The Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19)

Dependency Grammar Induction with a Neural Variational Transition-Based Parser

Bowen Li, Jianpeng Cheng, Yang Liu, Frank Keller
Institute for Language, Cognition and Computation
School of Informatics, University of Edinburgh
10 Crichton Street, Edinburgh EH8 9AB, UK
{bowen.li, jianpeng.cheng, yang.liu2}@ed.ac.uk, keller@inf.ed.ac.uk

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 possible 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:

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

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:

$$\begin{aligned} p(x, a) &= p(a)p(x|a) \\ &= \prod_{t=1}^{|a|} p(a_t|u_t)p(x_t|u_t)^{I(a_t=\text{GEN})} \end{aligned} \tag{2}$$

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:

$$\log p(x) = \log \sum_{a} p(x, a) \tag{3}$$

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

$$\begin{aligned} \log p(x) &\geqslant \log p(x) - KL[q(a)||p(a|x)] \\ &= \mathbb{E}_{q(a)}[\log \frac{p(x,a)}{q(a)}] = \mathcal{L}_x \end{aligned} \tag{4}$$

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 θ can be directly updated by gradient descent via Monte Carlo simulation:

$$\begin{aligned} \frac{\partial \mathcal{L}_x}{\partial \theta} &= \mathbb{E}_{q_\omega(a|x)}[\frac{\partial \log p_\theta(x,a)}{\partial \theta}] \\ &\approx \frac{1}{M} \sum_{m} \frac{\partial \log p_\theta(x, a^{(m)})}{\partial \theta} \end{aligned} \tag{5}$$

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

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

$$\begin{aligned} \frac{\partial \mathcal{L}_x}{\partial \omega} &= \mathbb{E}_{q_\omega(a|x)}[l(x,a)\frac{\partial \log q_\omega(a|x)}{\partial \omega}] \\ &\approx \frac{1}{M} \sum_{m} l(x, a^{(m)}) \frac{\partial \log q_\omega(a^{(m)}|x)}{\partial \omega} \end{aligned} \tag{6}$$

where l is known as the score function and computed as:

$$l(x,a) = \log \frac{p_\theta(x,a)}{q_\omega(a|x)} \tag{7}$$

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 parses 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:

$$Q = \{q(a) : \exists \boldsymbol{\xi},\ \mathbb{E}_q[\boldsymbol{\phi}(x,a)] - \boldsymbol{b} \leqslant \boldsymbol{\xi};\ ||\boldsymbol{\xi}||_\beta \leqslant \varepsilon\} \tag{8}$$

where $\boldsymbol{\phi}(x, a)$ is a vector of feature functions, $\boldsymbol{b}$ is a vector of given negative expectations, $\boldsymbol{\xi}$ is a vector of slack variables, ε 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 $\boldsymbol{\phi}(x, a)$) 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:

$$\begin{aligned} \mathcal{L}_Q &= \max_{q \in Q} \mathcal{L}_x \\ &= \log p(x) - \min_{q \in Q} KL[q(a) \ || \ p(a|x)] \end{aligned} \tag{9}$$

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:

$$\begin{aligned} q(a) &= \operatorname*{argmax}_{q \in Q} \mathcal{L}_Q \\ &= \operatorname*{argmin}_{q \in Q} KL[q(a) \ || \ q_\omega(a|x)] \end{aligned} \tag{10}$$

Formally, the optimization problem in the E$'$-step can be described as:

$$\begin{aligned} &\min_{q,\boldsymbol{\xi}} \quad KL[q(a) \ || \ q_\omega(a|x)] \\ &s.t. \quad \mathbb{E}_q[\boldsymbol{\phi}(x,a)] - \boldsymbol{b} \leqslant \boldsymbol{\xi};\ \ ||\boldsymbol{\xi}||_\beta \leqslant \varepsilon \end{aligned} \tag{11}$$

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 $\prod_{t=1}^{|a|} q_\omega(a_t|v_t)$ in (1). Thus, the factored primal solution can be written as:

$$q(a) = \frac{q_\omega(a|x)}{Z(\boldsymbol{\lambda}^*)} \exp(-\boldsymbol{\lambda}^{*T} \boldsymbol{\phi}(x,a)) \tag{12}$$

where $\boldsymbol{\lambda}$ is the Lagrangian multiplier whose solution is given as $\boldsymbol{\lambda}^* = \operatorname{argmax}_{\boldsymbol{\lambda} \geqslant 0} -\boldsymbol{b}^T\boldsymbol{\lambda} - \log Z(\boldsymbol{\lambda}) - \varepsilon||\boldsymbol{\lambda}||_{\beta^*}$[2] and $Z(\boldsymbol{\lambda})$ is given as:

$$Z(\boldsymbol{\lambda}) = \sum_{a} q_\omega(a|x) \exp(-\boldsymbol{\lambda}^T \boldsymbol{\phi}(x,a)) \tag{13}$$

We also define the multiplier computed by PR as:

$$\gamma(a,x) = \frac{1}{Z(\boldsymbol{\lambda})} \exp(-\boldsymbol{\lambda}^T \boldsymbol{\phi}(x,a)) \tag{14}$$

[2] $||\cdot||_{\beta^*}$ is the dual norm of $||\cdot||_\beta$. Here we use ℓ_2 norm for both primal norm $||\cdot||_\beta$ and dual norm $||\cdot||_{\beta^*}$.

In our case, computing the normalization term $Z(\boldsymbol{\lambda})$ is intractable for transition-based dependency parsing systems. To address this problem, we view $Z(\boldsymbol{\lambda})$ as an expectation and estimate it by Monte Carlo simulation as:

$$\begin{aligned} Z(\boldsymbol{\lambda}) &= \mathbb{E}_{q_\omega(a|x)}[\exp(-\boldsymbol{\lambda}^T \boldsymbol{\phi}(x,a))] \\ &\approx \frac{1}{M}\sum_m \exp(-\boldsymbol{\lambda}^T \boldsymbol{\phi}(x,a^{(m)})) \end{aligned} \tag{15}$$

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

$$\begin{aligned} \frac{\partial \mathcal{L}_x}{\partial \theta} &= \frac{1}{M}\sum_m \gamma(x,a^{(m)}) l(x,a^{(m)}) \frac{\partial \log p_\theta(x,a^{(m)})}{\partial \theta} \\ \frac{\partial \mathcal{L}_x}{\partial \omega} &= \frac{1}{M}\sum_m \gamma(x,a^{(m)}) l(x,a^{(m)}) \frac{\partial \log q(a^{(m)})}{\partial \omega} \end{aligned} \tag{16}$$

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:

$$l_{\text{RL-BL}}(x,a) = l(x,a) - b(x) - b \tag{17}$$

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 $l_{\text{RL-SN}}(x,a)$:

$$l_{\text{RL-SN}}(x,a) = \frac{l(x,a) - \bar{l}(x,a)}{\max(1, \sqrt{\mathrm{Var}[l(x,a)]})} \tag{18}$$

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 $l_{\text{RL-SN}}(x,a)$ 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 $l_{\text{RL-SN}}(x,a)$ with the number of rules $s(x,a) = -\text{SUM}[\boldsymbol{\phi}(x,a)]$ that occur in the induced dependency structure, where $\text{SUM}[]$ returns the sum of vector elements. The refined score function is:

$$l_{\text{RL-PC}}(x,a) = \begin{cases} |l_{\text{RL-SN}}(x,a)| & \hat{s}(x,a) \geqslant 0 \\ -|l_{\text{RL-SN}}(x,a)| & \hat{s}(x,a) < 0 \end{cases} \tag{19}$$

where $\hat{s}(x,a) = \frac{s(x,a) - \bar{s}(x,a)}{\sqrt{\mathrm{Var}[s]}}$.

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

$$l_{\text{RL-C}}(x,a) = \hat{s}(x,a) \tag{20}$$

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 ≤ 10 are used for training, and we report directed dependency accuracy (DDA) on test sentences of length ≤ 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 ≤ 10 and report DDA on test sentences of length ≤ 15 and ≤ 40. We found that training on short sentences generally increased performance compared to training on longer sentences (e.g., length ≤ 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*.

Model	WSJ-10	WSJ
Random	19.1	16.4
Left branching	36.2	30.2
Right branching	20.1	20.6
UNSUPERVISED	33.3 (39.0)	29.0 (30.5)
L-UNSUPERVISED	34.9 (36.4)	28.0 (30.2)

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.

Setup To avoid a scenario in which REINFORCE has to work with an arbitrarily initialized encoder and decoder, our posterior regularized neural variational dependency parser is pretrained with the direct reward from PR. (This will be discussed later; for more details on the training, see Supplementary Material.)

We use AdaGrad (Duchi, Hazan, and Singer 2011) to optimize the parameters of the encoder and decoder, as well as the projected gradient descent algorithm (Bertsekas 1999) to optimize the parameters of posterior regularization.

We use GloVe embeddings (Pennington, Socher, and Manning 2014) to initialize English word vectors and FastText embeddings (Bojanowski et al. 2016) for the other languages. Across all experiments, we test both unlexicalized and lexicalized versions of our models. The unlexicalized versions use gold POS tags as model inputs, while the lexicalized versions additionally use word tokens (Le and Zuidema 2015). We use Brown clustering (Brown et al. 1992) to obtain additional features in the lexicalized versions (Buys and Blunsom 2015).

We report average DDA and best DDA over five runs for our main parsing results.

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 without posterior regularization. This model is trained with variational 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 posterior regularization.

Table 1 shows the results for the unsupervised model, together 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 unconstrained. 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 function l, which in return relies on the model parameters. At the beginning of training we cannot obtain an accurately approximated l for updating model parameters. To alleviate this problem, one approach is to ignore the score function in the gradient update at the early stage. In this case, both the encoder and decoder are trained with the direct reward from PR (detailed equations can be found in the Supplementary Material). We test the effectiveness of this approach, which we call *pretraining*.

	WSJ-10	WSJ
No Pretraining	47.5 (59.8)	36.7 (46.3)
Pretraining	64.8 (67.1)	42.0 (43.7)

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.

	RL-BL	RL-SN	RL-C	RL-PC
μ	58.7	60.8	64.4	66.7
σ	1.8	0.6	0.3	0.7

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

Table 2 shows the results of a standard posterior-regularized model compared to one only with pretraining. Both models use the unlexicalized setup. We find that the posterior-regularized model benefits a lot from pretraining, which therefore is a useful way to avoid cold start.

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 a 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

Model	WSJ-10	WSJ
UNIVERSALRULES	**54.7** (58.2)	**37.8** (39.3)
L-UNIVERSALRULES	54.7 (56.3)	36.8 (38.1)
WEAKLYSUPERVISED	66.7 (67.6)	43.6 (45.0)
L-WEAKLYSUPERVISED	**68.2** (71.1)	**48.6** (50.2)

Table 4: Comparison of uniformly initialized (UNIVERSALRULES) and empirically estimated (WEAKLYSUPERVISED) rule expectation on the WSJ.

Model	WSJ-10	WSJ
Graph-based models		
Convex-MST	60.8	**48.6**
HDP-DEP	**71.9**	–
Transition-based models		
RF	37.3 (40.7)	32.1 (33.1)
RF+H1+H2	51.0 (52.7)	37.2 (37.6)
UNIVERSALRULES	54.7 (58.2)	37.8 (39.3)
L-WEAKLYSUPERVISED	68.2 (71.1)	**48.6** (50.2)

Table 5: Comparison of our models (UNIVERSALRULES and L-WEAKLYSUPERVISED) with previous work on the English Penn Treebank. H1 and H2 are two heuristics used in Rasooli and Faili (2012).

UNIVERSALRULES.

However, we can initialize the rule expectation non-uniformly, which allows us to introduce a degree of supervision into the PR scheme. Here, we explore one way of doing this: we assume a training set that is annotated with dependency rules (the training portion of the WSJ), based on which we estimate expectations for the universal rules. We call this setting WEAKLYSUPERVISED.

The results of an experiment comparing these two settings is shown in Table 4. In both cases we use pretraining and the best performing score function RL-PC. Here we report results using both unlexicalized and lexicalized settings. It can be seen that the best performing UNIVERSALRULES model is the unlexicalized one, while the best WEAKLYSUPERVISED model is lexicalized. Overall, WEAKLYSUPERVISED outperforms UNIVERSALRULES, which demonstrates that our posterior regularized parser is able to effectively use weak supervision in the form of an empirical initialization of the rule expectations.

Parsing Results

English Penn Treebank We compare our unsupervised UNIVERSALRULES model and its WEAKLYSUPERVISED variant with (1) the state-of-the-art unsupervised transition-based parser of Rasooli and Faili (2012) [3], denoted as RF, and (2) two state-of-the-art unsupervised graph-based parsers with universal linguistic rules: Convex-MST (Grave and Elhadad 2015) and HDP-DEP (Naseem et al. 2010). Both of these are not transition-based, and thus not directly comparable to our approach, but are useful for reference.

[3]Since we used different preprocessing, we re-implemented their model for a fair comparison.

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, Alshawi, 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 *Noun* $\rightarrow$ *Verb*, *Adjective* $\rightarrow$ *Verb*, and *Adjective* $\rightarrow$ *Noun* rule by 0.1. These heuristics seem fairly adhoc; they presumably bias the probability estimates towards more linguistically plausible values.

As the results in Table 5 show, our 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-WEAKLYSUPERVISED model achieves a further improvement over 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 UNIVERSALRULE model and lexicalized L-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 UNIVERSALRULES and L-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

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

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

Sentence length	$\leq$15	$\leq$40	All
LC-DMV	663	193	74
Our Unlexicalized	1192	1194	1191
Our Lexicalized	939	938	983

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

2005), and learning initial parameters from shorter sentences (Spitkovsky, Alshawi, and Jurafsky 2009). Among graph-based models, there is also some work conceptually related to our approach. Jiang, Han, and Tu (2017) combine a discriminative and a generative unsupervised parser using dual decomposition. Cai, Jiang, and Tu (2017) use CRF autoencoder for unsupervised parsing. In contrast to these two approaches, we use neural variational inference to combine discriminative and generative models.

For transition-based models, Daumé III (2009) introduces a structure prediction approach and Rasooli and Faili (2012) propose a model with simple features based on this approach. Recently, Shi, Huang, and Lee (2017) and Gomez-Rodriguez, Shi, and Lee (2018) show that practical dynamic programming-like global inference is possible for transition-based systems using a minimal set of bidirectional LSTM features. These models achieve competitive performance for projective or non-projective supervised dependency parsing but have not been applied to unsupervised parsing.

Conclusions

In this work, we propose a neural variational transition-based model for dependency grammar induction. The model consists of a generative RNNG for generation, and a discriminative RNNG for parsing and inference. We train the model on unlablled corpora with an integration of neural variational inference, posterior regularization and variance reduction techniques. This allows us to use a small amount of universal linguistic rules as prior knowledge to regularize the latent space. We show that it is straightforward to integrate weak supervision into our model in the form of rule expectations. Our parser obtains a new state of the art for unsupervised transition-based dependency parsing, with linear time complexity and significantly faster parsing speed compared to graph-based models.

In future, we plan to conduct a larger-sclae of grammar induction experiment with our model. We will also explore better training and optimization techiniques 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

Bertsekas, D. P. 1999. *Nonlinear programming*. Athena scientific Belmont.

Bojanowski, P.; Grave, E.; Joulin, A.; and Mikolov, T. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

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

and Lai, J. C. 1992. Class-based n-gram models of natural language. *Computational linguistics* 18(4):467–479.

Buys, J., and Blunsom, P. 2015. Generative incremental dependency parsing with neural networks. In *Proceedings of the ACL Conference*.

Cai, J.; Jiang, Y.; and Tu, K. 2017. Crf autoencoder for unsupervised dependency parsing. In *Proceedings of the EMNLP Conference*.

Cheng, J.; Lopez, A.; and Lapata, M. 2017. A generative parser with a discriminative recognition algorithm. In *Proceedings of the ACL Conference*.

Daumé III, H. 2009. Unsupervised search-based structured prediction. In *Proceedings of the ICML Conference*.

Duchi, J.; Hazan, E.; and Singer, Y. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR* 12(Jul):2121–2159.

Dyer, C.; Ballesteros, M.; Ling, W.; Matthews, A.; and Smith, N. A. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the ACL Conference*.

Dyer, C.; Kuncoro, A.; Ballesteros, M.; and Smith, N. A. 2016. Recurrent neural network grammars. In *Proceedings of the NAACL Conference*.

Ganchev, K.; Gillenwater, J.; Taskar, B.; et al. 2010. Posterior regularization for structured latent variable models. *JMLR* 11(Jul):2001–2049.

Gomez-Rodriguez, C.; Shi, T.; and Lee, L. 2018. Global transition-based non-projective dependency parsing. In *Proceedings of the ACL Conference*.

Grave, E., and Elhadad, N. 2015. A convex and feature-rich discriminative approach to dependency grammar induction. In *Proceedings of the ACL Conference*.

Headden III, W. P.; Johnson, M.; and McClosky, D. 2009. Improving unsupervised dependency parsing with richer contexts and smoothing. In *Proceedings of the NAACL Conference*.

Jiang, Y.; Han, W.; and Tu, K. 2016. Unsupervised neural dependency parsing. In *Proceedings of the EMNLP Conference*.

Jiang, Y.; Han, W.; and Tu, K. 2017. Combining generative and discriminative approaches to unsupervised dependency parsing via dual decomposition. In *Proceedings of the EMNLP Conference*.

Kiperwasser, E., and Goldberg, Y. 2016. Simple and accurate dependency parsing using bidirectional lstm feature representations. In *TACL*.

Klein, D., and Manning, C. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the ACL Conference*.

Le, P., and Zuidema, W. 2015. Unsupervised dependency parsing: Let's use supervised parsers. In *Proceedings of the NAACL Conference*.

Marcus, M. P.; Marcinkiewicz, M. A.; and Santorini, B. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics* 19(2):313–330.

McDonald, R.; Petrov, S.; and Hall, K. 2011. Multi-source transfer of delexicalized dependency parsers. In *Proceedings of the EMNLP Conference*.

Miao, Y., and Blunsom, P. 2016. Language as a latent variable: Discrete generative models for sentence compression. In *Proceedings of the EMNLP Conference*.

Miao, Y.; Yu, L.; and Blunsom, P. 2016. Neural variational inference for text processing. In *Proceedings of the ICML Conference*.

Mnih, A., and Gregor, K. 2014. Neural variational inference and learning in belief networks. In *Proceedings of the ICML Conference*.

Naseem, T.; Chen, H.; Barzilay, R.; and Johnson, M. 2010. Using universal linguistic knowledge to guide grammar induction. In *Proceedings of the EMNLP Conference*.

Nivre, J.; de Marneffe, M.-C.; Ginter, F.; Goldberg, Y.; Hajic, J.; Manning, C. D.; McDonald, R. T.; Petrov, S.; Pyysalo, S.; Silveira, N.; et al. 2016. Universal dependencies v1: A multilingual treebank collection. In *LREC*.

Noji, H.; Miyao, Y.; and Johnson, M. 2016. Using left-corner parsing to encode universal structural constraints in grammar induction. In *Proceedings of the EMNLP Conference*.

Pennington, J.; Socher, R.; and Manning, C. 2014. Glove: Global vectors for word representation. In *Proceedings of the EMNLP Conference*.

Rasooli, M. S., and Faili, H. 2012. Fast unsupervised dependency parsing with arc-standard transitions. In *Proceedings of the Joint Workshop on Unsupervised and Semi-Supervised Learning in NLP*.

Shi, T.; Huang, L.; and Lee, L. 2017. Fast(er) exact decoding and global training for transition-based dependency parsing via a minimal feature set. In *Proceedings of the EMNLP Conference*.

Smith, N. A., and Eisner, J. 2005. Guiding unsupervised grammar induction using contrastive estimation. In *Proceedings of IJCAI Workshop on Grammatical Inference Applications*.

Spitkovsky, V. I.; Alshawi, H.; and Jurafsky, D. 2009. Baby steps: How "less is more" in unsupervised dependency parsing. *NIPS: Grammar Induction, Representation of Language and Language Learning*.

Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8(3-4):229–256.