Spa: On the Sparsity of Virtual Adversarial Training for Dependency Parsing

Chao Lou[1], **Wenjuan Han**[2]*, **Kewei Tu**[1]*
[1]School of Information Science and Technology, ShanghaiTech University
Shanghai Engineering Research Center of Intelligent Vision and Imaging
[2]Beijing Jiaotong University, Beijing, China
{louchao,hanwj,tukw}@shanghaitech.edu.cn

Abstract

Virtual adversarial training (VAT) is a powerful approach to improving robustness and performance, leveraging both labeled and unlabeled data to compensate for the scarcity of labeled data. It is adopted on lots of vision and language classification tasks. However, for tasks with structured output (*e.g.*, dependency parsing), the application of VAT is nontrivial due to the intrinsic proprieties of structures: (1) the non-sparse problem and (2) exponential complexity. Against this background, we propose the **S**parse **P**arse **A**djustment algorithm (Spa) and successfully applied VAT to the dependency parsing task. Spa refers to the learning algorithm which combines the graph-based dependency parsing model with VAT in an exact computational manner and enhances the dependency parser with controllable and adjustable sparsity. Empirical studies show that the TreeCRF parser optimized using Spa outperforms other methods without sparsity regularization.

1 Introduction

Dependency parsing is a fundamental structured prediction task in natural language processing that aims to capture syntactic structures in sentences in the form of dependency relations between words. The application of dependency structures is mainly reflected in discourse parsing (Nishida and Nakayama, 2020; Zhang et al., 2021), machine translation (Shen et al., 2008), and many other tasks. While supervised learning is the ideal technique used to learn a dependency parser automatically, it requires the training sentences to be manually annotated with their gold parse trees. This brings the main bottleneck for learning a practical dependency parser — the lack of adequate training corpora with dependency trees. Annotations are both laborious and time costly. Multiple research directions (*i.e.*, unsupervised learning, semi-supervised learning and transfer learning, *etc.*) try to eliminate this bottleneck (Han et al., 2020a).

Virtual adversarial training (VAT) (Miyato et al., 2018), as a semi-supervised learning approach, utilizes both annotated training sentences and unlabeled data to compensate for the scarcity of labeled data. It extends adversarial training (AT) (Goodfellow et al., 2015) to unlabeled data. VAT encourages the output distributions to be similar on both an unlabeled sample and corresponding adversarial examples by adding a Kullback-Leibler (KL) divergence term in the training loss. In this way, VAT improves the performance and robustness of many tasks (Akhtar and Mian, 2018; Berthelot et al., 2019; Chen et al., 2020).

However, multiple technical challenges are faced by applying VAT on dependency parsing. Except for the general challenges related to gradient computation of discrete inputs, grammatical correctness, and meaning preservation (Zhang et al., 2019; Jia and Liang, 2017; Wang et al., 2019; Cheng et al., 2019, 2020) faced by all adversarial example generators, two potential but critical challenges exist because of the propriety of structured prediction: (1) the non-sparse problem and (2) exponential complexity. The non-sparse problem is naturally connected to unambiguity (Tu and Honavar, 2012), both highlighting that the number of plausible parses of a natural language sentence is relatively small compared with the huge number of possible parses. We are interested in predicting probabilities as small as possible for these unlikely trees rather than having an estimation of their actual probabilities. The fact that the Viterbi expectation-maximization algorithm (EM) outperforms Standard EM in previous work (Poon and Domingos, 2011; Tu and Honavar, 2012; Spitkovsky et al., 2010, 2011) also provides evidence of the advantage of implicitly utilizing the sparsity property. Although Chen et al. (2020) make VAT compatible with a linear-chain structured prediction model by

*Corresponding Author

Findings of the Association for Computational Linguistics: AACL-IJCNLP 2022, pages 11–21
November 20–23, 2022.

considering the probabilities of K most possible label sequences, which is a sparse approximation of the original distribution, they did not quantitatively investigate the impact of sparsity in the application of VAT. For the complexity challenge, different from conventional classification tasks with a fixed number of classes, computing the KL divergence of parse tree distributions by enumerating all possible parses is intractable because the number of possible parses for each sentence is exponential w.r.t. the sentence length. Therefore, conventional approaches can only estimate the KL divergence in the VAT loss rather than compute it exactly.

Against this background, we propose **S**parse **P**arse **A**djustment algorithm (`Spa`) and successfully applied VAT to dependency parsing. `Spa` refers to the learning algorithm which combines the graph-based dependency parsing model with VAT in an exact computational manner, overcoming the problem of enumerating, and enhances the dependency parser with controllable and adjustable sparsity. We applied VAT to a state-of-the-art parsing model: the Tree Conditional Random Field (TreeCRF) parser (Zhang et al., 2020). `Spa` incorporates into TreeCRF an inductive bias in favor of models that lead to a controllable sparsity. Adjusting the hyper-parameter can control sparsity to ease the non-sparse problem. Empirical studies show that the TreeCRF parser optimized using `Spa` outperforms other semi-supervised methods without sparsity regularization. Within `Spa`, our exact computational manner achieves competitive performance and enables faster training compared to the top-K approximate approach (Chen et al., 2020). The code can be found at: `https://github.com/LouChao98/struct-vat`.

2 Related Work

2.1 Semi-Supervised Learning

Semi-supervised learning is an important branch of machine learning to improve model performance when there is insufficient labeled data, which utilizes unlabeled data to get more information that might be beneficial for supervised tasks. A common semi-supervised learning approach is to train a generative model (Hinton et al., 2006; Maaløe et al., 2016; Wang and Tu, 2020a) which achieve state-of-the-art performance. However, these methods require additional hyperparameters, and the conditions under which the generative model will provide good supervised learning performance are poorly understood (Miyato et al., 2017b).

Self-training (Yarowsky, 1995) is another approach to semi-supervised learning, which has been successfully applied to natural language processing tasks. In self-training, the model acts as teacher and student iteratively. Recent approaches use soft targets from one or multiple teachers' output (Hinton et al., 2015), such as in tri-training (Zhi-Hua Zhou and Ming Li, 2005; Ruder and Plank, 2018).

Consistency training is also a branch of semi-supervised learning, forcing the model to make consistent predictions on different views of the same data. Cross-view training (CVT) (Clark et al., 2018) works on bidirectional LSTMs and constructs views by masking out neurons of one direction. R-drop (Liang et al., 2021) constructs views by *dropout-twice*, thus is compatible with transformers. Unsupervised Data Augmentation (Xie et al., 2019) changes the input tokens instead of hidden representations with the help of external models, e.g., a back-translator. Unlike others, VAT constructs views using a gradient-based attacker. Next, we will introduce VAT in detail.

2.2 Virtual Adversarial Training

Adversarial training (Goodfellow et al., 2015) is a method to improve model robustness, in which models are trained using not only labeled data but also perturbed samples generated by an adversarial attacker. As a consequence, model predictions would be consistent regardless of the perturbations. AT was demonstrated to be more effective than random attackers since its perturbations maximize model loss in a constrained length. Many previous works (Goodfellow et al., 2015; Miyato et al., 2017a; Yasunaga et al., 2018; Han et al., 2020c; Zhang et al., 2022) proved the effectiveness of AT on computer vision tasks and language tasks. To introduce AT into semi-supervised settings, Miyato et al. (2018) proposed virtual adversarial training. The idea of VAT can be seen as the combination of self-training (Yarowsky, 1995) and AT if we treat predictions on clean input as labels in AT. VAT can be applied to both labeled and unlabeled data because ground-truth labels are not required. VAT achieved state-of-the-art performance for image classification tasks (Miyato et al., 2018) and proved to be more efficient than previous semi-supervised approaches, such as entropy minimization (Grandvalet and Bengio, 2005) and self-training (Yarowsky, 1995). Chen et al. (2020)

proposed SeqVAT, which successfully makes VAT compatible with the linear-chain conditional random field (LinearChainCRF), and showed that VAT benefits from structure information. It combines VAT with LinearChainCRF and achieves significant improvements in sequence labeling. They estimate the KL divergence by only considering the K most possible label sequences and report that the performance of VAT on LinearChainCRF is better than that of VAT on token-level categorical distributions, which is used by works before SeqVAT. In this paper, we show VAT can be applied to dependency parsing with TreeCRF, which is a more complex structure.

3 Model

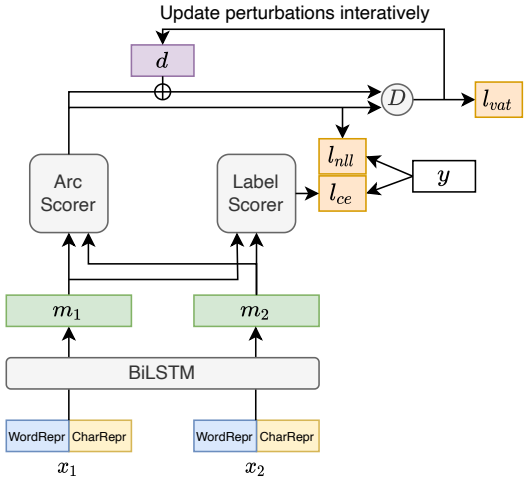

Figure 1: Model Architecture

Our model architecture is illustrated in Fig. 1. It adopts the basic architecture of the TreeCRF parser. We concatenate word embeddings with character features extracted by a LSTM layer as input features. Then, we feed input features into the scoring functions. Finally, TreeCRFs are constructed using scores.

Encoder The encoder includes both a word-based representation and a character-based representation inspired by character information capturing morphological features (Ma and Hovy, 2016; Zhang et al., 2020).

Word Representation We use 100-dimension GloVe (Pennington et al., 2014) as word representations for dependency parsing, following previous parsing work (Dozat and Manning, 2017; Zhang et al., 2020). Intuitively, a model could learn to make the perturbations in VAT insignificant by learning embeddings with a very large norm. To prevent this pathological situation, we follow the setting from Miyato et al. (2017b) and use normalized word embeddings $\hat{w}$ instead of raw vectors w. Formally, we use the representations as follows:

$$\hat{w}^{(i)} = \frac{w^{(i)} - \text{Mean}(w)}{\sqrt{\text{Var}(w)}}$$

$$\text{where} \quad \text{Mean}(w) = \frac{1}{n}\sum_{i=1}^{n} w^{(i)}$$

$$\text{and} \quad \text{Var}(w) = \frac{1}{n}\sum_{i=1}(w^{(i)} - \text{Mean}(w))^2$$

where n is the number of all tokens in the embedding space and $w^{(i)}$ is the embedding of the ith word in the vocabulary.

Character Representation Following Zhang et al. (2020), 50-dimension character embeddings and a bidirectional LSTM with 50 neurons per direction are used. Similar to word embeddings, we also apply normalization to the output vectors of the character LSTM.

Contextual Representation After transforming input tokens to vector representations, we use a three-layer bidirectional LSTM to capture contextual information with 400 neurons per direction. We also add Variational Dropout (Gal and Ghahramani, 2016) between LSTM layers for stable training.

Scoring Functions Following Zhang et al. (2020), we adopt a two-stage parsing strategy. The structure (whether arcs exist) and the labels of arcs are processed separately. The scores of structures are computed using deep biaffine functions. Let $m_{(\cdot)}$ be the output of LSTM and MLP be a multi-layer perceptron, the score of the arc from i to j is defined as follows:

$$h_{(\cdot)}^{h/d} = \text{MLP}^{h/d}(m_{(\cdot)})$$

$$\phi_{ij} = \text{Biaffine}(h_i^h, h_j^d)$$

Label scores ϕ_{ijl} of the arc from i to j with label l are defined similarly. Please refer to Zhang et al. (2020) for more details.

Decoder The arc scores are fed into TreeCRF, which defines the distribution over all possible trees of a sentence. For a tree y (a set of arcs) of sentence

x, its probability is defined as follows:

$$p(y|x) = \frac{\phi(y)}{Z}$$
$$\phi(y) = \prod_{(i,j)\in y} \phi_{ij}$$
$$Z = \sum_{y'\in\mathcal{Y}(x)} \phi(y'),$$

where $\phi(y)$ denotes the scores of the tree y, Z denotes the partition function and $\mathcal{Y}(x)$ denotes the set of possible trees of x. The supervised training loss L_{sup} consists of two parts. Negative log-likelihood L_{nll} is used as the supervised loss of structures and cross-entropy L_{ce} is used as the supervised loss of labels.

$$L_{nll} = \log Z - \log \phi(\bar{y})$$
$$L_{ce} = \sum_{(i,j)\in\bar{y}} \mathcal{CE}(\text{Softmax}(\phi_{ij\cdot}), l_{ij})$$
$$L_{sup} = L_{nll} + L_{ce},$$

where $\bar{y}$ is the gold tree and l_{ij} is the one-hot encoding of the gold label of arc from i to j.

4 Learning

4.1 Unsupervised Loss

In AT, the perturbations d_w, d_c bounded by δ_w, δ_c is generated by maximizing the training loss:

$$d_w = \underset{\epsilon,\|\epsilon\|\leqslant\delta_w}{\arg\max}\, D(y; P(w+\epsilon, c))$$
$$d_c = \underset{\epsilon,\|\epsilon\|\leqslant\delta_c}{\arg\max}\, D(y; P(w, c+\epsilon))$$

where D is an arbitrary distance measure or loss function, w, c are the normalized word and character representations respectively, and P is the model outputting TreeCRF distribution. AT can only be used in supervised settings because it requires y to generate the perturbations.

Miyato et al. (2018) proposed virtual adversarial training to extend AT to unlabeled data. Denote x, x_{adv} as a sample and its corresponding adversarial sample, and p_{orig}, p_{adv} as the distribution predicted by the model for x, x_{adv}. Then a natural choice of D in VAT is the KL divergence:

$$D(p_{orig}; p_{adv}) = \mathcal{KL}(P(w,c)||P(w+d_w, c+d_c))$$

Compared to AT, VAT can be seen as the "self-training" version of AT since VAT replaces the ground-truth y with the predicted p_{orig}. The perturbations d_w, d_c are now defined by:

$$d_w = \underset{\epsilon,\|\epsilon\|\leqslant\delta_w}{\arg\min}\, \mathcal{KL}(P(w,c)||P(w+\epsilon, c))$$
$$d_c = \underset{\epsilon,\|\epsilon\|\leqslant\delta_c}{\arg\min}\, \mathcal{KL}(P(w,c)||P(w, c+\epsilon))$$

Those two are still intractable for gradient descent. Miyato et al. (2018) propose to approximate perturbations by the second-order Taylor approximation and the power iteration method. The perturbations d_w, d_c can be estimated as follows:

$$d_w = \frac{g_w}{\|g_w\|}\delta_w \quad d_c = \frac{g_c}{\|g_c\|}\delta_c$$

where g_w, g_c are gradients of the distance w.r.t. perturbations:

$$g_w = \nabla_\epsilon \mathcal{KL}(P(w,c)||P(w+\epsilon, c))$$
$$g_c = \nabla_\epsilon \mathcal{KL}(P(w,c)||P(w, c+\epsilon))$$

We stop the gradient propagation through d_w, d_c when optimizing model parameters because they are adversarial attacks.

The full loss function of our model is a weighted summation of the supervised training loss and contrastive training loss:

$$L = L_{sup} + \alpha D(p_{orig}; p_{adv}). \quad (1)$$

Because p_{orig} is at least as good as p_{adv}, we do not want to optimize p_{orig} for the loss in terms of p_{adv}. In practical, we detach p_{orig} from the computational graph when optimizing the unsupervised loss, such that the entropy term $\mathcal{E}(p_{orig})$ in $D = \mathcal{KL}(p_{orig}||p_{adv}) = \mathcal{CE}(p_{orig}||p_{adv}) - \mathcal{E}(p_{orig})$ can be omitted because it will not contribute any gradients to trainable parameters.

4.2 Exact Computation

As Chen et al. (2020) mentioned, the computation of the KL divergence of two CRFs is nontrivial because of the exponential-size space. This section derives the polynomial-time exact computation for the TreeCRF using dynamic programming. A similar derivation for the entropy of constituency trees is documented in Hwa (2000).

We use the notation $N(i,j;y)$ to denote the quantity N (Tab. 1) of the tree y which covers the span $x_i \dots x_j$ and $N(i,j)$ to denote the quantity N of all possible trees covering it. Similarly, we use the notation $N(i,j,k;y)$ to denote the quantity N of the tree y which, additionally, can be split

N	$N(\cdot)$	$N(\cdot; y)$	Explanation
ϕ_d	•	•	the tree score
p_d	○	•	the tree probability
h_d	•	○	the cross entropy

Table 1: Notations. •/○ means the quantity is defined/undefined for this form. $d \in \{p, q\}$ is the identifier of two distributions. We abuse some notations.

into two sub-trees at the point k. The left sub-tree covering $x_i \dots x_k$ is named as y_l and the right one covering $x_{k+1} \dots x_j$ as y_r. We do not decorate y_l, y_r with the span indices (e.g., i, j, k) because they can be understood from the context. $N(i, j, k)$ is the aggregated version of $N(i, j, k; y)$.

The KL divergence consists of the entropy and the cross-entropy. As the full KL divergence can be derived with little effort from the cross-entropy, we only show the derivation of the cross-entropy, $h(1, n) \equiv \mathcal{CE}(p, q)$, for the sake of simplicity. $h(i, j)$ can be written as the form of enumerating all possible trees $y \in \mathcal{Y}$.

$$h(i,j) = -\sum_{y} p_p(i,j;y) \log p_q(i,j;y) \quad (2)$$

The first step (Eq. 6) is to decompose y into sub-trees y_l, y_r and also an arc connecting the two sub-trees' roots $\mathcal{A}(y_l, y_r) \in \{root(y_l) \rightarrow root(y_r), root(y_r) \rightarrow root(y_l)\}$[1] where γ_d ($d \in \{p, q\}$) is the normalizer[2] (Eq. 5). After breaking the log-terms about q into three terms, the summation of y_l, y_r reduces p_q terms to h terms (Eq. 8). Specifically, there are two types of reduction: (1) reducing to cross entropy of trees covering smaller spans (e.g., Eq. 3); (2) and reducing to marginalization of possible trees (e.g., Eq. 4):

$$-\sum_{y_l} p_p(i,k;y_l) \log p_q(i,k;y_l) \equiv h(i,k) \quad (3)$$

$$\sum_{y_r} p_p(k+1,j;y_r) = \sum_{y_r} p(y_r|x_{[k+1:j]}) = 1 \quad (4)$$

Eq.8 is the state transition equation of dynamic programming, in which $h(i, j, k)$ is in terms of $h(i, k), h(k + 1, j)$, which are smaller problems, and $\gamma(\mathcal{A}(y_l, y_r))$ is in terms of $\phi_d(i, k), \phi_d(k + 1, j)$ and the potential score of $a \rightarrow b$ in d ($\phi_{d,ab}$).

$$\gamma_d(\mathcal{A}(y_l,y_r), y_l, y_r) = \gamma_d(a \rightarrow b, y_l, y_r) = \phi_{d,ab}\phi_d(i,k)\phi_d(k+1,j)/\phi_d(i,j,k) \quad (5)$$

[1] We use the Eisner algorithm (Eisner, 2000) as the routine (Sec.4.3), in which $\mathcal{A} = \{i \rightarrow j, j \rightarrow i\}$.

[2] We denote $\gamma_d(\mathcal{A}(y_l, y_r), y_l, y_r)$ as $\gamma_d(\mathcal{A}(y_l, y_r))$ for simplicity. One can read i, j, k from y_l, y_r.

4.3 CrossEntropy Semiring

The semiring parsing framework (Goodman, 1999; Li and Eisner, 2009) enables us to decouple the semantics (e.g., cross-entropy and MAP inference) from the routine (e.g., the inside algorithm).

The semiring parsing framework is a generalization of the sum-product algorithm where operators $+, \times$ are generalized to abstract operators $\oplus, \otimes$. Plugging in different semirings allows us to query different properties, e.g., partition and mode. To illustrate the cross-entropy semiring, we define the elements of the semiring as triplets indexed by positions i, j:

$$s(i,j) = (\phi_p(i,j),\ \phi_q(i,j),\ h(i,j)) \quad (9)$$

Because the first two elements can be solved by the inside algorithms, we focus on the third term. An abstract product $\otimes$ combines two sub-structures. After reordering terms in Eq. 8, we observes that:

$$h(i,j,k) = A(h(i,k) + h(k+1,j)) + B \quad (10)$$

where A, B are in terms of γ_d but irrelative to $h(i, k), h(k + 1, j)$. γ_d is available only after summation due to $\phi_d(i, j, k)$[3] in Eq. 5 , therefore we delay to resolve A, B and perform the summation of $h(i, k)$ and $h(k + 1, j)$ only (Eq. 11).

An abstract summation $\oplus$ aggregates all possible structures at the same position. In our case, there are two jobs: (1) resolve A, B (2) aggregate $h(i, j, k)$ to get $h(i, j)$. The computation is defined as Eq. 12. Let $s = [s_1, s_2, \dots]$ be a list of triplets, the cross-entropy semiring is defined as follows:

$$\otimes\, s = (\prod s[0],\ \prod s[1],\ \sum s[2]) \quad (11)$$

$$\oplus\, s = (\sum s[0],\ \sum s[1],\ f) \quad (12)$$

$$f = \sum \left(\frac{s[0]}{\sum s[0]} \times \left(s[2] - \log \frac{s[1]}{\sum s[1]}\right)\right)$$

$$\mathbf{1} = (0, 0, 1), \quad \mathbf{0} = (-\infty, -\infty, 0)$$

4.4 Sparsity Regularization

Motivated by the sparsity property, we would like to incorporate into the model a flexibly adjustable button in favor of sparsity adjustment. In our approach, this button is a adjustable hyperparameter.

One natural measurement of sparsity is the number of parse trees considered in leaning. We denote the number of parse trees K as this adjustable

[3] $\phi_d(i, j, k)$ can be obtained by summing the first two items in triplets.

$$
\begin{aligned}
& h(i,j,k) \\
&= -\sum_{(y_l,y_r,\mathcal{A})} \big[p_p(i,k;y_l)p_p(k+1,j;y_r)\gamma_p(\mathcal{A}(y_l,y_r))\log[p_q(i,k;y_l)p_q(k+1,j;y_r)\gamma_q(\mathcal{A}(y_l,y_r))]\big] && (6)\\
&= \underbrace{\sum_{\mathcal{A}}\gamma_p(\mathcal{A}(y_l,y_r))}_{\substack{\text{Only depends on i,j}\\ \text{(Eq. 5 and Fn. 1.)}}} \underbrace{\sum_{y_r} p_p(k+1,j;y_r)}_{\text{Eq. 4}} \underbrace{\big(-\sum_{y_l} p_p(i,k;y_l)\log p_q(i,k;y_l)\big)}_{\text{Eq. 3}} && (7)\\
&\quad + \sum_{\mathcal{A}}\gamma_p(\mathcal{A}(y_l,y_r))\sum_{y_l} p_p(i,k;y_l)\big(-\sum_{y_r} p_p(k+1,j;y_r)\log p_q(k+1,j;y_r)\big) \\
&\quad + \sum_{y_l} p_p(i,k;y_l)\sum_{y_r} p_p(k+1,j;y_r)\big(-\sum_{\mathcal{A}}\gamma_p(\mathcal{A}(y_l,y_r))\log\gamma_p(\mathcal{A}(y_l,y_r))\big) \\
&= \sum_{\mathcal{A}}\big[\gamma_p(\mathcal{A}(y_l,y_r))h(i,k) + \gamma_p(\mathcal{A}(y_l,y_r))h(k+1,j) - \gamma_p(\mathcal{A}(y_l,y_r))\log\gamma_q(\mathcal{A}(y_l,y_r))\big] && (8)
\end{aligned}
$$

hyperparameter. Specifically, the sparsity of the model is controlled by the value of the non-negative parameter K. Following Chen et al. (2020), we provide an approximate probability distribution with "$K+1$ dimensions" to estimates the KL divergence. In addition to the K most possible label predictions, the rest predicted labels are represented as the additional $+1$ dimension. We could modify K in the objective function to favor different degrees of sparsity. We refer this sparsity regularization as *Top-K* approach.

While *Top-K* estimates the KL divergence by designing an approximate distribution, the full probability distributions actually can be exactly computed as shown in Sec. 4.2. We manipulate the sparsity degree based on the exact computation by temperature control following Hinton et al. (2015). Specifically, we divide the logits of probability distributions by a temperature in the objective. A higher temperature results in softer probability distributions and often results in better KD performance. However, there is an opposite view of temperature. Grandvalet and Bengio (2004) applied a low temperature to sharpen predictions, which leads to a lower entropy, and showed that regularizing the predictions to have low entropy could be beneficial. When setting the two temperatures T_{orig} and T_{adv} (which refer to the temperatures of p_{orig} and p_{adv} respectively), we could adjust the sparsity degree in a more flexible way. Specifically, the sparsity of our model is controlled by the value of the non-negative parameter T_{orig} and T_{adv}. A smaller value of T_{orig} corresponds to a stronger sparsity in favor of an unambiguous model. When T_{orig} is set to 1, the learning algorithm can be considered as the exact computation. When $T_{orig} < 1$, our approach becomes a sparse version. When $T_{orig} > 1$, our approach falls into a smoother version. Models do not have a fixed degree of sparsity when targeting different datasets. For a given dataset, different models should be set different hyperparameters. Therefore, it is unclear how to choose an optimal temperature. To make it more flexible, we use different temperatures T_{orig} and T_{adv} for the two terms in KL. We refer to this sparsity regularization as *Temp-(T_{orig},T_{adv})* approach. *ExactComp-(T_{orig},T_{adv})* denotes applying *Temp-(T_{orig},T_{adv})* on TreeCRF with exact computation and *HeadSelect-(T_{orig},T_{adv})* denotes applying *Temp-(T_{orig},T_{adv})* on the head selection model Dozat and Manning (2017).

5 Experiments

5.1 Dataset

We evaluate our methods on the Wall Street Journal (WSJ) corpus with default training/development/test split (Cohen et al., 2008) for dependency parsing by unlabeled and labeled attachment score (UAS/LAS) (Han et al., 2020b).

We use Stanford dependencies 3.3 (Manning et al., 2014) to preprocess the WSJ corpus as in previous work. We consider several settings including full labeled WSJ data with extra unlabeled BLLIP corpus[4], and $x\%$ in WSJ as labeled data and the rest $(1-x)\%$ as unlabeled data. We use BLLIP as the unlabeled data pool, which has the

[4]Brown Laboratory for Linguistic Information Processing (BLLIP) 1987-89 WSJ Corpus Release 1

Setting	Labeled	Unlabeled
WSJ(10%/90%)	3,983	35,849
WSJ(30%/70%)	11,950	27,882
WSJ(50%/50%)	19,916	19,916
WSJ+BLLIP	39,832	650,000

Table 2: Statistic analysis of labeled and unlabeled training data. WSJ($x\%/(1-x)\%$) means $x\%$ of sentences are annotated while the remaining $(1-x)\%$ are not.

same data source as WSJ but contains much more sentences than the WSJ corpus. We drop sentences in BLLIP with length >20 to speed up training and balance the number of labeled and unlabeled data. All dataset settings we used to evaluate our method are listed in Tab. 2.

5.2 Setting

We directly adopt most hyper-parameters from Zhang et al. (2020). We train our supervised baseline for 200 epochs. For other models, we run semi-supervised training for 100 epochs after 100 epochs of purely supervised training.

5.3 Main Results

We report the averaged score over four random restarts for each model[5] and compare our models on dependency parsing. We tune hyperparameters and choose models according to the LAS score on th validation set. The results of small training data are shown in Tab. 3 on WSJ test data, including two settings: supervised learning and semi-supervised learning.

We focus on the semi-supervised settings and list supervised learning[6] for reference. We have three strong baselines reported in previous work: (1) Self-Training is the conventional self-training approach that uses the predicted data as extra labeled training data; (2) NCRFAE is the semi-supervised version of a neural CRF autoencoder (Cai et al., 2017)[7]. (3) Arc-Factored Sup/Semi are the supervised/semi-supervised version of the model from Wang and Tu (2020b). It can be seen that two variants of Spa generally outperform these three baselines with a margin. For example, *Top-2* outperforms Self-Training by 1.04% and Arc-Factored Sup by 0.31%. *ExactComp-(*0.3,2*)* outperforms Self-Training by 1.0% and Arc-Factored Sup by 0.27%.

There are also some interesting observations from different settings. We also apply VAT on the head selection distribution of each token (Dozat and Manning, 2017) (denoted as *HeadSelect-(*1,1*)*), in the sense that TreeCRF is not used, to show the efficiency of adversarial training without the tree structure constraint. Here two 1 in *HeadSelect-(*1,1*)* mean that sparsity adjustment is not used. In semi-supervised settings, *HeadSelect-(*1,1*)* is competitive and even outperforms some baselines with the structure constraint by a large margin. We suspect that it may be because of our good hyperparameters. Then after we set $T_{orig} = 0.3$ and $T_{adv} = 2$, an improvement is observed from 92.23% to 92.60%. It reveals the benefit of sparsity bias in the head selection model.

The second evidence of the benefit of sparsity bias lays on the *Top-K* Sparsity rows. All variants of *Top-K* including *Top-2*, *Top-3*, *Top-5*, and *Top-7* outperform the strong baselines.

Finally, a similar improvement can also been seen from *ExactComp-(*1,1*)* to *ExactComp-(*0.3,2*)*. This empirical result provides another piece of evidence for the superiority of Spa. Results show that *ExactComp-(*0.3,2*)* with both exact computation and sparsity adjustment consistently performs well, regardless of the different settings. This demonstrates that the non-sparsity problem limits the power of VAT.

In Tab. 4, models are fed with sufficient labeled data as well as unlabeled data. Results show that VAT provides consistent improvement, especially the model without sparsity regularization, *ExactComp-(*1,1*)*. Later analysis (Tab 6) also shows that a large amount of labeled data weakens the significance of the sparsity regularization. We argue that in this case, we have high quality p_{orig} such that no much inaccurate information is required to be ruled out.

5.4 Results of Different K

The value of K in Spa is an important hyper-parameter. If the value of K is too large, the model may consider too much possibilities of parses and hence the model is very likely to be misled. If

[5] If a setting requires to sample data, e.g., WSJ(10%/90%), we randomly sample data twice and run models using two randomly chosen seeds for each data. Otherwise, we run models using four randomly chosen seeds.

[6] The TreeCRF parser in this paper is different from the original version by an additional embedding normalization.

[7] We develop this neural version of CRF autoencoder dependency parser by Cai et al. (2017). For the self-training setting, we use the parser to predict parse trees of the unlabeled data iteratively and use the pseudo labeled data to update the model.

Approach		UAS	LAS
Supervised Learning			
Arc-Factored VAE Sup* (Wang and Tu, 2020b)		92.00	-
TreeCRF (Zhang et al., 2020)		92.11	89.51
Semi-supervised Learning			
Self-Training*		91.82	-
NCRFAE*		91.94	-
Arc-Factored VAE Semi* (Wang and Tu, 2020b)		92.55	-
W/O Sparsity	*HeadSelect-*(1,1)	92.23	89.80
	ExactComp-(1,1)	92.36	89.99
Top-K Sparsity	*Top-2*	**92.86**	90.38
	Top-3	92.76	90.36
	Top-5	92.74	90.35
	Top-7	92.79	90.45
Temp-(T_{orig},T_{adv}) Sparsity	*HeadSelect-*(0.3,2)	92.60	90.19
	ExactComp-(0.3,2)	**92.82**	90.45

Table 3: Results on Test data for a typical semi-supervised setting – 10% labeled WSJ+90% unlabeled WSJ. W/O Sparsity: Without Sparsity Adjustment. Result with a star * are reported by Wang and Tu (2020b).

Approach	UAS
Supervised Learning	
Zhang et al. (2020)	95.82
Semi-supervised Learning	
Top-3	95.92
ExactComp-(1,1)	95.99
ExactComp-(0.3,2)	95.84

Table 4: UAS Results on Test data for the semi-supervised setting – WSJ+BLIIP650k.

the value of K is too small, the model loses the benefit of expressiveness. As Tab. 5 illustrates, value of $K = 3$ leads to the best parsing accuracy, while other values produce lower parsing accuracy probably because of inappropriate sparsity degrees.

6 Analysis

6.1 Ablation Study

In this section we study the effectiveness of our two sparsity adjustment on different settings: exact computation to ease the computation errors and sparsity adjustment to add a prior of sparsity property. As show in Tab. 6, the sparsity adjustment is not only successfully applied on low-resource setting, namely the 10%WSJ+90%WSJ setting, but also works on other settings (*i.e.*, 30%WSJ+70%WSJ and 50%WSJ+50%WSJ).

Exact computation is capable of improving

	10%+90%	**30%+70%**	**50%+50%**
Supervised Learning			
Sup	92.00	93.94	94.38
TreeCRF	92.11	94.43	95.28
Semi-supervised Learning			
Semi	92.55	94.15	94.41
Top-2	92.86	94.74	95.47
Top-3	92.76	95.00	95.54
Top-5	92.74	94.93	95.35
Top-7	92.79	94.76	95.51

Table 5: UAS Results of different K in various semi-supervised setting. **X%+Y%**: X% labled WSJ+Y% unlabled WSJ. Sup: Arc-Factored VAE Sup (Wang and Tu, 2020b). Semi: Arc-Factored VAE Semi (Wang and Tu, 2020b). TreeCRF: (Zhang et al., 2020).

		10%+90%	**30%+70%**	**50%+50%**
W/O	*HeadSelect-*(1,1)	92.23	94.65	95.11
	ExactComp-(1,1)	92.36	94.66	**95.47**
W	*HeadSelect-*(0.3,2)	92.60	94.83	95.31
	ExactComp-(0.3,2)	**92.82**	**94.85**	**95.47**

Table 6: With Sparsity *vs*. Without Sparsity. in various semi-supervised setting. W/O: W/O Sparsity. W: *Temp-*(T_{orig},T_{adv}). **X%+Y%**: X% labled WSJ+Y% unlabled WSJ.

the parsing result on all the settings (including 10%WSJ+90%WSJ, 30%WSJ+70%WSJ and 50%WSJ+50%WSJ). It shows that the model takes advantage of eliminating the approximation problem. When simultaneously combining the sparsity adjustment and the sparsity adjustment, we observe a further improvement on the final result in the *ExactComp-*(0.3,2) row.

We provide other results, including inspection of non-sparse problems and speed comparison, in the Appendix.

7 Conclusion and Further Work

In this paper, we propose **S**parse **P**arse **A**djustment algorithm (Spa). We successfully applied VAT to the dependency parsing task using this Spa algorithm. We use Spa to enhance the TreeCRF parser with exact computation and sparsity adjustment. Further empirical study indicates that Spa has strong effects in semi-supervised settings and time and space efficiency. Furthermore, this approach has broad applications on other structured prediction tasks. The exact computation for the TreeCRF can be easily transferred to general structured prediction architectures, *e.g.*, LinearChain-CRF. We will leave it as a further work.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (61976139). Wenjuan Han has been supported by the National Key R&D Program of China (2020AAA0108005) and the National Nature Science Foundation of China (No. 61976015, 61976016, 61876198 and 61370130). The authors would like to thank the anonymous reviewers for their valuable comments and suggestions to improve this paper.

References

Naveed Akhtar and Ajmal Mian. 2018. Threat of adversarial attacks on deep learning in computer vision: A survey. *Ieee Access*, 6:14410–14430.

David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin A Raffel. 2019. Mixmatch: A holistic approach to semi-supervised learning. *Advances in Neural Information Processing Systems*, 32.

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*, pages 1638–1643.

Luoxin Chen, Weitong Ruan, Xinyue Liu, and Jianhua Lu. 2020. SeqVAT: Virtual adversarial training for semi-supervised sequence labeling. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8801–8811, Online. Association for Computational Linguistics.

Yong Cheng, Lu Jiang, and Wolfgang Macherey. 2019. Robust neural machine translation with doubly adversarial inputs. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4324–4333, Florence, Italy. Association for Computational Linguistics.

Yong Cheng, Lu Jiang, Wolfgang Macherey, and Jacob Eisenstein. 2020. AdvAug: Robust adversarial augmentation for neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5961–5970, Online. Association for Computational Linguistics.

Kevin Clark, Minh-Thang Luong, Christopher D. Manning, and Quoc Le. 2018. Semi-supervised sequence modeling with cross-view training. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1914–1925, Brussels, Belgium. Association for Computational Linguistics.

Shay B Cohen, Kevin Gimpel, and Noah A Smith. 2008. Logistic normal priors for unsupervised probabilistic grammar induction. In *Advances in Neural Information Processing Systems*, pages 321–328.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing.

Jason Eisner. 2000. Bilexical grammars and their cubic-time parsing algorithms. In *Advances in probabilistic and other parsing technologies*, pages 29–61. Springer.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. *Advances in neural information processing systems*, 29.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples.

Joshua Goodman. 1999. Semiring parsing. *Computational Linguistics*, 25(4):573–606.

Yves Grandvalet and Yoshua Bengio. 2004. Semi-supervised learning by entropy minimization. In *Proceedings of the 17th International Conference on Neural Information Processing Systems*, NIPS'04, page 529–536, Cambridge, MA, USA. MIT Press.

Yves Grandvalet and Yoshua Bengio. 2005. Semi-supervised learning by entropy minimization. In *Advances in Neural Information Processing Systems*, volume 17, pages 529–536. MIT Press.

Wenjuan Han, Yong Jiang, Hwee Tou Ng, and Kewei Tu. 2020a. A survey of unsupervised dependency parsing. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2522–2533, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Wenjuan Han, Yong Jiang, Hwee Tou Ng, and Kewei Tu. 2020b. A survey of unsupervised dependency parsing. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2522–2533.

Wenjuan Han, Liwen Zhang, Yong Jiang, and Kewei Tu. 2020c. Adversarial attack and defense of structured prediction models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2327–2338, Online. Association for Computational Linguistics.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network.

Geoffrey E. Hinton, Simon Osindero, and Yee Whye Teh. 2006. A fast learning algorithm for deep belief nets. *Neural Comput.*, 18(7):1527–1554.

Rebecca Hwa. 2000. Sample selection for statistical grammar induction. In *2000 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 45–52, Hong Kong, China. Association for Computational Linguistics.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031.

Zhifei Li and Jason Eisner. 2009. First- and second-order expectation semirings with applications to minimum-risk training on translation forests. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 40–51, Singapore. Association for Computational Linguistics.

Xiaobo* Liang, Lijun* Wu, Juntao Li, Yue Wang, Qi Meng, Tao Qin, Wei Chen, Min Zhang, and Tie-Yan Liu. 2021. R-drop: Regularized dropout for neural networks. In *NeurIPS*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany. Association for Computational Linguistics.

Lars Maaløe, Casper Kaae Sønderby, Søren Kaae Sønderby, and Ole Winther. 2016. Auxiliary deep generative models.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60.

Takeru Miyato, Andrew M. Dai, and Ian Goodfellow. 2017a. Adversarial training methods for semi-supervised text classification.

Takeru Miyato, Andrew M. Dai, and Ian J. Goodfellow. 2017b. Adversarial training methods for semi-supervised text classification. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Takeru Miyato, Shin ichi Maeda, Masanori Koyama, and Shin Ishii. 2018. Virtual adversarial training: A regularization method for supervised and semi-supervised learning.

Noriki Nishida and Hideki Nakayama. 2020. Unsupervised discourse constituency parsing using viterbi em. *Transactions of the Association for Computational Linguistics*, 8:215–230.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Hoifung Poon and Pedro Domingos. 2011. Sum-product networks: A new deep architecture. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 689–690. IEEE.

Sebastian Ruder and Barbara Plank. 2018. Strong baselines for neural semi-supervised learning under domain shift. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1044–1054. Association for Computational Linguistics.

Libin Shen, Jinxi Xu, and Ralph Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proceedings of ACL-08: HLT*, pages 577–585.

Valentin I Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2011. Lateen em: Unsupervised training with multiple objectives, applied to dependency grammar induction. In *EMNLP*.

Valentin I Spitkovsky, Hiyan Alshawi, Daniel Jurafsky, and Christopher D Manning. 2010. Viterbi training improves unsupervised dependency parsing. In *CoNLL*.

Kewei Tu and Vasant 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*, pages 1324–1334.

Dilin Wang, Chengyue Gong, and Qiang Liu. 2019. Improving neural language modeling via adversarial training. In *International Conference on Machine Learning*, pages 6555–6565.

Ge Wang and Kewei Tu. 2020a. Semi-supervised dependency parsing with arc-factored variational autoencoding. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2485–2496, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Ge Wang and Kewei Tu. 2020b. Semi-supervised dependency parsing with arc-factored variational autoencoding. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2485–2496.

Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V Le. 2019. Unsupervised data augmentation for consistency training. *arXiv preprint arXiv:1904.12848*.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *33rd Annual Meeting of the Association for Computational Linguistics*, pages 189–196, Cambridge, Massachusetts, USA. Association for Computational Linguistics.

Michihiro Yasunaga, Jungo Kasai, and Dragomir Radev. 2018. Robust multilingual part-of-speech tagging via adversarial training. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 976–986, New Orleans, Louisiana. Association for Computational Linguistics.

Huangzhao Zhang, Hao Zhou, Ning Miao, and Lei Li. 2019. Generating fluent adversarial examples for natural languages. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5564–5569, Florence, Italy. Association for Computational Linguistics.

Liwen Zhang, Zixia Jia, Wenjuan Han, Zilong Zheng, and Kewei Tu. 2022. SHARP: Search-based adversarial attack for structured prediction. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 950–961, Seattle, United States. Association for Computational Linguistics.

Liwen Zhang, Ge Wang, Wenjuan Han, and Kewei Tu. 2021. Adapting unsupervised syntactic parsing methodology for discourse dependency parsing. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5782–5794, Online. Association for Computational Linguistics.

Yu Zhang, Zhenghua Li, and Min Zhang. 2020. Efficient second-order TreeCRF for neural dependency parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3295–3305, Online. Association for Computational Linguistics.

Zhi-Hua Zhou and Ming Li. 2005. Tri-training: exploiting unlabeled data using three classifiers. *IEEE Transactions on Knowledge and Data Engineering*, 17(11):1529–1541.

A Hyper-Parameters Setting

We adopt most hyperparameters of the TreeCRF parser (Zhang et al., 2020). We only list parameters different from them and VAT-specific parameters in Table 7.

Name	Value
Base model	
Maximum epochs	$\{200, 100 + 100\}$
VAT-specific	
Update steps for d_w, d_c	1
α	1
ξ in Miyato et al. (2018)	0.5
ϵ in Miyato et al. (2018)	0.1
Normalization on	Token
Temperature of p, q	$\{0.3, 0.7, 1, 2\}$

Table 7: Hyper-parameters of our methods.

B Other Results

B.1 Speed Comparison

Computing the *Top-K* distribution cost more and time than our exact computation, since the former has to record the *Top-K* candidates at each step in the routine. We report the training time per epoch of several methods (Tab. 8) on WSJ(10%/90%) running on one Nvidia RTX3090.

Method	Time/epoch
Supervised	27s
HeadSelect-(1,1)	1min9s
Top-3	2min36s
ExactComp-(1,1)	1min41s

Table 8: Training speed of *Top-K* and our exact computation with batch size 64.

B.2 Analysis of Sparsity

We conduct an experiments about the motivation of sparsity adjustment. Fig.2 shows the number of the gold parse tree in the *Top-K* beams. We can see that most of the gold parses are existed in the *Top-10* parse trees. Quantitatively, we find that the number of the parses increase roughly before 4. After K reaches a large number, *e.g.*, 7, the leaning may not be easy. This observation is consistent with our empirical experiments and further suggests that natural language parsing are indeed should be adjusted in favor of sparsity.

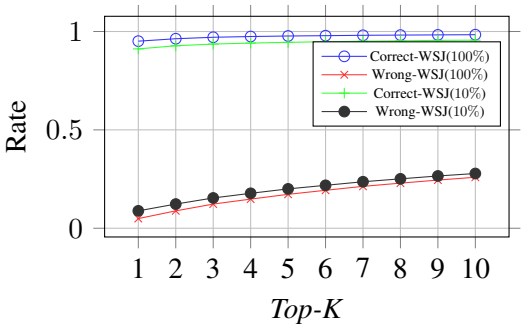

Figure 2: Correct-*: #(all arcs of *Top-K* trees ∩ gold arcs) / #tokens. Wrong-*: #(all arcs of *Top-K* trees - gold arcs) / #tokens. *-WSJ(100%)/*-WSJ(10%): the model train on the full/10% WSJ training set. We only count sentences with length⩾ 5.