Semi-Supervised Dependency Parsing with Arc-Factored Variational Autoencoding

Ge Wang, Kewei Tu*
School of Information Science and Technology, ShanghaiTech University, Shanghai, China
Shanghai Engineering Research Center of Intelligent Vision and Imaging
Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences
University of Chinese Academy of Sciences
{wangge,tukw}@shanghaitech.edu.cn

Abstract

Mannual annotation for dependency parsing is both labourious and time costly, resulting in the difficulty to learn practical dependency parsers for many languages due to the lack of labelled training corpora. To compensate for the scarcity of labelled data, semi-supervised dependency parsing methods are developed to utilize unlabelled data in the training procedure of dependency parsers. In previous work, the autoencoder framework is a prevalent approach for the utilization of unlabelled data. In this framework, training sentences are reconstructed from a decoder conditioned on dependency trees predicted by an encoder. The tree structure requirement brings challenges for both the encoder and the decoder. Sophisticated techniques are employed to tackle these challenges at the expense of model complexity and approximations in encoding and decoding. In this paper, we propose a model based on the variational autoencoder framework. By relaxing the tree constraint in both the encoder and the decoder during training, we make the learning of our model fully arc-factored and thus circumvent the challenges brought by the tree constraint. We evaluate our model on datasets across several languages and the results demonstrate the advantage of our model over previous approaches in both parsing accuracy and speed.

1 Introduction

Dependency parsing is the task of finding syntactic dependency relations between words in sentences (Kübler et al., 2009). For each sentence, the dependency arcs between words are constrained to form a tree structure. A main bottleneck for learning a practical dependency parser is the lack of adequate training corpora as labelling raw text with dependency trees is both labourious and time costly. Semi-supervised dependency parsing utilizes unlabelled data to compensate the scarcity of labelled data in training dependency parsers (Sagae and Tsujii, 2007; Chen et al., 2009; Suzuki et al., 2011; Li et al., 2014).

In previous work, the autoencoder framework is a prevalent approach for the utilization of unlabelled data (Ammar et al., 2014; Kingma and Welling, 2014). When this framework is applied to dependency parsing, training sentences are reconstructed from a decoder conditioned on dependency trees predicted by an encoder. Concretely, the autoencoder approaches to dependency parsing are mainly divided into two categories: the Conditional Random Field (CRF) Autoencoder (Cai et al., 2017) and the Variational Autoencoder (VAE) (Corro and Titov, 2019; Li et al., 2019). The CRF autoencoder predicts the dependency structure with the encoder and tries to reconstruct the input sentence based on the predicted structure. The variational autoencoder assumes the decoder is a generative model which generates the observed sentence from a group of latent variables containing information of the dependency tree, while the encoder tries to infer the posterior of the latent variables from the observed sentence.

The tree structure constraint of the dependency parsing brings challenges to approaches of both categories. For the CRF autoencoder, the encoder predicts the dependency tree structure using dynamic programming with the time complexity of $O(n^3)$ (Eisner, 1996), which is quite time-consuming in practice. For the VAE, the learning objective function contains an expectation over the posterior of the parse

* Corresponding author.

Proceedings of the 28th International Conference on Computational Linguistics, pages 2485–2496
Barcelona, Spain (Online), December 8-13, 2020

tree modeled by the encoder and sophisticated sampling techniques have to be applied to approximate it in previous work.

Another problem in previous approaches is related to the incorporation of contextual information in the decoding procedure. The decoder of the CRF autoencoder of Cai *et al.* (2017) independently generates each word of the sentence using only information from its dependency head, which is linguistically unrealistic. The decoder in the VAE approaches uses complicated models such as Long Short-Term Memory (LSTM) network (Hochreiter and Schmidhuber, 1997) and Graph Convolutional Network (GCN) (Kipf and Welling, 2017) to leverage context information, but only part of the context can be used as the generation is constrained to follow the left-to-right order.

In this work, to handle the challenges mentioned above, we propose our method based on the VAE framework with the following features. We relax the tree constraint in both encoding and decoding during training and make the learning of our model fully arc-factored. For the encoder, the relaxation of the tree restriction transforms the task of finding the most probable tree to head selection for each token, hence eliminating the need for the time-consuming dynamic programming parsing algorithms as well as the need to sample tree structures when computing the expectation in VAE training. This makes training faster and stabler. The decoder is also arc-factored similar to that used by the CRF autoencoder, but an important difference is that we introduce a continuous latent variable for each head word representing its contextual information from both left and right to influence the generation of each child word.

We evaluate our model on datasets of several languages and the experiment results demonstrate that our model is effective in utilizing unlabelled data and achieves better parsing accuracy and faster speed than previous work.

2 Related Work

Dependency parsing is a classic research topic in the Natural Language Processing (NLP) community. Because of the difficulty in obtaining dependency annotations, approaches to dependency parsing with scarce or even no labelled training data have been widely studied. Most of such approaches(Naseem et al., 2010; Tu and Honavar, 2012; Jiang et al., 2016; Noji et al., 2016) are extended from Dependency Model with Valence (DMV) proposed by Klein and Manning (2004), a generative model adapting the Expectation-Maximization (EM) algorithm for its parameter optimization. Limited by strong context-free assumption, DMV and its variants fail to capture useful contextual information in the sentence when scoring dependency parses. There are attempts made to incorporate discriminative information with DMV to remedy this problem (Han et al., 2019), leading to models similar to autoencoders. Another line of research is to incorporate traditional machine learning methods for unlabelled data utilization, such as self-training and tri-training (McClosky et al., 2006; Clark et al., 2018; Søgaard and Rishøj, 2010). A third line of research is to utilize the autoencoder framework, as discussed in Section 1.

Previous work has already broadly applied the autoencoder framework in many NLP tasks other than dependency parsing such as Part-Of-Speech (POS) tagging (Zhang et al., 2017a) and sentence generation (Guu et al., 2018). Among them, VAE has been proved to be a useful tool in modelling problems with latent representations. However, compared with the CRF autoencoder, it is more difficult to use VAE in tasks involving latent structures. The main reason is that VAE requires marginalizing all latent variables (approximated by Monte Carlo sampling in implementation), which is intractable when both continuous and structural latent variables are present.

Corro and Titov (2019) alleviate the problem by first dividing the latent variables into two parts: the discrete ones containing structural information and the continuous ones containing sentence content information. Then they develop a sophisticated sampling method to the approximately marginalize the structural latent variables and employ the Gumbel-Softmax trick (Jang et al., 2017) to facilitate back-propagation in the training process of the structural variables while the continuous variables are treated with traditional VAE procedures. In the area of transition-based dependency parsing and semantic parsing, the REINFORCE algorithm is introduced to VAE-based models for the marginalization of structural latent variables (Yin et al., 2018; Li et al., 2019).

On the other hand, Chen *et al.* (2018), which apply VAE in structure related semi-supervised sequence

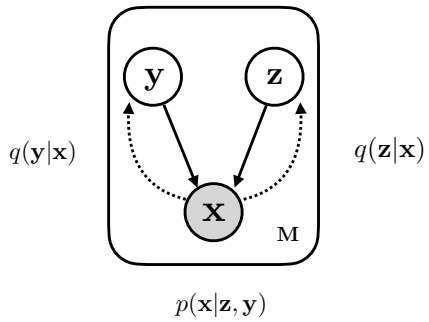

Figure 1: The graphical model representation. The observed variables are shaded. M is the number of sentences in the dataset.

labeling tasks, avoid the marginalization of the structural variables by completely ignoring the structural constraints in model learning. Their success suggests the possibility of trying a similar solution for dependency parsing. In fact, in fully supervised parsing, Zhang *et al.* (2017b) and Dozat and Manning (2017) have shown that it is possible to relax the tree structure constraint during the training of dependency parsers. Further studies (Zhang et al., 2019) demonstrate that dropping the tree constraint only causes minor impact to the parsing accuracy.

3 Model

Inspired by the previous work mentioned in section 2, we propose our semi-supervised dependency parsing model based on the VAE framework. Following the classic VAE setting, our model assumes that all the observed data are generated from a generative model based on a set of latent variables. The generative model forms the decoder part of VAE while the encoder part tries to infer the posterior of the latent variables.

Formally, for an observed sentence represented by a sequence of tokens $\mathbf{x} = x_{1:T}$, we assume it is generated from a sequence of continuous latent vectors $\mathbf{z} = z_{1:T}$ according to a latent dependency tree structure $\mathbf{y}$. $\mathbf{y}$ and $\mathbf{z}$ are assumed to be independent. The tree structure $\mathbf{y}$ is also represented as a sequence by defining $\mathbf{y} = y_{1:T}$, where y_i is the index of the dependency head for x_i. We generate each token x_i conditioned on the latent vector z_{y_i} that corresponds to the dependency head of x_i. We use $y_i = 0$ to denote the root token of the dependency tree, the probability of generating the whole sentence is the product of arc-wise generation probabilities: $P(\mathbf{x}|\mathbf{z},\mathbf{y}) = \prod_{i=1}^{T} p(x_i|z_{y_i})$. Note that although we follow Corro and Titov (2019) to divide the latent variables into discrete $\mathbf{y}$ and continuous $\mathbf{z}$, an important difference is that we define $\mathbf{z}$ as a sequence of continuous vectors respectively assigned to each token in the sentence, while Corro and Titov (2019) define a single continuous vector representing the whole sentence. Our definition of $\mathbf{z}$ is intended to make arc-factored autoencoding possible.

Maximizing the log likelihood of the observed sentence $\log P_{\boldsymbol{\Theta}}(\mathbf{x})$ with distribution parameter Θ requires marginalizing all the latent variables, which is intractable with the presence of continuous latent variables. Therefore, VAE seeks to instead maximize the lower bound of the likelihood by introducing an auxiliary variational distribution $Q_{\Phi}(\mathbf{z},\mathbf{y}|\mathbf{x})$ that is parameterized by Φ. We assume that Q_{Φ} is also arc-factored: $Q(\mathbf{z},\mathbf{y}|\mathbf{x}) = Q(\mathbf{z}|\mathbf{x})Q(\mathbf{y}|\mathbf{x}) = \prod_{i=1}^{T} q(y_i|\mathbf{x})q(z_i|\mathbf{x})$. Note that by assuming arc-factorization with respect to $\mathbf{y}$, we can no longer enforce that $\mathbf{y}$ represents a valid tree structure. This relaxation, however, leads to tractable computation. We illustrate the graphical model representation of $\mathbf{x}$,$\mathbf{y}$,$\mathbf{z}$ in Fig. 1. The derivation of the lower bound, formally known as the Evidence Lower Bound Objective

(ELBO), follows that of the vanilla VAE with the additional discrete latent variable $\mathbf{y}$:

$$\begin{aligned}\log P(\mathbf{x}) &= \sum_{\mathbf{y}} \int Q(\mathbf{z},\mathbf{y}|\mathbf{x}) \log P(\mathbf{x},\mathbf{y},\mathbf{z}) d_{\mathbf{z}} - \sum_{\mathbf{y}} \int Q(\mathbf{z},\mathbf{y}|\mathbf{x}) \log P(\mathbf{z},\mathbf{y}|\mathbf{x}) d_{\mathbf{z}} \\ &\geq \mathop{\mathbb{E}}_{Q(\mathbf{z},\mathbf{y}|\mathbf{x})} \log P(\mathbf{x}|\mathbf{z},\mathbf{y}) - \mathbb{KL}(Q(\mathbf{z},\mathbf{y}|\mathbf{x})||P(\mathbf{z},\mathbf{y})) \\ &= \sum_{t}^{T} \mathop{\mathbb{E}}_{Q(z_{y_t}|\mathbf{x})} \mathop{\mathbb{E}}_{Q(y_t|\mathbf{x})} \log P(x_t|z_{y_t}, y_t) - \mathbb{KL}(Q(y_t|\mathbf{x})Q(z_{y_t}|\mathbf{x})||P(y_t)P(z_{y_t}))\end{aligned} \tag{1}$$

The prior for $\mathbf{z}$ and $\mathbf{y}$ are set to be the standard normal distribution and uniform distribution respectively, so closed-form computation for the Kullback-Leibler (KL) divergence can be achieved.

Biaffine Encoder

The variational distribution $Q_\Phi(\mathbf{z},\mathbf{y}|\mathbf{x})$ with parameter set Φ forms the encoder part of VAE as it encodes the observed sentence into the space of latent variables $\mathbf{z}$ and $\mathbf{y}$.

We first run a Bi-LSTM network over the input sentence, where each token $\mathbf{x}_i$ is represented by the concatenated embedding of the word $\mathbf{e}_i(word)$ and its POS tag $\mathbf{e}_i(tag)$:

$$\mathbf{x}_i = \mathbf{e}_i(word) \oplus \mathbf{e}_i(tag)$$

$$\mathbf{h}_i = BiLSTM(\mathbf{x}_i)$$

Here we include POS tag information because we find that it is beneficial to learning when labelled training data is relatively scarce. The POS tag input for each token is obtained directly from the gold annotations in the datasets to avoid the parsing results from being influenced by the performance of POS taggers. For simplicity of the model, except aforementioned word and POS tag embeddings, we do not include any other representations of the tokens in the input sentences such as the character-based embedding outputted by a Convolutional Neural Network (CNN) encoder. The hidden state $\mathbf{h}_i$ produced by Bi-LSTM network is then used to compute $q(z_i|\mathbf{x})$ and $q(y_i|\mathbf{x})$ respectively.

For $q(y_i|\mathbf{x})$, since $\mathbf{y}$ represents a dependency tree, we follow Dozat and Manning (2017) and first pass $\mathbf{h}_i$ through two Multi-Layer Perceptrons (MLP), producing representations of the token as a dependency head and a dependant respectively, denoted as $\mathbf{h}_i^{head}$ and $\mathbf{h}_i^{dep}$:

$$\mathbf{h}_i^{head} = MLP^{head}(\mathbf{h}_i), \mathbf{h}_i^{dep} = MLP^{dep}(\mathbf{h}_i)$$

Then a biaffine function is used to compute the score s_{ij} for the dependency arc directed from x_j to x_i:

$$s_{ij} = \mathbf{h}_j^{head} \mathbf{W} \mathbf{h}_i^{dep} + b_{ij}$$

where W is a $d \times d$ square matrix, d is the dimension of $\mathbf{h}_j^{head}$ and $\mathbf{h}_i^{dep}$, and b is bias scalar. Finally we compute the probability $q(y_i = j|\mathbf{x})$ by applying softmax to all the scores of possible heads for x_i:

$$q(y_i = j|\mathbf{x}) = \frac{\exp(s_{ij})}{\sum_{t=0}^{T} \exp(s_{it})}$$

For $Q(z_i|\mathbf{x})$, we follow the inference networks in vanilla VAE and assume that $q(z_i|\mathbf{x})$ is assumed to be a Gaussian distribution with its mean $m_i \in \mathbb{R}^k$ and variance $\sigma_i^2 \in \mathbb{R}^{k \times k}$ (k is the dimension of z_i) being the outputs of two MLPs denoted as MLP^{VAE_m} and MLP^{VAE_σ} that take $\mathbf{h}_i$ as inputs. We sample z_i from $q(z_i|\mathbf{x})$ and use the reparameterization trick to pave the way for the backpropagation in learning. The process is formulated below:

$$m_i = MLP^{VAE_m}(\mathbf{h}_i) \quad \sigma_i = MLP^{VAE_\sigma}(\mathbf{h}_i) \quad z_i \sim \mathcal{N}(m_i, \sigma_i)$$

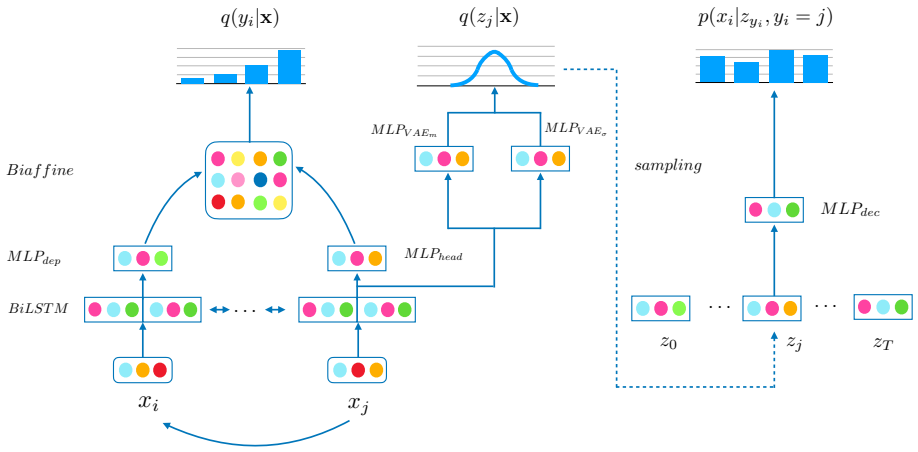

Figure 2: The overall encoding-decoding procedure illustrated by the reconstruction of token x_i from its dependency head x_j.

Arc-Factored Decoder

The decoder part models the generative distribution $P_\Theta(\mathbf{x}|\mathbf{y}, \mathbf{z})$, which is factorized into $p(x_i|y_i, z_{y_i})$ and computed as follows:

$$\mathbf{r}_i = MLP^{dec}(z_i)$$

$$p(x_i|z_{y_i}, y_i) = \frac{\exp(\mathbf{r}_{y_i}^T \mathbf{e}_i(x_i))}{\sum_{w \in V} \exp(\mathbf{r}_{y_i}^T \mathbf{e}_i(w))}$$

where V is the vocabulary and MLP^{dec} denotes layers in the decoding network. Note that different from previous work, our decoder generates each word based on z_{y_i}, which contains contextual information of the head word from both left and right.

As shown in Eq. 1, the ELBO computation involves expectation over the latent variables $\mathbf{y}$ and $\mathbf{z}$. We first follow the traditional VAE procedure to approximate the expectation over z_t by sampling z_t from the inference distribution $q(z_t|\mathbf{x})$. Then we compute the exact expectation over y_t by enumerating all possible $T + 1$ dependency heads.

We illustrate the whole encoding-decoding procedure with the reconstruction of token x_i from its dependency head x_j in Fig. 2. For each training sentence, the ELBO loss is computed with the distributions listed in Fig. 2 for each possible depedency arc and the KL divergence terms.

4 Semi-Supervised Learning

In the semi-supervised scenario, we divide the training dataset $\mathcal{D}$ into labelled data $\mathbb{L}$ and unlabelled data $\mathbb{U}$. Gold dependency parse trees are known for $\mathbb{L}$ only. The loss function $\mathcal{L}(\mathcal{D})$ thus consists of labelled loss $\mathcal{L}_l(\mathbb{L})$ and unlabelled loss $\mathcal{L}_u(\mathbb{U})$:

$$\mathcal{L}(\mathcal{D}) = \alpha \mathcal{L}_l(\mathbb{L}) + (1 - \alpha)\mathcal{L}_u(\mathbb{U}) \tag{2}$$

where α is the hyperparameter used to balance the importance of the two parts.

The labelled loss is further divided into the parser loss and the reconstruction loss. Recall that our encoder share the same configuration with the biaffine parser. When the gold parse tree $\mathbf{y}^*$ is available, we can optimize it with the probability of correctly predicting the gold parse tree $q(\mathbf{y}^*|\mathbf{x})$, which forms the parser loss. The reconstruction loss is the direct application of the ELBO in Eq. 1 which ignores the gold parse tree. The unlabelled loss is solely ELBO since gold parse trees are unknown. We rewrite the

loss function as the sum of three terms:

$$\mathcal{L}_{\theta,\phi}(\mathcal{D}) = -\alpha \sum_{(\mathbf{x},\mathbf{y}^*)\in\mathbb{L}} q_\phi(\mathbf{y}^*|\mathbf{x}) - \alpha \sum_{\mathbf{x}\in\mathbb{L}} \mathcal{E}_{\phi,\theta}(\mathbf{x}) - (1-\alpha) \sum_{\mathbf{x}\in\mathbb{U}} \mathcal{E}_{\phi,\theta}(\mathbf{x}) \quad (3)$$

where $\mathcal{E}$ denotes the ELBO formulation of Eq 1. The training process to minimize $\mathcal{L}$ is fully end-to-end with the reparameterization trick used to facilitate backpropagation of ELBO as mentioned earlier.

Directly training autoencoders unsupervisedly may cause the learned latent structures to diverge from the linguistically correct structures (Williams et al., 2018). Therefore, in each training epoch, we start with optimizing the parsing loss on the labelled data to pretrain the encoder and then optimize the complete loss function on both labelled and unlabelled data.

Note that the tree constraint over $\mathbf{y}$ is not enforced in training. Our definition of $\mathbf{y}$ only enforces the head selection constraint that each token has exactly one head. On the other hand, we enforce the tree constraint during the validation and the test phases when the encoder of our model works as a dependency parser, predicting the most probable tree with commonly used Maximum Spanning Tree (MST) decoding algorithms such as Eisner's algorithm based on the score computed for each dependency arc. The decoder part does not participate in the prediction.

5 Evaluation

Settings

We evaluate the parsing performance of our model on datasets across 7 languages: English, French, German, Italian, Spanish, Swedish and Hindi. To make our evaluation comparable with that of Corro and Titov (2019), for English and French we choose corpora from Stanford Dependency conversion (De Marneffe and Manning, 2008) of the Penn Treebank (Marcus et al., 1993) and the French Treebank distributed for the SPMRL 2013 shared task (Abeillé et al., 2000). The corpora for the rest five languages are all from the Universal Dependencies (UD) v2.0 (Zeman et al., 2017). In Table 1 we list the number of sentences in training, validation and test sets.

	Training	Development	Test
English	39832	1700	2416
French	14759	1235	2541
German	14118	799	977
Italian	12838	564	482
Swedish	4303	504	1219
Hindi	13304	1659	1684
Spanish	14187	1400	426

Table 1: The number of sentences in treebanks used for our evaluation. The split of training, development and test sets follow the default split setting for each treebank.

We adopt the default split of training, development and test sets for the SPMRL and UD corpora and the commonly used split for the Penn Treebank corpus (section 2-22 for training, section 22 for development and section 23 for test).

Following Corro and Titov (2019), we split the training set of each dataset into a labelled set and an unlabelled set with the ratio of 1:9 and choose a sentence as labelled one if its index modulo 10 equals zero.

The dimensions for the word embeddings and POS tag embeddings are set to 100 and 25. The dimensions in LSTMs and MLPs are uniformly set to 200 except for the layer before word generation, whose dimension is set to 100 in order to fit the word embedding size. The word embeddings used in the encoder part and the decoder part share the same initialization. Pre-trained word embeddings are used

for all the datasets. For English, we use the 100 dimensions version of GLOVE embedding (Pennington et al., 2014). For all the other languages, the embeddings released for the 2017 CoNLL Shared Task on Universal Dependency Parsing are used. The network parameters are optimized by Adam (Kingma and Ba, 2015) with the setting of learning rate 0.002, $\beta_1 = 0.9$, and $\beta_2 = 0.9$. The hyperparameter α varies for different datasets: $\alpha = 0.2$ for English and French, $\alpha = 0.9$ for Spanish and $\alpha = 0.8$ for the other corpora. The number of epochs taken to train the parser until convergence also varies in different datasets, but none of them is above 150 epochs. All the hyperparameters are tuned on the development set for each treebank.[1]

	English	French	German	Italian	Spanish	Hindi	Swedish
Ours-Sup	92.00	84.58	80.41	87.11	84.63	92.22	80.29
Self-Training	91.82	85.27	81.33	87.62	85.08	91.74	78.33
NCRFAE	91.94	84.83	80.70	87.33	84.31	92.49	80.33
Ours-Semi	**92.55**	**85.57**	**81.52**	**88.58**	**85.46**	**92.56**	**80.85**

Table 2: The UAS results of our semi-supervised model compared with three baseline models on the datasets of 7 languages. The best result for each column is shown in bold.

Results

In Table 2 we list the parsing results of our model and three baseline models. **Ours-Sup** stands for the encoder of our model as a supervised dependency parser trained on labelled data only. **Self-Train** is the traditional self-training method that uses the prediction on unlabelled data as extra labelled training samples (we follow the common self-training implementation that uses the parser to iteratively predict parse trees of the unlabelled data and uses them to update the model). **NCRFAE** is the neural version of a semi-supervisedly trained CRF autoencoder. We develop this neural version based on the implementation of CRF autoencoder dependency parser by Cai *et al.* (2017) which runs in the fully unsupervised scenario. In our neural version we replace the original discrete feature extractor with a BiLSTM network and neuralize all the computations involved. **Ours-Semi** is our semi-supervised model. The evaluation metric that we report is Unlabelled Attachment Score (UAS), which measures the percentage of dependency heads that are correctly found. We do not evaluate Labelled Attachment Score (LAS) because our model focuses on learning parse tree structures. Both the baselines and our model are evaluated after being fine-tuned on the development set for each treebank to ensure that the differences in results do not depend on the hyperparameter settings.

In Table 3, we also compare our model with the method proposed in Corro and Titov (2019) denoted as **C&T**. Our model is not directly comparable with C&T because our encoder is stronger than theirs in terms of supervised parsing accuracy. Therefore, we report the evaluation results after weakening our encoder by removing the POS input and applying weaker scoring functions (the one used in Kiperwasser and Goldberg (2016) instead of Dozat and Manning (2017)).

	English	French
C&T-Sup	88.79	84.09
Ours-Sup (weakened)	88.58	84.05
C&T	89.50	84.69
Ours-Semi(weakened)	**89.67**	**84.94**

Table 3: The UAS results of our semi-supervised model compared with the method of Corro and Titov (2019). "-Sup" stands for the encoder of the model being used as a supervised dependency parser trained on labelled data only . "weakened" means our encoder is deliberately weakened to make it comparable with that of C&T. The best result for each column is shown in bold.

[1]Our code is available at `https://github.com/mikufan/SemiVariationalParser`.

From Table 2 we observe moderate but consistent boost of performance by our model over the baseline models on all the languages evaluated, which proves the effectiveness of our model in utilizing unlabelled data to improve parsing accuracy. The results in Table 3 further demonstrate that our model can achieve results on par with a state-of-the-art model that employs more complex model structures and learning techniques.

6 Analysis

In this section we conduct a series of analysis on the English Penn Treebank corpus to shed more insights into our model and reveal where its advantage lies compared with other models.

Varying the Proportion of Labelled Data

In our evaluation in section 5, we uniformly set the ratio of labelled and unlabelled data to 1:9. However, since our motivation is to use unlabelled data to remedy the scarcity of labelled data, we are interested to know whether our model still works when there is far less labelled data. On the other hand, we also want to know how much labelled data is enough to diminish the utility of unlabelled data.

To analyze the impact brought by the amount of labelled data, we vary the proportion of labelled data from 1% to 50% on the Penn Treebank training corpus and observe the changes in the UAS score evaluated on the corresponding test set. The results are shown in Table 4.

Ratio	Ours-Sup	Ours-Semi	Δ UAS
0.01:0.99	85.72	86.21	+0.49
0.1:0.9	92.00	92.55	+0.55
0.3:0.7	93.94	94.15	+0.21
0.5:0.5	94.38	94.41	+0.03

Table 4: The UAS results of our supervised model and semi-supervised model when the proportion of labelled data varies. Δ UAS denotes the UAS difference between the "Ours-Semi" column and the "Ours-Sup" column.

From the results in Table 4, we see a clear trend that as the proportion of labelled data increases, the UAS goes up for both the semi-supervised model and the supervised model. However, the advantage in performance of the semi-supervised model quickly diminishes with the increase of labelled proportion. When the proportion rises to 50%, the UAS difference is almost reduced to none. The results accord with the intuition that the more substantial the labelled data is, the more unnecessary it is to utilize extra training data. We also notice that even when the labelled data proportion is set to as small as 1%, our semi-supervised model still robustly outperforms the supervised model.

Time Efficiency

As our model is arc-factored without the tree constraint, the time efficiency in both encoding and decoding is one of our prominent advantages.

For encoding, given a sentence of length T, the time complexity for our encoder is $O(T^2)$, while the two counterpart methods NCRFAE and C&T employ variants of Eisner's algorithm with time complexity of $O(T^3)$. We evaluate the running speed of our encoder and the NCRFAE encoder on the training set of English Penn Treebank in the environment of NVIDIA Titan V servers. The results are shown in Table 5, which reflects the difference in the theoretical time complexity. Here we do not report the running time of encoder comparison with C&T since it is not fully open-sourced and difficult to reimplement, but it is straightforward to deduce that our model should be more time efficient than C&T: except for the sampling part, the encoder of C&T is almost the same as that of NCRFAE, which has been shown to be much slower than the encoder of our model by the experimental results.

As for the decoder, NCRFAE employs a decoder similar to ours in structure and hence has similar time efficiency. In C&T, however, the reconstruction of each sentence moves strictly from left to right,

while in our model it is arc-factored and can be implemented parallelly. It is thus reasonable to suggest that our model still holds the advantage in time efficiency in decoding.

Note that there still exist potential ways to further boost our encoding speed: we may replace the bi-LSTM in our encoder with more parallelizable feature extractors such as the transformer (Vaswani et al., 2017). We leave this option for future work.

Encoder	Sec/Sen
Ours	0.142
NCRFAE	2.365

Table 5: The time cost for encoders on each sentence

Ablation Test on Latent Continuous Variables

Previous work applying the autoencoder framework to dependency parsing does not make full use of contextual information in decoding. NCRFAE reconstructs each input token from the embedding of its dependency head alone. The decoding for C&T leverages LSTM and GCN networks to utilize contextual information in the sentence being generated to the left of each generated word, but cannot incorporate contextual information to the right of the generated word because of the left-to-right generative process.

We believe utilizing the complete contextual information in decoding is important. Intuitively is that what dependent to choose should be determined by the dependency head and its full context from both the left and the right. In our decoder, the contextual information from the input sentence is conveyed by latent continuous variables z_i. To verify the importance of full contextual information, we design an ablation test for the latent continuous variables by restricting their encoder so that they contain no contextual information or partial contextual information. In the no context case, we directly assign word embeddings to the latent variables, so the generation of each word depends on its dependency head word alone, which is similar to the decoder of NCRFAE. In the partial context case, we use a left-to-right LSTM instead of a bi-LSTM to produce the Gaussian distributions over the latent variables. The results are listed in Table 6. In Table 6, we see a clear drop of performance after contextual information is

Encoder	UAS	Δ
Full Context	92.55	+0.0
Partial Context	92.23	-0.22
No Context	91.71	-0.84

Table 6: The ablation test results. The Δ column shows the relative changes of UAS compared with the full context baseline.

removed or partially removed from the latent variables. This proves the importance of the full contextual information in our model.

7 Conclusion

In this paper, we presented a semi-supervised dependency parsing model based on the VAE framework. We tackle the main challenges brought by the tree structure constraint of dependency parsing by relaxing the tree constraint during training and making the learning of our model fully arc-factored. The experiments show that the simplicity of the model does not hinder its performance and it achieves better parsing accuracy and faster speed than previous work. In future work we plan to extend our model to other tasks such as semantic dependency parsing and apply it to fully unsupervised scenarios.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (61976139).

References

Anne Abeillé, Lionel Clément, and Alexandra Kinyon. 2000. Building a treebank for french. In *Proceedings of LREC 2000, 31 May - June 2, 2000, Athens, Greece*.

Waleed Ammar, Chris Dyer, and Noah A Smith. 2014. Conditional random field autoencoders for unsupervised structured prediction. In *Advances in Neural Information Processing Systems*, pages 3311–3319.

Jiong Cai, Yong Jiang, and Kewei Tu. 2017. CRF autoencoder for unsupervised dependency parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 1638–1643.

Wenliang Chen, Daisuke Kawahara, Kiyotaka Uchimoto, Yujie Zhang, and Hitoshi Isahara. 2009. Using short dependency relations from auto-parsed data for chinese dependency parsing. *ACM Transactions on Asian Language Information Processing (TALIP)*, 8(3):10.

Mingda Chen, Qingming Tang, Karen Livescu, and Kevin Gimpel. 2018. Variational sequential labelers for semi-supervised learning. In *Proceedings of EMNLP2018, Brussels, Belgium, October 31 - November 4, 2018*, pages 215–226.

Kevin Clark, Minh-Thang Luong, Christopher D. Manning, and Quoc V. Le. 2018. Semi-supervised sequence modeling with cross-view training. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 1914–1925. Association for Computational Linguistics.

Caio Corro and Ivan Titov. 2019. Differentiable perturb-and-parse: Semi-supervised parsing with a structured variational autoencoder. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*.

Marie-Catherine De Marneffe and Christopher D Manning. 2008. Stanford typed dependencies manual. Technical report, Technical report, Stanford University.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Jason Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *16th International Conference on Computational Linguistics, Proceedings of the Conference, COLING 1996, Center for Sprogteknologi, Copenhagen, Denmark, August 5-9, 1996*, pages 340–345.

Kelvin Guu, Tatsunori B Hashimoto, Yonatan Oren, and Percy Liang. 2018. Generating sentences by editing prototypes. *Transactions of the Association for Computational Linguistics*, 6:437–450.

Wenjuan Han, Yong Jiang, and Kewei Tu. 2019. Enhancing unsupervised generative dependency parser with contextual information. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 5315–5325.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Yong Jiang, Wenjuan Han, and Kewei Tu. 2016. Unsupervised neural dependency parsing. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 763–771.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Diederik P. Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *TACL*, 4:313–327.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Dan Klein and Christopher D. Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics, 21-26 July, 2004, Barcelona, Spain*, pages 478–485.

Sandra Kübler, Ryan T. McDonald, and Joakim Nivre. 2009. *Dependency Parsing*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Zhenghua Li, Min Zhang, and Wenliang Chen. 2014. Ambiguity-aware ensemble training for semi-supervised dependency parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 457–467, Baltimore, Maryland, June. Association for Computational Linguistics.

Bowen Li, Jianpeng Cheng, Yang Liu, and Frank Keller. 2019. Dependency grammar induction with a neural variational transition-based parser. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 6658–6665.

Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 152–159, New York City, USA, June. Association for Computational Linguistics.

Tahira Naseem, Harr Chen, Regina Barzilay, and Mark Johnson. 2010. Using universal linguistic knowledge to guide grammar induction. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1234–1244, Cambridge, MA, October. Association for Computational Linguistics.

Hiroshi Noji, Yusuke Miyao, and Mark Johnson. 2016. Using left-corner parsing to encode universal structural constraints in grammar induction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 33–43, Austin, Texas, November. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543.

Kenji Sagae and Jun'ichi Tsujii. 2007. Dependency parsing and domain adaptation with LR models and parser ensembles. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 1044–1050, Prague, Czech Republic, June. Association for Computational Linguistics.

Anders Søgaard and Christian Rishøj. 2010. Semi-supervised dependency parsing using generalized tri-training. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 1065–1073, Beijing, China, August. Coling 2010 Organizing Committee.

Jun Suzuki, Hideki Isozaki, and Masaaki Nagata. 2011. Learning condensed feature representations from large unsupervised data sets for supervised learning. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 636–641, Portland, Oregon, USA, June. Association for Computational Linguistics.

Kewei Tu and Vasant G. Honavar. 2012. Unambiguity regularization for unsupervised learning of probabilistic grammars. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLP-CoNLL 2012, July 12-14, 2012, Jeju Island, Korea*, pages 1324–1334.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 5998–6008.

Adina Williams, Andrew Drozdov, and Samuel R. Bowman. 2018. Do latent tree learning models identify meaningful structure in sentences? *TACL*, 6:253–267.

Pengcheng Yin, Chunting Zhou, Junxian He, and Graham Neubig. 2018. StructVAE: Tree-structured latent variable models for semi-supervised semantic parsing. In *The 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, Melbourne, Australia, July.

Daniel Zeman, Martin Popel, and Milan Straka *et al.* 2017. Conll 2017 shared task: Multilingual parsing from raw text to universal dependencies. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies, Vancouver, Canada, August 3-4, 2017*, pages 1–19.

Xiao Zhang, Yong Jiang, Hao Peng, Kewei Tu, and Dan Goldwasser. 2017a. Semi-supervised structured prediction with neural CRF autoencoder. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 1701–1711.

Xingxing Zhang, Jianpeng Cheng, and Mirella Lapata. 2017b. Dependency parsing as head selection. In *Proceedings of the 15th EACL: Volume 1, Long Papers*, pages 665–676, Valencia, Spain, April. Association for Computational Linguistics.

Zhisong Zhang, Xuezhe Ma, and Eduard H. Hovy. 2019. An empirical investigation of structured output modeling for graph-based neural dependency parsing. In *Proceedings of ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 5592–5598.