Transition-Based Parsing with Multiword Expressions

Joakim Nivre, Uppsala University, WG3

### Statistical Dependency Parsing
- Map sentences to dependency trees
- Learn mapping from (labeled) corpora
- Approaches
  - Graph-based – score trees, factored into subgraphs
  - Transition-based – score derivations, factored into transitions
- The spanning tree assumption
  - Input is a sequence of tokens $w_1 \cdots w_n$
  - Output is a spanning tree over input tokens
  - Every input token is a tree node (and vice versa)
- Problematic for MWEs (and many other phenomena)

### Transition-Based Parsing
- Transition system
  - Abstract state machine for deriving dependency trees
  - Configurations = parser states
  - Transitions = parser actions
- Scoring model
  - Statistical model for scoring transitions out of a configuration
  - Usually a linear model learned from treebank derivations
- Search algorithm
  - Algorithm for finding the optimal sequence of transitions
  - Usually approximate search (greedy search, beam search)

### Arc-Standard Transition System
- Configuration: $(S, B, A)$ [$S =$ Stack, $B =$ Buffer, $A =$ Arcs]
- Initial: $(\varepsilon, w_1 \cdots w_n, \{\})$
- Terminal: $(w, \varepsilon, A)$
- Shift: $(S, w[B, A]) \Rightarrow (S[w, B, A])$
- Right-Arc: $(S[w][w', B, A]) \Rightarrow (S[w, B, A[w \rightarrow w']])$
- Left-Arc: $(S[w][w', B, A]) \Rightarrow (S[w', B, A[w' \rightarrow w]])$

### Example Derivation
<table>
<thead>
<tr>
<th>Transition</th>
<th>Stack</th>
<th>Buffer</th>
<th>Arcs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shift</td>
<td>she</td>
<td>found the word</td>
<td>she looked up the word</td>
</tr>
<tr>
<td>Shift</td>
<td>she found</td>
<td>the word</td>
<td>looked up the word</td>
</tr>
<tr>
<td>Left-Arc</td>
<td>found</td>
<td>the word</td>
<td>[looked up] → [she]</td>
</tr>
<tr>
<td>Shift</td>
<td>found the word</td>
<td></td>
<td>word</td>
</tr>
<tr>
<td>Left-Arc</td>
<td>found</td>
<td>word</td>
<td>[looked up] → [word]</td>
</tr>
<tr>
<td>Right-Arc</td>
<td>found</td>
<td>word</td>
<td>[looked up] → [word]</td>
</tr>
</tbody>
</table>

### Adding Multiword Expressions
- Multiword expressions can be encoded as pseudo-dependencies
  - Structure is (often) arbitrary
  - Dependency tree features are uninformative
  - Lexical features are potentially misleading
- New approach
  - Integrate MWE recognition into parsing
  - Make a distinction between input tokens and tree nodes
  - Add transitions to merge tokens into MWEs

### VPC Example
- She looked up the word

### A New Transition System
- Tree nodes and input tokens are now different
  - Tree nodes are lists of input tokens
  - The buffer $B$ holds input tokens
  - The stack $S$ holds tree nodes
- There are two transitions for consuming tokens from the buffer
  - Shift adds the next token to a new singleton list on the stack
  - Chunk appends the next token to the list on top of the stack
- Multiword expressions can be treated as first-class citizens
  - Can enter directly into dependency relations
  - Can have holistic features distinct from their components

### New Transition System
- Configuration: $(S, B, A)$ [$S =$ Stack, $B =$ Buffer, $A =$ Arcs]
- Initial: $(\varepsilon, w_1 \cdots w_n, \{\})$
- Terminal: $(v, \varepsilon, A)$
- Shift: $(S, w[B, A]) \Rightarrow (S[w, B, A])$
- Chunk: $(S[u, w][B, A]) \Rightarrow (S[u][w], B, A)$
- Right-Arc: $(S[u][v, B, A]) \Rightarrow (S[u, B, A[u \rightarrow v]])$
- Left-Arc: $(S[u][v, B, A]) \Rightarrow (S[v, B, A[u \rightarrow v]])$

### Example Derivation
<table>
<thead>
<tr>
<th>Transition</th>
<th>Stack</th>
<th>Buffer</th>
<th>Arcs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shift</td>
<td>[she]</td>
<td>she looked up the word</td>
<td>looked up the word</td>
</tr>
<tr>
<td>Shift</td>
<td>[she] [looked up]</td>
<td>up the word</td>
<td>word</td>
</tr>
<tr>
<td>Chunk</td>
<td>[she] [looked up]</td>
<td>the word</td>
<td>[looked up] → [she]</td>
</tr>
<tr>
<td>Shift</td>
<td>[looked up] [the]</td>
<td>word</td>
<td>[looked up] → [word]</td>
</tr>
<tr>
<td>Shift</td>
<td>[looked up] [the]</td>
<td>word</td>
<td>[looked up] → [word]</td>
</tr>
<tr>
<td>Right-Arc</td>
<td>[looked up]</td>
<td>[word]</td>
<td>[looked up] → [word]</td>
</tr>
</tbody>
</table>