COS 584

Advanced Natural Language Processing

P8: Dependency Parsing

Spring 2021
Main take-aways:

• Use bidirectional LSTMs to “build” features for dependency parsing
• It is applied to both transition-based dependency parsing and graph-based dependency parsing
• End-to-end training of a structured prediction model with neural feature extractors
Dependency parsing

Dependency parsing is the task of recognizing a sentence and assigning a dependency structure to it.

I prefer the morning flight through Denver

I prefer the morning flight through Denver
Two families of algorithms

Transition-based dependency parsing
• Also called “shift-reduce parsing”

Graph-based dependency parsing
Transition-based Dependency Parsing

- A configuration consists of a stack $s$, a buffer $b$ and a set of dependency arcs $A$: $c = (s, b, A)$

- Initially, $s = \text{[ROOT]}$, $b = \{w_1, w_2, \ldots, w_n\}$, $A = \emptyset$

- Three types of transitions:
  - LEFT-ARC ($r$), RIGHT-ARC ($r$), SHIFT

- A configuration is terminal if $s = \text{[ROOT]}$ and $b = \emptyset$

- **Inference**: let the classifier predict the next transition repeatedly until we reach a terminal configuration

Greedy transition-based dependency parsing
Graph-based Dependency Parsing

- **Basic idea**: let’s predict the dependency tree directly
  
  \[ Y^* = \text{argmax}_{y \in \Phi(X)} \text{score}(X, Y) \]

  - X: sentence, Y: any possible dependency tree

- **Factorization**:
  
  \[ \text{score}(X, Y) = \sum_{e \in Y} \text{score}(e) = \sum_{e \in Y} w^T f(e) \]

  e: \( h \rightarrow m \), first-order

\[ \max(0, 1 + \max_{y' \neq y} \text{score}(x, y') - \text{score}(x, y)) \]
Graph-based Dependency Parsing

- **Inference**: finding maximum spanning tree (MST) for weighted, directed graph

The Chu-Liu Edmonds algorithm $O(n^3)$
Feature Functions

• Transition-based dependency parsing
  • Extract features from the configuration $c = (s, b, A)$

• Graph-based dependency parsing
  • Extract features for each edge $(h, m)$

Example: The word and POS of the head and modifier items, as well as POS tags of the items around the head and modifier, POS tags of items between the head and modifier, and the distance and direction between the head and modifier.
A History of Dependency Parsing

• Transition-based dependency parsing started from ~2004

Incrementality in Deterministic Dependency Parsing

Joakim Nivre
School of Mathematics and Systems Engineering
Växjö University
SE-35195 Växjö
Sweden
joakim.nivre@msi.vxu.se

• Graph-based dependency parsing started from ~2005

Online Large-Margin Training of Dependency Parsers

Ryan McDonald    Koby Crammer    Fernando Pereira
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA
{ryantm,crammer,pereira}@cis.upenn.edu

All these methods are based on millions of sparse indicator features
A History of Dependency Parsing

- 2004-2013: a lot of improvements focus on
  - Constructing better features for both families of algorithms
  - Solutions to handle non-projective parse trees
  - Transition-based:
    - Arc-standard, Arc-eager, **Arc-hybrid**, Easy-first, ...
    - Better search strategies: beam search, **dynamic oracle**
  - Graph-based:
    - From first-order to second-order/third-order
    - Better inference algorithms

- (Chen and Manning, 2014) introduced neural networks in dependency parsing
  - The features are built based on word/part-of-speech tag/label embeddings of 18 different elements and let FFNNs learn the composition of these elements
This paper: BiLSTM vectors as minimal features

\[ x_i = e(w_i) \circ e(p_i) \]

\[ v_i = BI\text{LSTM}(x_{1:n}, i) \]

- Transition-based DP

\[ \phi(e) = v_{s_2} \circ v_{s_1} \circ v_{s_0} \circ v_{b_0} \]
This paper: BiLSTM vectors as minimal features

\[ x_i = e(w_i) \circ e(p_i) \quad v_i = \text{BiLSTM}(x_{1:n}, i) \]

- Graph-based DP

\[ \text{score}(h, m) = \text{MLP}(v_h \circ v_m) \]
## Experimental results

<table>
<thead>
<tr>
<th>System</th>
<th>Method</th>
<th>Representation</th>
<th>Emb</th>
<th>PTB-YM UAS</th>
<th>PTB-SD UAS/LAS</th>
<th>CTB UAS/LAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>This work</td>
<td>graph, 1st order</td>
<td>2 BiLSTM vectors</td>
<td>YES</td>
<td>93.1</td>
<td>91.0</td>
<td>86.6</td>
</tr>
<tr>
<td>This work</td>
<td>transition (greedy, dyn-oracle)</td>
<td>4 BiLSTM vectors</td>
<td>YES</td>
<td>93.1</td>
<td>91.0</td>
<td>86.2</td>
</tr>
<tr>
<td>This work</td>
<td>transition (greedy, dyn-oracle)</td>
<td>11 BiLSTM vectors</td>
<td>YES</td>
<td>93.2</td>
<td>91.2</td>
<td>86.5</td>
</tr>
<tr>
<td>ZhangNivre11</td>
<td>transition (beam)</td>
<td>large feature set (sparse)</td>
<td>YES</td>
<td>92.9</td>
<td></td>
<td>86.0</td>
</tr>
<tr>
<td>Martins13 (TurboParser)</td>
<td>graph, 3rd order+</td>
<td>large feature set (sparse)</td>
<td>YES</td>
<td>92.8</td>
<td>93.1</td>
<td>84.4</td>
</tr>
<tr>
<td>Pei15</td>
<td>graph, 2nd order</td>
<td>large feature set (dense)</td>
<td>YES</td>
<td>93.0</td>
<td></td>
<td>85.7</td>
</tr>
<tr>
<td>Dyer15</td>
<td>transition (greedy)</td>
<td>Stack-LSTM + composition</td>
<td>YES</td>
<td>92.4</td>
<td>90.0</td>
<td>84.1</td>
</tr>
<tr>
<td>Ballesteros16</td>
<td>transition (greedy, dyn-oracle)</td>
<td>Stack-LSTM + composition</td>
<td>YES</td>
<td>92.7</td>
<td>90.6</td>
<td>86.1</td>
</tr>
<tr>
<td>This work</td>
<td>graph, 1st order</td>
<td>2 BiLSTM vectors</td>
<td>YES</td>
<td>93.0</td>
<td>90.9</td>
<td>86.5</td>
</tr>
<tr>
<td>This work</td>
<td>transition (greedy, dyn-oracle)</td>
<td>4 BiLSTM vectors</td>
<td>YES</td>
<td>93.6</td>
<td>91.5</td>
<td>87.4</td>
</tr>
<tr>
<td>This work</td>
<td>transition (greedy, dyn-oracle)</td>
<td>11 BiLSTM vectors</td>
<td>YES</td>
<td>93.9</td>
<td>91.9</td>
<td>87.6</td>
</tr>
<tr>
<td>Weiss15</td>
<td>transition (greedy)</td>
<td>large feature set (dense)</td>
<td>YES</td>
<td>93.2</td>
<td>91.2</td>
<td></td>
</tr>
<tr>
<td>Weiss15</td>
<td>transition (beam)</td>
<td>large feature set (dense)</td>
<td>YES</td>
<td>94.0</td>
<td>92.0</td>
<td></td>
</tr>
<tr>
<td>Pei15</td>
<td>graph, 2nd order</td>
<td>large feature set (dense)</td>
<td>YES</td>
<td>93.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dyer15</td>
<td>transition (greedy)</td>
<td>Stack-LSTM + composition</td>
<td>YES</td>
<td>93.1</td>
<td>90.9</td>
<td>87.1</td>
</tr>
<tr>
<td>Ballesteros16</td>
<td>transition (greedy, dyn-oracle)</td>
<td>Stack-LSTM + composition</td>
<td>YES</td>
<td>93.6</td>
<td>91.4</td>
<td>87.6</td>
</tr>
<tr>
<td>LeZuidema14</td>
<td>reranking /blend</td>
<td>inside-outside recursive net</td>
<td>YES</td>
<td>93.1</td>
<td>93.8</td>
<td></td>
</tr>
<tr>
<td>Zhu15</td>
<td>reranking /blend</td>
<td>recursive conv-net</td>
<td>YES</td>
<td>93.8</td>
<td></td>
<td>85.7</td>
</tr>
</tbody>
</table>
Breakout discussion

• Group 1 (Danqi)
  • How can we further improve the models presented in this paper?
• Group 2 (Zexuan)
  • Compare the pros and cons of transition-based and graph-based dependency parsing
• Group 3 (Shunyu)
  • Why do you think bidirectional LSTMs build good features for dependency parsing? What features are important for making parsing decisions?
• Group 4 (Kaiyu)
  • Is there anything else interesting in this paper (that we haven’t covered yet)?

Use the remaining time for free-form discussion!!!