Bayesian Learning for Neural Dependency Parsing

Ehsan Shareghi,[♣] Yingzhen Li,[♠] Yi Zhu,[♣] Roi Reichart,[◇] Anna Korhonen[♣]
[♣] Language Technology Lab, DTAL, University of Cambridge
[♠]Microsoft Research Cambridge, [◇]Technion, IIT
[♣]{es776, yz568, alk23}@cam.ac.uk
[♠]Yingzhen.Li@microsoft.com, [◇]roiri@technion.ac.il

Abstract

While neural dependency parsers provide state-of-the-art accuracy for several languages, they still rely on large amounts of costly labeled training data. We demonstrate that in the small data regime, where uncertainty around parameter estimation and model prediction matters the most, Bayesian neural modeling is very effective. In order to overcome the computational and statistical costs of the approximate inference step in this framework, we utilize an efficient sampling procedure via stochastic gradient Langevin dynamics to generate samples from the approximated posterior. Moreover, we show that our Bayesian neural parser can be further improved when integrated into a multi-task parsing and POS tagging framework, designed to minimize task interference via an adversarial procedure. When trained and tested on 6 languages with less than $5k$ training instances, our parser consistently outperforms the strong BiLSTM baseline (Kiperwasser and Goldberg, 2016). Compared with the BiAFFINE parser (Dozat et al., 2017) our model achieves an improvement of up to 3% for Vietnamese and Irish, while our multi-task model achieves an improvement of up to 9% across five languages: Farsi, Russian, Turkish, Vietnamese, and Irish.

1 Introduction

Dependency parsing is essential for many Natural Language Processing (NLP) tasks (Levy and Goldberg, 2014; Angeli et al., 2015; Toutanova et al., 2016; Hadiwinoto and Ng, 2017; Marcheggiani et al., 2017). While earlier work on dependency parsing required careful feature engineering (McDonald et al., 2005b; Koo et al., 2008), this has become less of a concern in recent years with the emergence of deep neural networks (Kiperwasser and Goldberg, 2016; Dozat et al., 2017). Nonetheless, an accurate parser still requires a large amount of labeled data for training, which is costly to obtain, while the lack of data often causes overfitting and poor generalization.

Several approaches for parsing in the small data regime have been proposed. These include augmenting input data with pretrained embedding (Dozat et al., 2017; Che et al., 2018), leveraging unannotated data via semi-supervised learning (Corro and Titov, 2018), predicting based on a pool of high probability trees (Niculae et al., 2018; Keith et al., 2018), and transferring annotation or model across languages (Agic et al., 2016; Lacroix et al., 2016; Rasooli and Collins, 2017). Despite the empirical success of these approaches, an inherent problem still holds: The maximum likelihood parameter estimation (MLE) in deep neural networks (DNNs) introduces statistical challenges at both estimation (training), due to the risk of overfitting, and at test time as the model ignores the uncertainty around the estimated parameters. When training data is small these challenges are more pronounced.

The Bayesian paradigm provides a statistical framework which addresses both challenges by (i) including prior knowledge to guide the learning in the absence of sufficient data, and (ii) predicting under the full posterior distribution of model parameters which offers the desired degree of uncertainty by exploring the posterior space during inference. However, this solution comes with a high computational cost, specifically in DNNs, and is often replaced by regularization techniques such as dropout (Srivastava et al., 2014) as well as ensemble learning and prediction averaging (Liu et al., 2018; Che et al., 2018).

Bayesian neural networks (BNNs) have attracted some attention (Welling and Teh, 2011; Hernández-Lobato and Adams, 2015; Li et al., 2016; Gong et al., 2018). Yet, its current application to NLP is limited to language modeling (Gan et al., 2017), and BNNs have not been developed for structured

prediction tasks such as dependency parsing.

In this paper we aim to close this gap and propose the first BNN for dependency parsing (BNNP). To address the costs of inference step, we apply an efficient sampling procedure via stochastic gradient Langevin dynamics (SGLD) (Welling and Teh, 2011). At training, samples from the posterior distribution of the parser parameters are generated via controlled noise injection to the maximum a posteriori (MAP) gradient update. The generated samples are then used during the inference step to create multiple viable parses based on which the final dependency parse is generated.

Another means of directing a model towards more accurate predictions in the small data regime is via multi-task learning (Caruana, 1997). Such a framework allows models for multiple tasks to reinforce each other towards more accurate joint solutions, which is particularly useful when training data is scarce. We hence present a multi-task framework where our BNNP is integrated with a POS tagger through an adversarial procedure designed to guide the two models towards improved joint solutions.

Our experiments with monolingual and delexicalized cross-lingual parsing using the Universal Dependency treebank (Zeman et al., 2017) demonstrate the effectiveness of our approach. Particularly, our BNNP consistently outperforms the single task BiLSTM baseline (Kiperwasser and Goldberg, 2016), while outperforming the BiAFFINE parser (Dozat et al., 2017) by up to 3% on Vietnamese and Irish. Additionally, our multi-task model achieves an improvement of up to 9% over the BiAFFINE parser for five low-resource languages: Farsi, Russian, Turkish, Vietnamese, and Irish.

2 BiLSTM Dependency Parsing

Our parser extends the graph-based BiLSTM parser of Kiperwasser and Goldberg (2016). We briefly review their work and its notable extensions, and then discuss our extension of their architecture and the limitations of MLE training.

Model Given an input word sequence $w_{1:n}$,[1] and its corresponding POS tag sequence $p_{1:n}$, each input word is encoded as a d-dimensional representation,

$$w_i = e(w_i) \circ e(p_i) \circ \hat{e}(w_i),$$

where $e(.)$ denotes trainable embedding, $\hat{e}$ denotes pretrained external embedding, and $\circ$ denotes concatenation. The context dependent representation of each word is generated through a BiLSTM,

$$v_i = \overrightarrow{\text{LSTM}}_{\zeta}(w_{1:i}) \circ \overleftarrow{\text{LSTM}}_{\phi}(w_{n:i}).$$

Here, ζ and ϕ denote the LSTMs' parameters. The resulting sequence of l-dimensional vectors, $v_{1:n}$, is then used for computing the ARC-SCORE matrix $(n \times n)$ where cell ARC-SCORE(i, j) is computed as,

$$V \times \tanh\Big(\underbrace{W^{(arc-head)} v_i}_{\text{head specialized } w_i} + \underbrace{W^{(arc-mod)} v_j}_{\text{modifier specialized } w_j} + b \Big),$$

indicating the likelihood of having the arc from w_i to w_j. Here, V is $(1 \times h)$, b is $(h \times 1)$, and $W^{(.)}$ is a $(h \times l)$ matrix used for creating specialized versions of words in $w_{1:n}$. The h-dimensional specialized representation of w_i as a head or a modifier is generated via multiplication with $W^{(arc-head)}$ and $W^{(arc-mod)}$, respectively. Following a first-order dependence assumption between the arcs, the parsing is done via a dynamic programming solution that finds the dependency parse T^* such that,

$$T^* = \operatorname*{argmax}_T \Big(\text{SCORE}(T) = \sum_{(i,j) \in T} \text{ARC-SCORE}(i, j) \Big).$$

Next, given T^*, for each arc $(i, j) \in T^*$ the LABEL-SCORE(i, j, r) is computed as:

$$\Big(V' \times \tanh(\underbrace{W'^{(arc-head)} v_i}_{\text{head specialized } w_i} + \underbrace{W'^{(arc-mod)} v_j}_{\text{modifier specialized } w_j} + b') \Big)[r],$$

where r is a dependency relation type, $r \in \{r_z\}_{z=1}^{\mathbf{r}}$, V' is $(\mathbf{r} \times h)$, b' is $(\mathbf{r} \times 1)$, and $W'^{(.)}$ is $(h \times l)$. The label for each dependency arc (i, j) is then chosen by a max operation.

Dozat et al. (2017) proposed an extension of the BiLSTM parser by replacing the non-linear transformation of ARC-SCORE and LABEL-SCORE with a linear transformation (BiAFFINE). This was further extended by Che et al. (2018) who utilized contextualized word embeddings (Peters et al., 2018). Both extensions showed success in dependency parsing shared tasks (Zeman et al., 2017, 2018).

Our Neural Parser Architecture Our network architecture extends the BiLSTM model with an additional BiLSTM layer and input signals. While our

[1] We overload the notation w_i, and it refers to both a word and its d-dimensional representation.

architecture is not the core contribution of this paper, we aim to implement our BNNP on a strong architecture. In §5.3 we demonstrate the contribution of these additions to our final results.

In our BNNP, each word is represented as,

$$w_i = e(w_i) \circ e(p_i) \circ \hat{e}(w_i) \circ e(c_i),$$

where, similar to the BIAFFINE parser, $e(c_i)$ is a character-level representation of the word w_i, generated by a BILSTM:

$$e(c_i) = \text{LS}\vec{\text{T}}\text{M}_\alpha(c_{1:|w_i|}) \circ \text{LS}\overleftarrow{\text{T}}\text{M}_\beta(c_{|w_i|:1}).$$

We consider a maximum of two layers of BILSTMs, denoted as 1^{st} and 2^{nd}.[2] The output of the 2^{nd} BILSTM is denoted as,

$$u_i = \text{LS}\vec{\text{T}}\text{M}_\kappa(v_{1:i}) \circ \text{LS}\overleftarrow{\text{T}}\text{M}_\delta(v_{n:i}),$$

where v_i is the encoding generated by the 1^{st} layer of the BILSTMs.

Learning For structured prediction, the objective function is to maximize the margin between the gold structure $\mathbb{T}$ and the other structures. To achieve this, the cost augmented hinge loss is used, where the fixed margin m is replaced by a cost function, $\text{COST}(\mathbb{T}, T)$, sensitive to the local mistakes (here, each local mistake has a constant cost of 1), and computed as:

$$\mathcal{L}_{arc} = \max\Big(0, \text{COST}(\mathbb{T}, T) + \text{SCORE}(T) - \text{SCORE}(\mathbb{T})\Big)$$

For label prediction, hinge loss with $m = 1$ is used (denoted by $\mathcal{L}_{rel}$). We refer to the parser loss as:

$$\mathcal{L}_{parse} = \mathcal{L}_{arc} + \mathcal{L}_{rel}.$$

Beyond MLE Training The point-estimate of DNN parameters is computationally efficient, but ignores the uncertainty around model parameters during learning. This results in an overconfidence over model predictions during the inference phase. A common generic practice to incorporate a degree of uncertainty is to consider an ensemble of models. Indeed, for dependency parsing ensemble learning has shown to improve accuracy (Surdeanu and Manning, 2010; Kuncoro et al., 2016; Che et al., 2018). However, ensembles are computationally demanding due to the large number of participating models. The de-facto approach to overcome this has been to randomly perturb the structure of the network for each training instance by switching off connections between the nodes, a practice known as dropConnect (Wan et al., 2013), or eliminating the nodes entirely, which is known as dropout (Srivastava et al., 2014).

Hinton et al. (2012) demonstrated that dropping out a node with probability ρ from a neural model $\mathcal{M}$ at training, and scaling the same node's output at test time (in the fully connected $\mathcal{M}$), performs very well in practice. While this scaling trick avoids the need for running multiple models at test time, it lacks theoretical guarantees when applied to DNNs. Gal and Ghahramani (2016) showed that dropout can be casted as integrating out the posterior of model parameters, and in this regard is a form of Bayesian approximation. Still, dropout is not as effective when applied to the small data regime (Goodfellow et al., 2016). We hence propose a fully Bayesian approach.

3 Bayesian Neural Parsing

The Bayesian paradigm is appealing in its ability to capture uncertainty around parameters, avoid overfitting, and incorporate prior knowledge to compensate for the lack of sufficient training data in resource lean settings. Dependency parsing may be a natural candidate for Bayesian modeling since a typical sentence can have multiple viable parses corresponding to different grammatical ambiguities, and considering these possibilities have shown to improve predictive accuracy (Niculae et al., 2018; Keith et al., 2018). In the same spirit, Bayesian inference can be interpreted as a statistically grounded means to generate this pool of high quality trees.

In detail, Bayesian inference for parsing computes the *predictive distribution* $P(T|s, \mathcal{D})$[3] of the dependency parse tree T for an input sentence s given the training data $\mathcal{D}$ by *posterior averaging*:

$$P(T|s, \mathcal{D}) = \int P(T|s, \theta) P(\theta|\mathcal{D}) d\theta,$$

where $P(\theta|\mathcal{D}) \propto p(\mathcal{D}|\theta)p(\theta)$ denotes the posterior distribution of the BNNP parameters θ given observations $\mathcal{D}$. Here the prior distribution $p(\theta)$ is a standard Gaussian $\mathcal{N}(0, I)$, and, by assuming i.i.d training datapoints, the likelihood term

[2] Stacking BILSTMs is believed to be helpful. In our case, the addition of a third layer led to overfitting.

[3] When $P(T|s, \theta)$ is replaced by the unnormalized score from the first-order arc-factored model, $P(T|s, \mathcal{D})$ can be interpreted as the expected score assigned to a dependency tree under the full posterior distribution of model parameters.

is $P(\mathcal{D}|\theta) = \prod_{x_i \in D} P(x_i|\theta)$ where $x_i = (s_i, T_i)$ is a training instance.

The above integral is intractable, however an empirical estimate can be computed using $\mathcal{K}$ samples from the posterior,

$$P(T|s,\mathcal{D}) \approx \frac{1}{\mathcal{K}} \sum_{\{\theta_k \sim P(\theta|D)\}_{k=1}^{\mathcal{K}}} P(T|s,\theta_k). \quad (1)$$

Therefore, efficient generation of posterior samples is the key for applying the Bayesian treatment. Since the posterior is intractable in DNNs, additional machineries such as Markov Chain Monte Carlo (MCMC) (Robert and Casella, 2010) are required for sampling. In this paper we explore an efficient class of Stochastic Gradient Markov Chain Monte Carlo (SG-MCMC) methods, designed for NNs by adjusting stochastic gradient descent with properly scaled Gaussian noise to generate posterior samples.

3.1 Stochastic Gradient Langevin Dynamics

We first consider the Stochastic Gradient Langevin Dynamics (SGLD) (Welling and Teh, 2011) sampler to generate posterior samples. Intuitively, this mechanism guides SGD to explore the posterior distribution rather than finding the maximum a posteriori (MAP) solution. More concretely, at each time step t we first compute a stochastic estimate of the gradient for the negative log-posterior distribution on a mini-batch b_t:

$$\nabla_{\theta_t}\mathcal{L} \approx -\left(\nabla_{\theta_t} \log P(\theta_t) + \frac{|\mathcal{D}|}{|b_t|}\sum_{x_i \in b_t} \nabla_{\theta_t} \log P(x_i|\theta_t)\right),$$

and then update the BNNP parameters by SGD with noise injection,

$$\theta_{t+1} \leftarrow \theta_t - \frac{\eta_t}{2}\nabla_{\theta_t}\mathcal{L} + \xi_t, \quad \xi_t \sim \mathcal{N}(0, \eta_t \mathbf{I}) \quad (2)$$

Theoretically, the distribution of θ_t converges to the true posterior $p(\theta|\mathcal{D})$ when $b_t = \mathcal{D}$, $t \to \infty$ and $\eta_t \to 0$ (see Teh et al. (2016) for SGLD convergence rate proofs and analysis).

In practice, the learning rate $\eta_t = a(b+t)^{-\gamma}$, starting high and decaying, plays a crucial role in eqn. 2 dynamics. Intuitively, while the variance of the injected noise is η_t, the variance of the gradient is of $\mathcal{O}(\eta_t^2)$. In the earlier stages of the optimization, the $\frac{\eta_t}{2}\nabla_\theta \mathcal{L}$ term is dominant as gradients are of greater impact and the learning rate is high, simulating stochastic gradient descent. As the optimization proceeds, the impact of the gradient shrinks and the injected Gaussian noise becomes the dominant term, transitioning from SGD to Langevin Monte Carlo (Neal, 2011).

Since $\eta_t \to 0$ might cause slow mixing in practice, we also set a minimum threshold for the learning rate to allow mixing at later stages. SGLD and SGD have the same time complexity as sampling occurs along with parameter updates with a negligible overhead of drawing ξ_ts.

3.2 Preconditioned SGLD

As SGLD relies on a single learning rate along all dimensions of θ, it has the same potential inefficiencies of SGD for optimizing functions with different curvatures along each dimension. Inspired by more advanced optimization techniques that adjust to the geometry of parameter space, Li et al. (2016) proposed preconditioning of SGLD (similar to RMSprop (Tieleman and Hinton, 2012)) which scales the gradients with respect to the weighted average of the gradients along each dimension, such that a unified learning rate is sufficient. The proposed preconditioner diagonal matrix is computed as follows,

$$G(\theta_t) = \text{diag}(1 \oslash (\lambda 1 + \sqrt{V(\theta_t)})),$$
$$V(\theta_t) = \alpha V(\theta_{t-1}) + (1-\alpha)\nabla_{\theta_t}\mathcal{L} \odot \nabla_{\theta_t}\mathcal{L},$$

where $\odot, \oslash$ are element-wise matrix product and division, and λ, α are hyperparameters controlling the extremes of the curvature, and weighting average of previous and current gradients, respectively. The updated preconditioned eqn. 2 is,

$$\theta_{t+1} \leftarrow \theta_t - \frac{\eta_t}{2}G(\theta_t)\nabla_{\theta_t}\mathcal{L} + \xi_t, \quad \xi_t \sim \mathcal{N}(0, \eta_t G(\theta_t)).$$

Here, replacing the preconditioner matrix $G(\theta_t)$ with an identity matrix will recover the SGLD update formulation. In case of a diagonal preconditioner, it has the same time complexity of SGLD, although the constant factors in the complexity figure can be slightly higher due to additional computations involved in $G(\theta_t)$ and $V(\theta_t)$.

While SGLD is designed to explore different modes of the posterior distribution, we found (§5.4) that in practice we still require dropout to facilitate modes exploration via random perturbation of the model structure.

3.3 Prediction

The generated posterior samples can be used to compute the approximate expectation of eqn. 1 by scoring all plausible parses via explicit model averaging.

As this solution is not computationally feasible, we use the sampled parameters and follow a procedure that minimizes the Bayes risk (MBR) (Goodman, 1996). Given each sampled parameter, first we generate the maximum scoring parse using the arc-factored decomposition (McDonald et al., 2005a) and dynamic programming (Eisner, 1996). This can be done concurrently for all samples, resulting in a running time identical to the non-Bayesian approach. For each labelled edge, we replace its score in the ARC-SCORE matrix with its occurrence count in the collection of sampled trees and infer the final tree using counts as scores. The predicted structure is then passed to the label predictor, which assigns labels to the edges (§2). This decoding approach, while selecting the global structure with the highest probability under the approximate posterior, could potentially allow for additional corrections of the highest scoring tree in the pool of samples (Shareghi et al., 2015; Kuncoro et al., 2016).

4 Multi-Task Learning

Multi-task learning (Caruana, 1997) lends itself as a natural choice for low-resource settings as it aims at leveraging the commonality between tasks to improve their performance in the absence of sufficient amount of training data. This framework hence naturally complements Bayesian modeling in dealing with the challenges of the small data regime.

We couple our BNNP with POS tagging due to the strong connection between the two tasks (Rush et al., 2010) and the availability of joint training data in several languages (Zeman et al., 2017). While multi-task frameworks have shown success in some areas (Reichart et al., 2008; Finkel and Manning, 2009; Liu et al., 2016; Malca and Reichart, 2018), in our case we found that our two tasks interfered with each other and degraded the parser performance (see similar findings for other tasks at Søgaard and Goldberg (2016); Plank and Alonso (2017)).

To minimize task interference, an approach shown effective (Ganin and Lempitsky, 2015; Kim et al., 2017; Chen et al., 2017; ZareMoodi and Haffari, 2018) is to implicitly guide the update signals during training via an adversarial procedure that avoids shared parameters contamination. We adapt this idea to our multi-task learning.

Architecture Given an input word, w_i, the score for a POS tag p is computed as $\tanh(W''v_i + b'')[p]$, where $p \in \{p_z\}_{z=1}^{\mathbf{p}}$, W'' is $(\mathbf{p} \times l)$, and b'' is $(\mathbf{p} \times 1)$. To train the POS tagger, the cross-entropy loss is used (denoted by $\mathcal{L}_{pos}$). Taking w_is as input, the *shared* BiLSTM strictly encodes a task-agnostic l-dimensional representation of the input (explained in the next subsection),

$$s_i = \overrightarrow{\text{LSTM}}_\rho(w_{1:i}) \circ \overleftarrow{\text{LSTM}}_\chi(w_{n:i}).$$

The input to each task is then defined as the concatenation of task-agnostic and task-specific representations. We considered a basic architecture where both tasks have their separate parameters (identical number of layers, dimensions, etc.) and they only share the *shared* BiLSTM (denoted as **MULTI TASK** in Table 1). [4]

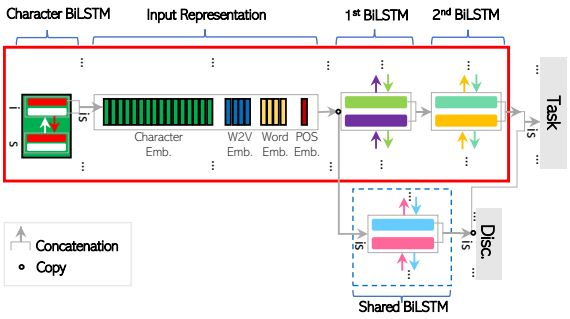


Figure 1: The **MULTI TASK** architecture flow for a single word in a sequence.

Adversarial Training The *shared* BiLSTM output, s_i, is meant to encode a task-agnostic representation of the input word w_i. In order to enforce this criterion, we apply an adversarial training procedure. The shared representation, s_i, is forwarded to a task discriminator (denoted as *Disc.* in Figure 1) through a gradient reversal layer. The task discriminator predicts the task identity for each word in the input via a linear transformation $(W'''s_i + b''')[\tau]$ followed by a softmax, where $\tau \in \{\texttt{parser}, \texttt{tagger}\}$ and W''' and b''' are $(2 \times l)$ and (2×1), respectively. To train the discriminator, a sum of the cross-entropy losses for $w_{1:n}$ is used (denoted by $\mathcal{L}_{disc}$). As the parameters of the discriminators are being updated, the gradient signals to minimize the discriminator's error are backpropagated with an opposite sign to the *shared* BiLSTM layer, which adversarially encourages the *shared* BiLSTM to fool the discriminator.

Our training schedule alternates between the two modes, in one mode optimizing the shared and task-specific parameters based on $\mathcal{L}_{parse}$ and $\mathcal{L}_{pos}$ (in

[4] We also tried layer-wise placements of tasks (Søgaard and Goldberg, 2016; Hashimoto et al., 2017) and the results were slightly worse. Details are omitted for space reason.

random order), while in the other mode optimizing $\mathcal{L}_{disc}$ which includes the shared and discriminator-specific parameters.

5 Experiments and Results

We experiment with mono-lingual and cross-lingual dependency parsing using the treebanks of the CoNLL 2017 shared task on parsing to Universal Dependencies (UD) (Zeman et al., 2017).[5]

5.1 Experimental Setup

We use the UDPipe baseline outputs for segmentation and POS tagging of the raw test data (released along with the raw test data). While segmentation and POS errors substantially impact the quality of the final predicted parse, their exploration is beyond our scope. Our evaluation metric is Labeled Attachment Score (LAS), computed by the shared task evaluation script. Statistical significance, when mentioned, is computed over 20 runs, via the Kolmogorov-Smirnov test (Reimers and Gurevych, 2017; Dror et al., 2018) with $p = 0.01$.

Mono-Lingual Experiments We experiment with Persian (fa), Korean (ko), Russian (ru), Turkish (tr), Vietnamese (vi) and Irish (ga), all with less than $5k$ training sentences (Table 1). For comparison we report the scores published by the top system of the CoNLL 2017 shared task, BIAFFINE (Dozat et al., 2017), noting the following differences between their input and output and ours. The BIAFFINE parser: (i) uses the UDPipe outputs for segmentation but corrects POS errors before parsing, (ii) includes both language specific and universal POS tags in the input layer while we only include the universal POS tags, and (iii) applies post-process correction for non-projective languages.

Cross-Lingual Experiments We use the English (en), French (fr), Russian (ru), and Persian (fa) datasets of the UD treebanks as our training and test data, with the addition of 3 languages for which we did not have any training data: Kurmanji (kmr), Buriat (bxr), and Northern Sami (sme). We also report the results for each language, where the combination of training datasets for the rest of the languages (marked as +) was used for training. The cross-lingual experiments are done on delexicalized parses after replacing the words with their Universal POS tags.

Models and Baselines - Single-Task We consider the following models: **BASE** is the BILSTM model of Kiperwasser and Goldberg (2016); **BASE++** extends BASE by having 2 layers of BILSTMs and using 1 layer of character level BILSTM (§2); **+SHARED** includes an additional BILSTM (dashed box in Figure 1). We included this to provide a fair comparison (in terms of the number of parameters) with the multi-task experiments but we apply a higher dropout rate to resolve overfitting; **ENSEMBLE** denotes a collection of 9 +SHARED models each randomly initialized (Reimers and Gurevych, 2017) and trained for MLE with MBR (§3.3) applied for prediction; **MAP** denotes the +SHARED model optimized for MAP instead of MLE; **+SGLD** denotes Bayesian learning and prediction (§3.1), and **+PRECOND** denotes preconditioned SGLD (§3.2). **+SGLD** and **+PRECOND** are applied to the +SHARED model.

Models and Baselines - Multi-Task We consider the same models as in the single-task setup, with the following changes. **BASEMT** is the variant of **MULTI TASK** architecture where the *shared* BILSTM is removed and all components except for decoders are shared between the two tasks. **+SHARED** and **+ADV** denote the results without and with the adversarial training (§4), respectively.

5.2 Hyperparameters

Training is done for 330 epochs with early stopping, using successive[6] mini-batches of 5000 words,[7] with drop-out rate 0.33 unless stated otherwise. SGLD is only applied for learning the parameters of the parser, with learning rate decaying from 0.01 to 0.0001 ($\gamma = 0.5$), and the preconditioned matrix hyperparameters are $\lambda = 10^{-5}, \alpha = 0.99$. The rest of the parameters are updated using Adam (Kingma and Ba, 2014) with default DyNet parameters (Neubig et al., 2017). The size and selection of the samples used in Bayesian inference are tuned on Irish (see §6.1 of the main paper). In all experiments, training stops based on loss convergence of a single model on the dev set. We also tried GRUs and our results were substantially worse than LSTMs.

We consider the following sizes within the models: **BASE** is 1 layer of BILSTM with 125 hidden

[5] For train and dev sets (1-1983), test set (1-2184), and pretrained embeddings (1-1989) see: https://lindat.mff.cuni.cz/repository/xmlui/handle/11234/{1-1983,1-2184,1-1989}

[6] We found that random batches perform slightly worse.

[7] When the 5000 words limit lands in the middle of a sentence, the entire sentence is included.

units, 100-D word embeddings, 25-D POS embeddings, and dropout rate of 0.33; **BASE++** extends BASE by having 2 layers of BiLSTMs with 200-D hidden units, 100-D external word embeddings, and using 1 layer of character level BiLSTM with 200-D hidden units to generate 100-D character embeddings (§2); **+SHARED** includes an additional BiLSTM with 200-D hidden units in BASE++ to provide a fair comparison (in terms of the number of parameters) with the multi-task experiments but we apply a higher dropout rate of 0.66 to resolve overfitting.

5.3 Results

Mono-Lingual Parsing Table 1 summarizes the training data statistics and the LAS results of the various models, with bold-font marking cases where a model outperforms the BiAFFINE parser.

1. SINGLE TASK Comparison between BASE++ and +SHARED reveals that the additional BiLSTM in the +SHARED does not improve the results. As expected, model averaging in ENSEMBLE improved the results over the +SHARED model. Additionally, the MAP results show slight improvements compared with +SHARED. Both of these findings suggest that the quality of predictions is likely to improve if some notion of uncertainty (via averaging, or prior insertion) is included.

For Bayesian solutions, both +SGLD and +PRECOND consistently (and in a statistically significant manner) outperform the ENSEMBLE models and the strong BiLSTM baselines on all languages, while PRECOND outperforms BiAFFINE by up to 3% on two languages (vi, and ga). The consistency of improvements as we incorporate richer means of capturing the uncertainty suggests that these gains are independent of our specific choice of neural architecture and that they might also hold for future parsing architectures.

2. MULTI TASK As explained earlier (§4), comparing the parser performances under +SHARED models in single and multi task settings indicates that parser quality was degraded with the inclusion of a POS tagging task, possibly due to interference between the tasks. This issue was alleviated with the inclusion of task discriminator and the adversarial training. The +ADV model consistently improves the results (in a statistically significant manner), while outperforming the **SINGLE TASK (+SHARED)** by 2.2% for Persian (fa), and 5.2% for Irish (ga). This is an indicator that low-resource languages could potentially benefit more from multi-task learning. Other variants of +ADV are also reported in Table 1 and similar patterns to the single task setting are observed.

	fa	ko	ru	tr	vi	ga
TRAIN(sen.)	4798	4400	3850	3685	1400	566
	Labeled Attachment Score (LAS)					
BiAFFINE	86.31	**82.49**	83.65	62.79	42.13	70.06
BASE	80.97	64.76	75.45	52.64	39.36	62.50
SINGLE TASK						
BASE++	83.15	76.70	79.44	58.92	41.03	66.58
+SHARED	83.11	76.32	79.62	58.72	41.10	66.42
ENSEMBLE	84.12	77.28	80.17	59.36	41.89	68.12
MAP	83.59	76.61	79.78	59.13	41.33	66.95
+SGLD	84.98	78.91	80.86	60.53	**43.12**	69.51
+PRECOND	85.76	79.83	81.9	61.71	**44.52**	**70.91**
MULTI TASK						
BASEMT	81.54	75.78	78.61	57.12	40.08	60.51
+SHARED	81.03	75.08	78.64	57.09	40.04	65.24
+ADV	84.93	78.12	81.23	60.66	**43.11**	69.89
ENSEMBLE	85.01	78.31	81.56	60.92	**43.3**	70.00
MAP	85.02	78.26	81.59	60.72	**43.25**	**70.36**
+SGLD	85.89	79.50	83.06	62.13	**44.57**	**72.52**
+PRECOND	**86.75**	80.97	**84.51**	**63.24**	**45.96**	**74.12**

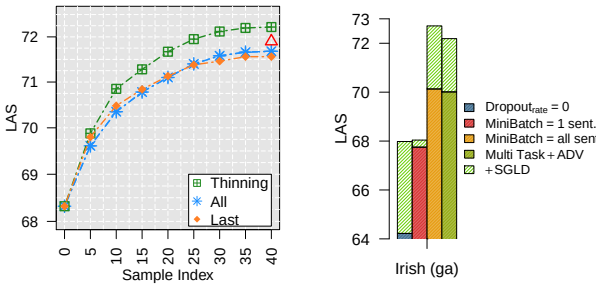
Table 1: Mono-Lingual results. Models that outperform the BiAFFINE parser are highlighted in bold.

Next, we test the effectiveness of Bayesian learning and inference using $\mathcal{K} = 9$ thinned samples (§5.4). For all languages in the multi-task setting the gain from including +SGLD in +ADV is statistically significant. However, the gain decays as the amount of training data increases: from 3.8% on Irish (ga) to 1.1% on Persian (fa). This verifies our expectation that Bayesian learning helps parameter estimation and prediction in the small data regime. Compared to +SGLD, +PRECOND provides further improvements (all are statistically significant) of 2.2% on Irish and 1% on Persian, showing a similar generic negative correlation with data size.

To summarize our mono-lingual results, the Bayesian framework shows its merit in improving predictive quality, and multi-task learning introduces new and informative signal via a related task which allows for better parameter estimation. The integration of a Bayesian parser into a multi-task learning framework outperforms the BiAFFINE parser on up to 5 languages, while we observe decreasing improvements as the data size grows. Also, since our Bayesian approach builds on the BASE model, it is subject to the shortcomings of this model: For instance, on Korean (ko) the difference between BASE and BiAFFINE is too large to be closed without further adjustment of model design,

	sen.	Labeled Attachment Score (LAS)						
		en	fr	ru	fa	kmr	bxr	sme
en	13k	81.04 +0.94	64.34 +1.34	44.61 +0.99	24.57 +1.05	27.62 +1.21	24.97 +1.39	33.42 +1.16
fr	15k	57.71 +0.89	79.14 +1.21	47.72 +0.82	29.23 +1.02	28.06 +1.39	18.04 +1.02	29.72 +1.01
ru	3k	52.25 +3.26	62.30 +2.58	77.55 +3.41	46.47 +2.61	27.80 +3.14	39.29 +2.75	32.40 +3.36
fa	5k	37.53 +1.30	50.94 +1.97	43.41 +1.82	78.28 +1.01	45.29 +1.66	21.65 +1.36	23.94 +1.41
+	36k	82.79 +0.78	80.92 +1.16	77.19 +0.87	78.25 +0.78	43.26 +0.96	20.57 +0.94	35.00 +0.82

Table 2: Cross-Lingual Parsing Evaluation. LAS of an ensemble of $\mathcal{K} = 9$ **SINGLE TASK** (BASE++) models, and the performance gain by the Bayesian **MULTI TASK** (+PRECOND), placed as subscript.



(a) Performance of different sampling strategies.

(b) Dropout, SGLD, and mini-batch size interdependence.

Figure 2: Ablation analysis for Irish. Similar patterns are observed for the other languages.

or its inputs and outputs.

Cross-Lingual Delexicalized Parsing We tested our most successful mono-lingual model, **MULTI TASK**(+PRECOND), in the cross-lingual setup against an ensemble of $\mathcal{K} = 9$ randomly initialized **SINGLE TASK** (BASE++) models. Table 2 summarizes the training data size (rounded up) and the LAS results of both models.

As expected, the ensemble models perform best when the training and test data come from the same language or language family (as in Buryat to Russian and Kurmanji to Persian). Showing a similar pattern to mono-lingual parsing, in all cases the performance gain of the Bayesian **MULTI TASK** (+PRECOND) setting, reported as under-text in the table, consistently outperforming the ENSEMBLE (in a statistically significant manner) and the impact decays with the training set size.

5.4 Ablation Analysis

We aim to answer two questions: (i) how many of the generated samples are used during inference? and (ii) how much the success of SGLD depends on other sources of noise during training?

1. UTILIZING SAMPLES Once the training stops, different strategies could be applied to utilize the collection of $\mathcal{M}$ posterior samples to recompute ARC-SCORE matrix (see §3.3). An extreme option is including all samples (denoted by *All*), another option is to only use the last sample (denoted by *Last*). An intermediate alternative is to choose $\mathcal{K}$ samples with interval $\lfloor \frac{\mathcal{M}}{\mathcal{K}} \rfloor$, an approach known as *Thinning* (Gan et al., 2017).

Comparing the results in Figure 2a, *Thinning* consistently outperforms the other strategies. Interestingly, when we compared *Thinning* with a variant (denoted by Δ in Figure 2a) that just uses the last 5 samples , it appears that *Thinning* which includes earlier samples is still superior. This is likely to be an indicator that a larger (and more diverse) set of trees can potentially improve MBR decoding. In practice, we use *Thinned* samples.

2. DROPOUT, SGLD, AND MINI-BATCH We would next like to better understand the interdependence between SGLD and two existing sources of noise: dropout and mini-batch size. Results are reported in Figure 2b. In all four stacked bars, the top segment demonstrates the gain when including +SGLD. The right-most stacked bar illustrates the performance of **MULTI TASK (+ADV)** without (bottom part of the bar) and with +SGLD. This is our reference and in the following three experiments, we make a single change to this model while keeping everything else fixed.

In the left-most stacked bar, we exclude dropout. This significantly hurts the performance, an indication that the random perturbation of model by dropout can provide a positive complimentary effect for better exploration of the posterior modes via SGLD. Comparing the top parts of all stacked bars (representing the gain with +SGLD) reveals that this gain is at its maximum when dropout is switched off. An interpretation can be that in the absence of dropout, SGLD becomes the main component for countering overfitting.

In the second stacked bar from the right, we remove the noise caused by mini-batching and use the entire dataset as one batch. This slightly improves the results, although this improvement is not statistically significant. Based on eqn. 2, full training data gradient updates cause $\nabla_\theta \mathcal{L} \rightarrow 0$ in fewer steps. Consequently, the learning rate (which controls the variance of the noise) is at a higher

value when the gradient is close to zero, giving a better chance for the sampler to mix via ξ_t.

In the second stacked bar from left, parameters are updated for each training sentence. In this setting the learning rate, η_t, is consumed much faster and can potentially reach zero on the MAP solution, discarding the effect of the injected noise and +SGLD. Hence, the gain for including SGLD is at its minimum. We speculate that adjusting the decaying speed of the learning rate (which we left untouched) could allow for this extreme case to perform better.

6 Conclusion

We proposed a Bayesian framework for neural dependency parsing (BNNP). We employ efficient SG-MCMC approximate inference mechanisms through stochastic gradient Langevin dynamics to generate posterior samples during optimization. The collected samples are then used via a minimum Bayes risk parsing algorithm to generate the final parse tree. In mono-lingual and cross-lingual experiments in the small data regime, where Bayesian learning in expected to be most effective, our BNNP consistently outperformed the strong BiLSTM baselines. Moreover, when integrating the BNNP into a multi-task learning framework, utilized to prevent task interference, we outperformed the BiAFFINE parser (best system of the CoNLL17 shared task) on 5 low-resource languages by up to 9% LAS.

In future work, we intend to investigate other types of priors over the network parameters (e.g., sparse priors (Lobacheva et al., 2017)). We would also like to explicitly quantify the uncertainty captured in our framework under different sampling strategies or MCMC-SG methods (e.g., similar to McClure and Kriegeskorte (2016); Teye et al. (2018)).

Acknowledgments

This work is supported by the ERC Consolidator Grant LEXICAL (648909). The first author would like to thank Costanza Conforti, Victor Prokhorov, and Gamal Crichton for their comments on the presentation of this work. The authors would like to thank the anonymous reviewers for their helpful suggestions.

References

Zeljko Agic, Anders Johannsen, Barbara Plank, Héctor Martínez Alonso, Natalie Schluter, and Anders Søgaard. 2016. Multilingual projection for parsing truly low-resource languages. *TACL*.

Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D. Manning. 2015. Leveraging linguistic structure for open domain information extraction. In *Proceedings of ACL*.

Rich Caruana. 1997. Multitask learning. *Machine learning*.

Wanxiang Che, Yijia Liu, Yuxuan Wang, Bo Zheng, and Ting Liu. 2018. Towards better UD parsing: Deep contextualized word embeddings, ensemble, and treebank concatenation. In *Proceedings of CoNLL*.

Xinchi Chen, Zhan Shi, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-criteria learning for chinese word segmentation. In *Proceedings of ACL*.

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

Timothy Dozat, Peng Qi, and Christopher D. Manning. 2017. Stanford's graph-based neural dependency parser at the conll 2017 shared task. In *Proceedings of CoNLL*.

Roten Dror, Gili Baumer, Segev Shlomov, and Roi Reichart. 2018. The hitchhiker's guide to testing statistical significance in natural language processing. In *Proceedings of ACL*.

Jason M Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of COLING*.

Jenny Rose Finkel and Christopher D Manning. 2009. Joint parsing and named entity recognition. In *Proceedings of NAACL*.

Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of ICML*.

Zhe Gan, Chunyuan Li, Changyou Chen, Yunchen Pu, Qinliang Su, and Lawrence Carin. 2017. Scalable bayesian learning of recurrent neural networks for language modeling. In *Proceedings of ACL*.

Yaroslav Ganin and Victor S. Lempitsky. 2015. Unsupervised domain adaptation by backpropagation. In *Proceedings of ICML*.

Wenbo Gong, Yingzhen Li, and José Miguel Hernández-Lobato. 2018. Meta-learning for stochastic gradient MCMC. *CoRR*.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press.

Joshua Goodman. 1996. Parsing algorithms and metrics. In *Proceedings of ACL*.

Christian Hadiwinoto and Hwee Tou Ng. 2017. A dependency-based neural reordering model for statistical machine translation. In *Proceedings of AAAI*.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A joint many-task model: Growing a neural network for multiple NLP tasks. In *Proceedings of EMNLP*.

José Miguel Hernández-Lobato and Ryan P. Adams. 2015. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *Proceedings of ICML*.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*.

Katherine A. Keith, Su Lin Blodgett, and Brendan T. O'Connor. 2018. Monte carlo syntax marginals for exploring and using dependency parses. In *Proceedings of NAACL*.

Joo-Kyung Kim, Young-Bum Kim, Ruhi Sarikaya, and Eric Fosler-Lussier. 2017. Cross-lingual transfer learning for POS tagging without cross-lingual resources. In *Proceedings of EMNLP*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *TACL*.

Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *Proceedings of ACL*.

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, and Noah A. Smith. 2016. Distilling an ensemble of greedy dependency parsers into one MST parser. In *Proceedings of EMNLP*.

Ophélie Lacroix, Lauriane Aufrant, Guillaume Wisniewski, and François Yvon. 2016. Frustratingly easy cross-lingual transfer for transition-based dependency parsing. In *Proceedings of NAACL*.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of ACL*.

Chunyuan Li, Changyou Chen, David E. Carlson, and Lawrence Carin. 2016. Preconditioned stochastic gradient langevin dynamics for deep neural networks. In *Proceedings of AAAI*.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Recurrent neural network for text classification with multi-task learning. In *Proceedings of IJCAI*.

Yijia Liu, Yi Zhu, Wanxiang Che, Bing Qin, Nathan Schneider, and Noah A. Smith. 2018. Parsing tweets into universal dependencies. In *Proceedings of NAACL*.

Ekaterina Lobacheva, Nadezhda Chirkova, and Dmitry P. Vetrov. 2017. Bayesian sparsification of recurrent neural networks. *CoRR*.

Rivka Malca and Roi Reichart. 2018. Neural transition based parsing of web queries: An entity based approach. In *Proceedings of EMNLP*.

Diego Marcheggiani, Anton Frolov, and Ivan Titov. 2017. A simple and accurate syntax-agnostic neural model for dependency-based semantic role labeling. In *Proceedings of CoNLL*.

Patrick McClure and Nikolaus Kriegeskorte. 2016. Representation of uncertainty in deep neural networks through sampling. *CoRR*.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005a. Online large-margin training of dependency parsers. In *Proceedings of ACL*.

Ryan T. McDonald, Koby Crammer, and Fernando C. N. Pereira. 2005b. Online large-margin training of dependency parsers. In *Proceedings of ACL*.

Radford M Neal. 2011. MCMC using hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, et al. 2017. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*.

Vlad Niculae, André F. T. Martins, Mathieu Blondel, and Claire Cardie. 2018. Sparsemap: Differentiable sparse structured inference. In *Proceedings of ICML*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL*.

Barbara Plank and Héctor Martínez Alonso. 2017. When is multitask learning effective? semantic sequence prediction under varying data conditions. In *Proceedings of EACL*.

Mohammad Sadegh Rasooli and Michael Collins. 2017. Cross-lingual syntactic transfer with limited resources. *TACL*.

Roi Reichart, Katrin Tomanek, Udo Hahn, and Ari Rappoport. 2008. Multi-task active learning for linguistic annotations. In *Proceedings of ACL*.

Nils Reimers and Iryna Gurevych. 2017. Reporting score distributions makes a difference: Performance study of lstm-networks for sequence tagging. In *Proceedings of EMNLP*.

Christian P. Robert and George Casella. 2010. *Monte Carlo Statistical Methods*.

Alexander M. Rush, David Sontag, Michael Collins, and Tommi S. Jaakkola. 2010. On dual decomposition and linear programming relaxations for natural language processing. In *Proceedings of EMNLP*.

Ehsan Shareghi, Gholamreza Haffari, Trevor Cohn, and Ann Nicholson. 2015. Structured prediction of sequences and trees using infinite contexts. In *Proceedings of ECML*.

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of ACL*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*.

Mihai Surdeanu and Christopher D. Manning. 2010. Ensemble models for dependency parsing: Cheap and good? In *Proceedings of NAACL*.

Yee Whye Teh, Alexandre H. Thiery, and Sebastian J. Vollmer. 2016. Consistency and fluctuations for stochastic gradient langevin dynamics. *JMLR*.

Mattias Teye, Hossein Azizpour, and Kevin Smith. 2018. Bayesian uncertainty estimation for batch normalized deep networks. In *Proceedings of ICML*.

Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*.

Kristina Toutanova, Victoria Lin, Wen-tau Yih, Hoifung Poon, and Chris Quirk. 2016. Compositional learning of embeddings for relation paths in knowledge base and text. In *Proceedings of ACL*.

Li Wan, Matthew D. Zeiler, Sixin Zhang, Yann LeCun, and Rob Fergus. 2013. Regularization of neural networks using dropconnect. In *Proceedings of ICML*.

Max Welling and Yee W Teh. 2011. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of ICML*.

Poorya ZareMoodi and Gholamreza Haffari. 2018. Neural machine translation for bilingually scarce scenarios: a deep multi-task learning approach. In *Proceedings of NAACL*.

Daniel Zeman, Jan Hajič, Martin Popel, Martin Potthast, Milan Straka, Filip Ginter, Joakim Nivre, and Slav Petrov. 2018. Conll 2018 shared task: Multilingual parsing from raw text to universal dependencies. In *Proceedings of the CoNLL*.

Daniel Zeman, Martin Popel, Milan Straka, Jan Hajic, Joakim Nivre, Filip Ginter, Juhani Luotolahti, Sampo Pyysalo, Slav Petrov, et al. 2017. Conll 2017 shared task: Multilingual parsing from raw text to universal dependencies. In *Proceedings of CoNLL*.