Semi-supervised Parsing with Variational Autoencoding Parser

Xiao Zhang
Department of Computer Science
Purdue University
zhang923@purdue.edu

Dan Goldwasser
Department of Computer Science
Purdue University
dgoldwas@purdue.edu

Abstract

We propose an end-to-end variational autoencoding parsing (VAP) model for semi-supervised graph-based projective dependency parsing. It encodes the input using continuous latent variables in a sequential manner by deep neural networks (DNN) that can utilize the contextual information, and reconstruct the input using a generative model. The VAP model admits a unified structure with different loss functions for labeled and unlabeled data with shared parameters. We conducted experiments on the WSJ data sets, showing the proposed model can use the unlabeled data to increase the performance on a limited amount of labeled data, on a par with a recently proposed semi-supervised parser with faster inference.

1 Introduction

Dependency parsing captures bi-lexical relationships by constructing directional arcs between words, defining a head-modifier syntactic structure for sentences, as shown in Figure 1. Dependency trees are fundamental for many downstream tasks such as semantic parsing (Reddy et al., 2016; Marcheggiani and Titov, 2017), machine translation (Bastings et al., 2017; Ding and Palmer, 2007), information extraction (Culotta and Sorensen, 2004; Liu et al., 2015) and question answering (Cui et al., 2005). Recently, efficient parsers (Kiperwasser and Goldberg, 2016; Dozat and Manning, 2017; Dozat et al., 2017; Ma et al., 2018) have been developed using various neural architectures.

While supervised approaches have been very successful, they require large amounts of labeled data, particularly when neural architectures are used, which usually are over-parameterized. Syntactic annotation is notoriously difficult and requires specialized linguistic expertise, posing a serious challenge for low-resource languages. Semi-supervised parsing aims to alleviate this problem by combining a small amount of labeled data and a large amount of unlabeled data, to improve parsing performance on using labeled data alone. Traditional semi-supervised parsers use unlabeled data to generate additional features in order to assist the learning process (Koo et al., 2008), together with different variants of self-training (Søgaard, 2010). However, these approaches are usually pipe-lined and error-propagation may occur.

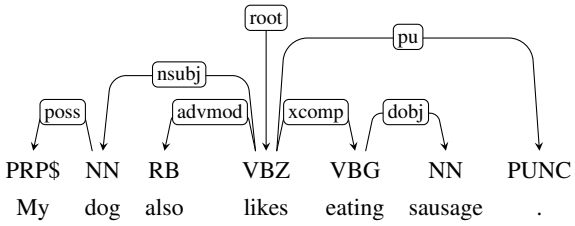


Figure 1: A dependency tree: directional arcs represent head-modifier relation between words.

In this paper, we propose Variational Autoencoding Parser, or VAP, extends the idea of VAE, illustrated in Figure 3. The VAP model uses unlabeled examples to learn continuous latent variables of the sentence, which can be used to support tree inference by providing an enriched representation.

We summarize our contributions as follows:

1. We proposed a Variational Autoencoding Parser (VAP) for semi-supervised dependency parsing;
2. We designed a unified loss function for the proposed parser to deal with both labeled and unlabeled data.
3. We show improved performance of the proposed model with unlabeled data on the WSJ data sets, and the performance is on a par with a recently proposed semi-supervised parser (Corro and Titov, 2019), with faster inference.

2 Related Work

Most dependency parsing studies fall into two major groups: graph-based and transition-based

Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task, pages 40–47

(Kubler et al., 2009). Graph-based parsers (McDonald, 2006) regard parsing as a structured prediction problem to find the most probable tree, while transition-based parsers (Nivre, 2004, 2008) treat parsing as a sequence of actions at different stages leading to a dependency tree.

While earlier works relied on manual feature engineering, in recent years the hand-crafted features were replaced by embeddings and deep neural network architectures were used to learn representation for scoring structural decisions, leading to improved performance in both graph-based and transition-based parsing (Nivre, 2014; Pei et al., 2015; Chen and Manning, 2014; Dyer et al., 2015; Weiss et al., 2015; Andor et al., 2016; Kiperwasser and Goldberg, 2016; Wiseman and Rush, 2016).

The annotation difficulty for this task, has also motivated work on unsupervised (grammar induction) and semi-supervised approaches to parsing (Tu and Honavar, 2012; Jiang et al., 2016; Koo et al., 2008; Li et al., 2014; Kiperwasser and Goldberg, 2015; Cai et al., 2017; Corro and Titov, 2019). It also leads to advances in using unlabeled data for constituent grammar (Shen et al., 2018b,a)

Similar to other structured prediction tasks, directly optimizing the objective is difficult when the underlying probabilistic model requires marginalizing over the dependency trees. Variational approaches are a natural way to alleviate this difficulty, as they try to improve the lower bound of the original objective, and have been applied in several recent NLP works (Stratos, 2019; Chen et al., 2018; Kim et al., 2019b,a). Variational Autoencoder (VAE) (Kingma and Welling, 2014) is particularly useful for latent representation learning, and has been studied in semi-supervised context as the Conditional VAE (CVAE) (Sohn et al., 2015). Note our work differs from VAE as VAE is designed for tabular data but not for structured prediction, as the input towards VAP is the sequence of sentential tokens and the output is the dependency tree.

3 Graph-based Dependency Parsing

A dependency graph of a sentence can be regarded as a directed tree spanning all the words of the sentence, including a special "word"–the ROOT–to originate out. Assuming a sentence of length l, a dependency tree can be denoted as $\mathcal{T} = (< h_1, m_1 >, \ldots, < h_{l-1}, m_{l-1} >)$, where h_t is the index in the sequence of the head word of the dependency connecting the tth word m_t as a modifier.

Our graph-based VAP parser is constructed based on the following standard structured prediction paradigm (McDonald et al., 2005; Taskar et al., 2005). In inference, based on the scoring function $\mathcal{S}_{\boldsymbol{\Lambda}}$ with parameter $\boldsymbol{\Lambda}$, the parsing problem is formulated as finding the most probable directed spanning tree for a given sentence $\boldsymbol{x}$:

$$\mathcal{T}^* = \arg\max_{\tilde{\mathcal{T}} \in \mathbb{T}} \mathcal{S}_{\boldsymbol{\Lambda}}(\boldsymbol{x}, \tilde{\mathcal{T}}),$$

where $\mathcal{T}^*$ is the highest scoring parse tree and $\mathbb{T}$ is the set of all valid trees for the sentence $\boldsymbol{x}$.

It is common to factorize the score of the entire graph into the summation of its substructures–the individual arc scores (McDonald et al., 2005):

$$\mathcal{S}_{\boldsymbol{\Lambda}}(\boldsymbol{x}, \tilde{\mathcal{T}}) = \sum_{(h,m) \in \tilde{\mathcal{T}}} s_{\boldsymbol{\Lambda}}(h, m) = \sum_{t=1}^{l} s_{\boldsymbol{\Lambda}}(h_t, m_t),$$

where $\tilde{\mathcal{T}}$ represents the candidate parse tree, and $s_{\boldsymbol{\Lambda}}$ is a function scoring individual arcs. $s_{\boldsymbol{\Lambda}}(h, m)$ describes the likelihood of an arc from the head h to its modifier m in the tree. Throughout this paper, the scoring is based on individual arcs, as we focus on *first-order* parsing.

3.1 Scoring Function Using Neural Architecture

We used the same neural architecture as that in Kiperwasser and Goldberg (2016)'s study. We first use a bi-LSTM model to take as input $\boldsymbol{u}_t = [\boldsymbol{p}_t; \boldsymbol{e}_t]$ at position t to incorporate contextual information, by feeding the word embedding $\boldsymbol{e}_t$ concatenated with the POS tag embeddings $\boldsymbol{p}_t$ of each word. The bi-LSTM then projects $\boldsymbol{u}_t$ as $\boldsymbol{o}_t$.

Subsequently a nonlinear transformation is applied on these projections. Suppose the hidden states generated by the bi-LSTM are $[\boldsymbol{o}_{root}, \boldsymbol{o}_1, \boldsymbol{o}_2, \ldots, \boldsymbol{o}_t, \ldots, \boldsymbol{o}_l]$, for a sentence of length l, we compute the arc scores by introducing parameters $\boldsymbol{W}_h$, $\boldsymbol{W}_m$, $\boldsymbol{w}$ and $\boldsymbol{b}$, and transform them as follows:

$$\boldsymbol{r}_t^{h-arc} = \boldsymbol{W}_h \boldsymbol{o}_t; \quad \boldsymbol{r}_t^{m-arc} = \boldsymbol{W}_m \boldsymbol{o}_t,$$
$$s_{\boldsymbol{\Lambda}}(h, m) = \boldsymbol{w}^\intercal(\tanh(\boldsymbol{r}_h^{h-arc} + \boldsymbol{r}_m^{m-arc} + \boldsymbol{b})).$$

In this formulation, we first use two parameters to extract two different representations that carry two different types of information: a head seeking for its modifier (h-arc) and a modifier seeking for its head (m-arc). Then a nonlinear function maps them to an arc score.

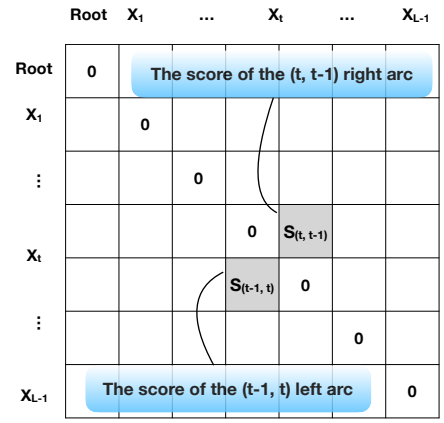
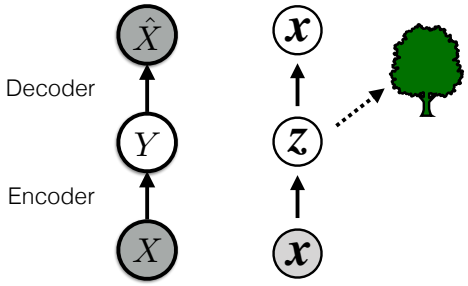


Figure 2: In this illustration of the arc scoring matrix, each entry represents the $(h(head) \rightarrow m(modifier))$ score.



(a) VAE (b) VAP

Figure 3: Illustration of variational autoencoder (VAE)(left) and variational autoencoding parser (VAP)(right).

For a single sentence, we can form a scoring matrix as shown in Figure 2, by filling each entry in the matrix using the score we obtained. Therefore, the scoring matrix is used to represent the head-modifier arc score for all the possible arcs connecting two tokens in a sentence (Zheng, 2017). Using this scoring arc matrix, we build our graph-based parser.

4 Variational Autoencoding Parser

VAP (illustrated in Figure 3b) is a semi-supervised parser able to make use of unlabeled data in addition to labeled data, extending the idea of variational autoencoder (VAE, illustrated in Figure 3a) to dependency parsing.

VAP learns, using both labeled and unlabeled data, a continuous latent variables representation, designed to support the parsing task by creating contextualized token-representations that capture properties of the full sentence. Typically, each token in the sentence is represented by its latent variable z_t, which is a high-dimensional Gaussian variable, to be aggregated as a group of latent variables $\boldsymbol{z}$. This configuration ensures the continuous latent variable retains the contextual information from lower-level neural models to assist finding its head or its modifier; as well as forcing the representation of similar tokens to be closer. The latent variable group $\boldsymbol{z}$ is modeled via $P(\boldsymbol{z}|\boldsymbol{x})$. In addition, we model the process of reconstructing the input sentence from the latent variable through a generative story $P(\boldsymbol{x}|\boldsymbol{z})$.

We adjust the original VAE setup in our semi-supervised task by considering examples with labels, similar to recent conditional variational formulations (Sohn et al., 2015; Miao and Blunsom, 2016; Zhou and Neubig, 2017). We propose a full probabilistic model for a given sentence $\boldsymbol{x}$, with the unified objective to maximize for both supervised and unsupervised parsing as follows:

$$\mathcal{J} = \log P_{\boldsymbol{\theta}}(\boldsymbol{x}) P^{\epsilon}_{\boldsymbol{\omega}}(\mathcal{T}|\boldsymbol{x}), \quad \epsilon = \begin{cases} 1, & \text{if } \mathcal{T} \text{ exists,} \\ 0, & \text{otherwise.} \end{cases}$$

This objective can be interpreted as follows: if the training example has a golden tree $\mathcal{T}$ with it, then the objective is the log joint probability $P_{\boldsymbol{\theta},\boldsymbol{\omega}}(\mathcal{T}, \boldsymbol{x})$; if the golden tree is missing, then the objective is the log marginal probability $P_{\boldsymbol{\theta}}(\boldsymbol{x})$. The probability of a certain tree is modeled by a tree-CRF with parameters $\boldsymbol{\omega}$ as $P_{\boldsymbol{\omega}}(\mathcal{T}|\boldsymbol{x})$. Given the assumed generative process $P_{\theta}(\boldsymbol{x}|\boldsymbol{z})$, directly optimizing this objective is intractable, thus instead we optimize its Evidence Lower BOund (ELBO):

$$\begin{aligned} \mathcal{J}_{lap} = & \mathop{\mathbb{E}}_{\boldsymbol{z}\sim Q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})} \left[\log P_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})\right] \\ & - \mathbb{KL}(Q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})||P_{\boldsymbol{\theta}}(\boldsymbol{z})) \\ & + \epsilon \mathop{\mathbb{E}}_{\boldsymbol{z}\sim Q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})} \left[\log P_{\boldsymbol{\omega}}(\mathcal{T}|\boldsymbol{z})\right]. \end{aligned}$$

We show J_{lap} is the ELBO of J in the appendix A.1.

In practice, similar as VAE-style models, $\mathop{\mathbb{E}}_{\boldsymbol{z}\sim Q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})} \left[\log P_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})\right]$ is approximated by $\frac{1}{N}\sum_{j=1}^{N} \log P_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}_j)$ and $\mathop{\mathbb{E}}_{\boldsymbol{z}\sim Q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})} \left[\log P_{\boldsymbol{\omega}}(\mathcal{T}|\boldsymbol{z})\right]$ by $\frac{1}{N}\sum_{j=1}^{N} \log P_{\boldsymbol{\omega}}(\mathcal{T}|\boldsymbol{z}_j)$, where $\boldsymbol{z}_j$ is the j-th sample of N samples sampled from $Q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$. At prediction stage, we simply use $\mu_{\boldsymbol{z}}$ rather than sampling $\boldsymbol{z}$.

4.1 Incorporating POS and External Embeddings

In previous studies (Chen and Manning, 2014; Dozat and Manning, 2017; Dozat et al., 2017;

Kiperwasser and Goldberg, 2016) exploring parsing using neural architectures, POS tags and external embeddings have been shown to contain important information characterizing the dependency relationship between a head and a child. Therefore, in addition to the variational autoencoding framework taking as input the randomly initialized word embeddings, optionally we can build the same structure for POS to reconstruct tags and for external embeddings to reconstruct words as well, whose variational objectives are $\mathcal{U}_p$ and $\mathcal{U}_e$ respectively. Hence, the final variational objective can be a combination of three: $\mathcal{U} = \mathcal{U}_w$(The original $\mathcal{U}$ in Lemma A.1)$+\mathcal{U}_p+\mathcal{U}_e$ (or just $\mathcal{U} = \mathcal{U}_w + \mathcal{U}_p$ if external embeddings are not used).

5 Experiments

5.1 Experimental Settings

Data sets We compared our models' performance with strong baselines on the WSJ data set, which is the Stanford Dependency conversion (De Marneffe and Manning, 2008) of the Penn Treebank (Marcus et al., 1993). We used the standard section split: 2-21 for training, 22 for development and 23 for testing.

To simulate the low-resource language environment, we used 10% of the whole training set as the annotated, and the rest 90% as the unlabeled.

Input Representation and Architecture We describe the details of the architecture as follows: The internal word embeddings have dimension 100 and the POS embeddings have dimension 25. The hidden layer of the bi-LSTM layer is of dimension 125. The nonlinear layers used to form the head and the modifier representation both have 100 dimension. We also used separate bi-LSTMs for words and POSs.

Training In the training phase, we usedd Adam (Kingma and Ba, 2014) to learn all the parameters in the VAP model. We did not take efforts to tune models' hyper-parameters and they remained the same across all the experiments. To preventing over-fitting, we applied the "early stop" strategy by using the development set.

5.2 Semi-Supervised Dependency Parsing on WSJ Data Set

We evaluated our VAP model on the WSJ data set and compared the model performance with other semi-supervised parsing models, including CRFAE (Cai et al., 2017), which is originally designed for dependency grammar induction but can be adopted for semi-supervised parsing, and "differentiable Perturb-and-Parse" parser (DPPP) (Corro and Titov, 2019). To contextualize the results, we also experiment with the supervised neural margin-based parser (NMP) (Kiperwasser and Goldberg, 2016), neural tree-CRF parser (NTP) and the supervised version of VAP, with only the labeled data. To ensure a fair comparison, our experimental set up on the WSJ is identical as that in DPPP, using the same 100 dimension skip-gram word embeddings employed in an earlier transition-based system (Dyer et al., 2015). We show our experimental results in Table 1.

Model	UAS
DPPP(L)	88.79
DPPP(L+U)	**89.50**
CRFAE(L+U)	82.34
NMP(L)	89.64
NTP (L)	89.63
Self-training (L+U)	87.81
VAP (L)	89.37
VAP (L+U)	**89.49**

Table 1: This table compares the model performance on the WSJ data set with 10% labeled data. "L" means only 10% labeled data is used, while "L+U" means both 10% labeled and 90% unlabeled data are used.

As shown in this table, our VAP model is able to utilize the unlabeled data to improve the overall performance on that with only using labeled data alone. Our VAP model performs slightly worse than the NMP model, which we attribute to the increased model complexity by incorporating extra encoder and decoders to deal with the latent variable. However, our VAP model achieved comparable results on semi-supervised parsing as the DPPP model, while our VAP model is simple and straightforward without inferencing the parse tree if it is unknown. Instead, the DPPP model has to apply Monte Carlo sampling from the posterior of the structure by using a "GUMBEL-MAX trick" to approximate the categorical distribution at each step, which is intensively computationally expensive, in order to form a dependency tree of high probability. Self-training using NMP with both labeled and unlabeled data is also included as a base-line, where the performance is deteriorated without appropriately using the unlabeled data.

6 Conclusion

In this study, we presented Variational Autoencoding Parser (VAP), an end-to-end parser, capable of utilizing the unlabeled data together with labeled data to improve the parsing performance, without any external resources. The proposed VAP model performs on a par with a recently published (Corro and Titov, 2019) semi-supervised parsing system on the WSJ data set, with faster inference, showing its potential for low-resource languages.

References

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proc. of the Annual Meeting of the Association Computational Linguistics (ACL)*.

Joost Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Simaan. 2017. Graph Convolutional Encoders for Syntax-aware Neural Machine Translation. In *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating Sentences from a Continuous Space. In *Proc. of the International Conference on Learning Representation (ICLR)*.

Jiong Cai, Yong Jiang, and Kewei Tu. 2017. CRF Autoencoder for Unsupervised Dependency Parsing. In *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*.

Danqi Chen and Christopher D Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*.

Mingda Chen, Qingming Tang, Karen Livescu, and Kevin Gimpel. 2018. Variational sequential labelers for semi-supervised learning. In *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*.

Caio Corro and Ivan Titov. 2019. Differentiable Perturb-and-Parse: Semi-Supervised Parsing with a Structured Variational Autoencoder. In *Proc. of the International Conference on Learning Representation (ICLR)*.

Hang Cui, Renxu Sun, Keya Li, Min-Yen Kan, and Tat-Seng Chua. 2005. Question Answering Passage Retrieval Using Dependency Relations. In *Proc. of the International Conference on Research and Development in Information Retrieval (SIGIR)*.

Aron Culotta and Jeffrey Sorensen. 2004. Dependency tree kernels for relation extraction. In *Proc. of the Annual Meeting of the Association Computational Linguistics (ACL)*.

Marie-Catherine De Marneffe and Christopher D Manning. 2008. The Stanford typed dependencies representation. In *Proc. of the International Conference on Computational Linguistics (COLING)*.

Yuan Ding and Martha Palmer. 2007. Machine translation using probabilistic synchronous dependency insertion grammars. In *Proc. of the Annual Meeting of the Association Computational Linguistics (ACL)*.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *Proc. of the International Conference on Learning Representation (ICLR)*.

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 the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-Based Dependency Parsing with Stack Long Short-Term Memory. In *Proc. of the Annual Meeting of the Association Computational Linguistics (ACL)*.

Yong Jiang, Wenjuan Han, and Kewei Tu. 2016. Unsupervised Neural Dependency Parsing. In *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*.

Yoon Kim, Chris Dyer, and Alexander Rush. 2019a. Compound probabilistic context-free grammars for grammar induction. In *Proc. of the Annual Meeting of the Association Computational Linguistics (ACL)*.

Yoon Kim, Alexander Rush, Lei Yu, Adhiguna Kuncoro, Chris Dyer, and Gábor Melis. 2019b. Unsupervised recurrent neural network grammars. In *Proc. of the Annual Meeting of the North American Association of Computational Linguistics (NAACL)*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *ArXiv*.

Diederik P Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *Proc. of the International Conference on Learning Representation (ICLR)*.

Eliyahu Kiperwasser and Yoav Goldberg. 2015. Semi-supervised dependency parsing using bilexical contextual features from auto-parsed data. In *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional lstm feature representations. *Transactions of the Association for Computational Linguistics (TACL)*.

Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple Semi-supervised Dependency Parsing. In *Proc. of the Annual Meeting of the Association Computational Linguistics (ACL)*.

Sandra Kubler, Ryan McDonald, Joakim Nivre, and Graeme Hirst. 2009. *Dependency Parsing*. Morgan and Claypool Publishers.

Zhenghua Li, Min Zhang, and Wenliang Chen. 2014. Ambiguity-aware ensemble training for semi-supervised dependency parsing. In *Proc. of the Annual Meeting of the Association Computational Linguistics (ACL)*.

Yang Liu, Furu Wei, Sujian Li, Heng Ji, Ming Zhou, and Houfeng Wang. 2015. A Dependency-Based Neural Network for Relation Classification. In *Proc. of the Annual Meeting of the Association Computational Linguistics (ACL)*.

Xuezhe Ma, Zecong Hu, Jingzhou Liu, Nanyun Peng, Graham Neubig, and Eduard Hovy. 2018. Stack-Pointer Networks for Dependency Parsing. In *Proc. of the Annual Meeting of the Association Computational Linguistics (ACL)*. Association for Computational Linguistics.

Diego Marcheggiani and Ivan Titov. 2017. Encoding Sentences with Graph Convolutional Networks for Semantic Role Labeling. In *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: the penn treebank. *Computational Linguistics*.

Ryan McDonald. 2006. *Discriminative learning and spanning tree algorithms for dependency parsing*. Ph.D. thesis, University of Pennsylvania.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proc. of the Annual Meeting of the Association Computational Linguistics (ACL)*.

Yishu Miao and Phil Blunsom. 2016. Language as a latent variable: Discrete generative models for sentence compression. In *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*.

Joakim Nivre. 2004. Incrementality in deterministic dependency parsing. In *Proceedings of the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together*.

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*.

Joakim Nivre. 2014. The inside-outside recursive neural network model for dependency parsing. In *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*.

Wenzhe Pei, Tao Ge, and Baobao Chang. 2015. An effective neural network model for graph-based dependency parsing. In *Proc. of the Annual Meeting of the Association Computational Linguistics (ACL)*.

Siva Reddy, Oscar Tackstrom, Michael Collins, Tom Kwiatkowski, Dipanjan Das, Mark Steedman, and Mirella Lapata. 2016. Transforming Dependency Structures to Logical Forms for Semantic Parsing. *Transactions of the Association for Computational Linguistics (TACL)*, pages 127–140.

Yikang Shen, Zhouhan Lin, Chin wei Huang, and Aaron Courville. 2018a. Neural language modeling by jointly learning syntax and lexicon. In *Proc. of the International Conference on Learning Representation (ICLR)*.

Yikang Shen, Shawn Tan, Alessandro Sordoni, and Aaron Courville. 2018b. Ordered neurons: Integrating tree structures into recurrent neural networks. In *Proc. of the International Conference on Learning Representation (ICLR)*.

Anders Søgaard. 2010. Simple semi-supervised training of part-of-speech taggers. In *Proc. of the Annual Meeting of the Association Computational Linguistics (ACL)*.

Kihyuk Sohn, Xinchen Yan, and Honglak Lee. 2015. Learning structured output representation using deep conditional generative models. In *Proc. of the Conference on Advances in Neural Information Processing Systems (NIPS)*.

Karl Stratos. 2019. Mutual Information Maximization for Simple and Accurate Part-Of-Speech Induction. In *Proc. of the Annual Meeting of the North American Association of Computational Linguistics (NAACL)*.

Toshiyuki Tanaka. 1999. A Theory of Mean Field Approximation. In *Proc. of the Conference on Advances in Neural Information Processing Systems (NIPS)*.

Ben Taskar, Vassil Chatalbashev, Daphne Koller, and Carlos Guestrin. 2005. Learning structured prediction models: A large margin approach. In *Proc. of the International Conference on Machine Learning (ICML)*.

Kewei Tu and Vasant Honavar. 2012. Unambiguity regularization for unsupervised learning of probabilistic grammars. In *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*.

David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Structured training for neural network transition-based parsing. In *Proc. of the Annual Meeting of the Association Computational Linguistics (ACL)*.

Sam Wiseman and Alexander M. Rush. 2016. Sequence-to-sequence learning as beam-search optimization. In *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*.

Xiaoqing Zheng. 2017. Incremental graph-based neural dependency parsing. In *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*.

Chunting Zhou and Graham Neubig. 2017. Multi-space variational encoder-decoders for semi-supervised labeled sequence transduction. In *Proc. of the Annual Meeting of the Association Computational Linguistics (ACL)*.

A Appendix

A.1 ELBO of LAP's Original Ojective

Given an input sequence $\boldsymbol{x}$, we prove that $\mathcal{J}_{lap}$ is the ELBO of $\mathcal{J}$:

Lemma A.1. *$\mathcal{J}_{lap}$ is the ELBO (evidence lower bound) of the original objective $\mathcal{J}$, with an input sequence $\boldsymbol{x}$.*

Denote the encoder Q is a distribution used to approximate the true posterior distribution $P_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$, parameterized by $\boldsymbol{\phi}$ such that Q encoding the input into the latent space $\boldsymbol{z}$.

Proof.

$$\log P_{\boldsymbol{\theta}}(\boldsymbol{x}) P^{\epsilon}_{\boldsymbol{\omega}}(\mathcal{T}|\boldsymbol{x}) = \underbrace{\log P_{\boldsymbol{\theta}}(\boldsymbol{x})}_{\mathcal{U}} + \underbrace{\epsilon \log P_{\boldsymbol{\omega}}(\mathcal{T}|\boldsymbol{x})}_{\mathcal{L}}$$

$$\begin{aligned}
\mathcal{U} &= \log \int_z Q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \frac{P_{\boldsymbol{\theta}}(\boldsymbol{x})}{Q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})} d_z \\
&\geq \mathop{\mathbb{E}}_{\boldsymbol{z}\sim Q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})} [\log P_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})] \\
&\quad - \mathop{\mathbb{E}}_{\boldsymbol{z}\sim Q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})} \left[\log \frac{Q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}{P_{\boldsymbol{\theta}}(\boldsymbol{x})}\right] \\
&= \mathop{\mathbb{E}}_{\boldsymbol{z}\sim Q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})} [\log P_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})] \\
&\quad - \mathbb{KL}\left(Q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) || P_{\boldsymbol{\theta}}(\boldsymbol{z})\right), \\
&\quad \text{(ELBO of traditional VAE)} \\
\mathcal{L} &= \epsilon \log P_{\boldsymbol{\omega}}(\mathcal{T}|\boldsymbol{x}) \\
&= \epsilon \log \int_z P_{\boldsymbol{\omega}}(\mathcal{T}|\boldsymbol{z}) Q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) d_z \\
&= \epsilon \log \mathop{\mathbb{E}}_{\boldsymbol{z}\sim Q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})} [P_{\boldsymbol{\omega}}(\mathcal{T}|\boldsymbol{z})] \\
&\geq \epsilon \mathop{\mathbb{E}}_{\boldsymbol{z}\sim Q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})} [\log P_{\boldsymbol{\omega}}(\mathcal{T}|\boldsymbol{z})].
\end{aligned}$$

Combining $\mathcal{U}$ and $\mathcal{L}$ leads to the fact:

$$\begin{aligned}
\mathcal{U} + \mathcal{L} \geq & \mathop{\mathbb{E}}_{\boldsymbol{z}\sim Q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})} [\log P_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})] \\
& - \mathbb{KL}(Q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) || P_{\boldsymbol{\theta}}(\boldsymbol{z})) \\
& + \epsilon \mathop{\mathbb{E}}_{\boldsymbol{z}\sim Q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})} [\log P_{\boldsymbol{\omega}}(\mathcal{T}|\boldsymbol{z})] = \mathcal{J}_{lap}
\end{aligned}$$

□

A.2 Mean Field Approximation and Annealing

Directly calculating the the auxiliary posterior $Q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ is difficult due to the complex model structure with deep neural networks, thus we used a mean field approximation (Tanaka, 1999) together with the conditional independence assumption by assuming $Q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) = \prod_{t=1}^{l} Q_{\boldsymbol{\phi}}(z_t|x_t)$ to compute it. Similarly, the generative model $P_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$, acting as a decoder parameterized by $\boldsymbol{\theta}$, tries to regenerate the input x_t at time step t given the latent variable z_t, assuming $P_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) = \prod_{t=1}^{l} P_{\boldsymbol{\theta}}(x_t|z_t)$. The encoder and the decoder are trained jointly using the traditional variational autoencoder framework, by minimizing the KL divergence between the approximated posterior and the true posterior.

We parameterize the encoder $Q_{\boldsymbol{\phi}}(z_t|x_t)$ in two steps: First a bi-LSTM is used to obtain a non-linear transformation h_t of the original x_t; then two separate MLPs are used to compute the mean $\boldsymbol{\mu}_{z_t}$ and the variance $\boldsymbol{\sigma}^2_{z_t}$. The generative story $P_{\boldsymbol{\theta}}(x_t|z_t)$ follows such parameterization: we used a MLP of two hidden layers in-between to take z_t as the input, and predict the word (or POS tag) over the vocabulary, such that the reconstruction probability can be measured.

Following traditional VAE training paradigms, we also apply the "re-parameterization" trick (Kingma and Welling, 2014) to circumvent the non-differentiable sampling procedure to sample z_t from the $Q_{\boldsymbol{\phi}}(z_t|x_t)$. Instead of directly sample from $\mathcal{N}(\boldsymbol{\mu}_{z_t}, \boldsymbol{\sigma}^2_{z_t})$, we form $z_t = \boldsymbol{\mu}_{z_t} + \boldsymbol{\epsilon} \odot \boldsymbol{\sigma}^2_{z_t}$ by sampling $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. In addition, to avoid hindering learning during the initial training phases, following previous works (Chen et al., 2018; Bowman et al., 2016), we anneal the temperature on the KL divergence term from a small value to 1.

A.3 Empirical Bayesian Treatment

From an empirical Bayesian perspective, it is beneficial to estimate the prior distribution directly from the data by treating prior's parameters part of the model parameters, rather than fixing the prior using some certain distributions. Similar to the approach used in the previous study (Chen et al., 2018), LAP also learns the priors from the data by updating them iteratively. We initialize the priors from a standard Gaussian distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$, where $\mathbf{I}$ is an identity matrix. During the training, the current priors are updated using the last optimized posterior, following the rule:

$$\pi^k_{\boldsymbol{\theta}}(\boldsymbol{z}) = \sum_{\boldsymbol{x}} Q^{k-1}_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) P(\boldsymbol{x}),$$

where $P(\boldsymbol{x})$ represents the empirical data distribution, and k the iteration step. Empirical Bayesian is also named as "maximum marginal likelihood", such that our approach here is to marginalize over the missing observation as a random variable.