Dependency Parsing as Head Selection

Xingxing Zhang, Jianpeng Cheng, Mirella Lapata

Institute for Language, Cognition and Computation
University of Edinburgh

x.zhang@ed.ac.uk

April 6, 2017
Dependency Parsing is the task of transforming a sentence $S = (\text{ROOT}, w_1, w_2, \ldots, w_N)$ into a directed tree originating out of $\text{ROOT}$.

- Parsing Algorithms
  - Transition-based Parsing
  - Graph-based Parsing

Dependency Parsing

Dependency Parsing is the task of transforming a sentence $S = (\text{ROOT}, w_1, w_2, \ldots, w_N)$ into a directed tree originating out of \text{ROOT}.

- Parsing Algorithms
  - Transition-based Parsing
  - Graph-based Parsing

- Our parser is neither Transition-based nor Graph-based (during training)
Transition-based Parsing

- Data Structure
  - Buffer, Stack, Arc Set

- Parsing:
  - Choose an action from
    - SHIFT
    - REDUCE-Left
    - REDUCE-Right
Graph-based Parsing

- A Sentence → A Directed Complete Graph

(Graphs from Kubler et al., 2009)

- Parsing: Finding Maximum Spanning Tree
  - Chu-Liu-Edmond algorithm (Chu and Liu, 1965)
  - Eisner algorithm (Eisner 1996)
Mostly replacing discrete features with Neural Network features.

- **Transition-based Parsers**
  - Feed-Forward NN features (Chen and Manning, 2014)
  - Bi-LSTM features (Kiperwasser and Goldberg, 2016)
  - Stack LSTM: Buffer, Stack and Action Sequences modeled by Stack-LSTMs (Dyer et al., 2015)

- **Graph-based Parsers**
  - Tensor Decomposition features (Lei et al., 2014)
  - Feed-Forward NN features (Pei et al., 2015)
  - Bi-LSTM features (Kiperwasser and Goldberg, 2016)
Do we need a transition system or graph algorithm?

An important fact: Every word has only one head!

Why not just learn to select the head?

Zhang et al. (Univ. of Edinburgh)
Do we need a transition system or graph algorithm?

- An important fact: Every word has only one head!
Do we need a transition system or graph algorithm?

- An important fact: Every word has only one head!
- Why not just learn to select the head?
Dependency Parsing as Head Selection

DeNSE: Dependency Neural Selection

\[
\text{DeNSE: } \begin{array}{c}
\text{ROOT} \\
kids \\
love \\
candy
\end{array}
\]
Dependency Parsing as Head Selection

DeNSE: Dependency Neural Selection

\[
\text{DeNSE: Dependency Neural Selection}
\]

\[
\text{Dependency Parsing as Head Selection}
\]

\[
\begin{align*}
\text{DeNSE: Dependency Neural Selection} & \quad \text{April 6, 2017 7 / 18}
\end{align*}
\]
**Dependency Parsing as Head Selection**

**DeNSE:** Dependency Neural Selection

\[
P_{head}(\text{ROOT}|\text{love}, S) = \exp(\text{MLP}(a_{\text{ROOT}}, a_{\text{love}})) \sum_{k=0}^{3} \exp(\text{MLP}(a_{k}, a_{\text{love}}))
\]

Zhang et al. (Univ. of Edinburgh)
Dependency Parsing as Head Selection

**DeNSE: Dependency Neural Selection**

\[
P_{\text{head}}(\text{ROOT}|\text{love}, S) = \frac{\exp(\text{MLP}(a_{\text{ROOT}}, a_{\text{love}}))}{\sum_{k=0}^{3} \exp(\text{MLP}(a_k, a_{\text{love}}))}
\]

Zhang et al. (Univ. of Edinburgh)
Dependency Parsing as Head Selection

**DeNSe: Dependency Neural Selection**

\[
P_{\text{head}}(\text{ROOT}|\text{love}, S) = \frac{\exp(MLP(a_{\text{ROOT}}, a_{\text{love}}))}{\sum_{k=0}^{3} \exp(MLP(a_{k}, a_{\text{love}}))}
\]
Decoding

- Greedy Decoding: The output may not be a (projective) tree!
Decoding

Greedy Decoding: The output may not be a (projective) tree!

<table>
<thead>
<tr>
<th>Dataset</th>
<th>#Sent (Dev)</th>
<th>Greedy Decoding Tree</th>
<th>Greedy Decoding Proj</th>
</tr>
</thead>
<tbody>
<tr>
<td>PTB (English)</td>
<td>1,700</td>
<td>95.1</td>
<td>86.6</td>
</tr>
<tr>
<td>CTB (Chinese)</td>
<td>803</td>
<td>87.0</td>
<td>73.1</td>
</tr>
<tr>
<td>Czech</td>
<td>374</td>
<td>87.7</td>
<td>65.5</td>
</tr>
<tr>
<td>German</td>
<td>367</td>
<td>96.7</td>
<td>67.3</td>
</tr>
</tbody>
</table>
Decoding

- **Greedy Decoding**: The output may not be a (projective) tree!

<table>
<thead>
<tr>
<th>Dataset</th>
<th>#Sent (Dev)</th>
<th>Greedy Decoding Tree</th>
<th>Greedy Decoding Proj</th>
</tr>
</thead>
<tbody>
<tr>
<td>PTB (English)</td>
<td>1,700</td>
<td>95.1</td>
<td>86.6</td>
</tr>
<tr>
<td>CTB (Chinese)</td>
<td>803</td>
<td>87.0</td>
<td>73.1</td>
</tr>
<tr>
<td>Czech</td>
<td>374</td>
<td>87.7</td>
<td>65.5</td>
</tr>
<tr>
<td>German</td>
<td>367</td>
<td>96.7</td>
<td>67.3</td>
</tr>
</tbody>
</table>

- **Decoding with a Maximum Spanning Tree Algorithm (relatively rare)**
  - Projective Parsing: Eisner Algorithm
  - Non-projective Parsing: Chu-Liu-Edmond Algorithm
Labelled Parser

A two-layer Rectifier Network (Glorot et al., 2011)

- **Dependent Word:**
  - Bi-LSTM Feature
  - Word Embedding
  - PoS Embedding

- **Head Word:**
  - Bi-LSTM Feature
  - Word Embedding
  - PoS Embedding
Experiments
Projective Parsing Results (PTB; English)

NN (Chen & Manning, 2014); S-LSTM (Dyer et al., 2015); Bi-LSTM (Kiperwasser & Goldberg, 2016); SynNet (Andor et al. 2016)
NN (Chen & Manning, 2014); S-LSTM (Dyer et al., 2015); Bi-LSTM (Kiperwasser & Goldberg, 2016); SynNet (Andor et al., 2016)
Projective Parsing Results (PTB; English)

NN (Chen & Manning, 2014); S-LSTM (Dyer et al., 2015); Bi-LSTM (Kiperwasser & Goldberg, 2016); SynNet (Andor et al. 2016)
Projective Parsing Results (PTB; Chinese)

NN (Chen & Manning, 2014); S-LSTM (Dyer et al., 2015); Bi-LSTM (Kiperwasser & Goldberg, 2016); 3rd-cubic (Zhang & McDonald 2014)
Non-projective Parsing Results (German)

MST-1st, MST-2nd (McDonald et al., 2005) Turbo-1st, Turbo-3rd (Martins et al., 2013) RBG-1st RBG-3rd (Martins et al. 2013)
Non-projective Parsing Results (German)

MST-1st, MST-2nd (McDonald et al., 2005) Turbo-1st, Turbo-3rd (Martins et al., 2013) RBG-1st RBG-3rd (Martins et al. 2013)
Non-projective Parsing Results (Czech)

MST-1st, MST-2nd ((McDonald et al., 2005) Turbo-1st, Turbo-3rd (Martins et al., 2013) RBG-1st RBG-3rd (Martins et al. 2013)
Non-projective Parsing Results (Czech)

MST-1st, MST-2nd ((McDonald et al., 2005) Turbo-1st, Turbo-3rd (Martins et al., 2013) RBG-1st RBG-3rd (Martins et al. 2013)
### Unlabeled Exact Match

Table: UEM results on PTB and CTB.

<table>
<thead>
<tr>
<th>Parser</th>
<th>PTB</th>
<th>CTB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Dev</td>
<td>Test</td>
</tr>
<tr>
<td>C&amp;M14</td>
<td>43.35</td>
<td>40.93</td>
</tr>
<tr>
<td>Dyer15</td>
<td>51.94</td>
<td>50.70</td>
</tr>
<tr>
<td>DeNSE</td>
<td>51.24</td>
<td>49.34</td>
</tr>
<tr>
<td>DeNSE+E</td>
<td><strong>52.47</strong></td>
<td><strong>50.79</strong></td>
</tr>
</tbody>
</table>

*Note: The table compares the performance of different parsers on PTB and CTB datasets. The best scores are highlighted in bold.*
UAS v.s. Length

![Graph showing the relationship between UAS and sentence length for different models.]

- C&M14
- DeNSe+E
- Dyer15

Zhang et al. (Univ. of Edinburgh)
Conclusions

- We propose a dependency parser as greedily selecting the head of each word in sentence.
- Combine the greedy model with a MST algorithm can further increase the performance.
- Code available: https://github.com/XingxingZhang/dense_parser
Thanks

Q & A