Semi-Supervised Semantic Dependency Parsing Using CRF Autoencoders

Zixia Jia, Youmi Ma†, Jiong Cai, Kewei Tu

◊ ShanghaiTech University
† Tokyo Institute of Technology

{jiazx, caijiong, tukw}@shanghaitech.edu.cn
youni.ma@nlp.c.titech.ac.jp

June 16, 2020
Overview

1 Problem and Motivation
2 Model
3 Experiments and Results
4 Conclusion
Semantic dependency parsing

- Semantic dependency parsing graph

**DM**
- The company also adopted an anti-takeover plan.

**PAS**
- The company also adopted an anti-takeover plan.

**PSD**
- The company also adopted an anti-takeover plan.
Motivation

- Due to the rich relationships in SDP, the annotation of semantic dependency graphs is expensive and difficult.
- Neural approaches are more data-hungry and susceptible to over-fitting when lacking training data.
- While a lot of work has been done on supervised SDP, the research of unsupervised and semi-supervised SDP is still lacking.
A CRF autoencoder aims to produce a reconstruction of the input \( \hat{X} \) from the original input \( X \) with an intermediate latent structure \( Y \).

\[
P(\hat{X} = X|X) = \sum_Y P(\hat{X}, Y|X) = \sum_Y P(Y|X)P(\hat{X}|Y)
\]

The encoder is a CRF, exploit features from input \( X \), can keep exact inference tractable (to get \( Y \)), computes \( P(Y|X) \).

The decoder is reconstruction part, generate a copy of input conditioned on the latent \( Y \), computes \( P(\hat{X}|Y) \).
Model: overview

\[ s = (\text{Top}, s_1, s_2, \ldots, s_m) \quad \hat{s} = (\hat{s}_1, \hat{s}_2, \ldots, \hat{s}_m) \]

![Diagram of a graph with nodes S, Y, and \hat{S} connected by arrows labeled encoding and reconstruction.]

\[ Y \in \{0, 1\}^{(m+1) \times (m+1)} \]

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

- Our encoder with parameters \( \Theta \) computes \( P_{\Theta}(Y|s) \), the probability of generating a dependency parse graph \( Y \) given a sentence \( s \).
- Our decoder with parameters \( \Lambda \) computes \( P_{\Lambda}(\hat{s}|Y) \), the probability of reconstructing sentence \( \hat{s} \) conditioned on the parse graph \( Y \).
Model: encoder

- A discriminative encoder, independently labeling each arc in a directed complete graph.

Figure: Illustration of the encoder, following the design of dozat.
SDP allows a word to have multiple heads. If generate a word conditioned on multiple heads, need enumerate, intractable inference and learning.

Arc-factored, generate a word for multiple times, each time conditioned on a different head.

\[ [y_0; y_1; y_2; \ldots; y_m] \rightarrow (\hat{s}^0, \hat{s}^1, \hat{s}^2, \ldots, \hat{s}^m) \]
Model: decoder

- Generative probability with a dependency

**Figure:** Illustration of the decoder generating $\hat{s}^k$ from the $k$-th neural generator, guided by sub-graph $y_k$. 
Model: decoder

- Generative probability without a dependency

**Figure**: Illustration of the decoder generating $\hat{s}^k$ from the $k$-th neural generator, guided by sub-graph $y_k$. 
Model: decoder

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

$$P_\Lambda(\hat{s}^0, \hat{s}^1, \ldots, \hat{s}^m|Y) = \prod_{k=0}^{m} P_\Lambda(\hat{s}^k|y_k) = \prod_{k=0}^{m} \prod_{i=1}^{m} P_\Lambda(\hat{s}_i^k|y_k)$$

- To balance the encoder and decoder, take the geometric mean of the $m+1$ probabilities

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

- Note that this is not a properly normalized probability distribution.
We can parse a sentence $s$ by finding a $Y \in \mathcal{Y}(s)$ which maximizes probability $P(\hat{s} = s, Y|s)$.

$$Y^* = \arg \max_{Y \in \mathcal{Y}(s)} \log P_{\Theta, \Lambda}(\hat{s}, Y|s)$$

$$= \arg \max_{Y \in \mathcal{Y}(s)} \log P_{\Lambda}(\hat{s}|Y) P_{\Theta}(Y|s)$$

$$= \arg \max_{Y \in \mathcal{Y}(s)} \sum_{i,j} \left( \frac{1}{m+1} \log P_{\Lambda}(\hat{s}_j|y_{i,j}) + \log P_{\Theta}(y_{i,j}|s) \right)$$

The probability is arc-factored. Independently determine the existence of each arc, picking the value of $y_{i,j}$ that maximizes the corresponding term. $O(m^2)$ time complexity.
Learning

- Supervised loss: for any labeled sentence \((s, Y^*)\), where \(s\) stands for a sentence and \(Y^*\) stands for a gold parse graph.

\[ \mathcal{L}_l(s) = -\log P_{\Theta,\Lambda}(\hat{s} = s, Y^*|s) \]

- Unsupervised loss: for any unlabeled sentence \(s\), maximize the conditional reconstruction probability \(P(\hat{s} = s|s)\).

\[ \mathcal{L}_u(s) = -\log P_{\Theta,\Lambda}(\hat{s}|s) \]

\[ = -\sum_{i,j} \log \sum_{y_{i,j} \in \{0,1\}} \left( P_{\Lambda}(y_{i,j}|s) \times \sqrt[\text{m+1}]{P_{\Theta}(\hat{s}_j|y_{i,j})} \right) \]

- The overall loss function is defined as a combination of supervised loss \(\mathcal{L}_l\) and unsupervised loss \(\mathcal{L}_u\).

\[ \mathcal{L}(s) = \iota(s) \times \mathcal{L}_l(s) + (1 - \iota(s)) \times \rho \mathcal{L}_u(s) \]
Experiments: Varying Size of Unlabeled Data

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

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

Figure: Results with fixed amount of labeled data and varying amount of unlabeled data.
### Experiments: Varying Proportion of Unlabeled Data

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

**Figure:** Experimental results with varying proportions of labeled and unlabeled data.
Conclusion

- We proposed a semi-supervised learning model for semantic dependency parsing using 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.
- Our model works in an arc-factored fashion, promising end-to-end learning and efficient parsing.
- Applying our model to low-resource languages and cross-domain settings may be the interesting future directions.
The End. Thanks!