Transition-based Dependency Parsing

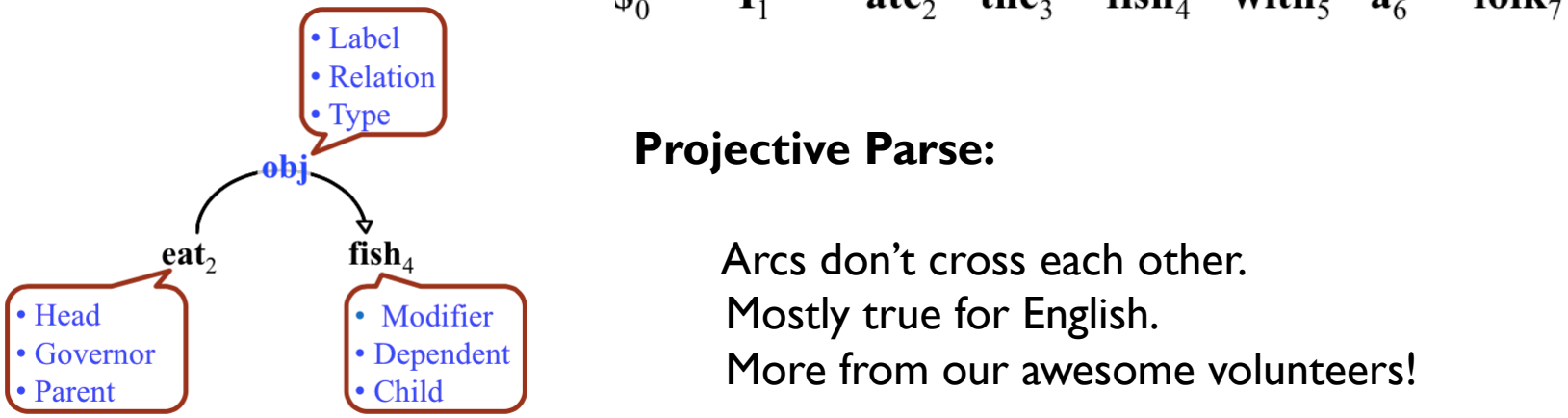
Niranjan Balasubramanian

September 01, 2015

Slides Adapted from Nivre and Manning.

Dependency Parsing

- **Dependency tree --** A dependency tree is a tree structure composed of the input words and meets a few constraints:
  - Single-head
  - Connected
  - Acyclic



Projective Parse:

Arcs don't cross each other.
Mostly true for English.
More from our awesome volunteers!

Transition-based Parsing
Arc-Eager [Nivre 2003]

Configuration:	(S, B, A)		$[S = \text{Stack}, B = \text{Buffer}, A = \text{Arcs}]$	
Initial:	$([\,], [0, 1, \ldots, n], \{\,\})$			
Terminal:	$(S, [\,], A)$			
Shift:	$(S, i\|B, A)$	$\Rightarrow$	$(S\|i, B, A)$	
Reduce:	$(S\|i, B, A)$	$\Rightarrow$	(S, B, A)	$h(i, A)$
Right-Arc(k):	$(S\|i, j\|B, A)$	$\Rightarrow$	$(S\|i\|j, B, A \cup \{(i, j, k)\})$	
Left-Arc(k):	$(S\|i, j\|B, A)$	$\Rightarrow$	$(S, j\|B, A \cup \{(j, i, k)\})$	$\neg h(i, A) \wedge i \neq 0$

Notation:
- $S|i$ = stack with top i and remainder S
- $j|B$ = buffer with head j and remainder B
- $h(i, A)$ = i has a head in A

Rules of the game!
-- Keep move items from buffer to stack.
-- If the top item on stack is a dependent of the top buffer item output dependency relation and drop the item from stack.
-- If the top buffer item is a dependent of any item in stack, move buffer item to stack, but keep the head in stack.

Example Transition Sequence

$[\text{ROOT}]_S$ [Economic, news, had, little, effect, on, financial, markets, .]$_B$

ROOT	Economic	news	had	little	effect	on	financial	markets	.
	adj	noun	verb	adj	noun	prep	adj	noun	.

Assume we have some black-box that takes two words and magically gives you the dependency relation between them if one exists.

Example Transition Sequence

$[\text{ROOT}]_S$ [Economic, news, had, little, effect, on, financial, markets, $.]_B$

ROOT	Economic	news	had	little	effect	on	financial	markets	.
	adj	noun	verb	adj	noun	prep	adj	noun	.

Example Transition Sequence

Shift:
Move Economic to stack.

$[\text{ROOT}, \text{Economic}]_S \quad [\text{news}, \text{had}, \text{little}, \text{effect}, \text{on}, \text{financial}, \text{markets}, .]_B$

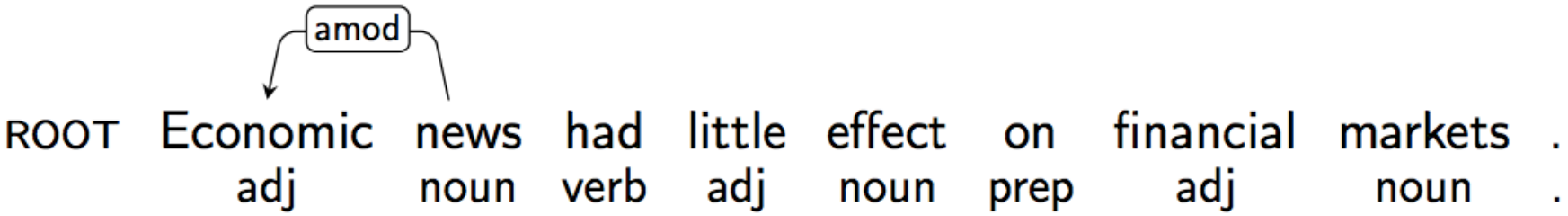
ROOT	Economic	news	had	little	effect	on	financial	markets	.
	adj	noun	verb	adj	noun	prep	adj	noun	.

Example Transition Sequence

Left Arc:

Add left-arc *amod*(news, Economic) to A.
Remove Economic from stack since it now has head in A.
NOTE: Left-arc was possible only as Economic did not previously have a head in A.

$[\text{ROOT}]_S$ $[\text{news, had, little, effect, on, financial, markets, .}]_B$

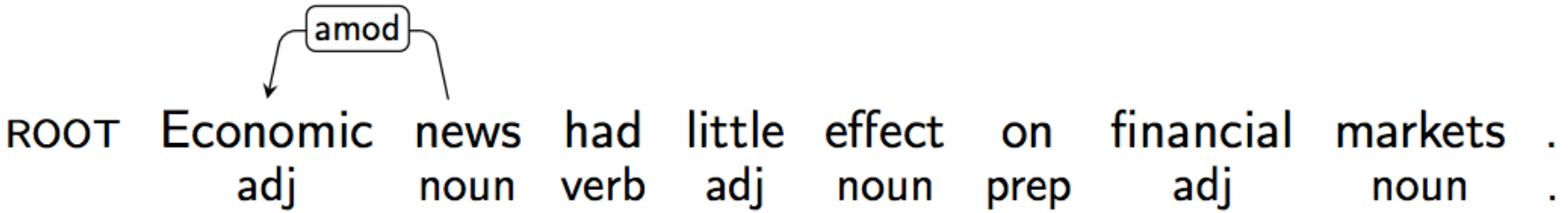


Example Transition Sequence

Shift

Move news to stack.

$[\text{ROOT}, \text{news}]_S \quad [\text{had}, \text{little}, \text{effect}, \text{on}, \text{financial}, \text{markets}, .]_B$

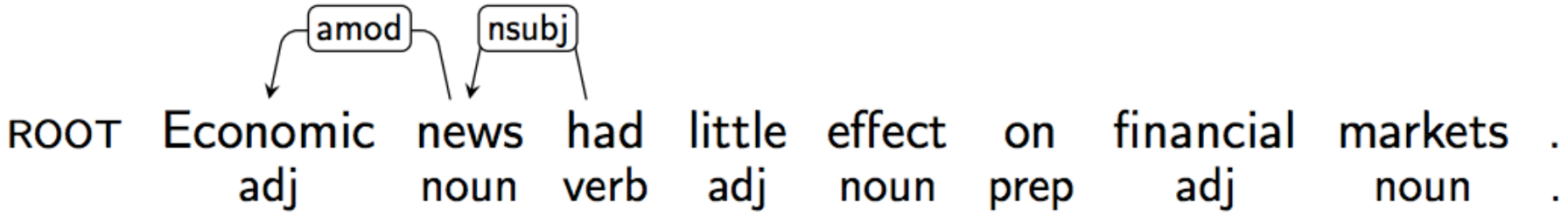


Example Transition Sequence

Left Arc:

Add left-arc *nsubj*(had, news) to A.
Remove news from stack since it now has head in A.

$[\text{ROOT}]_S$ $[\text{had, little, effect, on, financial, markets, .}]_B$

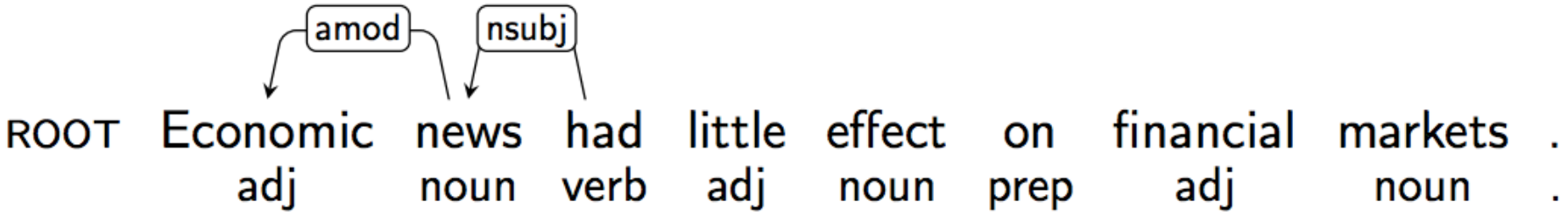


Example Transition Sequence

Shift

Move had to stack.

[ROOT, had]$_S$ [little, effect, on, financial, markets, .]$_B$

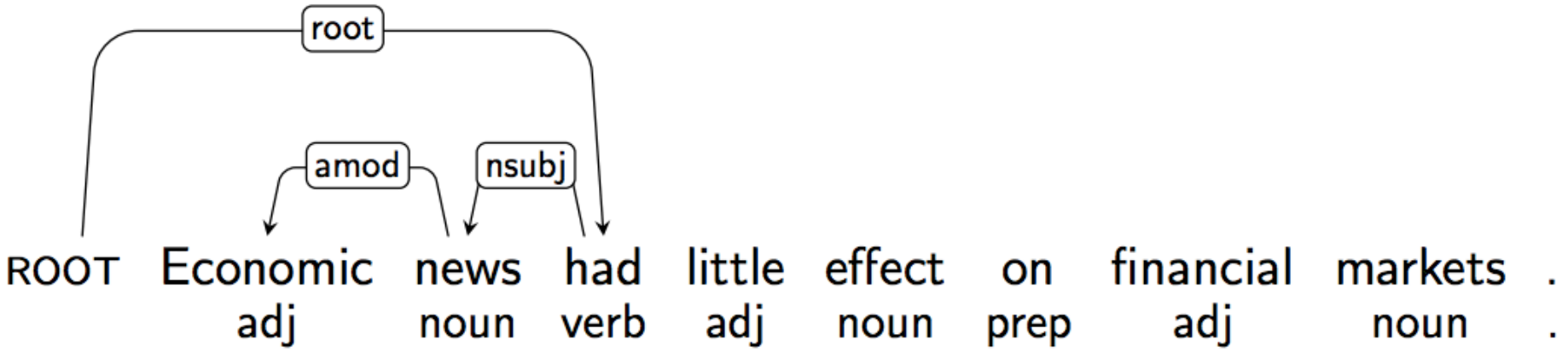


Example Transition Sequence

Right Arc:

Add right-arc *root*(ROOT, had) to A.
Keep had in stack.
NOTE: We are keeping had because it can have other dependents on the left.

$[\text{ROOT, had}]_S$ $[\text{little, effect, on, financial, markets, .}]_B$

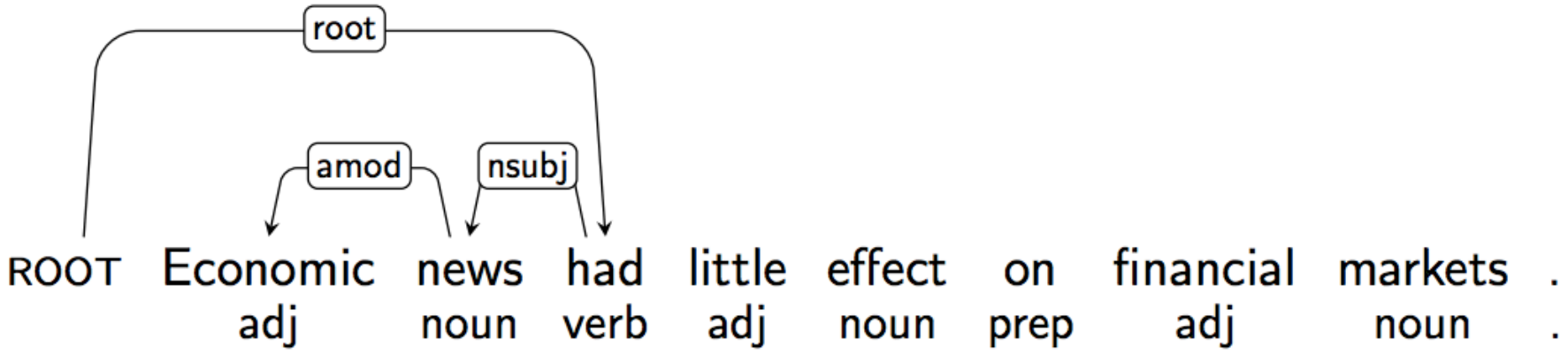


Example Transition Sequence

Shift:

Black-box did not find any dependence relation for had and little.

$[\text{ROOT}, \text{had}, \text{little}]_S$ $[\text{effect}, \text{on}, \text{financial}, \text{markets}, .]_B$

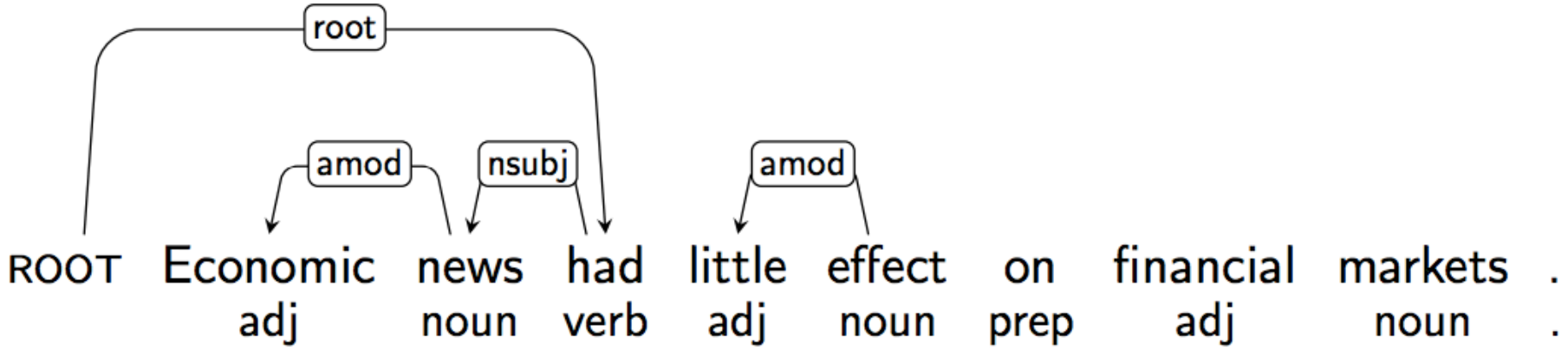


Example Transition Sequence

Left-arc:

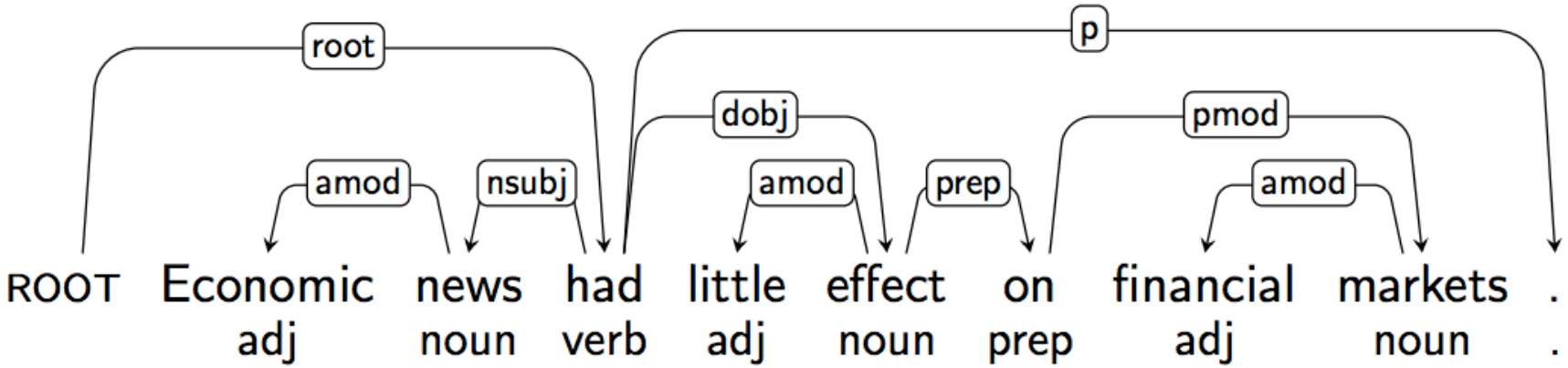
Add amod(effect, little) to A.
Remove little from stack.

$[\text{ROOT, had}]_S$ $[\text{effect, on, financial, markets, .}]_B$



And on it goes until …

[ROOT, had, .]$_S$ [ ]$_B$



As a supervised classification task.

- Given the current state (i.e., stack, buffer and A) predict the next action.
- Can be viewed as a supervised learning problem.
  - Four way classification (if un-typed dependencies)
  - m-way classification, where m = 2 x number of types + 2
- Features
  - Compute features of the current configuration of the stack, buffer and A.
  - Word in stack, POS of word, Word in buffer and POS of Word in buffer.
  - Other features: Length of dependency arc
- Greedy classifier (no search involved)
  - At each stage ask the classifier to predict the next transition.
  - Select the best legal transition and apply it.
  - Works quite well, close to PCFG.
- Quite fast!
  - O(N) in length of sentence.