Semi-Supervised Semantic Dependency Parsing Using CRF Autoencoders

Zixia Jia[◇], **Youmi Ma**[†], **Jiong Cai**[◇], **Kewei Tu**[◇*]
[◇]School of Information Science and Technology, ShanghaiTech University
[◇]Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences
[◇]University of Chinese Academy of Sciences
[◇]Shanghai Engineering Research Center of Intelligent Vision and Imaging
[†]Tokyo Institute of Technology
{jiazx, caijiong, tukw}@shanghaitech.edu.cn
youmi.ma@nlp.c.titech.ac.jp

Abstract

Semantic dependency parsing, which aims to find rich bi-lexical relationships, allows words to have multiple dependency heads, resulting in graph-structured representations. We propose an approach to semi-supervised learning of semantic dependency parsers based on the CRF autoencoder framework. Our encoder is a discriminative neural semantic dependency parser that predicts the latent parse graph of the input sentence. Our decoder is a generative neural model that reconstructs the input sentence conditioned on the latent parse graph. Our model is arc-factored and therefore parsing and learning are both tractable. Experiments show our model achieves significant and consistent improvement over the supervised baseline.

1 Introduction

Semantic dependency parsing (SDP) is a task aiming at discovering sentence-internal linguistic information. The focus of SDP is the identification of predicate-argument relationships for all content words inside a sentence (Oepen et al., 2014, 2015). Compared with syntactic dependencies, semantic dependencies are more general, allowing a word to be either unattached or the argument of multiple predicates. The set of semantic dependencies within a sentence form a directed acyclic graph (DAG), distinguishing SDP from syntactic dependency parsing tasks, where dependencies are usually tree-structured. Extraction of such high-level structured semantic information potentially benefits downstream NLP tasks (Reddy et al., 2017; Schuster et al., 2017).

Several supervised SDP models are proposed in the recent years by modifying syntactic dependency parsers. Their parsing mechanisms are either transition-based (Kanerva et al., 2015; Wang et al., 2018) or graph-based (Martins and Almeida, 2014; Peng et al., 2017; Dozat and Manning, 2018; Wang et al., 2019).

One limitation of supervised SDP is that labeled SDP data resources are limited in scale and diversity. Due to the rich relationships in SDP, the annotation of semantic dependency graphs is expensive and difficult, calling for professional linguists to design rules and highly skilled annotators to annotate sentences. This limitation becomes more severe with the rise of deep learning, because neural approaches are more data-hungry and susceptible to over-fitting when lacking training data. To alleviate this limitation, we investigate semi-supervised SDP capable of learning from both labeled and unlabeled data.

While a lot of work has been done on supervised SDP, the research of unsupervised and semi-supervised SDP is still lacking. Since parsing results of semantic dependencies are DAGs without the tree-shape restriction, most existing successful unsupervised (Klein and Manning, 2004; I. Spitkovsky et al., 2010; Jiang et al., 2016; Cai et al., 2017) and semi-supervised (Koo et al., 2008; Druck et al., 2009; Suzuki et al., 2009; Corro and Titov, 2019) learning models for syntactic dependency parsing cannot be applied to SDP directly and it would be non-trivial to extend these models for SDP. There also exist several unsupervised (Poon and Domingos, 2009; Titov and Klementiev, 2011) and semi-supervised (Das and Smith, 2011; Kočiskỳ et al., 2016; Yin et al., 2018) methods for semantic parsing, but these models are designed for semantic representations different from dependency graphs, making their adaptation to SDP difficult.

In this work, we propose an end-to-end neural semi-supervised model leveraging both labeled and unlabeled data to learn a dependency graph parser. Our model employs the framework of Conditional

* Corresponding author.

Random Field Autoencoder (Ammar et al., 2014), modeling the conditional reconstruction probability given the input sentence with its dependency graph as the latent variable. Our encoder is the supervised model of Dozat and Manning (2018), formulating an SDP task as labeling each arc in a directed graph with a simple neural network. Analogous to a CRF model (Sutton et al., 2012), our encoder is capable of computing the probability of a dependency graph conditioned on the input sentence. The decoder is a generative model based on recurrent neural network language model (Mikolov et al., 2010), which formulates the probability of generating the input sentence, but we take into account the information given by the dependency parse graphs when generating the input.

Our model is arc-factored, i.e., the encoding, decoding and reconstructing probabilities can all be factorized into the product of arc-specific quantities, making both learning and parsing tractable. A unified learning objective is defined that takes advantage of both labeled and unlabeled data. Besides, compared with previous semi-supervised approaches based on Variational Autoencoder (Kingma and Welling, 2013), our learning process does not involve sampling, promising better stability.

We evaluate our model on SemEval 2015 Task 18 Dataset (English) (Oepen et al., 2015) and find that our model consistently outperforms the supervised baseline. We also conduct detailed analysis showing the benefits of different amounts of unlabeled data.

2 Model

Our model is based on the CRF autoencoder framework (Ammar et al., 2014) which provides a unified fashion for structured predictors to leverage both labeled and unlabeled data. A CRF autoencoder aims to produce a reconstruction of the input $\hat{\mathrm{X}}$ from the original input X with an intermediate latent structure Y. It is trained to maximize the conditional reconstruction probability $P(\hat{\mathrm{X}} = \mathrm{X}|\mathrm{X})$ with the latent variable Y marginalized. Ideally, successful reconstruction implies that the latent structure captures important information of the input.

We adopt the following notations when describing our model. We represent a vector in lowercase bold, e.g., $\mathbf{s}$, and use a superscript for indexing, e.g., $\mathbf{s}^i$ for the i-th vector. We represent a scalar in lowercase italics, e.g., s, and use a subscript for indexing, e.g., s_i for the i-th element of vector $\mathbf{s}$. An uppercase italic letter such as Y denotes a matrix. A lower case letter with a subscript pair such as $y_{i,j}$ refers to the element of matrix Y at row i and column j. An uppercase bold letter, e.g., $\mathbf{U}$, stands for a tensor. We maintain this convention when indexing, e.g., $\mathbf{y}_i$ is the i-th row of matrix Y.

In our model, the input is a natural language sentence consisting of a sequence of words. A sentence with m words is represented by $\mathbf{s} = (s_0, s_1, s_2, \ldots, s_m)$, where s_0 is a special token TOP. The latent variable produced by our encoder is a dependency parse graph of the input sentence, represented as a matrix of booleans $Y \in \{0,1\}^{(m+1)\times(m+1)}$, where $y_{i,j} = 1$ indicates that there exists an dependency arc pointing from word s_i to word s_j. The reconstructed output generated by our decoder is a word sequence $\hat{\mathbf{s}} = (\hat{s}_1, \hat{s}_2, \ldots, \hat{s}_m)$.

Our encoder with parameters Θ computes $P_\Theta(Y|\mathbf{s})$, the probability of generating a dependency parse graph Y given a sentence $\mathbf{s}$. Our decoder with parameters Λ computes $P_\Lambda(\hat{\mathbf{s}}|Y)$, the probability of reconstructing sentence $\hat{\mathbf{s}}$ conditioned on the parse graph Y. The encoder and decoder in combination specify the following conditional distribution.

$$P_{\Theta,\Lambda}(\hat{\mathbf{s}}, Y|\mathbf{s}) = P_\Theta(Y|\mathbf{s})P_\Lambda(\hat{\mathbf{s}}|Y)$$

To compute the conditional probability $P(\hat{\mathbf{s}}|\mathbf{s})$, we sum out the latent variable Y.

$$P_{\Theta,\Lambda}(\hat{\mathbf{s}}|\mathbf{s}) = \sum_{Y \in \mathcal{Y}} P_{\Theta,\Lambda}(\hat{\mathbf{s}}, Y|\mathbf{s})$$

where $\mathcal{Y}$ is the set of all possible dependency parse graphs of $\mathbf{s}$. During training, we set $\hat{\mathbf{s}} = \mathbf{s}$ and maximize the conditional reconstruction probability $P(\hat{\mathbf{s}}|\mathbf{s})$.

Note that throughout our model, we only consider dependency arc predictions (i.e., whether an arc exists between each word pair). Arc-labels will be learned separately as described in Section 3. We leave the incorporation of arc-label prediction in our model for future work.

2.1 Encoder

Our encoder can be any arc-factored discriminative SDP model. Here we adopt the model of Dozat and Manning (2018), one of the best-performing SDP models, which formulates the semantic dependency parsing task as independently labeling each arc in

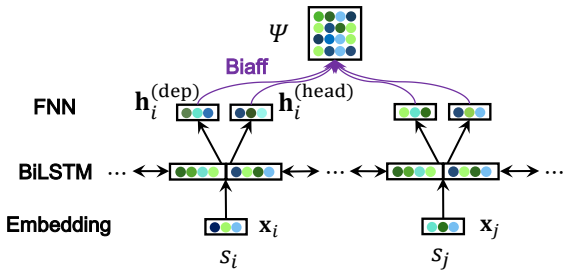


Figure 1: Illustration of the encoder, following the design of Dozat and Manning (2018).

a directed complete graph. To predict whether or not a directed arc (s_i, s_j) exists, the model computes contextualized representations of s_i and s_j and feeds them into a binary classifier.

The architecture of our encoder is shown in Figure 1. Word, part-of-speech tag (for short, POS tag), and lemma embeddings[1] of each word in the input sentence are concatenated and fed into a multi-layer bi-directional LSTM to get a contextualized representation of the word.

$$\mathbf{x}_i = \mathbf{e}_i^{(\text{word})} \oplus \mathbf{e}_i^{(\text{tag})} \oplus \mathbf{e}_i^{(\text{lemma})} \quad (1)$$
$$R = \mathbf{BiLSTM}(X)$$

where $\mathbf{e}_i^{(\text{word})}$, $\mathbf{e}_i^{(\text{tag})}$ and $\mathbf{e}_i^{(\text{lemma})}$ are notations for the word, POS tag and lemma embedding respectively, concatenated ($\oplus$) to form an embedding $\mathbf{x}_i$ for word s_i. Stacking $\mathbf{x}_i$ for $i = 0, 1, \ldots, m$ forms matrix X.

The contextualized word representation is then fed into two single-layer feedforward neural networks (FNN) with different parameters to produce two vectors: one for the representation of the word as a dependency head and the other for the representation of the word as a dependent. They are denoted as $\mathbf{h}_i^{(\text{head})}$ and $\mathbf{h}_i^{(\text{dep})}$ respectively.

$$\mathbf{h}_i^{(\text{head})} = \mathbf{FNN}^{(\text{enc-head})}(\mathbf{r}_i)$$
$$\mathbf{h}_i^{(\text{dep})} = \mathbf{FNN}^{(\text{enc-dep})}(\mathbf{r}_i)$$

Finally, a *biaffine* function is applied to every arc between word pairs (s_i, s_j) to obtain an arc-existence score $\psi_{i,j}$.

$$\psi_{i,j} = \mathbf{h}_i^{(\text{head})\top} W \mathbf{h}_j^{(\text{dep})} + b$$

where W is a square matrix of size $d \times d$ (d is the size of vector $\mathbf{h}_i^{(\text{head})}$ and $\mathbf{h}_j^{(\text{dep})}$) , and b is a scalar.

The likelihood of every arc's presence given a sentence, $P(y_{i,j} = 1|\mathbf{s})$, can be computed by applying a *sigmoid* function on score $\psi_{i,j}$. The arc-absence probability $P(y_{i,j} = 0|\mathbf{s})$ is evidently $1 - P(y_{i,j} = 1|\mathbf{s})$.

To conclude, the probability of producing a dependency parse graph Y from the encoder given an input sentence $\mathbf{s}$ can be computed as below.

$$P(Y|\mathbf{s}) = \prod_{i,j} P(y_{i,j}|\mathbf{s})$$

2.2 Decoder

Our generative decoder is based on recurrent neural network language models (Mikolov et al., 2010), but we take dependency relationships into account during reconstruction. Our inspiration sources from the decoder with a Graph Convolutional Network (GCN) used by Corro and Titov (2019) to incorporate tree-structured syntactic dependencies when generating sentences, but our decoder differs significantly from theirs in that ours handles parse graphs and is arc-factored.

As mentioned above, semantic dependency parsing allows a word to have multiple dependency heads. If we generate a word conditioned on multiple heads, then it becomes difficult if not impossible to make the decoder arc-factored and hence we may have to enumerate all parse graphs during parsing and learning, which is intractable. Instead, we propose to generate a word for multiple times, each time conditioned on a different head, which leads to a fully arc-factored generative decoder and hence tractable parsing and learning. Specifically, we split dependency graph Y of a sentence $\mathbf{s} = (s_0, s_1, \ldots, s_m)$ with m words and a TOP token into $m + 1$ parts:

$$Y = [\mathbf{y}_0; \mathbf{y}_1; \mathbf{y}_2; \ldots; \mathbf{y}_m]$$

Each $\mathbf{y}_i$ is the i-th row of Y, representing a sub-graph where arcs are rooted at the i-th word of the sentence $\mathbf{s}$. Mathematically, we have $\mathbf{y}_i = \{y_{i,j}|j \in (1, 2, ..., m)\}$.

We then generate $m + 1$ sentences $(\hat{\mathbf{s}}^0, \hat{\mathbf{s}}^1, \hat{\mathbf{s}}^2, \ldots, \hat{\mathbf{s}}^m)$ using $m + 1$ neural generators. The generation of sentence $\hat{\mathbf{s}}^i$ is guided by the i-th sub-graph $\mathbf{y}_i$. Each generator is a left-to-right LSTM language model and computes $P_\Lambda(\hat{s}_i^k|\hat{\mathbf{s}}_{0:i-1}^k, y_{k,i})$, the probability of generating

[1] The latest experimental results in Dozat and Manning (2018) show that using lemma embedding improves performance even further while including character-level word embedding produces little effect. Thus unless stated otherwise, our model makes use of lemma embeddings by default.

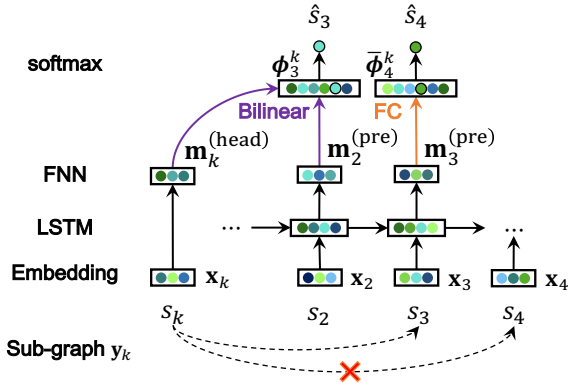


Figure 2: Illustration of the decoder generating $\hat{\mathbf{s}}^k$ from the k-th neural generator, guided by sub-graph $\mathbf{y}_k$. Dashed arcs at the bottom represent dependency arcs. A cross over arc (s_k, s_4) indicates the absence of this arc.

each word conditioned on its preceding words and whether $\mathbf{y}_k$ contains a dependency arc to the word. We share parameters among all the $m+1$ generators.

Figure 2 shows an example for computing the generative probability of $\hat{\mathbf{s}}^k$ by the k-th generator ($k \in \{0, 1, \ldots, m\}$) that incorporates the information of the k-th sub-graph $\mathbf{y}_k$. Recall that $\mathbf{y}_k$ contains only dependencies rooted at s_k. Below we describe how to compute the generative probability of each word $\hat{s}_i^k$ with and without the dependency arc (s_k, s_i) respectively.

Generative probability with a dependency Suppose there is a dependency arc from s_k to s_i, we need to compute the generative probability $P_\Lambda(\hat{s}_i^k | \hat{\mathbf{s}}_{0:i-1}^k, y_{k,i} = 1)$. The LSTM in the k-th generator takes the embedding of the previous word s_{i-1} computed through Eq.1 as its input and outputs the hidden state $\mathbf{g}_{i-1}$, which is fed into an FNN to produce a representation $\mathbf{m}_{i-1}^{(\text{pre})}$. Meanwhile, the embedding of the k-th word (also computed through Eq.1) is fed into another FNN to get its representation $\mathbf{m}_k^{(\text{head})}$ as a dependency head.

$$G = \mathbf{LSTM}(X)$$
$$\mathbf{m}_{i-1}^{(\text{pre})} = \mathbf{FNN}^{(\text{dec}-\text{pre})}(\mathbf{g}_{i-1}) \quad (2)$$
$$\mathbf{m}_k^{(\text{head})} = \mathbf{FNN}^{(\text{dec}-\text{head})}(\mathbf{x}_k) \quad (3)$$

$\mathbf{m}_k^{(\text{head})}$ and $\mathbf{m}_{i-1}^{(\text{pre})}$ are fed into a *bilinear* function to obtain a vocabulary-size score vector $\boldsymbol{\phi}_i^k$.

$$\boldsymbol{\phi}_i^k = \mathbf{m}_k^{(\text{head})\top} \mathbf{U} \mathbf{m}_{i-1}^{(\text{pre})} \quad (4)$$

Here, $\mathbf{U}$ is a tensor of size $d \times V \times d$, where V is the vocabulary size and d is the size of vector $\mathbf{m}_k^{(\text{head})}$ and $\mathbf{m}_{i-1}^{(\text{pre})}$. To conserve parameters, the tensor $\mathbf{U}$ is diagonal (i.e., $u_{i,k,j} = 0$ wherever $i \neq j$). A *softmax* function can then be applied to $\boldsymbol{\phi}_i^k$, from which we pick the generative probability of $\hat{s}_i^k$.

Generative probability without a dependency Suppose there is no dependency arc from s_k to s_i. In this case, reconstruction of $\hat{s}_i^k$ resembles a normal recurrent neural network language model. The representation $\mathbf{m}_{i-1}^{(\text{pre})}$ from Eq.2 is fed into a fully connected layer to get $\bar{\boldsymbol{\phi}}_i^k$, a vector of vocabulary size containing generative scores of all the words.

$$\bar{\boldsymbol{\phi}}_i^k = \mathbf{FC}(\mathbf{m}_{i-1}^{(\text{pre})}) \quad (5)$$

The generative probability $P_\Lambda(\hat{s}_i^k | \hat{\mathbf{s}}_{0:i-1}^k, y_{k,i} = 0)$ can then be computed by applying a *softmax* function on $\bar{\boldsymbol{\phi}}_i^k$ and selecting the corresponding probability of $\hat{s}_i^k$. Since we simply reconstruct word s_i without considering the dependency arc information, this probability is exactly the same in the $m+1$ generators and only needs to be computed once.

To conclude the overall design of our decoder, it is worth noting that in $m+1$ generation processes, parameters among all LSTMs are shared, as well as those among all FNNs[2] and FCs. Still, embeddings in Eq.1 are shared among both encoder and decoder.

With $P(\hat{s}_i^k | \hat{\mathbf{s}}_{0:i-1}^k, y_{k,i})$ computed for $i = 1, \ldots, m$, $k = 0, 1, \ldots, m$, the probability of generating $\hat{\mathbf{s}}^0, \hat{\mathbf{s}}^1, \hat{\mathbf{s}}^2, \ldots, \hat{\mathbf{s}}^m$ from dependency graph Y can be computed through:

$$P_\Lambda(\hat{\mathbf{s}}^0, \hat{\mathbf{s}}^1, \ldots, \hat{\mathbf{s}}^m | Y) = \prod_{k=0}^{m} P_\Lambda(\hat{\mathbf{s}}^k | \mathbf{y}_k)$$
$$= \prod_{k=0}^{m} \prod_{i=1}^{m} P_\Lambda(\hat{s}_i^k | \hat{\mathbf{s}}_{0:i-1}^k, y_{k,i})$$

In our model, we are only interested in the case where all the $m+1$ sentences are the same. In addition, to balance the influence of the encoder and the decoder, we take the geometric mean of the $m+1$ probabilities. The final decoding probability is defined as follows.

$$P_\Lambda(\hat{\mathbf{s}} | Y) := \prod_{i=1}^{m} \prod_{k=0}^{m} \sqrt[m+1]{P_\Lambda(\hat{s}_i^k | \hat{\mathbf{s}}_{0:i-1}^k, y_{k,i})}$$

[2] $\mathbf{FNN}^{(\text{dec}-\text{pre})}$ and $\mathbf{FNN}^{(\text{dec}-\text{head})}$ never share parameters between each other, since their usages are different.

Note that this is not a properly normalized probability distribution, but in practice we find it sufficient for semi-supervised SDP.

2.3 Parsing

Given parameters $\{\Theta, \Lambda\}$ of our encoder and decoder, we can parse a sentence $\mathbf{s}$ by finding a $Y \in \mathcal{Y}(\mathbf{s})$ which maximizes probability $P(\hat{\mathbf{s}} = \mathbf{s}, Y|\mathbf{s})$, where $\mathcal{Y}(\mathbf{s})$ is the set of all parse graphs of sentence $\mathbf{s}$.

$$\begin{aligned} Y^* &= \arg\max_{Y \in \mathcal{Y}(\mathbf{s})} \log P_{\Theta,\Lambda}(\hat{\mathbf{s}}, Y|\mathbf{s}) \\ &= \arg\max_{Y \in \mathcal{Y}(\mathbf{s})} \log P_\Theta(Y|\mathbf{s}) P_\Lambda(\hat{\mathbf{s}}|Y) \\ &= \arg\max_{Y \in \mathcal{Y}(\mathbf{s})} \sum_{i,j} \Big(\log P_\Theta(y_{i,j}|\mathbf{s}) \\ &\quad + \frac{1}{m+1} \log P_\Lambda(\hat{s}_j|\hat{\mathbf{s}}_{0:j-1}, y_{i,j}) \Big) \end{aligned} \tag{6}$$

Since the probability is arc-factored, we can determine the existence of each dependency arc independently by picking the value of $y_{i,j}$ that maximizes the corresponding term. The time complexity of our parsing algorithm is $O(m^2)$ for a sentence with m words.

3 Learning

Since we want to train our model in a semi-supervised manner, we design loss functions for labeled and unlabeled data respectively. For each training sentence $\mathbf{s}$, the overall loss function is defined as a combination of supervised loss $\mathcal{L}_l$ and unsupervised loss $\mathcal{L}_u$.

$$\mathcal{L}(\mathbf{s}) = \iota(\mathbf{s}) * \mathcal{L}_l(\mathbf{s}) + (1 - \iota(\mathbf{s})) * \rho \mathcal{L}_u(\mathbf{s}) \tag{7}$$

where an indicator $\iota(\mathbf{s}) \in \{0, 1\}$ specifies whether training sentence $\mathbf{s}$ is labeled or not and a tunable constant ρ balances the two losses.

Supervised Loss For any labeled sentence $(\mathbf{s}, Y^*)$, where $\mathbf{s}$ stands for a sentence and Y^* stands for a gold parse graph, we can compute the discriminative loss.

$$\mathcal{L}_l(\mathbf{s}) = -\log P_{\Theta,\Lambda}(\hat{\mathbf{s}} = \mathbf{s}, Y^*|\mathbf{s}) \tag{8}$$

Following the derivation of Eq.6, we have:

$$\begin{aligned} \log P_{\Theta,\Lambda}(\hat{\mathbf{s}}, Y^*|\mathbf{s}) = \sum_{i,j} \Big(&\log P_\Theta(y^*_{i,j}|\mathbf{s}) \\ &+ \frac{1}{m+1} \log P_\Lambda(\hat{s}_j|\hat{\mathbf{s}}_{0:j-1}, y^*_{i,j}) \Big) \end{aligned}$$

Usage	Source	Sentences	Tokens
train	WSJ Sec.00-20	35,656	802,717
test (id)	WSJ Sec.21	1,410	31,948
test (ood)	Brown	1,849	31,583

Table 1: The sources and scale of the SDP 2014 & 2015 (English) dataset. We extract WSJ Section 20 (1,692 sentences) from the train set for development purpose. **id** stands for in-domain testing, while **ood** stands for out-of-domain testing.

Gold parses also provide a label for each dependency. We follow Dozat and Manning (2018) and model dependency labels with a purely supervised module on top of the BiLSTM layer of the encoder. Its parameters are learned by optimizing a cross-entropy loss function.

Unsupervised Loss For any unlabeled sentence $\mathbf{s}$, we maximize the conditional reconstruction probability $P(\hat{\mathbf{s}} = \mathbf{s}|\mathbf{s})$. The unsupervised loss is:

$$\begin{aligned} \mathcal{L}_u(\mathbf{s}) &= -\log P_{\Theta,\Lambda}(\hat{\mathbf{s}}|\mathbf{s}) \\ &= -\log \sum_{Y \in \mathcal{Y}(\mathbf{s})} P_{\Theta,\Lambda}(Y, \hat{\mathbf{s}}|\mathbf{s}) \\ &= -\log \sum_{Y \in \mathcal{Y}(\mathbf{s})} P_\Theta(Y|\mathbf{s}) P_\Lambda(\hat{\mathbf{s}}|Y) \\ &= -\sum_{i,j} \log \sum_{y_{i,j} \in \{0,1\}} \Big(P_\Theta(y_{i,j}|\mathbf{s}) \\ &\quad \times \sqrt[m+1]{P_\Lambda(\hat{s}_j|\hat{\mathbf{s}}_{0:j-1}, y_{i,j})} \Big) \end{aligned} \tag{9}$$

Derivations of Eq.9 are provided in the Appendix A.

Given a dataset containing both labeled and unlabeled sentences, our model can be trained end-to-end by optimizing the loss function Eq.7 over the combined dataset using any gradient based method.

4 Experiments

4.1 Settings

Dataset We examine the performance of our model on the English corpus of the SDP 2014 & 2015: Broad Coverage Semantic Dependency Parsing dataset (Oepen et al., 2015). The corpus is composed of three distinct and parallel semantic dependency annotations (DM, PAS, PSD) of Sections 00-21 of the WSJ Corpus, as well as a balanced sample of twenty files from the Brown Corpus. The scale of this dataset is shown in Table 1.

Hidden Layer	Hidden Sizes
Word/GloVe/POS/Lemma/Char	100
GloVe Linear	125
Encoder BiLSTM	3*600
Encoder FNN(head)	1*600
Encoder FNN(dep)	1*600
Decoder UniLSTM	1*600
Decoder FNN(head)	1*400
Decoder FNN(pre)	1*400
Dropouts	**Dropout Prob.**
Word/GloVe/POS/Lemma	20%
Encoder/Decoder FNN	25 %
BiLSTM (FF/recur)	45%/25%
Optimizer & Loss	**Value**
Adam β_1	0
Adam β_2	0.95
Learning rate	$1e^{-3}$
L2 regularization	$3e^{-9}$

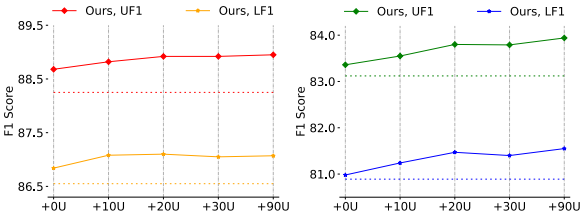
Table 2: Summary of hyper-parameters.

We evaluate the performance of models through two metrics: Unlabeled F1 score (UF1) and Labeled F1 score (LF1). UF1 measures the accuracy of the binary classification of arc existence, while LF1 measures the correctness of each arc-label as well. Unless stated otherwise, we report scores averaged over three runs.

Network Configuration For our encoder, we adopt the hyper-parameters of Dozat and Manning (2018). Following Dozat and Manning (2018), we concatenate pre-trained 100-dimensional GloVe embeddings (Pennington et al., 2014) linearly transformed to 125-dimension into our input word embeddings. Words or lemmas whose occurrences are less than 7 times within the training set are treated as UKN.

For our decoder, we set the number of layer(s) of uni-directional LSTM to 1, whose recurrent hidden size is 600. For $\mathbf{FNN}^{(\mathrm{dec-head})}$ and $\mathbf{FNN}^{(\mathrm{dec-pre})}$, the output sizes are both 400, activated by a $tanh(\cdot)$ function.

Learning Our loss function (Eq.7) is optimized by the Adam+AMSGrad optimizer (Reddi et al., 2018), with hyper-parameters β_1, β_2 kept the same as those of Dozat and Manning (2018). The interpolation constant ρ is tuned with the size of unlabeled data. A detailed table of hyper-parameters is provided in Table 2. The training time for one batch with our autoencoder is 2–3 times of that of Dozat and Manning (2018) because of the extra decoder.



(a) UF1 and LF1 for in-domain tests on DM.

(b) UF1 and LF1 for out-of-domain tests on DM.

Figure 3: Results with fixed amount of labeled data and varying amount of unlabeled data. **+10U** represents using 10% unlabeled data, and so on. Numbers on Y-axes represent F1 scores. Dashed horizontal lines are results of the supervised baseline (Dozat and Manning, 2018) trained on labeled data only. Solid lines are our results.

4.2 Varying Size of Unlabeled Data

In our first experiment (with the DM annotations only), we fix the amount of labeled data and continuously incorporate more unlabeled data into the training set.

Specifically, we randomly sample 10% of the whole dataset as labeled data. Unlabeled data are then sampled from the remaining part (with their gold parses removed), with a proportion increasing from 0% to 90% of the complete dataset.

For unlabeled data, we find that long sentences do not help in improving F1 scores and therefore in this and all the subsequent experiments we remove unlabeled sentences longer than 20 to reduce the running time and memory usage.

Experimental results are visualized in Figure 3. It is observed that in the purely supervised setting (i.e., +0% unlabeled data), our model already outperforms the baseline (Dozat and Manning, 2018). Since our encoder is exactly the baseline model, this shows the benefit of adding the decoder for joint learning and parsing even in the supervised setting. With an increasing size of unlabeled data, we can see the increase in performance of our model, especially when evaluated on out-of-domain data, suggesting the benefit of semi-supervised learning with our model.

4.3 Varying Proportion of Unlabeled Data

In our second experiment (again with the DM annotations), we use the full training set and vary the proportion of labeled and unlabeled data.

Experimental results are shown in Table 3. Our

	Models	Labeled:Unlabeled 0.1:9.9 UF1	0.1:9.9 LF1	1:9 UF1	1:9 LF1	3:7 UF1	3:7 LF1	5:5 UF1	5:5 LF1	10:0 UF1	10:0 LF1
id	D&M	75.21	70.70	88.32	86.60	91.65	90.52	92.81	91.90	94.11	93.38
	Ours-Sup	75.52	70.59	88.58	86.74	91.88	90.73	**92.99**	**92.05**	**94.30**	**93.55**
	Ours-Semi	**76.73**	**72.16**	**88.98**	**87.11**	**92.04**	**90.92**	**93.02**	**92.07**	-	-
ood	D&M	70.51	65.63	83.15	80.87	86.91	85.17	88.35	86.93	90.01	88.87
	Ours-Sup	70.53	65.48	83.33	80.92	87.16	85.45	**88.63**	**87.24**	**90.22**	**89.05**
	Ours-Semi	**72.18**	**67.30**	**83.93**	**81.48**	**87.43**	**85.70**	**88.67**	**87.28**	-	-

Table 3: Experimental results with varying proportions of labeled and unlabeled data. **D&M** stands for the supervised model of Dozat and Manning (2018) trained on labeled data only. **Ours-Sup** stands for our model trained on labeled data only. **Ours-Semi** stands for our model trained on both labeled and unlabeled data. All scores in this table are averaged over 10 runs. We do paired permutation test ($p < 0.05$). Two different scores being both boldfaced means that they are not significantly different.

semi-supervised model shows the largest advantage over the supervised models with the 0.1:9.9 proportion (which contains only 339 labeled sentences), indicating the strength of our mode in low resource setting.

With the increased proportion of labeled data, the performance of all the models goes up, but the advantage of our semi-supervised model diminishes. This is consistent with the tendency of many semi-supervised approaches to work well when given small labeled data but have diminishing effectiveness when adding more labeled data.

Another worth-noting observation is that the superiority of our semi-supervised model is much stronger on the out-of-domain tests, which suggests good generalizability of our semi-supervised model.

4.4 On All Representations

In the previous two experiments, we evaluate our model on the DM representation. Here we evaluate our model on all the three representations: DM, PAS and PSD. We slightly tune the hyper-parameters based on the optimal values from the previous experiments of the DM representation. We use 10% of the sentences as labeled data and the rest 90% of the sentences as unlabeled data. For the completeness of our experiment, we follow Dozat and Manning (2018) and examine four different word representations: **basic** (i.e., using only word and POS tag embeddings), **+Lemma** (i.e., using word, POS tag and lemma embeddings), **+Char** (i.e., using word, POS tag and character embeddings) and **+Lemma+Char** (i.e. using word, POS tag, lemma and character embeddings).

Table 4 shows the experimental results of **+Lemma**, the default word representation. The results of the other word representations show very similar trends (see the Table 7 in Appendix B). We observe significant improvement of our semi-supervised model over the two supervised baselines on both DM and PSD representations. However, it is surprising to find that on the PAS representation, our semi-supervised model exhibits little advantage over its supervised counterpart. One possible explanation, as Dozat and Manning (2018) also noted, is that PAS is the easiest of the three representations (as can be seen by comparing the scores of the three representations in Table 4) and our supervised model may already reach the performance ceiling.

4.5 Analysis of Our Decoder

We empirically study alternative structures of our decoder. In the first variant, we remove the LSTM layer of our decoder, so each word s_i is generated without access to the generation history before s_{i-1}. In the second variant, we replace the bilinear function in Eq.4 with a fully connected layer that takes as input either the concatenation or the summation of $\mathbf{m}_k^{(\text{head})}$ and $\mathbf{m}_{i-1}^{(\text{pre})}$. All the other settings are the same as in Section 4.4 on the DM annotation. Experimental results are shown in Table 5. We can see that these alternatives lead to worse scores, which verifies the effectiveness of our decoder design.

4.6 Stability of Our Model

To test the stability of our model, we repeat the experiment of Section 4.4 on the DM annotation for three times (without tuning hyper-parameters),

	Models	DM		PAS		PSD		Avg	
		UF1	LF1	UF1	LF1	UF1	LF1	UF1	LF1
id	D&M	88.32	86.60	91.89	90.57	88.17	73.42	89.46	83.53
	Ours-Sup	88.58	86.74	**92.14**	**90.91**	88.49	73.34	89.74	83.66
	Ours-Semi	**88.98**	**87.11**	**92.07**	**90.84**	**88.62**	**73.68**	**89.89**	**83.88**
ood	D&M	83.15	80.87	88.34	86.32	85.10	71.30	85.53	79.50
	Ours-Sup	83.33	80.92	**88.57**	**86.68**	85.09	71.11	85.66	79.57
	Ours-Semi	**83.93**	**81.48**	**88.61**	**86.68**	**85.30**	**71.46**	**85.95**	**79.87**

Table 4: Experimental results on all the three representations. All scores in this table are averaged over 10 runs. We do paired permutation test ($p < 0.05$). Two different scores being both boldfaced means that they are not significantly different.

	ID		OOD	
	UF1	LF1	UF1	LF1
Default	**88.95**	**87.07**	**83.94**	**81.55**
−LSTM	88.72	86.88	83.58	81.20
Concat	88.75	86.77	83.72	81.26
Sum	86.13	83.90	79.08	76.34

Table 5: Experimental results on different structures of our decoder. **Default** is our default semi-supervised model, **−LSTM** means removing the LSTM layer of our decoder, **Concat** stands for the concatenation setting, and **Sum** stands for the summation setting, as stated in Section 4.5.

each time with respect to different labeled data sampled from the training dataset. Table 6 shows the results. We observe consistent advantage of our model over the baseline on all the three datasets.

5 Related Work

Work on unsupervised or semi-supervised dependency parsing, to the best of our knowledge, is dominated by tree-structured parsing (Koo et al., 2008; Druck et al., 2009; Suzuki et al., 2009). Recently, Corro and Titov (2019) introduced an approximate inference method with a Variational Autoencoder (Kingma et al., 2014) for semi-supervised syntactic dependency parsing. Our decoder is inspired by their work, but differs from theirs in that our decoder handles parse graphs and is arc-factored. Cai et al. (2017) used the framework of CRF Autoencoder (Ammar et al., 2014) to perform unsupervised syntactic dependency parsing. The same framework has been used by Zhang et al. (2017) for semi-supervised sequence labeling. Our work also adopts the CRF Autoencoder framework, but with both the encoder and the decoder redesigned for semantic dependency parsing.

Existing unsupervised and semi-supervised approaches to semantic parsing focused on semantic representations different from dependency graphs, e.g., general-purpose logic forms (Sondheimer and Nebel, 1986) and formal meaning representations (Bordes et al., 2012). Poon and Domingos (2009) presented the first unsupervised semantic parser to transform dependency trees into quasi-logical forms with Markov logic. Following this work, Titov and Klementiev (2011) proposed a non-parametric Bayesian model for unsupervised semantic parsing using hierarchical PitmanYor process (Teh, 2006). Das and Smith (2011) described a semi-supervised approach to frame-semantic parsing. Kočiskỳ et al. (2016) proposed a semi-supervised semantic parsing approach making use of unpaired logical forms with sentence being unobserved. Recently, Yin et al. (2018) proposed a variational autoencoding model for semi-supervised semantic parsing of tree-structured semantic representations. Take Yin et al. (2018) for example. To extend their approach for SDP, one needs to design a different transition system for their encoder for graph parsing and design a graph linearization method for their sequence-to-sequence decoder. In addition, SDP-specific constraints (e.g., the graph structure contains exactly the same set of words as the sentence) shall be incorporated into their model. Therefore, previous semi-supervised semantic parsing models cannot be applied to SDP directly and modifying them for SDP is non-trivial. We leave for future work such modification and extension of previous semi-supervised semantic parsing approaches to SDP.

6 Conclusion

In this work, we proposed a semi-supervised learning model for semantic dependency parsing using

	Models	Data1		Data2		Data3		Avg	
		UF1	LF1	UF1	LF1	UF1	LF1	UF1	LF1
id	D&M	88.25	86.55	88.70	87.09	88.49	86.85	88.48	86.83
	Ours-Sup	88.68	86.84	88.79	86.96	88.71	86.97	88.73	86.92
	Ours-Semi	**88.95**	**87.07**	**89.24**	**87.45**	**88.97**	**87.19**	**89.05**	**87.24**
ood	D&M	83.12	80.89	83.30	81.01	83.62	81.26	83.35	81.05
	Ours-Sup	83.36	80.98	83.46	81.05	84.00	81.62	83.60	81.22
	Ours-Semi	**83.94**	**81.55**	**83.87**	**81.51**	**84.11**	**81.68**	**83.97**	**81.58**

Table 6: Experimental results on three randomly sampled datasets.

CRF Autoencoders. Our model is composed of a discriminative neural encoder producing a dependency graph conditioned on an input sentence, and a generative neural decoder for input reconstruction based on the dependency graph. The model works in an arc-factored fashion, promising end-to-end learning and efficient parsing.

We evaluated our model under both full-supervision settings and semi-supervision settings. Our model outperforms the baseline on multiple target representations. By adding unlabeled data, our model exhibits further performance improvements. In particular, our semi-supervised model performs well in the low resource setting and on the out-of-domain test set. This points to future directions of applying our model to low-resource languages and cross-domain settings. Our code is publicly available at `https://github.com/JZXXX/Semi-SDP`.

Acknowledgments

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

References

Waleed Ammar, Chris Dyer, and Noah A. Smith. 2014. Conditional Random Field Autoencoders for Unsupervised Structured Prediction. *Advances in Neural Information Processing Systems*, 4:3311–3319.

Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. 2012. Joint Learning of Words and Meaning Representations for Open-text Semantic Parsing. In *Artificial Intelligence and Statistics*, pages 127–135.

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)*.

Caio Corro and Ivan Titov. 2019. Differentiable Perturb-and-Parse: Semi-Supervised Parsing with a Structured Variational Autoencoder. In *International Conference on Learning Representations*.

Dipanjan Das and Noah A Smith. 2011. Semi-Supervised Frame-Semantic Parsing for Unknown Predicates. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1435–1444. Association for Computational Linguistics.

Timothy Dozat and Christopher D. Manning. 2018. Simpler but More Accurate Semantic Dependency Parsing. In *ACL*.

Gregory Druck, Gideon S. Mann, and Andrew McCallum. 2009. Semi-Supervised Learning of Dependency Parsers using Generalized Expectation Criteria. In *IJCNLP-ACL*.

Valentin I. Spitkovsky, Hiyan Alshawi, Daniel Jurafsky, and Christoper Manning. 2010. Viterbi Training Improves Unsupervised Dependency Parsing. *CoNLL 2010 - Fourteenth Conference on Computational Natural Language Learning, Proceedings of the Conference*, pages 9–17.

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*, pages 763–771.

Jenna Kanerva, Juhani Luotolahti, and Filip Ginter. 2015. Turku: Semantic Dependency Parsing as A Sequence Classification. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 965–969.

Diederik Kingma and Max Welling. 2013. Auto-Encoding Variational Bayes. *ICLR*.

Durk P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. 2014. Semi-Supervised Learning with Deep Generative Models. In *Advances in neural information processing systems*, pages 3581–3589.

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 on Association for Computational Linguistics, page 478. Association for Computational Linguistics.

Tomáš Kočiskỳ, Gábor Melis, Edward Grefenstette, Chris Dyer, Wang Ling, Phil Blunsom, and Karl Moritz Hermann. 2016. Semantic Parsing with Semi-Supervised Sequential Autoencoders. In *EMNLP*.

Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple Semi-Supervised Dependency Parsing. In *In Proceedings of ACL-08*.

André FT Martins and Mariana SC Almeida. 2014. Priberam: A Turbo Semantic Parser with Second Order Features. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 471–476.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent Neural Network Based Language Model. In *Eleventh annual conference of the international speech communication association*.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinkov, Dan Flickinger, Jan Haji, and Zdeka Ureov. 2015. SemEval 2015 Task 18: Broad-Coverage Semantic Dependency Parsing. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 915–926, Denver, CO, USA.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Dan Flickinger, Jan Hajic, Angelina Ivanova, and Yi Zhang. 2014. Semeval 2014 task 8: Broad-Coverage Semantic Dependency Parsing. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 63–72.

Hao Peng, Sam Thomson, and Noah A Smith. 2017. Deep Multitask Learning for Semantic Dependency Parsing. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

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. 2009. Unsupervised Semantic Parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 1–10. Association for Computational Linguistics.

Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. 2018. On the Convergence of Adam and Beyond. In *International Conference on Learning Representations*.

Siva Reddy, Oscar Tackstrom, Slav Petrov, Mark Steedman, and Mirella Lapata. 2017. Universal Semantic Parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Sebastian Schuster, Éric Villemonte de La Clergerie, Marie Candito, Benoît Sagot, Christopher Manning, and Djamé Seddah. 2017. Paris and Stanford at EPE 2017: Downstream Evaluation of Graph-based Dependency Representations. In *EPE 2017-The First Shared Task on Extrinsic Parser Evaluation*, pages 47–59.

Norman K Sondheimer and Bernhard Nebel. 1986. A Logical-Form and Knowledge-Base Design for Natural Language Generation. In *Proceedings of the workshop on Strategic computing natural language*, pages 231–241. Association for Computational Linguistics.

Charles Sutton, Andrew McCallum, et al. 2012. An Introduction to Conditional Random Fields. *Foundations and Trends® in Machine Learning*, 4(4):267–373.

Jun Suzuki, Hideki Isozaki, Xavier Carreras, and Michael Collins. 2009. An Empirical Study of Semi-Supervised Structured Conditional Models for Dependency Parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*, pages 551–560. Association for Computational Linguistics.

Yee Whye Teh. 2006. A Hierarchical Bayesian Language Model Based on Pitman-Yor Processes. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 985–992. Association for Computational Linguistics.

Ivan Titov and Alexandre Klementiev. 2011. A Bayesian Model for Unsupervised Semantic Parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1445–1455. Association for Computational Linguistics.

Xinyu Wang, Jingxian Huang, and Kewei Tu. 2019. Second-order semantic dependency parsing with end-to-end neural networks. *ACL*.

Yuxuan Wang, Wanxiang Che, Jiang Guo, and Ting Liu. 2018. A Neural Transition-Based Approach for Semantic Dependency Graph Parsing. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Pengcheng Yin, Chunting Zhou, Junxian He, and Graham Neubig. 2018. StructVAE: Tree-Structured Latent Variable Models for Semi-Supervised Semantic Parsing. In *ACL*.

Xiao Zhang, Yong Jiang, Hao Peng, Kewei Tu, and Dan Goldwasser. 2017. Semi-Supervised Structured Prediction with Neural CRF Autoencoder. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1701–1711.

	Models	DM		PAS		PSD		Avg	
		UF1	LF1	UF1	LF1	UF1	LF1	UF1	LF1
id, basic	D&M	87.48	85.36	91.82	90.46	87.9	72.74	89.07	82.86
	Ours-Sup	87.57	85.4	91.92	90.61	88.00	72.70	89.16	82.90
	Ours-Semi	**88.27**	**85.96**	**92.16**	**90.85**	**88.27**	**73.06**	**89.57**	**83.23**
ood, basic	D&M	82.32	79.6	88.30	86.29	84.35	70.18	84.99	78.69
	Ours-Sup	82.34	79.59	88.51	86.56	84.39	70.25	85.08	78.80
	Ours-Semi	**83.19**	**80.29**	**88.65**	**86.71**	**84.56**	**70.45**	**85.47**	**79.15**
id, +Char	D&M	87.66	85.68	91.93	90.58	87.85	72.74	89.15	83.00
	Ours-Sup	87.84	85.71	**92.23**	**90.99**	88.25	72.77	89.44	83.16
	Ours-Semi	**88.21**	**85.99**	92.20	90.93	**88.27**	**73.26**	**89.56**	**83.39**
ood, +Char	D&M	82.50	80.07	88.27	86.33	84.51	70.48	85.09	78.96
	Ours-Sup	82.60	79.99	88.71	**86.84**	84.59	70.46	85.30	79.09
	Ours-Semi	**83.23**	**80.41**	**88.74**	86.79	**84.84**	**71.00**	**85.60**	**79.40**
id, +Lemma+Char	D&M	88.47	86.87	92.10	90.83	88.3	73.54	89.62	83.74
	Ours-Sup	88.68	86.91	**92.23**	90.95	88.61	73.39	89.84	83.75
	Ours-Semi	**88.95**	**87.22**	**92.23**	**90.97**	**88.71**	**73.56**	**89.96**	**83.92**
ood, +Lemma+Char	D&M	83.42	81.31	88.55	86.59	85.23	71.34	85.73	79.75
	Ours-Sup	83.64	81.31	88.71	86.82	85.28	71.41	85.88	79.85
	Ours-Semi	**83.93**	**81.67**	**89.00**	**87.09**	**85.37**	**71.65**	**86.10**	**80.14**

Table 7: Experimental results on all the three representations. **D&M** stands for the supervised model of Dozat and Manning (2018). **Ours-Sup** stands for our model trained on labeled data only. **Ours-Semi** stands for our model trained on both labeled and unlabeled data.

A Detailed Derivation

Derivation of the marginalized probability over all possible dependency graphs of a sentence with m words for Eq.9 is shown below.

$$
\begin{aligned}
&\log \sum_{Y \in \mathcal{Y}} P(\hat{\mathbf{s}}, Y|\mathbf{s}) = \log \sum_{Y \in \mathcal{Y}} P(Y|\mathbf{s}) P(\hat{\mathbf{s}}|Y) \\
&= \log \sum_{Y \in \mathcal{Y}} \Big(\prod_{i,j} P(y_{i,j}|\mathbf{s}) \sqrt[m+1]{P(\hat{s}_j|\hat{\mathbf{s}}_{0:j-1}, y_{i,j})} \Big) \\
&= \log \sum_{y_{0,1}} ... \sum_{y_{i,j}} ... \sum_{y_{m,m}} \Big(\prod_{i,j} P(y_{i,j}|\mathbf{s}) \\
&\qquad \times \sqrt[m+1]{P(\hat{s}_j|\hat{\mathbf{s}}_{0:j-1}, y_{i,j})} \Big) \\
&= \log \Big(\Big(\sum_{y_{0,1}} P(y_{0,1}|\mathbf{s}) \sqrt[m+1]{P(\hat{s}_1|\hat{s}_0, y_{0,1})} \Big) \\
&\quad ... \Big(\sum_{y_{m,m}} P(y_{m,m}|\mathbf{s}) \sqrt[m+1]{P(\hat{s}_m|\hat{\mathbf{s}}_{0:m-1}, y_{m,m})} \Big) \Big) \\
&= \log \prod_{i,j} \Big(\sum_{y_{i,j}} P(y_{i,j}|\mathbf{s}) \sqrt[m+1]{P(\hat{s}_j|\hat{\mathbf{s}}_{0:j-1}, y_{i,j})} \Big) \\
&= \sum_{i,j} \log \Big(\sum_{y_{i,j}} \Big(P(y_{i,j}|\mathbf{s}) \\
&\qquad \times \sqrt[m+1]{P(\hat{s}_j)|\hat{\mathbf{s}}_{0:j-1}, y_{i,j})} \Big) \Big)
\end{aligned}
$$

B Experiments on All Representations

Results for experiments on DM, PAS and PSD under the setting of Section 4.4 (i.e., **basic**, **+Char** and **+Lemma+Char**) are summarized in Table 7.