An Improved Oracle for Dependency Parsing with Online Reordering

Joakim Nivre  Marco Kuhlmann  Johan Hall

Uppsala University
Sweden
Introduction

Transition-based dependency parsing:
- Deterministic parsing in linear time for projective trees
- Search through transition system guided by classifiers
- Classifiers trained using an oracle answering the question:
  - What sequence of transitions derives a given tree $T$?

Online reordering [Nivre 2009]:
- Extends transition-based parsing to non-projective trees
- Maintains linear time parsing on average

This paper:
- Improve parsing accuracy and efficiency by improving oracle
A dependency tree is a labeled directed tree $T$, consisting of:
- a set $V$ of nodes, labeled with words (including ROOT)
- a set $A$ of arcs, labeled with dependency types
- a linear precedence order $<$ on $V$

A dependency tree $T$ is projective iff every subtree has a contiguous yield.
A parser configuration is a triple $c = (S, B, A)$, where

- $S$ = a stack $[\ldots, w_i]_S$ of partially processed nodes
- $B$ = a buffer $[w_j, \ldots]_B$ of remaining nodes
- $A$ = a set of labeled arcs $(w_i, w_j, l)$

**Initialization:**
$([w_0]_S, [w_1, \ldots, w_n]_B, \{ \})$

**Termination:**
$([w_0]_S, [], B, A)$  

**NB:** $w_0 = \text{ROOT}$
Transition System: Transitions

- **Left-Arc(\(l\))**
  
  \[
  \frac{([\ldots, w_i, w_j]_S, B, A)}{([\ldots, w_j]_S, B, A \cup \{(w_j, w_i, l)\})}
  \quad [i \neq 0]
  \]

- **Right-Arc(\(l\))**
  
  \[
  \frac{([\ldots, w_i, w_j]_S, B, A)}{([\ldots, w_i]_S, B, A \cup \{(w_i, w_j, l)\})}
  \]

- **Shift**
  
  \[
  \frac{([\ldots]_S, [w_i, \ldots]_B, A)}{([\ldots, w_i]_S, [\ldots]_B, A)}
  \]

- **Swap**
  
  \[
  \frac{([\ldots, w_i, w_j]_S, [\ldots]_B, A)}{([\ldots, w_j]_S, [w_i, \ldots]_B, A)}
  \quad [i \neq 0, i < j]
  \]
Given a dependency tree $T = (V, A, <)$, let the projective order $<_p$ be the order defined by an inorder traversal of $T$ with respect to $<$.
Old Oracle

In a configuration $c = ([\ldots, w_i, w_j], B, A)$, do

- Left-Arc / if $w_i \leftarrow w_j$ and $w_i$ complete
- Right-Arc / if $w_i \rightarrow w_j$ and $w_j$ complete
- Swap if $w_i >_p w_j$
- Shift otherwise
Improving the Oracle

Problem:
- Old oracle swaps eagerly, whenever possible
- Many swaps are not necessary
- Is there a better lazy oracle?

Key concept:
- Maximal projective components of a dependency tree
- Connected components of oracle parse without swap
In a configuration $c = ([\ldots, w_i, w_j], B, A)$, do

- **Left-Arc** if $w_i \leftarrow w_j$ and $w_i$ complete
- **Right-Arc** if $w_i \rightarrow w_j$ and $w_j$ complete
- **Swap** if $w_i >_p w_j$, $B = [w_k, \ldots]$ and $\text{mpc}(w_j) \neq \text{mpc}(w_k)$
- **Shift** otherwise
## Experiments

- **Number of swaps (training set):**

<table>
<thead>
<tr>
<th>Oracle</th>
<th>Arabic</th>
<th>Czech</th>
<th>Danish</th>
<th>Slovene</th>
<th>Turkish</th>
</tr>
</thead>
<tbody>
<tr>
<td>Old</td>
<td>1416</td>
<td>57011</td>
<td>8296</td>
<td>2191</td>
<td>2828</td>
</tr>
<tr>
<td>New</td>
<td>229</td>
<td>26208</td>
<td>1497</td>
<td>690</td>
<td>1253</td>
</tr>
</tbody>
</table>

- **Parsing accuracy (labeled attachment score):**

<table>
<thead>
<tr>
<th>Oracle</th>
<th>Arabic</th>
<th>Czech</th>
<th>Danish</th>
<th>Slovene</th>
<th>Turkish</th>
</tr>
</thead>
<tbody>
<tr>
<td>Old</td>
<td>67.2</td>
<td>82.5</td>
<td>84.2</td>
<td>75.2</td>
<td>64.7</td>
</tr>
<tr>
<td>New</td>
<td>67.5</td>
<td>82.7</td>
<td>84.3</td>
<td>75.7</td>
<td>65.0</td>
</tr>
</tbody>
</table>
Conclusion

- Improved oracle for dependency parsing with online reordering:
  - Reduces number of swaps
  - Improves parsing accuracy
- Want to know more?
  - Come and see our poster!
  - Come to my talk on Friday!