Abstract
Dependency grammar induction is the task of learning dependency syntax without annotated training data. Traditional graph-based models with global inference achieve state-of-the-art results on this task but they require $O(n^3)$ run time. Transition-based models enable faster inference with $O(n)$ time complexity, but their performance still lags behind. In this work, we propose a neural transition-based parser for dependency grammar induction, whose inference procedure utilizes rich neural features with $O(n)$ time complexity. We train the parser with an integration of variational inference, posterior regularization and variance reduction techniques. The resulting framework outperforms previous unsupervised transition-based dependency parsers and achieves performance comparable to graph-based models, both on the English Penn Treebank and on the Universal Dependency Treebank. In an empirical comparison, we show that our approach substantially increases parsing speed over graph-based models.

Introduction
Grammar induction is the task of deriving plausible syntactic structures from raw text, without the use of annotated training data. In the case of dependency parsing, the syntactic structure takes the form of a tree whose nodes are the words of the sentence, and whose arcs are directed and denote head-dependent relationships between words. Inducing such a tree without annotated training data is challenging because of data sparseness and ambiguity, and because the search space of potential trees is huge, making optimization difficult.

Most existing approaches to dependency grammar induction have used inference over graph structures and are based either on the dependency model with valence (DMV) of Klein and Manning (2004) or the maximum spanning tree algorithm (MST) for dependency parsing by McDonald, Petrov, and Hall (2011). State-of-the-art representatives include LC-DMV (Noji, Miyao, and Johnson 2016) and Convex-MST (Grave and Elhadad 2015). Recently, researchers have also introduced neural networks for feature extraction in graph-based models (Jiang, Han, and Tu 2016; Cai, Jiang, and Tu 2017).

Though graph-based models achieve impressive results, their inference procedure requires $O(n^3)$ time complexity. Meanwhile, features in graph-based models must be decomposable over substructures to enable dynamic programming. In comparison, transition-based models allow faster inference with linear time complexity and richer feature sets. Although relying on local inference, transition-based models have been shown to perform well in supervised parsing (Kiperwasser and Goldberg 2016; Dyer et al. 2015). However, unsupervised transition parsers are not well-studied. One exception is the work of Rasooli and Faili (2012), in which search-based structure prediction (Daumé III 2009) is used with a simple feature set. However, there is still a large performance gap compared to graph-based models.

Recently, Dyer et al. (2016) proposed recurrent neural network grammars (RNNGs)—a probabilistic transition-based model for constituency trees. RNNG can be used either in a generative way as a language model or in a discriminative way as a parser. Cheng, Lopez, and Lapata (2017) use an autoencoder to integrate discriminative and generative RNNGs, yielding a reconstruction process with parse trees as latent variables and enabling the two components to be trained jointly on a language modeling objective. However, their work uses observed trees for training and does not study unsupervised learning.

In this paper, we make a more radical departure from the existing literature in dependency grammar induction, by proposing an unsupervised neural variational transition-based parser. Specifically, we first modify the transition actions in the original RNNG into a set of arc-standard actions for projective dependency parsing, and then build a dependency variant of the model of Cheng, Lopez, and Lapata (2017). Although this approach performs well for supervised parsing, when applied in an unsupervised setting, the performance decreases dramatically (see Experiments for details). We hypothesize that this is because the parser is fairly unconstrained without prior linguistic knowledge (Naseem et al. 2010; Noji, Miyao, and Johnson 2016). Therefore, we augment the model with posterior regularization, allowing us to seamlessly integrate linguistic knowledge in the shape of a small number of universal linguistic rules. In addition, we propose a novel variance reduction method for stabilizing neural variational inference with discrete latent variables. This yields the first known model that makes it possi-
able to use posterior regularization for neural variational inference with discrete latent variables. When evaluating on the English Penn Treebank and on eight languages from the Universal Dependency (UD) Treebank, we find that our model with posterior regularization outperforms the best unsupervised transition-based dependency parser (Rasooli and Faili 2012), and approaches the performance of graph-based models. We also show how a weak form of supervision can be integrated elegantly into our framework in the form of rule expectations. Finally, we present empirical evidence for the complexity advantage of transition-based models: our model attains a large speed-up compared to a state-of-the-art graph-based model. Code and Supplementary Material are available.\(^1\)

Background

RNNG is a top-down transition system originally proposed for constituency parsing and generation. There are two variants: the discriminative RNNG and the generative RNNG. The discriminative RNNG takes a sentence as input, and predicts the probability of generating a corresponding parse tree from the sentence. The model uses a buffer to store unprocessed terminal words and a stack to store partially completed syntactic constituents. It then follows top-down transition actions to shift words from the buffer to the stack to construct syntactic constituents incrementally.

The discriminative RNNG can be modified slightly to formulate the generative RNNG, an algorithm for incrementally producing trees and sentences in a generative fashion. In generative RNNG, there is no buffer of unprocessed words, but there is an output buffer for storing words that have been generated. Top-down actions are then specified to generate words and tree non-terminals in pre-order. Though not able to parse on its own, a generative RNNG can be used for language modeling as long as parse trees are sampled from a known distribution.

We modify the transition actions in the original RNNG into a set of arc-standard actions for projective dependency parsing. In the discriminative modeling case, the action space includes:

- **SHIFT** fetches the first word in the buffer and pushes it onto the top of the stack.
- **LEFT-REDUCE** adds a left arc in between the top two words of the stack and merges them into a single construct.
- **RIGHT-REDUCE** adds a right arc in between the top two words of the stack and merges them into a single construct.

In the generative modeling case, the **SHIFT** operation is replaced by a **GEN** operation:

- **GEN** generates a word and adds it to the stack and the output buffer.

\(^1\)https://github.com/libowen2121/VI-dependency-syntax

Methodology

To build our dependency grammar induction model, we follow Cheng, Lopez, and Lapata (2017) and propose a dependency-based, encoder-decoder RNNG. This model includes (1) a discriminative RNNG as the **encoder** for mapping the input sentence into a latent variable, which for the grammar induction task is a sequence of parse actions for building the dependency tree; (2) a generative RNNG as the **decoder** for reconstructing the input sentence based on the latent parse actions. The training objective is the likelihood of the observed input sentence, which is reformulated as an evidence lower bound (ELBO), and solved with neural variational inference. The REINFORCE algorithm (Williams 1992) is utilized to handle discrete latent variables in optimization. Overall, the encoder and decoder are jointly trained, inducing latent parse trees or actions from only unlabelled text data. To further regularize the space of parse trees with a linguistic prior, we introduce posterior regularization into the basic framework. Finally, we propose a novel variance reduction technique to train our posterior regularized framework more effectively.

Encoder

We formulate the encoder as a discriminative dependency RNNG that computes the conditional probability \(p(a|x)\) of the transition action sequence \(a\) given the observed sentence \(x\). The conditional probability is factorized over time steps, and parameterized by a transitional state embedding \(v\):

\[
p(a|x) = \prod_{t=1}^{a} p(a_t|v_t)
\]

where \(v_t\) is the transitional state embedding of the encoder at time step \(t\). The encoder is the actual component for parsing at run time.

Decoder

The decoder is a generative dependency RNNG that models the joint probability \(p(x,a)\) of a latent transition action sequence \(a\) and an observed sentence \(x\). This joint distribution can be factorized into a sequence of action and word (emitted by **GEN**) probabilities, which are parameterized by a transitional state embedding \(u\):

\[
p(x,a) = p(a)p(x|a) = \prod_{t=1}^{a} p(a_t|u_t)p(x_t|u_t)I(a_t=\text{GEN})
\]

where \(I\) is an indicator function and \(u_t\) is the state embedding at time step \(t\). The features and the modeling details of both the encoder and the decoder can be found in the Supplementary Material.

Training Objective

Consider a latent variable model in which the encoder infers the latent transition actions (i.e., the dependency structure) and the decoder reconstructs the sentence from these actions.
The maximum likelihood estimate of the model parameters is determined by the log marginal likelihood of the sentence:

\[ \log p(x) = \log \sum_a p(x, a) \]  

(3)

Since the form of the log likelihood is intractable in our case, we optimize the ELBO (by Jensen’s Inequality) as follows:

\[ \log p(x) \geq \log p(x) - KL[q(a)||p(a|x)] \]

\[ = E_{q(a)}[\log \frac{p(x, a)}{q(a)}] = \mathcal{L}_x \]  

(4)

where \( KL \) is the Kullback-Leibler divergence and \( q(a) \) is the variational approximation of the true posterior. This training objective is optimized with the EM algorithm. In the E-step, we approximate the variational distribution \( q(a) \) based on the encoder and the observation \( x \)—\( q(a) \) is parameterized as \( q_\omega(a|x) \). Similarly, the joint probability \( p(x, a) \) is parameterized by the decoder as \( p_\theta(x, a) \).

In the M-step, the decoder parameters \( \theta \) can be directly updated by gradient descent via Monte Carlo simulation:

\[ \frac{\partial \mathcal{L}_x}{\partial \theta} = E_{q_\omega(a|x)}[\frac{\partial \log p_\theta(x, a)}{\partial \theta}] \]

\[ \approx \frac{1}{M} \sum_m \frac{\partial \log p_\theta(x, a^{(m)})}{\partial \theta} \]  

(5)

where \( M \) samples \( a^{(m)} \sim q_\omega(a|x) \) are drawn independently to compute the stochastic gradient.

For the encoder parameters \( \omega \), since the sampling operation is not differentiable, we approximate the gradients using the REINFORCE algorithm (Williams 1992):

\[ \frac{\partial \mathcal{L}_x}{\partial \omega} = E_{q_\omega(a|x)}[l(x, a) \frac{\partial \log q_\omega(a|x)}{\partial \omega}] \]

\[ \approx \frac{1}{M} \sum_m l(x, a^{(m)}) \frac{\partial \log q_\omega(a^{(m)}|x)}{\partial \omega} \]  

(6)

where \( l \) is known as the score function and computed as:

\[ l(x, a) = \log \frac{p_\theta(x, a)}{q_\omega(a|x)} \]  

(7)

**Posterior Regularization**

As will become clear in the Experiments section, the basic model discussed previously performs poorly when used for unsupervised parsing, barely outperforming a left-branching baseline for English. We hypothesize the reason is that the basic model is fairly unconstrained: without any constraints to regularize the latent space, the induced parsers will be arbitrary, since the model is only trained to maximize sentence likelihood (Naseem et al. 2010; Noji, Miyao, and Johnson 2016).

We therefore introduce posterior regularization (PR; Ganchev et al. (2010)) to encourage the neural network to generate well-formed trees. Via posterior regularization, we can give the model access to a small amount of linguistic information in the form of universal syntactic rules (Naseem et al. 2010), which are the same for all languages. These rules effectively function as features, which impose soft constraints on the neural parameters in the form of expectations.

To integrate the PR constraints into the model, a set \( Q \) of allowed posterior distributions over the hidden variables \( a \) can be defined as:

\[ Q = \{ q(a) : \exists \xi, E_q[\phi(x, a)] - b \leq \xi; ||\xi||_\beta \leq \varepsilon \} \]  

(8)

where \( \phi(x, a) \) is a vector of feature functions, \( b \) is a vector of given negative expectations, \( \xi \) is a vector of slack variables, \( \varepsilon \) is a predefined small value and \( || \cdot ||_\beta \) denotes some norm. The PR algorithm only works if \( Q \) is non-empty.

In dependency grammar induction, \( \phi_k(x, a) \) (the \( k^{th} \) element in \( \phi(x, a) \)) can be set as the negative number of times a given rule (dependency arcs, e.g., Root \( \rightarrow \) Verb, Verb \( \rightarrow \) Noun) occurs in a sentence. We hope to bias the learning so that each sentence is parsed to contain these kinds of arcs more than a threshold in the expectation. The posterior regularized likelihood is then:

\[ \mathcal{L}_Q = \max_{q \in Q} \mathcal{L}_x \]

\[ = \log p(x) - \min_{q \in Q} KL[q(a) || p(a|x)] \]  

(9)

Equation (9) indicates that, in the posterior regularized framework, \( q(a) \) not only approximates the true posterior \( p(a|x) \) (estimated by the encoder network \( q_\omega(a|x) \)) but also belongs to the constrained set \( Q \). To optimize \( \mathcal{L}_Q \) via the EM algorithm, we get the revised \( E' \)-step as:

\[ q(a) = \arg \max_{q \in Q} \mathcal{L}_Q \]

\[ = \arg \min_{q \in Q} KL[q(a) || q_\omega(a|x)] \]  

(10)

Formally, the optimization problem in the \( E' \)-step can be described as:

\[ \min_{q, \xi} KL[q(a) || q_\omega(a|x)] \]

\[ s.t. \quad E_q[\phi(x, a)] - b \leq \xi; \quad ||\xi||_\beta \leq \varepsilon \]  

(11)

Following Ganchev et al. (2010), we can solve the optimization problem in (11) in its Lagrangian dual form. Since our transition-based encoder satisfies the decomposition property, the conditional probability \( q_\omega(a|x) \) can be factored as \( \prod_a q_\omega(a_t|x_t) \) in (1). Thus, the factored primal solution can be written as:

\[ q(a) = q_\omega(a|x) Z(\lambda^*) \exp(-\lambda^T \phi(x, a)) \]  

(12)

where \( \lambda^* \) is the Lagrangian multiplier whose solution is given as \( \lambda^* = \arg \max_{\lambda > 0} -b^T \lambda - \log Z(\lambda) - \varepsilon \|\lambda\|_\beta \) and \( Z(\lambda) \) is given as:

\[ Z(\lambda) = \sum_a q_\omega(a|x) \exp(-\lambda^T \phi(x, a)) \]  

(13)

We also define the multiplier computed by PR as:

\[ \gamma(a, x) = \frac{1}{Z(\lambda)} \exp(-\lambda^T \phi(x, a)) \]  

(14)

\[ ||\cdot||_\beta \] is the dual norm of \( || \cdot ||_\beta \). Here we use \( \ell_2 \) norm for both primal norm \( || \cdot ||_\beta \) and dual norm \( || \cdot ||_\beta^* \).
In our case, computing the normalization term $Z(\lambda)$ is intractable for transition-based dependency parsing systems. To address this problem, we view $Z(\lambda)$ as an expectation and estimate it by Monte Carlo simulation as:

$$Z(\lambda) = \mathbb{E}_{q_m(a|x)}[\exp(-\lambda^T \phi(x, a))]$$

$$\approx \frac{1}{M} \sum_m \exp(-\lambda^T \phi(x, a^{(m)}))$$

(15)

Finally, we compute the gradients for encoder and decoder in the M-step as follows:

$$\frac{\partial L_x}{\partial \theta} = \frac{1}{M} \sum_m \gamma(x, a^{(m)}) l(x, a^{(m)}) \frac{\partial \log p_\theta(x, a^{(m)})}{\partial \theta}$$

$$\frac{\partial L_x}{\partial \omega} = \frac{1}{M} \sum_m \gamma(x, a^{(m)}) l(x, a^{(m)}) \frac{\partial \log q(\phi^{(m)})}{\partial \omega}$$

(16)

where $l$ is the score function computed as in (7). Details of the derivation of the M-step can be found in the Supplementary Material.

Variance Reduction in the M-step

Training a neural variational inference framework with discrete latent variables is known to be a challenging problem (Mnih and Gregor 2014; Miao and Blunsom 2016; Miao, Yu, and Blunsom 2016). This is mainly caused by the sampling step of discrete latent variables which results in high variance, especially at the early stage of training when both encoder and decoder parameters are far from optimal. Intuitively, the score function $l(x, a)$ weights the gradient for each latent sample $a$, and its variance plays a crucial role in updating the parameters in the M-step.

To reduce the variance of the score function and stabilize learning, previous work (Mnih and Gregor 2014; Miao and Blunsom 2016; Miao, Yu, and Blunsom 2016) adopts the baseline method (RL-BL), re-defining the score function as:

$$l_{RL-BL}(x, a) = l(x, a) - b(x) - b$$

(17)

where $b(x)$ is a parameterized, input-dependent baseline (e.g., a neural language model in our case) and $b$ is the bias. The baseline method is able to reduce the variance to some extent, but also introduces extra model parameters that complicate optimization. In the following we propose an alternative generic method for reducing the variance of the gradient estimator in the M-step, as well as another task-specific method which results in further improvement.

1. Generic Method The intuition behind the generic method is as follows: the algorithm takes $M$ latent samples for each input $x$ and a score $l(x, a^{(m)})$ is computed for each sample $a^{(m)}$, hence the variance can be reduced by normalization within the group of samples. This motivates the following normalized score function $l_{RL-SN}(x, a)$:

$$l_{RL-SN}(x, a) = \frac{l(x, a) - \bar{l}(x, a)}{\max(1, \sqrt{\text{Var}[l(x, a)]})}$$

(18)

2. Task-Specific Method Besides the generic variance reduction method which applies to discrete neural variational inference in general, we further propose to enhance the quality of the score function $l_{RL-SN}(x, a)$ for the specific dependency grammar induction task.

Intuitively, the score function in (16) weights the gradient of a given sample $a$ by a positive or negative value, while $\gamma(x, a)$ only weights the gradient by a positive value. As a result, the score function plays a much more significant role in determining the optimization direction. Therefore, we propose to correct the polarity of our $l_{RL-SN}(x, a)$ with the number of rules $s(x, a) = -\text{SUM}[\phi(x, a)]$ that occur in the induced dependency structure, where $\text{SUM}[\cdot]$ returns the sum of vector elements. The refined score function is:

$$l_{RL-PC}(x, a) = \left\{ \begin{array}{ll}
\hat{s}(x, a) \\
-\hat{s}(x, a)
\end{array} \right. \quad s(x, a) \neq 0$$

(19)

where $\hat{s}(x, a)$ provides a natural corrective, we can obtain a simpler variant of (19) by directly using $\hat{s}(x, a)$ as the score function:

$$l_{RL-C}(x, a) = \hat{s}(x, a)$$

(20)

We will experimentally compare the different variance reduction techniques (or score functions) of the reinforcement learning objective.

Experiments

Datasets, Universal Rules, and Setup

English Penn Treebank We use the Wall Street Journal (WSJ) section of the English Penn Treebank (Marcus, Marcinkiewicz, and Santorini 1993). The dataset is preprocessed to strip off punctuation. We train our model on sections 2–21, tune the hyperparameters on section 22, and evaluate on section 23. Sentences of length $\leq 10$ are used for training, and we report directed dependency accuracy (DDA) on test sentences of length $\leq 10$ (WSJ-10), and on all sentences (WSJ).

Universal Dependency Treebank We select eight languages from the Universal Dependency Treebank 1.4 (Nivre et al. 2016). We train our model on training sentences of length $\leq 10$ and report DDA on test sentences of length $\leq 15$ and $\leq 40$. We found that training on short sentences generally increased performance compared to training on longer sentences (e.g., length $\leq 15$).

Universal Rules We employ the universal linguistic rules of Naseem et al. (2010) and Noji, Miyao, and Johnson (2016) for WSJ and the Universal Dependency Treebank, respectively (details can be found in the Supplementary Material). For WSJ, we expand the coarse rules defined in Naseem et al. (2010) with the Penn Treebank fine-grained part-of-speech tags. For example, Verb is expanded as VB, VBD, VBG, VBN, VBP and VBZ.
problem since no gold annotations exist to “warm up”
start. Unsupervised models in general face a
Pretraining posterior regularization in incorporating such knowledge. Table 2 (to be discussed next) reveals the effectiveness of
strained. A comparison with posterior-regularized results in
prior linguistic knowledge, the trained model is fairly uncon-
branching baseline. These results suggest that without any
lexicalized and lexicalized versions) fails to beat the left-
gether with the random and left- and right-branching base-
terior regularization.
To study the effectiveness of
Exploration of Model Variants
Posterior Regularization To study the effectiveness of posterior regularization in the neural grammar induction
model, we first implement a fully unsupervised model with-
out posterior regularization. This model is trained with vari-
tional inference, using the standard REINFORCE objective with a baseline (Mnih and Gregor 2014; Miao and Blunsom
2016; Miao, Yu, and Blunsom 2016) and employing no post-
erior regularization.
Table 1 shows the results for the unsupervised model, to-
gether with the random and left- and right-branching baselines. We observe that the unsupervised model (both the unlexicalized and lexicalized versions) fails to beat the left-
branching baseline. These results suggest that without any prior linguistic knowledge, the trained model is fairly uncon-
strained. A comparison with posterior-regularized results in Table 2 (to be discussed next) reveals the effectiveness of posterior regularization in incorporating such knowledge.

Pretraining Unsupervised models in general face a cold-
start problem since no gold annotations exist to “warm up”
the model parameters quickly. This can be observed in (16):
the gradient updates of the model are dependent on the score

<table>
<thead>
<tr>
<th>Model</th>
<th>WSJ-10</th>
<th>WSJ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>19.1</td>
<td>16.4</td>
</tr>
<tr>
<td>Left branching</td>
<td>36.2</td>
<td>30.2</td>
</tr>
<tr>
<td>Right branching</td>
<td>20.1</td>
<td>20.6</td>
</tr>
<tr>
<td>UNSUPERVISED</td>
<td>33.3 (39.0)</td>
<td>29.0 (30.5)</td>
</tr>
<tr>
<td>L-UNSUPERVISED</td>
<td>34.9 (36.4)</td>
<td>28.0 (30.2)</td>
</tr>
</tbody>
</table>

Table 1: Evaluation of the fully unsupervised model (without posterior regularization) on the English Penn Treebank. We report average DDA and the best DDA (in brackets) over five runs. “L-“ denotes the lexicalized version.

Table 2: Evaluation of the posterior-regularized model with and without pretraining on the WSJ. We report average DDA and best DDA (in brackets) over five runs.

<table>
<thead>
<tr>
<th></th>
<th>WSJ-10</th>
<th>WSJ</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Pretraining</td>
<td>47.5 (59.8)</td>
<td>36.7 (46.3)</td>
</tr>
<tr>
<td>Pretraining</td>
<td>64.8 (67.1)</td>
<td>42.0 (43.7)</td>
</tr>
</tbody>
</table>

Table 3: Comparison of models with different variance reduction techniques (or score functions) on the WSJ-10 test set. We report the average DDA $\mu$ and its standard deviation $\sigma$ over five runs.

<table>
<thead>
<tr>
<th></th>
<th>RL-BL</th>
<th>RL-SN</th>
<th>RL-C</th>
<th>RL-PC</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mu$</td>
<td>58.7</td>
<td>60.8</td>
<td>64.4</td>
<td>66.7</td>
</tr>
<tr>
<td>$\sigma$</td>
<td>1.8</td>
<td>0.6</td>
<td>0.3</td>
<td>0.7</td>
</tr>
</tbody>
</table>

Variance Reduction Previously, we described various variance reduction techniques, or modified score functions, for the reinforcement learning objective. These include the conventional baseline method (RL-BL), our sample normalization method (RL-SN), sample normalization with additional polarity correction (RL-PC), and a simplified version of the later (RL-C). We now compare these techniques; all experiments were conducted with pretraining and on the unlexicalized model.

The experimental results in Table 3 show that RL-SN outperforms RL-BL on average DDA, which indicates that sample normalization is more effective in reducing the variance of the gradient estimator. We believe the gain comes from the fact that sample normalization does not introduce extra model parameters, whereas RL-BL does. Polarity correction further boosts performance. However, polarity correction uses the number of universal rules present in an induced dependency structure, i.e., it is a task-specific method for variance reduction. Also RL-C (the simplified version of RL-PC) achieves competitive performance.

Universal Rules In our PR scheme, the rule expectations can be uniformly initialized. This approach does not require any annotated training data; the parser is furnished only with a small set of universal linguistic rules. We call this setting
Table 4: Comparison of uniformly initialized (\textsc{universalrules}) and empirically estimated (\textsc{weaklysupervised}) rule expectation on the WSJ.

<table>
<thead>
<tr>
<th>Model</th>
<th>WSJ-10</th>
<th>WSJ</th>
</tr>
</thead>
<tbody>
<tr>
<td>\textsc{universalrules}</td>
<td>54.7 (58.2)</td>
<td>37.8 (39.3)</td>
</tr>
<tr>
<td>L-\textsc{universalrules}</td>
<td>54.7 (56.3)</td>
<td>36.8 (38.1)</td>
</tr>
<tr>
<td>\textsc{weaklysupervised}</td>
<td>66.7 (67.6)</td>
<td>43.6 (45.0)</td>
</tr>
<tr>
<td>L-\textsc{weaklysupervised}</td>
<td>68.2 (71.1)</td>
<td>48.6 (50.2)</td>
</tr>
</tbody>
</table>

The parser of Rasooli and Faili (2012) is unlexicalized and count-based. To reach the best performance, the authors employed "baby steps" (i.e., they start training on short sentences and gradually add longer sentences (Spitkovsky, Al-shawi, and Jurafsky 2009)), as well as two heuristics called H1 and H2. H1 involves multiplying the probability of the last verb reduction in a sentence by $10^{-10}$, H2 involves multiplying each \textit{Noun} $\rightarrow$ \textit{Verb}, \textit{Adjective} $\rightarrow$ \textit{Verb}, and \textit{Adjective} $\rightarrow$ \textit{Noun} rule by 0.1. These heuristics seem fairly ad-hoc; they presumably bias the probability estimates towards more linguistically plausible values.

As the results in Table 5 show, our \textsc{universalrules} model outperforms RF on both WSJ-10 and full WSJ, achieving a new state of the art for transition-based dependency grammar induction. The RF model does not use universal rules, but its linguistic heuristics play a similar role, which makes our comparison fair. Note that our L-\textsc{weaklysupervised} model achieves a further improvement over \textsc{universalrules}, making it comparable with Convex-MST and HDP-DEP, demonstrating the potential of the neural, transition-based dependency grammar induction approach, which should be even clearer on large datasets.

**Universal Dependency Treebank** Our multilingual experiments use the UD treebank. There we evaluate the two models that perform the best on the WSJ: the unlexicalized \textsc{universalrules} model and lexicalized L-\textsc{weaklysupervised} model. We use the same hyperparameters as in the WSJ experiments. Again, we mainly compare our models with the transition-based model RF (with heuristics H1 and H2), but we also include the graph-based Convex-MST and LC-DMV models for reference.

Table 6 shows the UD treebank results. It can be observed that both \textsc{universalrules} and L-\textsc{weaklysupervised} significantly outperform the RF on both short and long sentences. The improvement of average DDA is roughly 20\% on sentences of length $\leq 40$. This shows that although the heuristic approach employed by Rasooli and Faili (2012) is useful for English, it does not generalize well across languages, in contrast to our posterior-regularized neural networks with universal rules.

** Parsing Speed** To highlight the advantage of our linear time complexity parser, we compare both lexicalized and unlexicalized variants of our parser with a representative DMV-based model (LC-DMV) in terms of parsing speed. The results in Table 7 show that our unlexicalized parser results in a 1.8-fold speed-up for short sentences (length $\leq 15$), and a speed-up of factor 16 for long sentences (full length). And our parser does not lose much parsing speed even in a lexicalized setting.

** Related Work**

In the family of graph-based models, besides LC-DMV, Convex-MST, and HDP-DEP, a lot of work has focused on improving the DMV, such as adding more types of valence (Headden III, Johnson, and McClosky 2009), training with artificial negative examples (Smith and Eisner 2015), and a speed-up of factor 16 for long sentences (full length). And our parser does not lose much parsing speed even in a lexicalized setting.

---

\(^3\)Since we used different preprocessing, we re-implemented their model for a fair comparison.
<table>
<thead>
<tr>
<th>Model</th>
<th>RF+H1+H2</th>
<th>LC-DMV</th>
<th>Conv-MST</th>
<th>L-WEAKLYSUP</th>
<th>UNIVRULES</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Length ≤ 15</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Basque</td>
<td>49.0 (51.0)</td>
<td>47.9</td>
<td>52.5</td>
<td><strong>55.2 (56.0)</strong></td>
<td>52.9 (55.1)</td>
</tr>
<tr>
<td>Dutch</td>
<td>26.6 (31.9)</td>
<td>35.5</td>
<td><strong>43.4</strong></td>
<td>38.7 (41.3)</td>
<td>39.6 (40.2)</td>
</tr>
<tr>
<td>French</td>
<td>33.2 (37.5)</td>
<td>52.1</td>
<td><strong>61.6</strong></td>
<td>56.6 (57.2)</td>
<td>59.9 (61.6)</td>
</tr>
<tr>
<td>German</td>
<td>40.5 (44.0)</td>
<td>51.9</td>
<td>54.4</td>
<td><strong>59.7 (59.9)</strong></td>
<td>57.5 (59.4)</td>
</tr>
<tr>
<td>Italian</td>
<td>33.3 (38.9)</td>
<td>73.1</td>
<td><strong>73.2</strong></td>
<td>58.5 (59.8)</td>
<td>59.7 (62.3)</td>
</tr>
<tr>
<td>Polish</td>
<td>46.8 (59.7)</td>
<td>66.2</td>
<td><strong>66.7</strong></td>
<td>61.8 (63.4)</td>
<td>57.1 (59.3)</td>
</tr>
<tr>
<td>Portuguese</td>
<td>35.7 (43.7)</td>
<td><strong>70.5</strong></td>
<td>60.7</td>
<td>52.5 (54.1)</td>
<td>52.7 (54.2)</td>
</tr>
<tr>
<td>Spanish</td>
<td>35.9 (38.3)</td>
<td><strong>65.5</strong></td>
<td>61.6</td>
<td>55.8 (56.2)</td>
<td>55.6 (56.8)</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td>37.6 (43.1)</td>
<td>57.8</td>
<td><strong>59.3</strong></td>
<td>54.9 (56.0)</td>
<td>54.4 (56.1)</td>
</tr>
<tr>
<td><strong>Length ≤ 40</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Basque</td>
<td>45.4 (47.6)</td>
<td>45.4</td>
<td>50.0</td>
<td><strong>51.0 (51.7)</strong></td>
<td>48.9 (51.5)</td>
</tr>
<tr>
<td>Dutch</td>
<td>23.1 (30.4)</td>
<td>34.1</td>
<td><strong>45.3</strong></td>
<td>42.2 (44.8)</td>
<td>42.5 (44.3)</td>
</tr>
<tr>
<td>French</td>
<td>27.3 (30.7)</td>
<td>48.6</td>
<td><strong>62.0</strong></td>
<td>46.4 (47.5)</td>
<td>55.4 (56.3)</td>
</tr>
<tr>
<td>German</td>
<td>32.5 (37.0)</td>
<td>50.5</td>
<td>51.4</td>
<td><strong>55.6 (56.3)</strong></td>
<td>54.2 (56.3)</td>
</tr>
<tr>
<td>Italian</td>
<td>27.7 (33.0)</td>
<td><strong>71.1</strong></td>
<td>69.1</td>
<td>54.1 (55.6)</td>
<td>55.7 (58.7)</td>
</tr>
<tr>
<td>Polish</td>
<td>43.3 (46.0)</td>
<td><strong>63.7</strong></td>
<td>63.4</td>
<td>57.3 (59.4)</td>
<td>51.7 (52.8)</td>
</tr>
<tr>
<td>Portuguese</td>
<td>28.8 (35.9)</td>
<td><strong>67.2</strong></td>
<td>57.9</td>
<td>44.6 (48.6)</td>
<td>45.3 (46.5)</td>
</tr>
<tr>
<td>Spanish</td>
<td>26.9 (28.8)</td>
<td><strong>61.9</strong></td>
<td>61.9</td>
<td>50.8 (54.0)</td>
<td>52.4 (53.9)</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td>31.9 (36.2)</td>
<td>55.3</td>
<td><strong>57.6</strong></td>
<td>50.3 (52.2)</td>
<td>50.8 (52.5)</td>
</tr>
</tbody>
</table>

Table 6: Evaluation on eight languages of the UD treebank with test sentences of length ≤ 15 and length ≤ 40.

<table>
<thead>
<tr>
<th>Sentence length</th>
<th>≤15</th>
<th>≤40</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>LC-DMV</td>
<td>663</td>
<td>193</td>
<td>74</td>
</tr>
<tr>
<td>Our Unlexicalized</td>
<td>1192</td>
<td>1194</td>
<td>1191</td>
</tr>
<tr>
<td>Our Lexicalized</td>
<td>939</td>
<td>938</td>
<td>983</td>
</tr>
</tbody>
</table>

Table 7: Parsing speed (tokens per second) on the French UD Treebank with test sentences of various lengths. All experiments were conducted on the same CPU platform.

In future, we plan to conduct a larger-scaled grammar induction experiment with our model. We will also explore better training and optimization techniques for neural variational inference with discrete autoregressive latent variables.

Acknowledgments

We gratefully acknowledge the support of the Leverhulme Trust (award IAF-2017-019 to FK). We also thank Li Dong and Jiangming Liu at ILCC for fruitful discussions, Yong Jiang at ShanghaiTech for sharing preprocessed WSJ dataset, and the anonymous reviewers for the constructive comments.

References


Brown, P. F.; Desouza, P. V.; Mercer, R. L.; Pietra, V. J. D.;


