acts as a guide on the generative model to learn meaningful trees through regularizing the posterior (Ganchev et al., 2010).

We experiment with URNNs on English and Chinese and observe that they perform well as language models compared to their supervised counterparts and standard neural LMs. In terms of grammar induction, they are competitive with recently-proposed neural architectures that discover tree-like structures through gated attention (Shen et al., 2018). Our results, along with other recent work on joint language modeling/structure learning with deep networks (Shen et al., 2018, 2019; Wiseman et al., 2018; Kawakami et al., 2018), suggest that it is possible learn generative models of language that model the underlying data well (i.e. assign high likelihood to held-out data) and at the same time induce meaningful linguistic structure.

2 Unsupervised Recurrent Neural Network Grammars

We use \( x = [x_1, \ldots, x_T] \) to denote a sentence of length \( T \), and \( z \in \mathcal{Z}_T \) to denote an unlabeled binary parse tree over a sequence of length \( T \), represented as a a binary vector of length \( 2T - 1 \). Here 0 and 1 correspond to shift and reduce actions, explained below.1 Figure 1 presents an overview of our approach.

2.1 Generative Model

An RNNG defines a joint probability distribution \( p_\theta(x, z) \) over sentences \( x \) and parse trees \( z \). We consider a simplified version of the original RNNG (Dyer et al., 2016) by ignoring constituent labels and only considering binary trees. The RNNG utilizes an RNN to parameterize a stack data structure (Dyer et al., 2015) of partially-completed constituents to incrementally build the parse tree while generating terminals. Using the current stack representation, the model samples an action (shift or reduce): shift generates a terminal symbol, i.e. word, and shifts it onto the stack,2 reduce pops the last two elements off the stack, composes them, and shifts the composed

\[
\text{Inference Network } q_\phi(z | x) \quad \text{Generative Model } p_\theta(x, z)
\]

Figure 1: Overview of our approach. The inference network \( q_\phi(z | x) \) (left) is a CRF parser which produces a distribution over binary trees (shown in dotted box). \( B_{ij} \) are random variables for existence of a constituent spanning \( i \)-th and \( j \)-th words, whose potentials are the output from a bidirectional LSTM (the global factor ensures that the distribution is only over valid binary trees). The generative model \( p_\theta(x, z) \) (right) is an RNNG which consists of a stack LSTM (from which actions/words are predicted) and a tree LSTM (to obtain constituent representations upon reduce). Training involves sampling a binary tree from \( q_\phi(z | x) \), converting it to a sequence of shift/reduce actions, and optimizing the log joint likelihood \( \log p_\theta(x, z) \) representation onto the stack.

Formally, let \( S = [[0, 0]] \) be the initial stack. Each item of the stack will be a pair, where the first element is the hidden state of the stack LSTM, and the second element is an input vector, described below. We use \( \top(S) \) to refer to the top pair in the stack. The push and pop operations are defined imperatively in the usual way. At each time step, the next action \( z_t \) (shift or reduce) is sampled from a Bernoulli distribution parameterized in terms of the current stack representation. Letting \( (h_{\text{prev}}, g_{\text{prev}}) = \top(S) \), we have

\[
z_t \sim \text{Bernoulli}(p_t), \quad p_t = \sigma(w^T h_{\text{prev}} + b).
\]

Subsequent generation depend on \( z_t \):

- If \( z_t = 0 \) (shift), the model first generates a terminal symbol via sampling from a categorical distribution whose parameters come from an affine transformation and a softmax,

\[
x \sim \text{softmax}(W h_{\text{prev}} + b).
\]

Then the generated terminal is shifted onto the stack using a stack LSTM,

\[
h_{\text{next}} = \text{LSTM}(e_x, h_{\text{prev}}),
\]

\[
\text{push}(S, (h_{\text{next}}, e_x)),
\]

where \( e_x \) is the word embedding for \( x \).

---

1The cardinality of \( \mathcal{Z}_T \subset \{0, 1\}^{2T-1} \) is given by the \((T - 1)\)-th Catalan number, \(|\mathcal{Z}_T| = \frac{\binom{2T-2}{T-1}}{T-1}\).

2A better name for shift would be generate (as in Dyer et al. (2016)), but we use shift to emphasize similarity with the shift-reduce parsing.
The joint log likelihood decomposes as a sum of
sentence symbol is generated. The parameters
the generation process continues until an end-of-
(2015) for the exact parameterization.
both) of the inputs to the tree LSTM is a word embedding,
log brevity, from here on we will use
Thus, the support of \( z_t \) is \( \{0, 1\}^{2T - 1} \), all binary vectors of length \( 2T - 1 \). To restrict our distribution to \( \mathcal{Z}_T \) (binary vectors which describe valid trees), we constrain \( z_t \) to be valid at each time step, which amounts to deterministically choosing \( z_t = 0 \) (SHIFT) if there are fewer than two elements (not counting the initial zero tuple) on the stack.

The action log likelihood is the sum of log conditional priors, which is obviously different from the unconditional log prior \( \log p_\theta(z) = \log \sum_x p_\theta(x, z) \).

In the supervised case where ground-truth \( z \) is available, we can straightforwardly perform gradient-based optimization to maximize the joint log likelihood \( \log p_\theta(x, z) \). In the unsupervised case, the standard approach is to maximize the log marginal likelihood,

\[
\log p_\theta(x) = \log \sum_{z' \in \mathcal{Z}_T} p_\theta(x, z').
\]

However this summation is intractable because \( z_t \) fully depends on all previous actions \( [z_1, \ldots, z_{t-1}] \). Even if this summation were tractable, it is not clear that meaningful latent structures would emerge given the lack of explicit independence assumptions in the RNNG (e.g. it is clearly not context-free). We handle these issues with amortized variational inference.

### 2.2 Amortized Variational Inference

Amortized variational inference (Kingma and Welling, 2014) defines a trainable inference network \( \phi \) that parameterizes \( q_\phi(z | x) \), a variational posterior distribution, in this case over parse trees \( z \) given the sentence \( x \). This distribution is used to form an evidence lower bound (ELBO) on the log marginal likelihood,

\[
\text{ELBO}(\theta, \phi; x) = \mathbb{E}_{q_\phi(z | x)} \left[ \log \frac{p_\theta(x, z)}{q_\phi(z | x)} \right].
\]

We maximize the ELBO with respect to both model parameters \( \theta \) and inference network parameters \( \phi \). The ELBO is still intractable to calculate exactly, but this formulation will allow us to obtain unbiased gradient estimators based on Monte Carlo sampling.

Observe that rearranging the ELBO gives the following optimization problem,

\[
\max_{\theta, \phi} \log p_\theta(x) - \text{KL}[q_\phi(z | x) \| p_\theta(z | x)].
\]

Thus, \( \phi \) is trained to match the variational posterior \( q_\phi(z | x) \) to the true posterior \( p_\theta(z | x) \), but \( \theta \) is also trained to match the true posterior to the variational posterior. Indeed, there is some evidence to suggest that generative models trained with amortized variational inference (i.e. variational autoencoders) learn posterior distributions that are close to the variational family (Cremer et al., 2018).

We can use this to our advantage with an inference network that injects inductive bias. We propose to do this by using a context-free model for the inference network, in particular, a neural CRF parser (Durrett and Klein, 2015). This choice
can seen as a form of posterior regularization that limits posterior flexibility of the overly powerful RNNG generative model.6,7

The parameterization of span scores is similar to recent works (Wang and Chang, 2016; Stern et al., 2017; Kitaev and Klein, 2018): we add position embeddings to word embeddings and run a bidirectional LSTM over the input representations to obtain the forward \([\hat{h}_1, \ldots, \hat{h}_T]\) and backward \([\hat{h}_1, \ldots, \hat{h}_T]\) hidden states. The score \(s_{ij} \in \mathbb{R}\) for a constituent spanning \(x_i\) to \(x_j\) is given by,

\[
s_{ij} = \text{MLP}(\hat{h}_{j+1} - \hat{h}_i; \hat{h}_{i-1} - \hat{h}_j).
\]

Letting \(B\) be the binary matrix representation of a tree (\(B_{ij} = 1\) means there is a constituent spanning \(x_i\) and \(x_j\)), the CRF parser defines a distribution over binary trees via the Gibbs distribution,

\[
q_\phi(B | x) = \frac{1}{Z_T(x)} \exp \left( \sum_{i,j} B_{ij} s_{ij} \right),
\]

where \(Z_T(x)\) is the partition function,

\[
Z_T(x) = \sum_{B \in B_T} \exp \left( \sum_{i,j} B_{ij} s_{ij} \right),
\]

and \(\phi\) denotes the parameters of the inference network (i.e., the bidirectional LSTM and the MLP). Calculating \(Z_T(x)\) requires a summation over an exponentially-sized set \(B_T \subset \{0,1\}^{T \times T}\), the set of all binary trees over a length \(T\) sequence. However, we can perform the summation in \(O(T^3)\) using the inside algorithm (Baker, 1979), shown in

Algorithm 1. This computation is itself differentiable and amenable to gradient-based optimization. Finally, letting \(f : B_T \rightarrow Z_T\) be the bijection between the binary tree matrix representation and a sequence of \text{SHIFT/REDUCE} actions, the inference network defines a distribution over \(Z_T\) via \(q_\phi(z | x) \triangleq q_\phi(f^{-1}(z) | x)\).

2.3 Optimization

For optimization, we use the following variant of the ELBO,

\[
\mathbb{E}_{q_\phi(z | x)} [\log p_\theta(x, z)] + \mathbb{H}[q_\phi(z | x)],
\]

where \(\mathbb{H}[q_\phi(z | x)] = \mathbb{E}_{q_\phi(z | x)} [-\log q_\phi(z | x)]\) is the entropy of the variational posterior. A Monte Carlo estimate for the gradient with respect to \(\theta\) is

\[
\nabla_\theta \text{ELBO}(\theta, \phi; x) \approx \frac{1}{K} \sum_{k=1}^K \nabla_\theta \log p_\theta(x, z^{(k)}),
\]

with samples \(z^{(1)}, \ldots, z^{(K)}\) from \(q_\phi(z | x)\). Sampling uses the intermediate values calculated during the inside algorithm to sample split points recursively (Goodman, 1998; Finkel et al., 2006), as shown in Algorithm 2. The gradient with respect to \(\phi\) involves two parts. The entropy term \(\mathbb{H}[q_\phi(z | x)]\) can be calculated exactly in \(O(T^3)\), again using the intermediate values from the inside algorithm (see Algorithm 3).8

\[
\nabla_\phi \mathbb{E}_{q_\phi(z | x)} [\log p_\theta(x, z)] = \mathbb{E}_{q_\phi(z | x)} [\log p_\theta(x, z) \nabla_\phi \log q_\phi(z | x)] 
\]

\[
\approx \frac{1}{K} \sum_{k=1}^K \log p_\theta(x, z^{(k)}) \nabla_\phi \log q_\phi(z^{(k)} | x).
\]

6While it has a similar goal, this formulation differs from posterior regularization as formulated by Ganchev et al. (2010), which constrains the distributional family via linear constraints on posterior expectations. In our case, the conditional independence assumptions in the CRF lead to a \textit{curved} exponential family where the vector of natural parameters has fewer dimensions than the vector of sufficient statistics of the full exponential family. This curved exponential family is a subset of the marginal polytope of the full exponential family, but it is an intersection of both linear and nonlinear manifolds, and therefore cannot be characterized through linear constraints over posterior expectations.

7In preliminary experiments, we also attempted to learn latent trees with a transition-based parser (which does not make explicit independence assumptions) that looks at the entire sentence. However we found that under this setup, the inference network degenerated into a local minimum whereby it always generated left-branching trees despite various optimization strategies. Williams et al. (2018) observe a similar phenomenon in the context of learning latent trees for classification tasks. However Li et al. (2019) find that it is possible to use a transition-based parser as the inference network for dependency grammar induction, if the inference network is constrained via posterior regularization (Ganchev et al., 2010) based on universal syntactic rules (Naseem et al., 2010).

8We adapt the algorithm for calculating tree entropy in PCFGs from Hwa (2000) to the CRF case.

9\(\nabla_\theta \mathbb{H}[q_\phi(z | x)]\) can also be computed using the inside-outside algorithm and a second-order expectation semiring (Li and Eisner, 2009), which has the same asymptotic runtime complexity but generally better constants.
Algorithm 2: Top-down sampling a tree from $q_\phi(\mathbf{z} \mid \mathbf{x})$

1: \textbf{procedure} \textsc{Sample}$((\beta) \triangleright \beta$ from running \textsc{Inside}(s)\textbf{)}
2: \textbf{B} = 0 \triangleright \text{binary matrix representation of tree}
3: \textbf{Q} = [[1, T]] \triangleright \text{queue of constituents}
4: \textbf{while} \textbf{Q} is not empty \textbf{do}
5: \hspace{1em} (i, j) = \text{pop}(\textbf{Q})
6: \hspace{1em} \tau = \sum_{i=1}^{j-1} \beta[i, k] \cdot \beta[k + 1, j]
7: \hspace{1em} \textbf{for} k := i \text{ to } j - 1 \textbf{ do} \triangleright \text{get distribution over splits}
8: \hspace{2em} u_{ij} = (\beta[i, k] \cdot \beta[k + 1, j]) / \tau
9: \hspace{1em} k \sim \text{Catt}((u_{i, i+1}, u_{j, j-1})) \triangleright \text{sample a split point}
10: \hspace{1em} \textbf{B}_{i,k} = 1, \textbf{B}_{k+1,j} = 1 \triangleright \text{update B}
11: \hspace{1em} \textbf{if} k > i \textbf{ then} \triangleright \text{if left child has width } > 1
12: \hspace{2em} \text{push}(\textbf{Q}, (i, k)) \triangleright \text{add to queue}
13: \hspace{1em} \textbf{if} k + 1 < j \textbf{ then} \triangleright \text{if right child has width } > 1
14: \hspace{2em} \text{push}(\textbf{Q}, (k + 1, j)) \triangleright \text{add to queue}
15: \hspace{1em} \mathbf{z} = f(\mathbf{B}) \triangleright f : \mathcal{T} \rightarrow \mathcal{Z}_T \text{ maps matrix representation of tree to sequence of actions.}
16: \textbf{return} \mathbf{z}

The above estimator is unbiased but typically suffers from high variance. To reduce variance, we use a control variate derived from an average of the other samples’ joint likelihoods (Mnih and Rezende, 2016), yielding the following estimator,

$$
\frac{1}{K} \sum_{k=1}^{K} (\log p_{\theta}(\mathbf{x}, \mathbf{z}^{(k)}) - r^{(k)}) \nabla_{\phi} \log q_{\phi}(\mathbf{z}^{(k)} \mid \mathbf{x}),
$$

where \( r^{(k)} = \frac{1}{\pi_{T}} \sum_{j \neq k} \log p_{\theta}(\mathbf{x}, \mathbf{z}^{(j)}) \). This control variate worked better than alternatives such as estimates of baselines from an auxiliary network (Mnih and Gregor, 2014; Deng et al., 2018) or a language model (Yin et al., 2018).

3 Experimental Setup

3.1 Data

For English we use the Penn Treebank (Marcus et al., 1993, PTB) with splits and preprocessing from Dyer et al. (2016) which retains punctuation and replaces singleton words with Berkeley parser’s mapping rules, resulting in a vocabulary of 23,815 word types.\(^{10}\) Notably this is much larger than the standard PTB LM setup from Mikolov et al. (2010) which uses 10K types.\(^{11}\) Also different from the LM setup, we model each sentence separately instead of carrying information across sentence boundaries, as the RNNG is a generative model of sentences. Hence our perplexity numbers are not comparable to the PTB LM results (Melis et al., 2018; Merity et al., 2018; Yang et al., 2018).

Since the PTB is rather small, and since the URNN does not require annotation, we also test our approach on a subset of the one billion word corpus (Chelba et al., 2013). We randomly sample 1M sentences for training and 2K sentences for validation/test, and limit the vocabulary to 30K word types. While still a subset of the full corpus (which has 30M sentences), this dataset is two orders of magnitude larger than PTB. Experiments on Chinese utilize version 5.1 of the Chinese Penn Treebank (CTB) (Xue et al., 2005), with the same splits as in Chen and Manning (2014). Singleton words are replaced with a single (UNK) token, resulting in a vocabulary of 17,489 word types.

3.2 Training and Hyperparameters

The stack LSTM has two layers with input/hidden size equal to 650 and dropout of 0.5. The tree LSTM also has 650 units. The inference network uses a one-layer bidirectional LSTM with 256 hidden units, and the MLP (to produce span scores \( s_{ij} \) for \( i \leq j \)) has a single hidden layer with a ReLU nonlinearity followed by layer normalization (Ba et al., 2016) and dropout of 0.5. We share word embeddings between the generative model and the inference network, and also tie weights between the input/output word embeddings (Press and Wolf, 2016).

Optimization of the model itself required standard techniques for avoiding posterior collapse in VAEs.\(^{12}\) We warm-up the ELBO objective by linearly annealing (per batch) the weight on the conditional prior \( \log p_{\theta}(z \mid x) \) and the entropy \( \mathbb{H}[q_{\phi}(z \mid x)] \) from 0 to 1 over the first two epochs (see equation (1) for definition of \( \log p_{\theta}(z \mid x) \)). This is analogous to KL-annealing in VAEs with continuous latent variables (Bowman et al., 2016; Sønderby et al., 2016). We train for 18 epochs (enough for convergence for all models) with a batch size of 16 and \( K = 8 \) samples for the Monte Carlo gradient estimators. The generative model is optimized with SGD with learning rate equal to 1,\

\(^{10}\)https://github.com/clab/rnng

\(^{11}\)Both versions of the PTB data can be obtained from http://demo.clab.cs.cmu.edu/cdyer/ptb-lm.tar.gz.

\(^{12}\)Posterior collapse in our context means that \( q_{\phi}(z \mid x) \) always produced trivial (always left or right branching) trees.
except for the affine layer that produces a distribution over the actions, which has learning rate 0.1. Gradients of the generative model are clipped at 5. The inference network is optimized with Adam (Kingma and Ba, 2015) with learning rate 0.0001, $\beta_1 = 0.9, \beta_2 = 0.999$, and gradient clipping at 1. As Adam converges significantly faster than SGD (even with a much lower learning rate), we stop training the inference network after the first two epochs. Initial model parameters are sampled from $U[-0.1, 0.1]$. The learning rate starts decaying by a factor of 2 each epoch after the first epoch at which validation performance does not improve, but this learning rate decay is not triggered for the first eight epochs to ensure adequate training. We use the same hyperparameters/training setup for both PTB and CTB. For experiments on (the subset of) the one billion word corpus, we use a smaller dropout rate of 0.1. The baseline RNNLM also uses the smaller dropout rate.

All models are trained with an end-of-sentence token, but for perplexity calculation these tokens are not counted to be comparable to prior work (Dyer et al., 2016; Kuncoro et al., 2017; Buys and Blunsom, 2018). To be more precise, the inference network does not make use of the end-of-sentence token to produce parse trees, but the generative model is trained to generate the end-of-sentence token after the final REDUCE operation.

### 3.3 Baselines

We compare the unsupervised RNNG (URNNG) against several baselines: (1) RNNLM, a standard RNN language model whose size is the same as URNNG’s stack LSTM; (2) Parsing Reading Predict Network (PRPN) (Shen et al., 2018), a neural language model that uses gated attention layers to embed soft tree-like structures into a neural network (and among the current state-of-the-art in grammar induction from words on the full corpus); (3) RNNG with trivial trees (left branching, right branching, random); (4) supervised RNNG trained on unlabeled, binarized gold trees. Note that the supervised RNNG also trains a discriminative parser $q_\theta(z|x)$ (alongside the generative model $p_\theta(x,z)$) in order to sample parse forests for perplexity evaluation (i.e. importance sampling). This discriminative parser has the same architecture as URNNG’s inference network. For all models, we perform early stopping based on validation perplexity.

### 4 Results and Discussion

#### 4.1 Language Modeling

Table 1 shows perplexity for the different models on PTB/CTB. As a language model URNNG outperforms an RNNLM and is competitive with the supervised RNNG. The left branching baseline performs poorly, implying that the strong performance of URNNG/RNNG is not simply due to the additional depth afforded by the tree LSTM composition function (a left branching tree, which always performs REDUCE when possible, is the “deepest” model). The right branching baseline is essentially equivalent to an RNNLM and hence performs similarly. We found PRPN with default hyperparameters (which obtains a perplexity of 62.0 in the PTB setup from Mikolov et al. (2010)) to not perform well, but tuning hyperparameters improves performance. The supervised RNNG performs well as a language model, despite being trained on the joint (rather than marginal) likelihood objective. This indicates that explicit

| Model                | PTB PPL | F1 | CTB PPL | F1 |
|----------------------|---------|----|---------|----|
| RNNLM                | 93.2    | –  | 201.3   | –  |
| PRPN (default)       | 126.2   | 32.9 | 290.9  | 32.9  |
| PRPN (tuned)         | 96.7    | 41.2 | 216.0  | 36.1  |
| Left Branching Trees | 100.9   | 10.3 | 223.6  | 12.4  |
| Right Branching Trees| 93.3    | 34.8 | 203.5  | 20.6  |
| Random Trees         | 113.2   | 17.0 | 209.1  | 17.4  |
| URNNG                | 90.6    | 40.7 | 195.7  | 29.1  |
| RNNG → URNNG         | 88.7    | 68.1 | 193.1  | 52.3  |
| Oracle Binary Trees  | –       | 82.5 | –      | –     |
| PRPN (tuned)         | 95.9    | 67.7 | 181.1  | 51.9  |

Table 1: Language modeling perplexity (PPL) and grammar induction $F_1$ scores on English (PTB) and Chinese (CTB) for the different models. Note that our PTB setup from Dyer et al. (2016) differs considerably from the usual language modeling setup (Mikolov et al., 2010) since we model each sentence independently and use a much larger vocabulary (see §3.1).

15For RNNG and URNNG we estimate the log marginal likelihood (and hence, perplexity) with $K = 1000$ importance-weighted samples. During evaluation only, we also flatten $q_\phi(z|x)$ by dividing span scores $s_{ij}$ by a temperature term 2.0 before feeding it to the CRF.

16Using the code from https://github.com/yikangshen/PRPN, we tuned model size, initialization, dropout, learning rate, and use of batch normalization.
modeling of syntax helps generalization even with richly-parameterized neural models. Encouraged by these observations, we also experiment with a hybrid approach where we train a supervised RNNG first and continue fine-tuning the model (including the inference network) on the URNNG objective (RNNG → URNNG in Table 1).\textsuperscript{17} This approach results in nontrivial perplexity improvements, and suggests that it is potentially possible to improve language models with supervision on parsed data. In Figure 2 we show perplexity by sentence length. We find that a standard language model (RNNLM) is better at modeling short sentences, but underperforms models that explicitly take into account structure (RNNG/URNNG) when the sentence length is greater than 10. Table 2 (top) compares our results against prior work on this version of the PTB, and Table 2 (bottom) shows the results on a 1M sentence subset of the one billion word corpus, which is two orders of magnitude larger than PTB. On this larger dataset URNNG still improves upon the RNNLM. We also trained an RNNG (and RNNG → URNNG) on this dataset by parsing the training set with the self-attentive parser from Kitaev and Klein (2018).\textsuperscript{18} These models improve upon the RNNLM but not the URNNG, potentially highlighting the limitations of using predicted trees for supervising RNNGs.

### 4.2 Grammar Induction

Table 1 also shows the $F_1$ scores for grammar induction. Note that we induce latent trees directly from words on the full dataset.\textsuperscript{19} For RNNG/URNNG we obtain the highest scoring $F_1$ score of 95.17 on the PTB test set.

Figure 2: Perplexity of the different models grouped by sentence length on PTB.

| PTB          | PPL   |
|--------------|-------|
| KN 5-gram    | 169.3 |
| RNNLM (Dyer et al., 2016) | 113.4 |
| Original RNNG (Dyer et al., 2016) | 102.4 |
| Stack-only RNNG (Kuncoro et al., 2017) | 101.2 |
| Gated-Attention RNNG (Kuncoro et al., 2017) | 100.9 |
| Generative Dep. Parser (Buys and Blunsom, 2015) | 138.6 |
| RNNLM (Buys and Blunsom, 2018) | 100.7 |
| Sup. Syntactic NLM (Buys and Blunsom, 2018) | 107.6 |
| Unsup. Syntactic NLM (Buys and Blunsom, 2018) | 125.2 |
| PRPN\textsuperscript{†} (Shen et al., 2018) | 96.7 |
| This work:  |       |
| RNNLM       | 93.2  |
| URNNG       | 90.6  |
| RNNG        | 88.7  |
| RNNG → URNNG | 85.9  |

| 1M Sentences | PPL   |
|--------------|-------|
| PRPN\textsuperscript{†} (Shen et al., 2018) | 77.7 |
| RNNLM       | 77.4  |
| URNNG       | 71.8  |
| RNNG\textsuperscript{‡} | 72.9  |
| RNNG\textsuperscript{‡} → URNNG | 72.0  |

Table 2: (Top) Comparison of this work as a language model against prior works on sentence-level PTB with preprocessing from Dyer et al. (2016). Note that previous versions of RNNG differ from ours in terms of parameterization and model size. (Bottom) Results on a subset (1M sentences) of the one billion word corpus. PRPN\textsuperscript{†} is the model from Shen et al. (2018), whose hyperparameters were tuned by us. RNNG\textsuperscript{‡} is trained on predicted parse trees from Kitaev and Klein (2018).

We confirm the replication study of Htut et al. (2018) and find that PRPN is a strong model for grammar induction. URNNG performs on par with PRPN on English but PRPN does better on Chinese; both outperform right branching baselines. Table 3 further analyzes the learned trees and shows the $F_1$ score of URNNG trees against

\textsuperscript{17}We fine-tune for 10 epochs and use a smaller learning rate of 0.1 for the generative model.

\textsuperscript{18}To parse the training set we use the benepar\_en2 model from https://github.com/nikitakit/self-attentive-parser, which obtains an $F_1$ score of 95.17 on the PTB test set.

\textsuperscript{19}Past work on grammar induction usually train/evaluate on short sentences and also assume access to gold POS tags (Klein and Manning, 2002; Smith and Eisner, 2004; Bod, 2006). However more recent works do train directly words (Jin et al., 2018; Shen et al., 2018; Drozdzov et al., 2019).

\textsuperscript{20}Available at https://nlp.cs.nyu.edu/evalb/. We evaluate with COLLINS.prm parameter file and LABELED option equal to 0. We observe that the setup for grammar induction varies widely across different papers: lexicalized vs. unlexicalized; use of punctuation vs. not; separation of train/test sets; counting sentence-level spans for evaluation vs. ignoring them; use of additional data; length cutoff for training/evaluation; corpus-level $F_1$ vs. sentence-level $F_1$; and, more. In our survey of twenty or so papers, almost no two papers were identical in their setup. Such variation makes it difficult to meaningfully compare models across papers. Hence, we report grammar induction results mainly for the models and baselines considered in the present work.
There are several limitations to our approach. For one, the URNNG takes considerably more time/memory to train than a standard language model due to the $O(T^3)$ dynamic program in the inference network, multiple samples to obtain low-variance gradient estimators, and dynamic computation graphs that make efficient batching impossible.

4.4 Syntactic Evaluation

We perform a syntactic evaluation of the different models based on the setup from Marvin and Linzen (2018): the model is given two minimally different sentences, one grammatical and one ungrammatical, and must identify the grammatical sentence by assigning it higher probability. Table 6 shows the accuracy results. Overall the supervised RNNG significantly outperforms the other models, indicating opportunities for further work in unsupervised modeling. While the URNNG does slightly outperform an RNNLM, the distribution of errors made from both models are similar, and thus it is not clear whether the out-performance is simply due to better perplexity or learning different structural biases.

4.5 Limitations

There are several limitations to our approach. Firstly, the URNNG takes considerably more time/memory to train than a standard language model due to the $O(T^3)$ dynamic program in the inference network, multiple samples to obtain low-variance gradient estimators, and dynamic computation graphs that make efficient batching impossible.

21We modify the publicly available dataset from https://github.com/BeckyMarvin/LM_syneval to only keep sentence pairs that did not have any unknown words with respect to our vocabulary, resulting in 80K sentence pairs for evaluation.
trees. Our work is also related to the recent line of work on learning latent trees as part of a deep model through supervision on other tasks, typically via differentiable structured hidden layers (Kim et al., 2017; Bradbury and Socher, 2017; Liu and Lapata, 2018; Tran and Bisk, 2018; Peng et al., 2018; Niculae et al., 2018; Liu et al., 2018), policy gradient-based approaches (Yogatama et al., 2017; Williams et al., 2018; Havrylov et al., 2019), or differentiable relaxations (Choi et al., 2018; Maillard and Clark, 2018).

The variational approximation uses amortized inference (Kingma and Welling, 2014; Mnih and Gregor, 2014; Rezende et al., 2014), in which an inference network is used to obtain the variational posterior for each observed \( x \). Since our inference network is structured (i.e., a CRF), it is also related to CRF autoencoders (Ammar et al., 2014) and structured VAEs (Johnson et al., 2016; Krishnan et al., 2017), which have been used previously for unsupervised (Cai et al., 2017; Drozdov et al., 2019; Li et al., 2019) and semi-supervised (Yin et al., 2018; Corro and Titov, 2019) parsing.

### 6 Conclusion

It is an open question as to whether explicit modeling of syntax significantly helps neural models. Strubell et al. (2018) find that supervising intermediate attention layers with syntactic heads improves semantic role labeling, while Shi et al. (2018) observe that for text classification, syntactic trees only have marginal impact. Our work suggests that at least for language modeling, incorporating syntax either via explicit supervision or as latent variables does provide useful inductive biases and improves performance.

Finally, in modeling child language acquisition, the complex interaction of the parser and the grammatical knowledge being acquired is the object of much investigation (Trueswell and Gleitman, 2007); our work shows that apparently grammatical constraints can emerge from the interaction of a constrained parser and a more general grammar learner, which is an intriguing but underexplored hypothesis for explaining human linguistic biases.

### Acknowledgments

We thank the members of the DeepMind language team for helpful feedback. YK is supported by a Google Fellowship. AR is supported by NSF Career 1845664.
References

Roee Aharoni and Yoav Goldberg. 2017. Towards String-to-Tree Neural Machine Translation. In Proceedings of ACL.

David Alvarez-Melis and Tommi S. Jaakkola. 2017. Tree-structured Decoding with Doubly-Reccurrent Neural Networks. In Proceedings of ICLR.

Waleed Ammar, Chris Dyer, and Noah A. Smith. 2014. Conditional Random Field Autoencoders for Unsupervised Structured Prediction. In Proceedings of NIPS.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer Normalization. In Proceedings of NIPS.

James K. Baker. 1979. Trainable Grammars for Speech Recognition. In Proceedings of the Spring Conference of the Acoustical Society of America.

Rens Bod. 2006. An All-Subtrees Approach to Unsupervised Parsing. In Proceedings of ACL.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyal, Andrew M. Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating Sentences from a Continuous Space. In Proceedings of CoNLL.

Jiong Cai, Yong Jiang, and Kewei Tu. 2017. CRF Autoencoder for Unsupervised Dependency Parsing. In Proceedings of EMNLP.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-Based Dependency Parsing with Stack Long Short-Term Memory. In Proceedings of ACL.

Ahmad Emami and Frederick Jelinek. 2005. A Neural Syntactic Language Model. Machine Learning, 60:195–227.

Akiko Eriguchi, Yoshimasa Tsuruoka, and Kyunghyun Cho. 2017. Learning to Parse and Translate Improves Neural Machine Translation. In Proceedings of ACL.

Jenny Rose Finkel, Alex Kleeman, and Christopher D. Manning. 2008. Efficient, Feature-based Conditional Random Field Parsing. In Proceedings of ACL.

Kuzman Ganchev, João Graça, Jennifer Gillenwater, and Ben Taskar. 2010. Posterior Regularization for Structured Latent Variable Models. Journal of Machine Learning Research, 11:2001–2049.

Jenny Rose Finkel, Christopher D. Manning, and Andrew Y. Ng. 2006. Solving the Problem of Cascading Errors: Approximate Bayesian Inference for Linguistic Annotation Pipelines. In Proceedings of EMNLP.

Kuzman Ganchev, João Graça, Jennifer Gillenwater, and Ben Taskar. 2010. Posterior Regularization for Structured Latent Variable Models. Journal of Machine Learning Research, 11:2001–2049.

Junyoung Chung, Sungjin Ahn, and Yoshua Bengio. 2017. Hierarchical Multiscale Recurrent Neural Networks. In Proceedings of ICLR.

Caio Corro and Ivan Titov. 2019. Differentiable Perturb-and-Parse: Semi-Supervised Parsing with a Structured Variational Autoencoder. In Proceedings of ICLR.

Chris Cremer, Xuechen Li, and David Duvenaud. 2018. Inference Suboptimality in Variational Autoencoders. In Proceedings of ICML.

Yuntian Deng, Yoon Kim, Justin Chiu, Demi Guo, and Alexander M. Rush. 2018. Latent Alignment and Variational Attention. In Proceedings of NIPS.

Ahmed Emami and Frederick Jelinek. 2005. A Neural Syntactic Language Model. Machine Learning, 60:195–227.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One Billion Word Benchmark for Measuring Progress in Statistical Language Modeling. arXiv:1312.3005.

Danqi Chen and Christopher D. Manning. 2014. A Fast and Accurate Dependency Parser using Neural Networks. In Proceedings of EMNLP.

Jinho Choi, Kang Min Yoo, and Sang goo Lee. 2018. Learning to Compose Task-Specific Tree Structures. In Proceedings of AAAI.
Jetic Gő, Hassan S. Shavarani, and Anoop Sarkar. 2018. Top-down Tree Structured Decoding with Syntactic Connections for Neural Machine Translation and Parsing. In Proceedings of EMNLP.

John Hale, Chris Dyer, Adhiguna Kuncoro, and Jonathan R. Brennan. 2018. Finding Syntax in Human Encephalography with Beam Search. In Proceedings of ACL.

Serhii Havrylov, Germán Kruszewski, and Armand Joulin. 2019. Cooperative Learning of Disjoint Syntax and Semantics. In Proceedings of NAACL.

Phu Mon Htut, Kyunghyun Cho, and Samuel R. Bowman. 2018. Grammar Induction with Neural Language Models: An Unusual Replication. In Proceedings of ACL.

Rebecca Hwa. 1999. Supervised Grammar Induction Using Training Data with Limited Constituent Information. In Proceedings of ACL.

Rebecca Hwa. 2000. Sample Selection for Statistical Grammar Induction. In Proceedings of EMNLP.

Lifeng Jin, Finale Doshi-Velez, Timothy Miller, William Schuler, and Lane Schwartz. 2018. Unsupervised Grammar Induction with Depth-bounded PCFG. In Proceedings of TACL.

Mark Johnson, Thomas L. Griffiths, and Sharon Goldwater. 2007. Bayesian Inference for PCFGs via Markov chain Monte Carlo. In Proceedings of NAACL.

Matthew Johnson, David K. Duvenaud, Alex Wiltschko, Ryan P. Adams, and Sandeep R. Datta. 2016. Composing Graphical Models with Neural Networks for Structured Representations and Fast Inference. In Proceedings of NIPS.

Kazuya Kawakami, Chris Dyer, and Phil Blunsom. 2018. Unsupervised Word Discovery with Segmental Neural Language Models. arXiv:1811.09353.

Yoon Kim, Carl Denton, Luong Hoang, and Alexander M. Rush. 2017. Structured Attention Networks. In Proceedings of ICLR.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In Proceedings of ICLR.

Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In Proceedings of ICLR.

Nikita Kitaev and Dan Klein. 2018. Constituency Parsing with a Self-Attentive Encoder. In Proceedings of ACL.

Dan Klein and Christopher Manning. 2002. A Generative Constituent-Context Model for Improved Grammar Induction. In Proceedings of ACL.

Rahul G. Krishnan, Uri Shalit, and David Sontag. 2017. Structured Inference Networks for Nonlinear State Space Models. In Proceedings of AAAI.

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, Graham Neubig, and Noah A. Smith. 2017. What Do Recurrent Neural Network Grammars Learn About Syntax? In Proceedings of EACL.

Adhiguna Kuncoro, Chris Dyer, John Hale, Dani Yogatama, Stephen Clark, and Phil Blunsom. 2018. LSTMs Can Learn Syntax-Sensitive Dependencies Well, But Modeling Structure Makes Them Better. In Proceedings of ACL.

Bowen Li, Jianpeng Cheng, Yang Liu, and Frank Keller. 2019. Dependency Grammar Induction with a Neural Variational Transition-based Parser. In Proceedings of AAAI.

Zhifei Li and Jason Eisner. 2009. First- and Second-Order Expectation Semirings with Applications to Minimum-Risk Training on Translation Forests. In Proceedings of EMNLP.

Yang Liu, Matt Gardner, and Mirella Lapata. 2018. Structured Alignment Networks for Matching Sentences. In Proceedings of EMNLP.

Yang Liu and Mirella Lapata. 2018. Learning Structured Text Representations. In Proceedings of TACL.

Jean Maillard and Stephen Clark. 2018. Latent Tree Learning with Differentiable Parsers: Shift-Reduce Parsing and Chart Parsing. In Proceedings of the Workshop on the Relevance of Linguistic Structure in Neural Architectures for NLP.

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

Rebecca Marvin and Tal Linzen. 2018. Targeted Syntactic Evaluation of Language Models. In Proceedings of EMNLP.

Takuya Matsuzaki, Yusuke Miyao, and Junichi Tsuji. 2005. Probabilistic CFG with Latent Annotations. In Proceedings of ACL.

Gábor Melis, Chris Dyer, and Phil Blunsom. 2018. On the State of the Art of Evaluation in Neural Language Models. In Proceedings of ICLR.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018. Regularizing and Optimizing LSTM Language Models. In Proceedings of ICLR.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Černocky, and Sanjeev Khudanpur. 2010. Recurrent Neural Network Based Language Model. In Proceedings of INTERSPEECH.
Pengcheng Yin, Chunting Zhou, Junxian He, and Graham Neubig. 2018. StructVAE: Tree-structured Latent Variable Models for Semi-supervised Semantic Parsing. In Proceedings of ACL.

Dani Yogatama, Phil Blunsom, Chris Dyer, Edward Grefenstette, and Wang Ling. 2017. Learning to Compose Words into Sentences with Reinforcement Learning. In Proceedings of ICLR.

Xiaodan Zhu, Parinaz Sobhani, and Hongyu Guo. 2015. Long Short-Term Memory Over Tree Structures. In Proceedings of ICML.