UPPSALA UNIVERSITET

Transition-Based Parsing with Multiword Expressions

Joakim Nivre, Uppsala University, WG3

Statistical Dependency Parsing

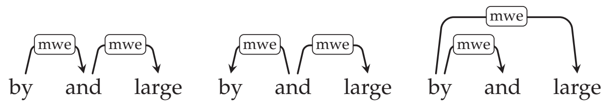
- Map sentences to dependency trees
- Learn mapping from (labeled) corpora
- Approaches
  - Graph-based – score trees, factored into subgraphs
  - Transition-based – score derivations, factored into transitions
- The spanning tree assumption
  - Input is a sequence of tokens $w_1 \cdots w_n$
  - Output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

Examples

French du = de le	1:m
French à cause de = à-cause-de	m:1
French à cause du = à-cause-de le	m:n

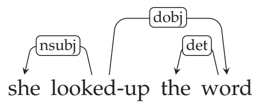
Adding Multiword Expressions

- 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



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: $(-, w_1 \cdots w_n, \{\,\})$

Terminal: $(w, -, A)$

Shift:	$(S, w\vert B, A)$	$\Rightarrow$	$(S\vert w, B, A)$
Right-Arc:	$(S\vert w\vert w', B, A)$	$\Rightarrow$	$(S\vert w, B, A[w \rightarrow w'])$
Left-Arc:	$(S\vert w\vert w', B, A)$	$\Rightarrow$	$(S\vert w', B, A[w' \rightarrow w])$

Example Derivation

Transition	Stack	Buffer	Arcs
	–	she found the word	
Shift	she	found the word	
Shift	she found	the word	
Left-Arc	found	the word	found → she
Shift	found the	word	
Shift	found the word	–	
Left-Arc	found word	–	word → the
Right-Arc	found	–	found → 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: $(-, w_1 \cdots w_n, \{\,\})$

Terminal: $(v, -, A)$

Shift:	$(S, w\vert B, A)$	$\Rightarrow$	$(S\vert[w], B, A)$
Chunk:	$(S\vert u, w\vert B, A)$	$\Rightarrow$	$(S\vert[u\vert w], B, A)$
Right-Arc:	$(S\vert u\vert v, B, A)$	$\Rightarrow$	$(S\vert u, B, A[u \rightarrow v])$
Left-Arc:	$(S\vert u\vert v, B, A)$	$\Rightarrow$	$(S\vert v, B, A[v \rightarrow u])$

Example Derivation

Transition	Stack	Buffer	Arcs
	–	she looked up the word	
Shift	[she]	looked up the word	
Shift	[she] [looked]	up the word	
Chunk	[she] [looked up]	the word	
Left-Arc	[looked up]	the word	[looked up] → [she]
Shift	[looked up] [the]	word	
Shift	[looked up] [the] [word]	–	
Left-Arc	[looked up] [word]	–	[word] → [the]
Right-Arc	[looked up]	–	[looked up] → [word]