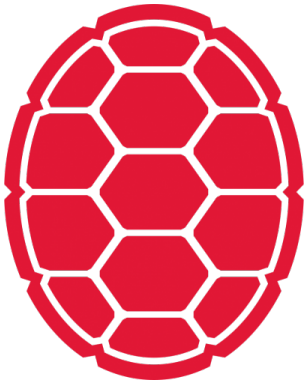


Dependency Parsing

Computational Linguistics: Jordan Boyd-Graber
University of Maryland

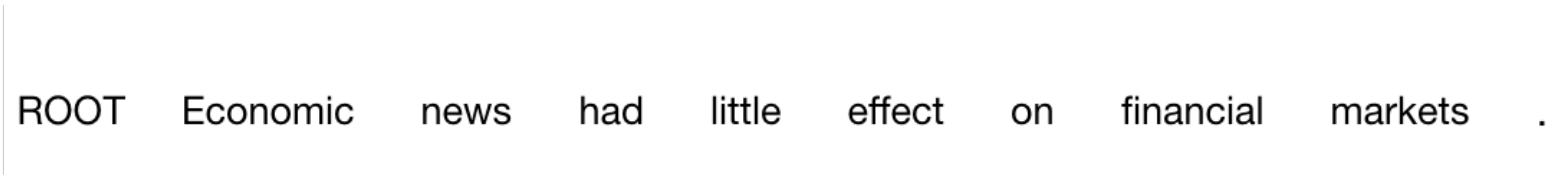
SHIFT-REDUCE

Adapted from material by Jimmy Lin and Jason Eisner

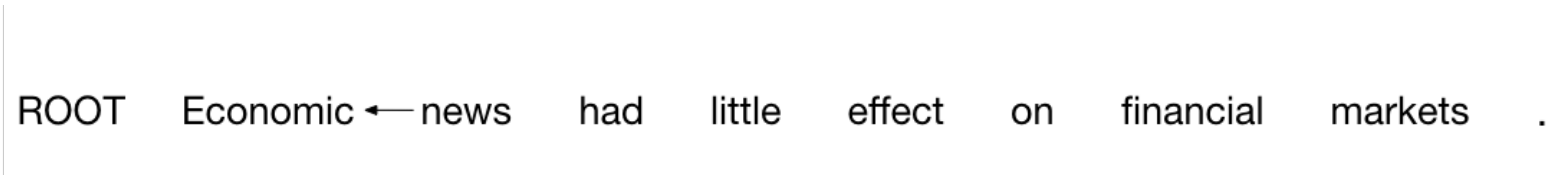
Shift-Reduce Parsing

- Alternative to arc-factored models
- Cognitively plausible
- Better at short-range dependencies

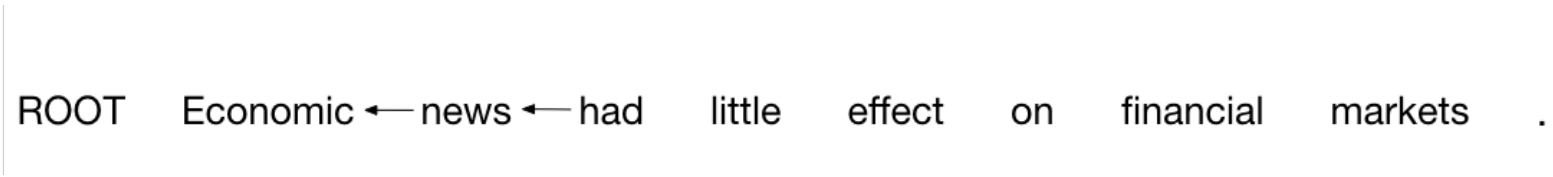
Example



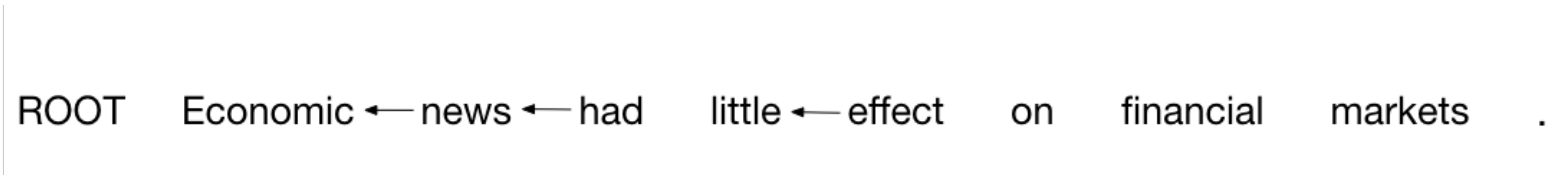
Example



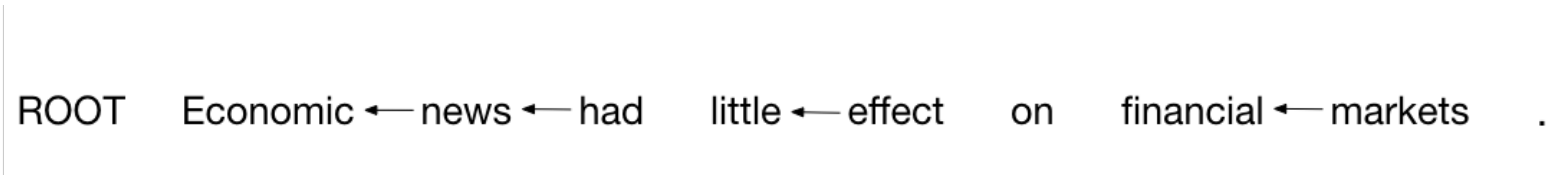
Example



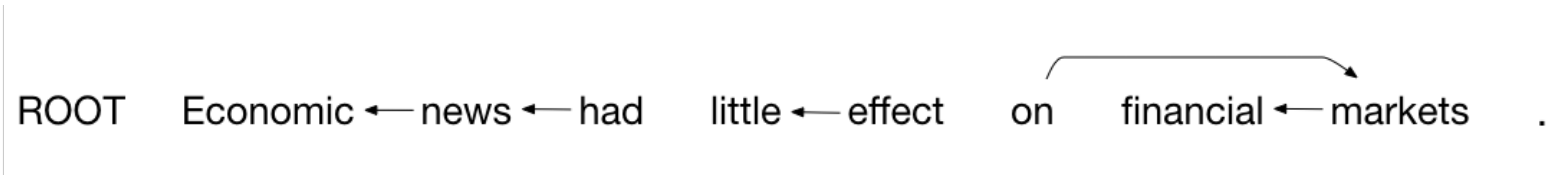
Example



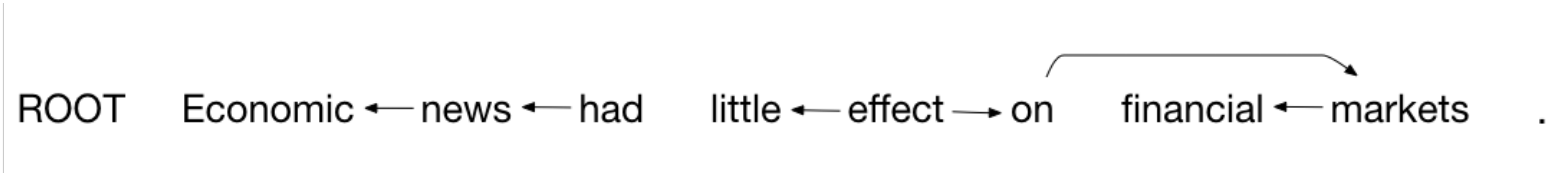
Example



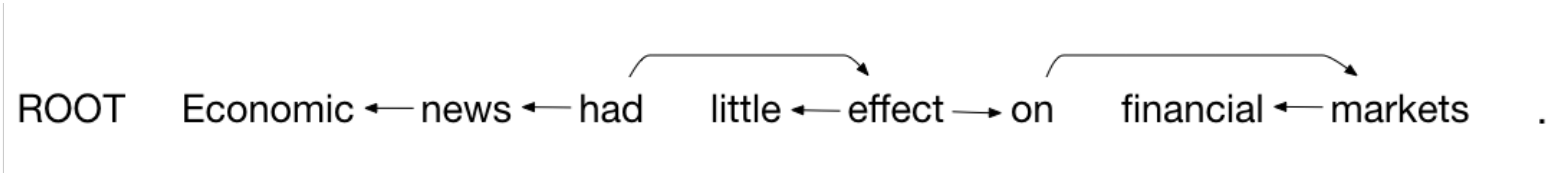
Example



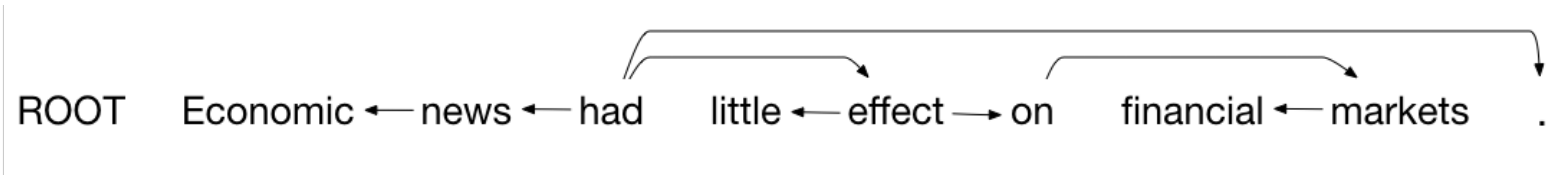
Example



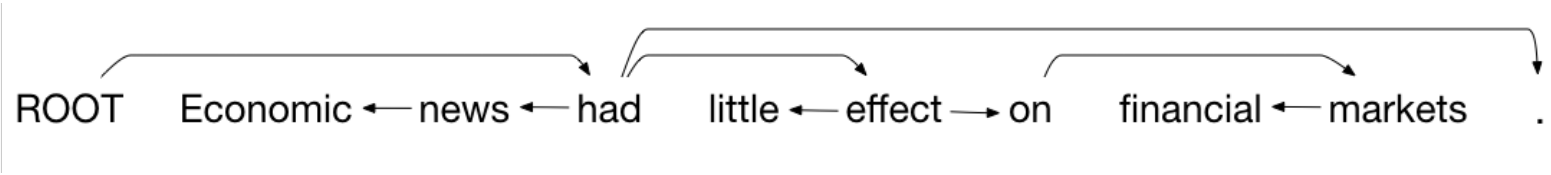
Example



Example



Example



Components

- Process a sentence word by word from a **buffer**
- You can temporarily place store words on a **stack**
- As you process you can either:

Components

- Process a sentence word by word from a **buffer**
- You can temporarily place store words on a **stack**
- As you process you can either:
  - Shift: Move a word from the buffer to the stack

Components

- Process a sentence word by word from a **buffer**
- You can temporarily place store words on a **stack**
- As you process you can either:
  - Shift: Move a word from the buffer to the stack
  - Left: The top of the stack is the child of the buffer's next word

Components

- Process a sentence word by word from a **buffer**
- You can temporarily place store words on a **stack**
- As you process you can either:
  - Shift: Move a word from the buffer to the stack
  - Left: The top of the stack is the child of the buffer's next word
  - Right: The buffer's next word is the child of the top of the stack

Initial and Final Conditions

- Initially the stack has root, the buffer has the sentence's words, and there are no edges
- At the end, the buffer must be empty

Action: Left

- Add an edge (w_j, w_i)
- w_i is the top of the stack
- w_j is the first word of the buffer
- Pop the stack

Action: Left

- Add an edge (w_j, w_i)
- w_i is the top of the stack
- w_j is the first word of the buffer
- Pop the stack
- Stack and buffer must be non-empty; w_i cannot be the root

Action: Right

- Add an edge (w_i, w_j)
- w_i is the top of the stack
- w_j is the first word in the buffer
- Pop the stack
- Replace w_j by w_i at the head of buffer

Action: Right

- Add an edge (w_i, w_j)
- w_i is the top of the stack
- w_j is the first word in the buffer
- Pop the stack
- Replace w_j by w_i at the head of buffer
- Stack and buffer must be non-empty

Shift

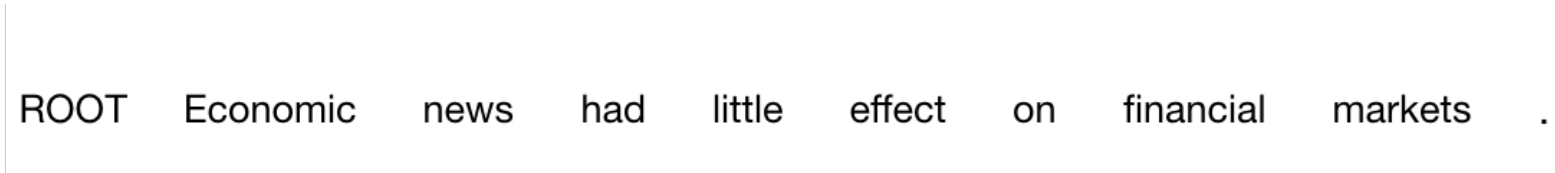
- Removes w_i from the buffer
- Places it on the stack

Shift

- Removes w_i from the buffer
- Places it on the stack
- Buffer must be non-empty

Shift Reduce Example

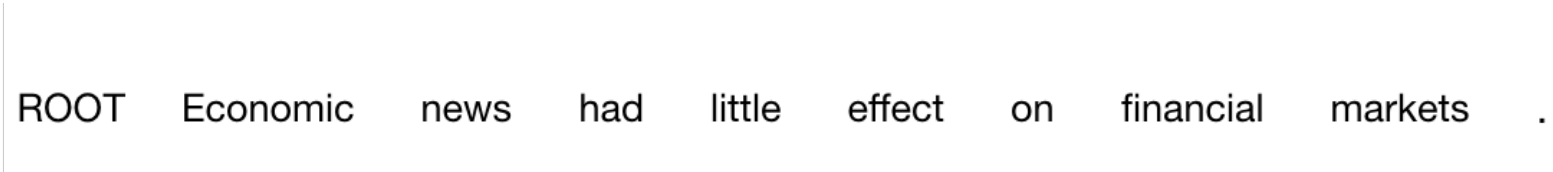
Stack	Buffer
[root]	[economic, news, had, little, effect, on, financial, markets, .]



Next transition: 1. Shift

Shift Reduce Example

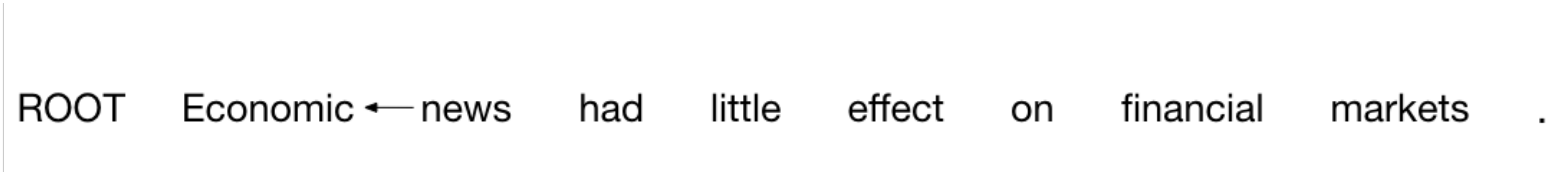
Stack	Buffer
[root , economic]	[news, had, little, effect, on, financial, markets, .]



Next transition: 2. Left

Shift Reduce Example

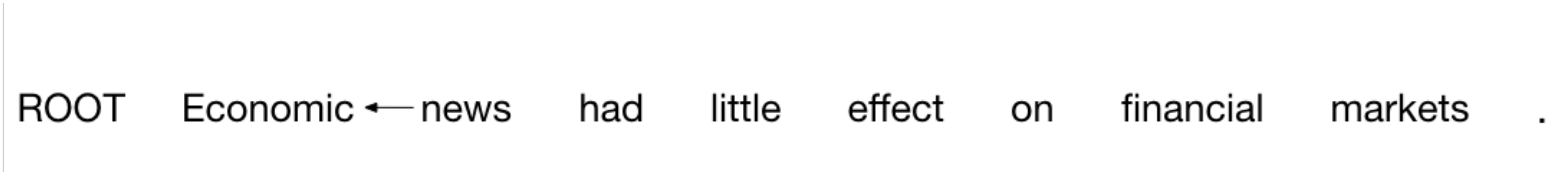
Stack	Buffer
[root]	[news, had, little, effect, on, financial, markets, .]



Next transition: 3. Shift

Shift Reduce Example

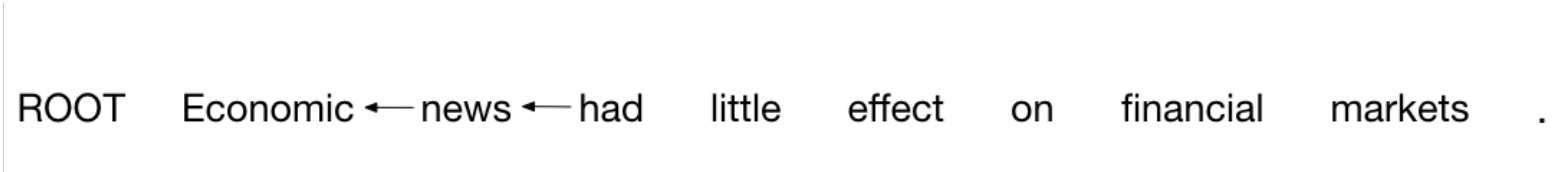
Stack	Buffer
[root , news]	[had, little, effect, on, financial, markets, .]



Next transition: 4. Left

Shift Reduce Example

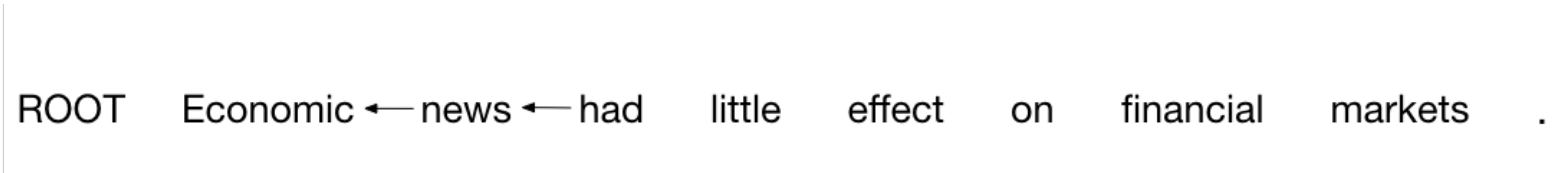
Stack	Buffer
[root]	[had, little, effect, on, financial, markets, .]



Next transition: 5. Shift

Shift Reduce Example

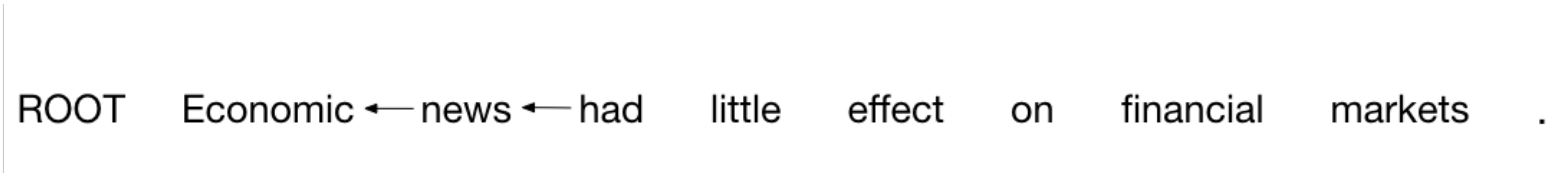
Stack	Buffer
[root , had]	[little, effect, on, financial, markets, .]



Next transition: 6. Shift

Shift Reduce Example

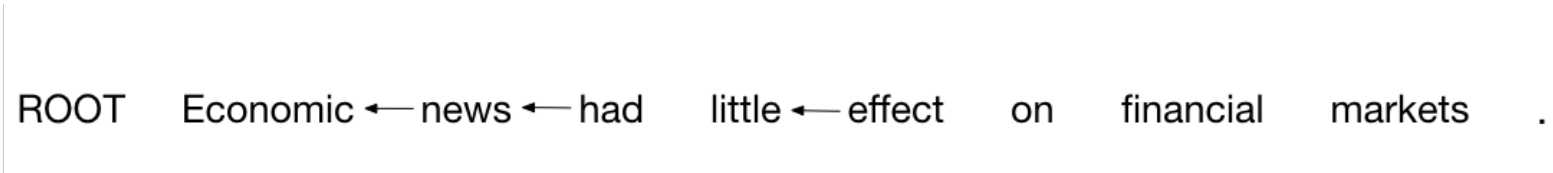
Stack	Buffer
[root , had , little]	[effect, on, financial, markets, .]



Next transition: 7. Left

Shift Reduce Example

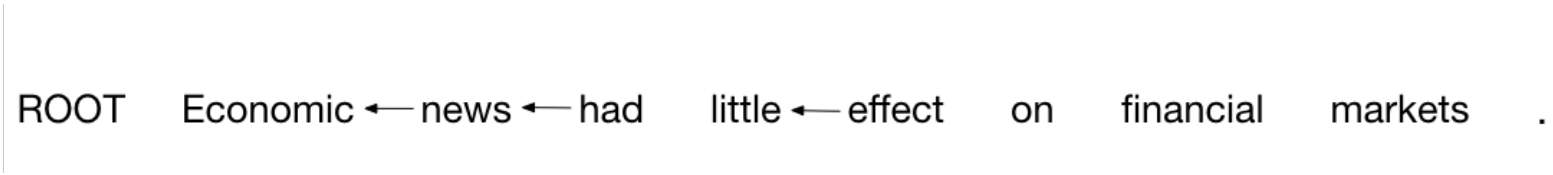
Stack	Buffer
[root , had]	[effect, on, financial, markets, .]



Next transition: 8. Shift

Shift Reduce Example

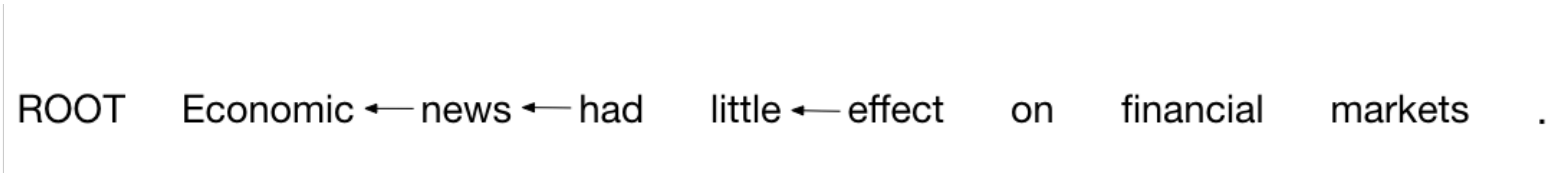
Stack	Buffer
[root , had , effect]	[on, financial, markets, .]



Next transition: 9. Shift

Shift Reduce Example

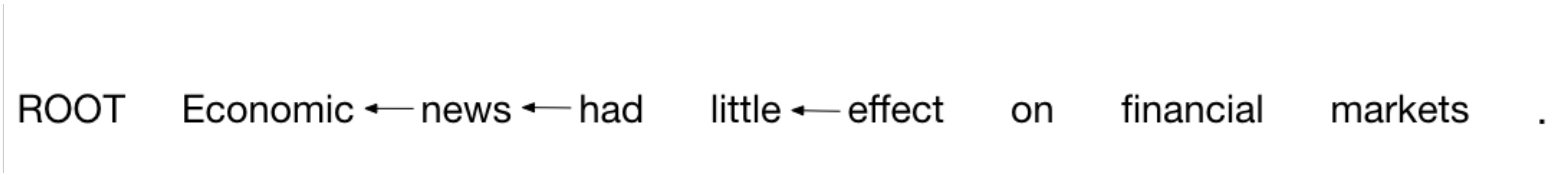
Stack	Buffer
[root , had , effect , on]	[financial, markets, .]



Next transition: 10. Shift

Shift Reduce Example

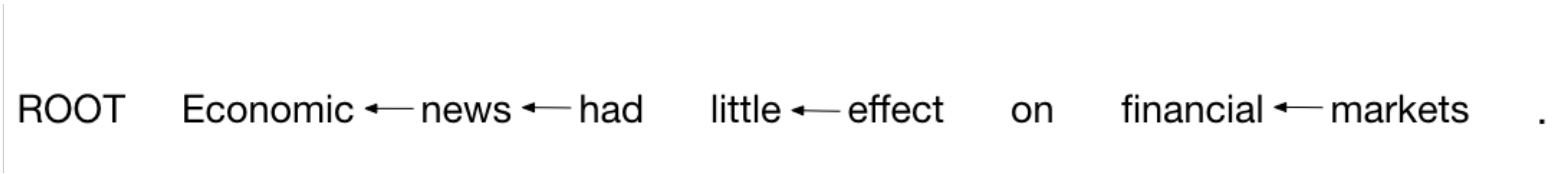
Stack	Buffer
[root , had , effect , on , financial]	[markets, .]



Next transition: 11. Left

Shift Reduce Example

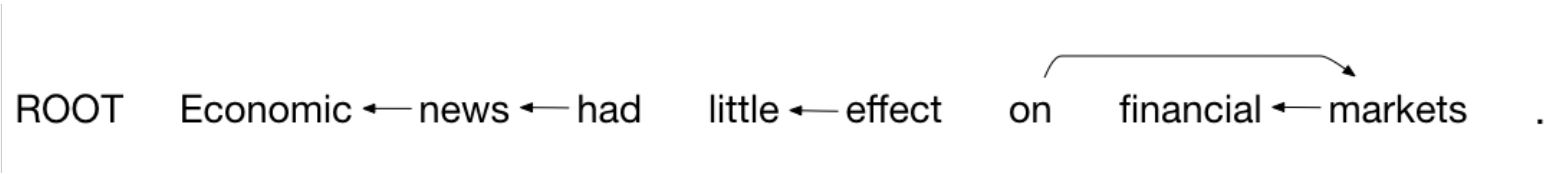
Stack	Buffer
[root , had , effect , on]	[markets, .]



Next transition: 12. Right

Shift Reduce Example

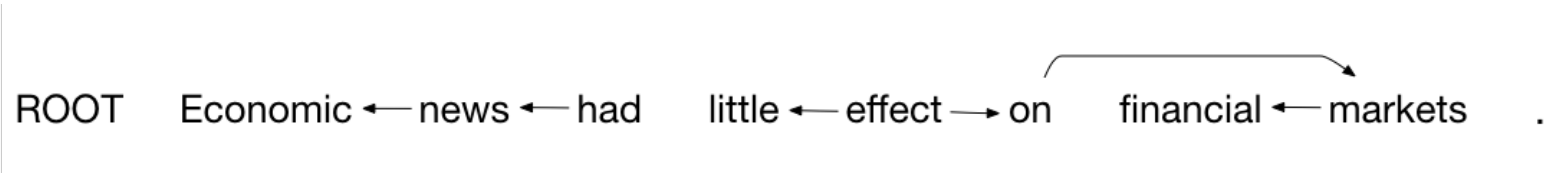
Stack	Buffer
[root , had , effect]	[on, .]



Next transition: 13. Right

Shift Reduce Example

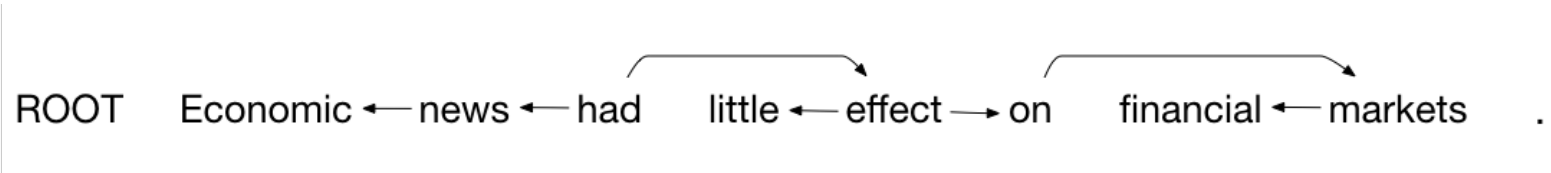
Stack	Buffer
[root , had]	[effect, .]



Next transition: 14. Right

Shift Reduce Example

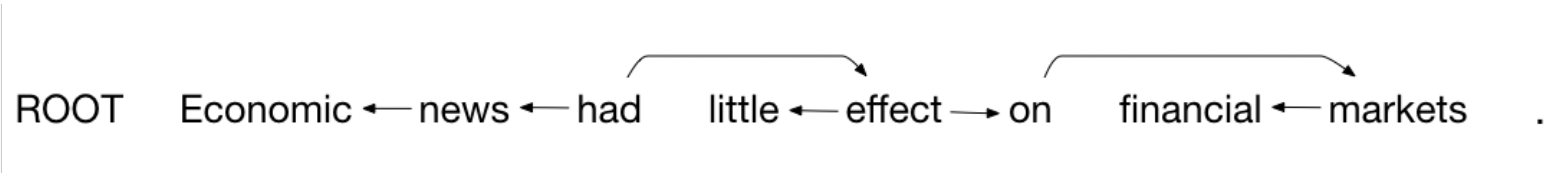
Stack	Buffer
[root]	[had, .]



Next transition: 15. Shift

Shift Reduce Example

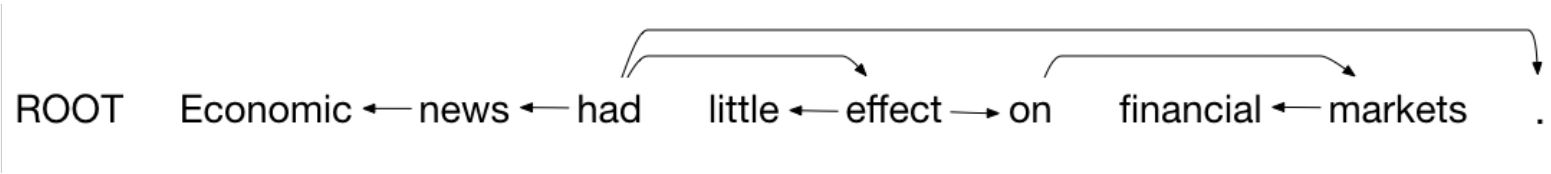
Stack	Buffer
[root , had]	[.]



Next transition: 16. Right

Shift Reduce Example

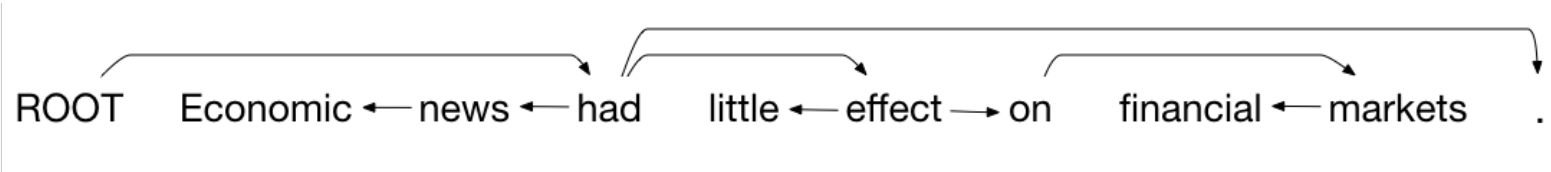
Stack	Buffer
[root]	[had]



Next transition: 17. Right

Shift Reduce Example

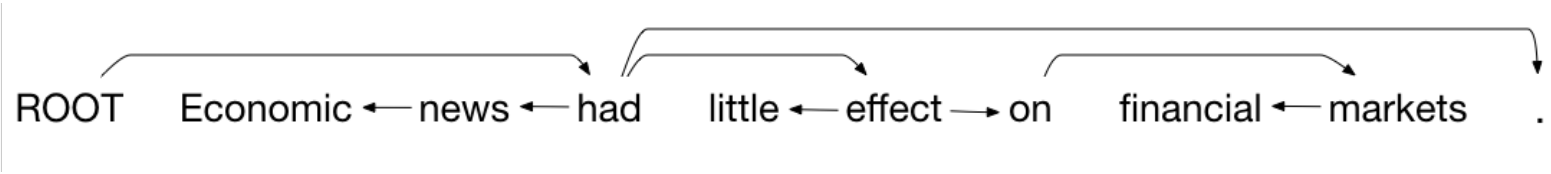
Stack	Buffer
[ ]	[root]



Next transition: 18. Shift

Shift Reduce Example

Stack	Buffer
[root]	[]



Next transition:

Transition Sequence Algorithm

- Start with root on stack, buffer with whole sentence
- If there's nothing on the stack, you must shift
- If the top of the stack is the child of the top of the buffer, then make a left edge
- If the top of the buffer is is a child of the top of the stack and the top of the buffer has no children that have yet to be added to the tree, then make a right

How to apply to data

- Create oracle for all sentences
- Create three-way classifier for each possible actions
- Features
  - The top of the stack
  - Top two words on buffer
  - The parts of speech of the words

Complexity

- A word can only enter the stack once
- So complexity is $O(2N)$

Comparison

- Shift-reduce parsers are faster
- Shift-reduce parsers do better at local (deeper) connections
- Arc-factored models do better at long-distance dependencies (e.g., verbs)