Relaxed RNNG VAEs

Sophia Sklaviadis

We study continuously relaxed discrete (CONCRETE)[1] latent variables of dependency grammar in variational autoencoder (VAE) language models. Distinct representations of syntax, $p(\mathbf{z})$, and semantics, $p(\mathbf{x})$, optimized jointly as $p(\mathbf{x}, \mathbf{z})$ share statistical strength. Since the posterior $p(\mathbf{z} \mid \mathbf{x})$ is intractable, we use a variational approximation $q(\mathbf{z} \mid \mathbf{x})$, which allows the coordination of a conditional random field (CRF) representation of globally normalized grammar in the encoder with a transition-based representation of a similar but locally normalized grammar in the decoder. Optimization of this VAE framework via EM is arguably a cognitively plausible representation of (the acquisition of) functional grammar, i.e. memorizing a globally normalized grammar in the expensive E-step over a batch, then optimizing for the best possible greedy decoding that this grammar facilitates with a cheap M-step, intuitively constrained by finite time and memory resources, and repeating, until you pass the quiz. This VAE setup also makes possible the cooperative optimization of global normalization in the encoder, and of local, domain- or language-specific regularization in the decoder e.g. through modeling of higher-order factors. The above approach to dependency grammar induction is, to our knowledge, new. The two most similar models, which we combine and relax, through Stochastic Softmax, have produced state-of-the-art unsupervised constituency Unlabeled Attachment Score (UAS) accuracies with a CRF econder (Kim et al. [2019]), and state-of-the-art unsupervised dependency UAS accuracies with posterior regularization (Li et al. [2019]).

Further, we evaluate the effect of recurrence on language model entropy, and on UAS accuracy via ablation studies on the VAE's encoder's and decoder's neural parametrizations, inspired by Andor et al. [2016]. That is, given a globally normalized CRF encoder, what is the relative benefit of using BiLSTM embeddings in the encoder?[2] Following Andor et al. [2016], we hypothesize that RNN-based embeddings are superfluous with a CRF E-step objective, while feed forward embeddings are faster and don't hurt performance. Conversely, in the jointly optimized decoder we test the hypothesis that the BiLSTM representations of the stack and buffer, contribute significantly to the improved accuracy of the grammar induction model. Our most interesting point is that the non-recurrent embedding parametrization of the (continuously relaxed) globally normalized CRF encoder allows for faster and more robust sampling. In turn, this leads to more efficient prevention of posterior collapse in the randomized dynamic programming style of Fu et al. [2022] who estimate the proposal distribution using successive pairs of randomized sums over subtrees, where for each subtree the topK paths are computed exactly, while a sample of the remaining paths is used for importance weighted Rao-Blackwellized variance reduction of the subtree estimator and bias correction.

1 Background: Recurrent Neural Network Grammars

Dyer et al. [2015] propose the stack LSTM data structure, a stack implemented with recurrent neural networks (RNNs). Dyer et al. [2016] use the stack LSTM to characterize recurrent neural network

[1] Maddison et al. [2016]

[2] This is to compare Kim et al. [2019]'s encoder arc-scores, $s_{ij} = MLP(BiLSTM(h_1, ..., h_T))$, with Andor et al. [2016]'s globally normalized feed forward CRF scores within a VAE framework.

grammars (RNNGs) which are a kind of generative language model: a joint probability distribution over sentences and their structure. The generative RNNG of Dyer et al. [2016] is based on a pretrained discriminative RNNG. Intuitively, the discriminative model functions as the latent parser. More precisely, the discriminative RNNG is a distribution over syntactic parse trees given the observed sentence. In Dyer et al. [2015] the discriminative RNNG represents a dependency grammar, while in Dyer et al. [2016] the discriminative RNNG represents a constituency grammar.

Using variational inference, Cheng et al. [2017] jointly train generative and discriminative RNNGs, modeling constituency parse trees as the latent variables.[3] Cheng et al. [2017] use training data to estimate the parameters of the discriminative RNNG. Li et al. [2019] use posterior regularization and variational inference to train an unsupervised RNNG (URNNG) whose discriminative component represents a dependency tree.[4] Kim et al. [2019] use a CRF approximation to learn a state of the art URRNG with a discriminative component that represents a binary constituency tree.

To generalize the underlying latent structure of Kim et al. [2019], we describe a URNNG using a neural CRF parametrization of an edge-factored dependency parser as the variational distribution. Using this dependency CRF proposal distribution, we combine Li et al. [2019]'s latent dependency model with Kim et al. [2019]'s variational inference approach. The CRF over dependency trees is the encoder's parser, while a stack LSTM is used as the decoder's transition-based parser. Estimating the CRF in E-step and the parameters of the RNNG decoder in the M-step is also appealing as a cognitively plausible model of language processing.

2 Methodology

A generative language model is a joint distribution over sentences and their latent dependency trees (spanning arborescences). We obtain the language model by marginalizing the structured latent variable z in the generative RNNG:

$$p(x) = \sum_{z} p(x, z) \tag{1}$$

where x is the observed sequence of words and z is an assignment to binary variables which corresponds to the directed arcs in x's *projective* dependency tree:

$$\begin{aligned} x &= < x_0 \equiv ROOT, x_1, ..., x_n > \\ z &= \{z_{ij} : i \neq j, 0 \leq i \leq n, 1 \leq j \leq n\} \end{aligned}$$

where $z_{ij} = 1$ means that the i-th word in x is the parent of the j-th word.

[3]Cheng et al. [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." (Li et al. [2019], p. 1)

[4]The posterior regularization of Li et al. [2019] represents universal constraints on the space of latent trees following Naseem et al. [2010], with methodology for posterior regularization from Ganchev et al. [2010], implemented with neural networks by Mnih and Gregor [2014].

A variational lower bound of the marginal log-likelihood is derived by introducing a neural CRF dependency parser, $q_\phi(z|x)$, as an inference network or *encoder* over the latent variables which approximates the true posterior, $p_\theta(z|x)$:

$$\begin{aligned} \log p_\theta(x) &= \mathbb{E}_{q_\phi(z|x)}\left[\log \frac{p_\theta(z,x)}{q_\phi(z|x)}\right] + KL\left[q_\phi(z|x) \parallel p_\theta(z|x)\right] && (2) \\ &\geq \mathbb{E}_{q_\phi(z|x)}\left[\log \frac{p_\theta(z,x)}{q_\phi(z|x)}\right] \equiv ELBO && (3) \end{aligned}$$

The transition-based generative RNNG or *decoder*, $p_\theta(z,x)$, and the graph-based inference model, $q_\phi(z|x)$, are trained simultaneously by optimizing the Evidence Lower Bound (ELBO) with respect to the parameters θ and ϕ.

2.1 Encoder: CRF graph-based dependency parser

We assume that the joint distribution over the latent variables, z, is associated with an undirected graph G over the set of random variables z, such that each link variable, z_{ij}, is a vertex of G. Let C be the set of cliques in G. The members of a clique are neighbors. The joint distribution over z's is a Markov random field (MRF) or undirected graphical model if the random variables z obey the Markov property with respect to the graph G. That is, every variable z_{ij} is conditionally independent of all the other variables in the graph given its neighbors. The graphical structure of G is used to factorize the joint distribution over z's into a product of *potential functions* or *local factors*. Each potential function operates on a set of random variables that form a clique in G. So, conditionally independent random variables do not appear in the same potential function.

A conditional random field (CRF) may be viewed as an MRF which is globally conditioned on the observed sequence, x. So a CRF makes independence assumptions among z's but not within x. A CRF is a conditional distribution, $p(z|x)$, with an associated graphical structure that describes the dependencies among the z's.[5] Note that while dependency tree links are directed arcs "the underlying graphical model used to compute the likelihood of a parse, is an undirected graphical model" (Smith [2010]). As long as the joint model over z's is a first-order factorization, it does not take into account potential functions that involve two or more directed arcs.

The arc factored model is a first-order factorization where we assume that the individual directed arcs of the dependency tree are independent. Assuming an arc factored model, in which $\sum_{ij} z_{ij} = n$ (McDonald et al. [2005]),[6] a CRF graph-based parser defines the conditional probability of a dependency

[5] Smith and Eisner [2008]; Smith [2010]

[6] Pei et al. [2015] and Wang and Chang [2016] implement a neural version of the original method of McDonald et al. [2005] and learn the parameters using a "max-margin" optimization criterion and training data.

A 2nd-order factorization model takes into account scores based on adjacent links (McDonald and Pereira [2006]). A 3rd-order model uses sibling and grandparent scores (Koo and Collins [2010]). Pei et al. [2015] comment that a neural 1st-order (edge-factored) model can be competitive with higher order factorization, as it is able to capture richer contextual information.

tree z given the sentence x through the Gibbs distribution (Smith [2010]):

$$q_\phi(z|x) = \frac{e^{score_\phi(z,x)}}{\sum\limits_{all\ valid\ z'} e^{score_\phi(z',x)}} \tag{4}$$

where the score of a tree, z, is the sum of the scores of all its edges, z_{ij}:

$$score_\phi(z,x) = \mathbb{1}\{z\ is\ valid\} \cdot \sum_{i \neq j} z_{ij} s_{ij} \tag{5}$$

and $s_{ij} = score_\phi(x_i \to x_j; x)$ represents the strength of the arc's association, or how likely word x_i is the head of word x_j, and it depends on the words x and parameters ϕ. The score s_{ij} is a log potential function that depends only on one link variable describing the edge $x_i \to x_j$, i.e. a first-order factorization. Below we describe the neural parametrization of the s_{ij}'s.

2.1.1 Neural arc scores

The neural parametrization of the arc scores described below is standard in current supervised parsers, however, as far as we know, it has not been used before to train an unsupervised dependency parser.

The arc scores, s_{ij}, can be parametrized with neural networks as follows:

1. Given word embeddings, $e_{w_i} \in \mathbb{R}^{d_e}$, and POS embeddings, $e_{p_i} \in \mathbb{R}^{d_e}$, we concatenate them as

$$x_i = g(W_e[e_{w_i} : e_{p_i}] + b_e)$$

where d_e is the embedding dimension, $b_e \in \mathbb{R}^{d_e}$ is a bias term, $W_e \in \mathbb{R}^{d_e \times 2d_e}$ is a weight matrix, g is an elementwise activation (ReLU), and x_i is the overall **word input representation**.

2. For each sentence, x, run a bidiretional LSTM over the ordered sequence of x_i's. We obtain the forward, $[\overrightarrow{h_1}, ..., \overrightarrow{h_n}]$, and backward, $[\overleftarrow{h_1}, ..., \overleftarrow{h_n}]$, hidden states of the BiLSTM and concatenate or add them:

$$v_i = \overrightarrow{h_i} + \overleftarrow{h_i} \qquad .$$

The output vectors of the BiLSTM, $v_i \in R^{d_e}$, are used as the **final word embeddings**.

3. The context information of a head-modifier pair, $x_i \to x_j$, results from dividing the sentence x in three parts: prefix, infix, suffix, and using LSTM-Minus (Wang and Chang [2016]) to obtain the **segment embeddings**:

$$x \ = \ [v_0 ... v_{i-1}]\ [v_i ... v_j]\ [v_{j+1} ... v_n] \tag{6}$$

The $prefix$ of head-modifier pair, $x_i \to x_j$, is represented by the LSTM's last hidden state of that segment, v_{i-1}. Correspondingly, $suffix(x_i \to x_j) = v_n - v_j$, and $infix(x_i \to x_j) = v_j - v_{i-1}$. When no prefix or suffix exists, the corresponding embedding is set to 0.

4. For each arc, $x_i \to x_j$, we concatenate the embeddings of head word, modifier word, and sentence segments:

$$a_{ij} \;=\; [v_i \;:\; v_j \;:\; prefix(x_i \to x_j) \;:\; infix(x_i \to x_j) \;:\; suffix(x_i \to x_j)]$$

   Then we model the **direction of the edge** using a direction-specific linear transformation of a_{ij}:

$$l = W_a^d a_{ij} + b_a^d$$

   where $d \in \{$LEFT, RIGHT$\}$ indicates the direction of the child word, and W_a^d, b_a^d are respectively a weight matrix and a bias term. There are separate parameters for each direction.

5. The **arc scores** are obtained by an elementwise activation function, g, applied to the last linear transformation:

$$s_{ij} = g(W_a^d a_{ij} + b_a^d)$$

   where $g(l) = \tanh(l^3 + l)$ is the tanh-cube function. The cube extension is added to enhance ability of capturing complex interactions.

Thus the variational distribution, $q_\phi(z|x)$, is parametrized by $\phi = (W_e, b_e, W_a^d, b_a^d)$.

2.2 Decoder: Transition-based RNNG

The transition-based dependency RNNG models the joint distribution, $p_\theta(x, z)$, of a latent tree, z, and an observed sequence, x, as a sequence of transition actions, $z_t \in \{$GENERATE, L-REDUCE, R-REDUCE$\}$. At time step t, the RNNG either generates words and adds them to a buffer, or combines the top two words of a stack with a left or right arc and merges them into a single construct (Dyer et al. [2016]; Cheng et al. [2017]; Li et al. [2019]; Kim et al. [2019]):

$$p_\theta(x, z) \;=\; \prod_{t=1}^{T} p(z_t|u_t) \cdot p(x_t|u_t)^{\mathbb{1}\{z_t = \text{GENERATE}\}} \tag{7}$$

where $p(z_t|u_t)$ is a Categorical distribution with three categories corresponding to the possible transition actions, and $p(x_t|u_t)$ is a Categorical distribution with $|W|$ categories where W is the vocabulary set.

The state embedding, u_t, represents the full history: all actions up to t, $z_{<t}$, and all generated words up to t in the *ouput buffer*, $x_{<t}$. Note that x_t is not the t-th word in the sequence x, rather we mean $x_{m(t)}$ where

$$m(t) \;=\; \sum_{t=1}^{t} \mathbb{1}\{z_t = \text{GENERATE}\} \qquad .$$

To compute the state embedding u_t, we concatenate $[s_t : b_t]$ and calculate

$$u_t = W_2 \tanh(W_1[s_t : b_t] + bias) \tag{8}$$

where s_t is the stack embedding obtained by a stack LSTM and b_t is the buffer embedding. More precisely, b_t is the final hidden state of an LSTM that reads the word embeddings of the generated words in the decoder's output buffer.

2.2.1 Stack LSTM

The stack embedding s_t is the hidden state of a stack LSTM (Dyer et al. [2015]) that reads the embeddings of the subtrees on the stack.

Whenever a REDUCE transition action is predicted, the items making up the tree fragment are composed into a single vector, by applying a function to the concatenated embeddings of the head and dependent:

$$c = \tanh(W_3[h:d] + bias) \qquad .$$

This composed vector is then fed as a single input into an LSTM that has *rewound* its hidden state to the previous element that was extended:

$$s_t = h_{next} = LSTM(c, h_{previous}) \qquad .$$

Thus the decoder, $p_\theta(x,z)$, is parametrized by $\theta = (W_1, W_2, W_3, bias)$.

2.3 Gumbel relaxation and Randomized Sampling

To optimize the ELBO of equation (3) both Li et al. [2019] and Kim et al. [2019] use a control variate to reduce the score function estimator's variance (see sections 3.1, 3.3 below). Alternatively, we can optimize the ELBO by using the Gumbel-Softmax approach to latent variables, applying Kingma and Welling [2013]'s reparametrization trick after converting the discrete variable over syntactic trees into a continuous one. In all cases we decompose the ELBO as $L_x = \mathbb{E}_{q_\phi(z)} \log p_\theta(x,z) + H(q(z))$ where the entropy $H(z) = -\mathbb{E}_{q_\phi(z)} \log q_\phi(z)$ is computed exactly; its partial derivative with respect to ϕ is computed by automatic differentiation. Instead of using Monte Carlo samples to estimate the partial derivative with respect to ϕ of the first term $\mathbb{E}_{q_\phi(z)} \log p_\theta(x,z)$ and computing the score function with REINFORCE, following the perturb-and-MAP approach of Papandreou and Yuille [2011], similarly with Corro and Titov [2018], we approximate $\mathbb{E}_{q_\phi(z)} \log p_\theta(x,z)$ with a single sample Monte Carlo estimate that uses a continuous relaxation $\tilde{z}$ of the discrete latent variable:

1. Perturb the arc scores by adding random Gumbel noise:

   $$s_{ij} + g_{ij}$$

   where $g_{ij} = -\log(-\log u_{ij})$ with $u_{ij} \sim Uniform(0,1)$

2. Draw a sample tree z^* by computing the most probable structure with the perturbed weights:

   $$z^* = \operatorname*{argmax}_z \mathbb{1}\{z\ is\ valid\} \sum_{i \neq j} z_{ij}(s_{ij} + g_{ij})$$

   The highest scoring dependency tree z^* is computed with the inside-outside algorithm.

3. Replacing the one-hot argmax operations of the inside-outside algorithm with a softmax, results in a continuous relaxation $\tilde{z}$ of the discrete z^* that is a differentiable function of ϕ.[7]

[7]Note that the perturbations are local and therefore the sample is approximate.

Randomized sampling can be used to compute the hard sample z^* and its soft relaxation $\tilde{z}$, using only the topK most probable subtree paths and a sample of the less likely paths. A differentiable inside algorithm can be derived where for each subtree the topK paths are computed exactly, while a sample of the remaining paths is used for importance weighted Rao-Blackwellized variance reduction of the subtree estimator and bias correction.[8]

3 Appendix

3.1 Evidence Lower Bound

Recall the marginal generative language model of equation (1):

$$\begin{aligned} \log\ p(x) &= \log \sum_z p(x,z) \\ &= \log \sum_z p(x) \cdot p(z|x) \end{aligned}$$

The true posterior $p(z|x)$ is intractable, so we use variatonal inference to learn approximate maximum likelihood estimates of the parameters, θ, ϕ. For any variational density, $q(z)$, the evidence lower bound (ELBO) is given by

$$L_x = \mathbb{E}_{q(z)} \log \frac{p(x,z)}{q(z)} = \mathbb{E}_{q(z)} \log p(x,z) + H(q(z)) \leq \log p(x) \tag{9}$$

where $H(z) = -\mathbb{E}_{q(z)} \log q(z)$ is the entropy. We take as variational density, $q(z)$, the encoder of section 2.1, $q_\phi(z|x)$, with parameters $\phi = (W_e, b_e, W_a^d, b_a^d)$, and jointly optimize the graph-based encoder and transition-based decoder by using Expectation Maximization to maximize the ELBO objective:

$$L_x = \mathbb{E}_{q_\phi(z|x)}\ \log\ \frac{p_\theta(x,z)}{q_\phi(z|x)} \tag{10}$$

3.2 Expectation Maximization (EM)

Repeat until convergence $\{$

(E-Step) For some initial value of ϕ, generate m samples $z^{(1)}, ..., z^{(m)}$ from the encoder

$$q_\phi(z^{(j)}|x;\theta) = \frac{e^{score_\phi(z^{(j)},x)}}{\sum\limits_{all\ valid\ z'} e^{score_\phi(z',x)}} \tag{11}$$

(M-Step) Maximize the ELBO with respect to θ and ϕ:

$$\theta := \underset{\theta}{\operatorname{argmax}} \sum_j \mathbb{E}_{q_\phi(z^{(j)}|x)} \log \frac{p_\theta(x, z^{(j)})}{q_\phi(z^{(j)}|x)} \tag{12}$$

$\}$

[8]This type of randomized dynamic programming VAE with latent trees architecture is mentioned in Fu et al. [2022], though not implemented or derived in detail.

3.3 Optimization

3.3.1 Gradient of ELBO with respect to θ

$$\begin{aligned}\frac{\partial L_x}{\partial \theta} &= \mathbb{E}_{q_\phi(z|x)} \frac{\partial}{\partial \theta} \log \frac{p_\theta(x,z)}{q_\phi(z|x)} \\ &= \mathbb{E}_{q_\phi(z|x)} \frac{\partial}{\partial \theta} \log p_\theta(x,z) \\ &\approx \frac{1}{M} \sum_m \frac{\partial}{\partial \theta} \log p_\theta(x, z^{(m)})\end{aligned}$$

3.3.2 Gradient of ELBO with respect to ϕ

The gradient L_x in equation (10) with respect to the inference network parameters ϕ involves two steps. The gradient of the entropy $-\mathbb{E}_{q(z)} \log q(z)$ can be obtained exactly through automatic differentiation. The gradient of $\mathbb{E}_{q(z)} \log p(x,z)$ is approximated through the score function as follows[9]:

$$\begin{aligned}\frac{\partial}{\partial \phi} \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x,z)] &= \frac{\partial}{\partial \phi} \sum_z q_\phi(z|x) \cdot \log p_\theta(x,z) \\ &= \sum_z \log p_\theta(x,z) \cdot \frac{\partial}{\partial \phi} q_\phi(z|x)\end{aligned}$$

Using the identity $\frac{\partial}{\partial \phi} q_\phi(z|x) = q_\phi(z|x) \frac{\partial}{\partial \phi} \log q_\phi(z|x)$, we get

$$\begin{aligned}\sum_z \log p_\theta(x,z) \cdot \frac{\partial}{\partial \phi} q_\phi(z|x) &= \sum_z q_\phi(z|x) \cdot \frac{\partial}{\partial \phi} \log q_\phi(z|x) \cdot [\log p_\theta(x,z)] \\ &= \mathbb{E}_{q_\phi(z|x)} \log p_\theta(x,z) \cdot \frac{\partial}{\partial \phi} \log q_\phi(z|x) \\ &\approx \frac{1}{M} \sum_m \log p_\theta(x, z^{(m)}) \cdot \frac{\partial}{\partial \phi} \log q_\phi(z^{(m)}|x)\end{aligned}$$

Kim et al. [2019] reduce the variance of the score function estimator by subtracting the control variate $b(x) = \frac{1}{M-1} \sum_{j \neq m} \log p_\theta\ (x, z^{(j)})$:

$$\begin{aligned}\nabla_\phi E_{q_\phi(z|x)} \log p_\theta(x,z) - b(x) &= E_{q_\phi(z|x)} [\log p_\theta(x,z) - b(x)] \nabla_\phi \log q_\phi(z|x) \\ &\approx \frac{1}{M} \sum_m \left[\log p_\theta(x, z^{(m)}) - b(x)\right] \nabla_\phi \log q_\phi(z^{(m)}|x)\end{aligned}$$

References

D. Andor, C. Alberti, D. Weiss, A. Severyn, A. Presta, K. Ganchev, S. Petrov, and M. Collins. Globally normalized transition-based neural networks. *arXiv preprint arXiv:1603.06042*, 2016.

[9]Cf. Mnih and Gregor [2014]

J. Cheng, A. Lopez, and M. Lapata. A generative parser with a discriminative recognition algorithm. *arXiv preprint arXiv:1708.00415*, 2017.

C. Corro and I. Titov. Differentiable perturb-and-parse: Semi-supervised parsing with a structured variational autoencoder. *arXiv preprint arXiv:1807.09875*, 2018.

C. Dyer, M. Ballesteros, W. Ling, A. Matthews, and N. A. Smith. Transition-based dependency parsing with stack long short-term memory. *arXiv preprint arXiv:1505.08075*, 2015.

C. Dyer, A. Kuncoro, M. Ballesteros, and N. A. Smith. Recurrent neural network grammars. *arXiv preprint arXiv:1602.07776*, 2016.

Y. Fu, J. Cunningham, and M. Lapata. Scaling structured inference with randomization. In *International Conference on Machine Learning*, pages 6811–6828. PMLR, 2022.

K. Ganchev, J. Graça, J. Gillenwater, and B. Taskar. Posterior regularization for structured latent variable models. *The Journal of Machine Learning Research*, 11:2001–2049, 2010.

Y. Kim, A. M. Rush, L. Yu, A. Kuncoro, C. Dyer, and G. Melis. Unsupervised recurrent neural network grammars. *arXiv preprint arXiv:1904.03746*, 2019.

D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

T. Koo and M. Collins. Efficient third-order dependency parsers. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1–11, 2010.

B. Li, J. Cheng, Y. Liu, and F. Keller. Dependency grammar induction with a neural variational transition-based parser. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6658–6665, 2019.

C. J. Maddison, A. Mnih, and Y. W. Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

R. McDonald and F. Pereira. Online learning of approximate dependency parsing algorithms. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 81–88, 2006.

R. McDonald, F. Pereira, K. Ribarov, and J. Hajic. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of human language technology conference and conference on empirical methods in natural language processing*, pages 523–530, 2005.

A. Mnih and K. Gregor. Neural variational inference and learning in belief networks. In *International Conference on Machine Learning*, pages 1791–1799. PMLR, 2014.

T. Naseem, H. Chen, R. Barzilay, and M. Johnson. Using universal linguistic knowledge to guide grammar induction. 2010.

G. Papandreou and A. L. Yuille. Perturb-and-map random fields: Using discrete optimization to learn and sample from energy models. In *2011 International Conference on Computer Vision*, pages 193–200. IEEE, 2011.

W. Pei, T. Ge, and B. Chang. An effective neural network model for graph-based dependency parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 313–322, 2015.

D. A. Smith. *Efficient inference for trees and alignments: modeling monolingual and bilingual syntax with hard and soft constraints and latent variables*. The Johns Hopkins University, 2010.

D. A. Smith and J. Eisner. Dependency parsing by belief propagation. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 145–156, 2008.

W. Wang and B. Chang. Graph-based dependency parsing with bidirectional lstm. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2306–2315, 2016.