Department of Computer Science
UNIVERSITY OF COLORADO **BOULDER**

Shift-Reduce Parsing

Natural Language Processing: Jordan Boyd-Graber
University of Colorado Boulder
6. OCTOBER 2014

Adapted from material by Jimmy Lin and Jason Eisner

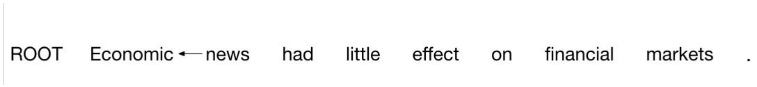
Shift-Reduce Parsing

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

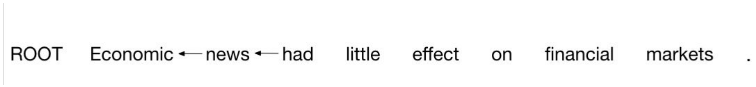
Example

ROOT Economic news had little effect on financial markets .

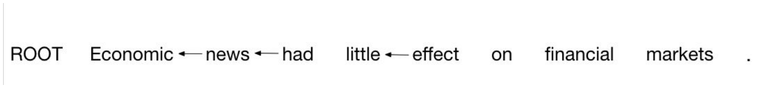
Example



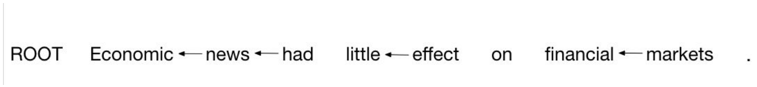
Example



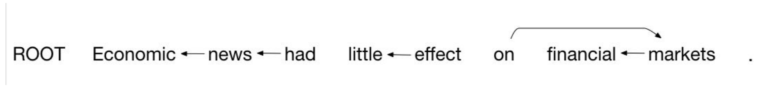
Example



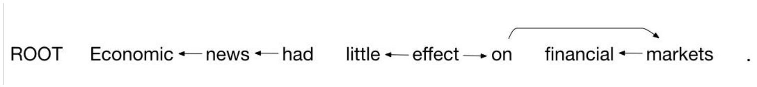
Example



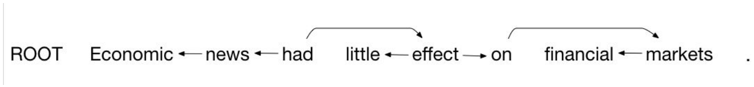
Example



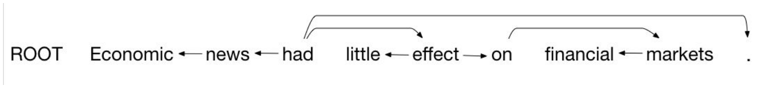
Example



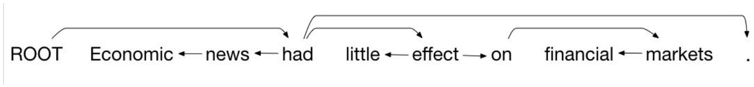
Example



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

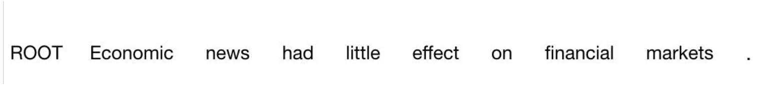
Shift Reduce Example

Stack

[ROOT]

Buffer

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



Next transition: 1. Shift

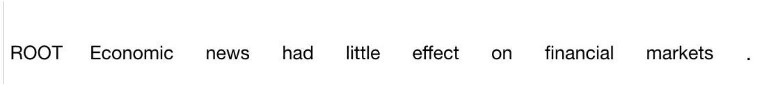
Shift Reduce Example

Stack

[ROOT , economic]

Buffer

[news, had, little, effect, on, financial, markets, .]



Next transition: 2. Left

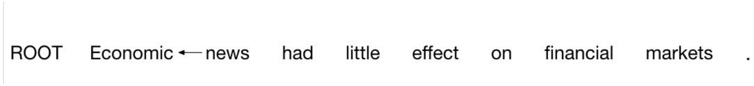
Shift Reduce Example

Stack

[ROOT]

Buffer

[news, had, little, effect, on, financial, markets, .]



Next transition: 3. Shift

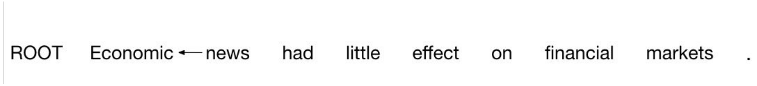
Shift Reduce Example

Stack

[ROOT , news]

Buffer

[had, little, effect, on, financial, markets, .]



Next transition: 4. Left

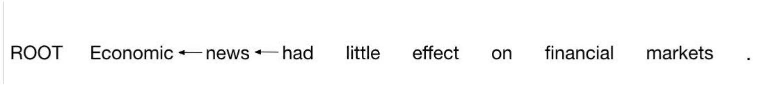
Shift Reduce Example

Stack

[ROOT]

Buffer

[had, little, effect, on, financial, markets, .]



Next transition: 5. Shift

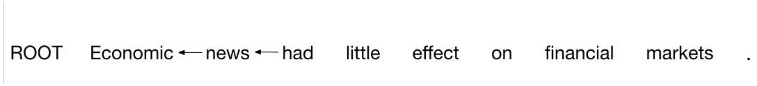
Shift Reduce Example

Stack

[ROOT , had]

Buffer

[little, effect, on, financial, markets, .]



Next transition: 6. Shift

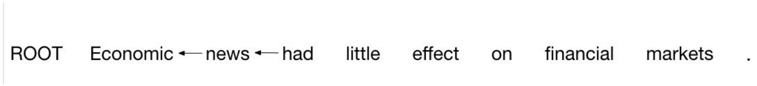
Shift Reduce Example

Stack

[ROOT , had , little]

Buffer

[effect, on, financial, markets, .]



Next transition: 7. Left

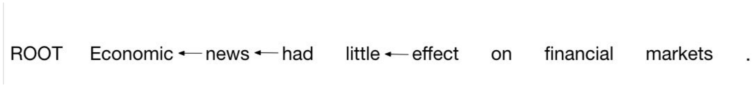
Shift Reduce Example

Stack

[ROOT , had]

Buffer

[effect, on, financial, markets, .]



Next transition: 8. Shift

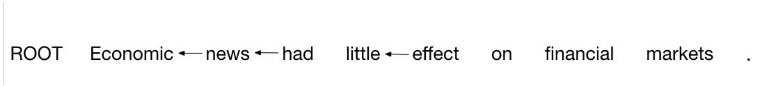
Shift Reduce Example

Stack

[ROOT , had , effect]

Buffer

[on, financial, markets, .]



Next transition: 9. Shift

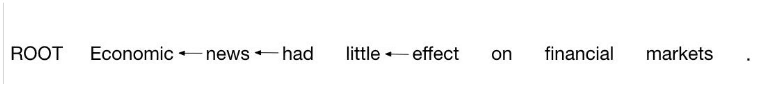
Shift Reduce Example

Stack

[ROOT , had , effect , on]

Buffer

[financial, markets, .]



Next transition: 10. Shift

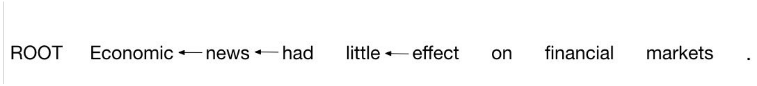
Shift Reduce Example

Stack

[ROOT , had , effect , on , financial]

Buffer

[markets, .]



Next transition: 11. Left

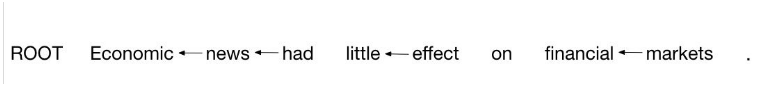
Shift Reduce Example

Stack

[ROOT , had , effect , on]

Buffer

[markets, .]



Next transition: 12. Right

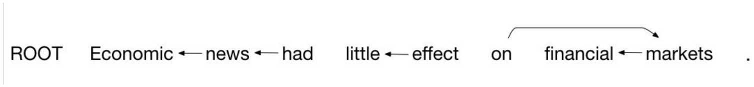
Shift Reduce Example

Stack

[ROOT , had , effect]

Buffer

[on, .]



Next transition: 13. Right

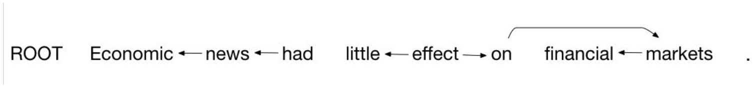
Shift Reduce Example

Stack

[ROOT , had]

Buffer

[effect, .]



Next transition: 14. Right

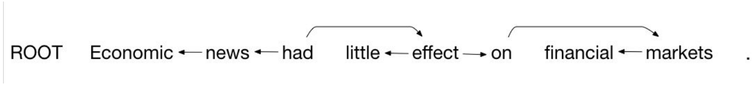
Shift Reduce Example

Stack

[ROOT]

Buffer

[had, .]



Next transition: 15. Shift

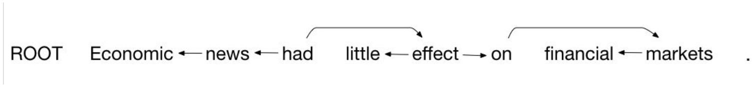
Shift Reduce Example

Stack

[ROOT , had]

Buffer

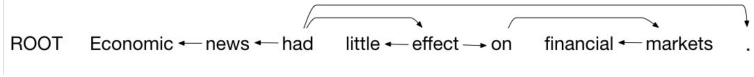
[.]



Next transition: 16. Right

Shift Reduce Example

Stack	Buffer
[ROOT]	[had]



Next transition: 17. Right

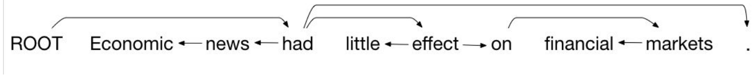
Shift Reduce Example

Stack

[ ]

Buffer

[ROOT]



Next transition: 18. Shift

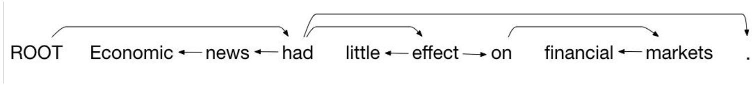
Shift Reduce Example

Stack

[ROOT]

Buffer

[]



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*

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*
- Part of Homework 6

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

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)

In Class

- Transition Sequence to Parse
- Parse to Transition Sequence

Stack

[ROOT]

Buffer

[I, am, the, very, model, of, a, modern, major, general]

Edges

Stack

[ROOT , I]

Buffer

[am, the, very, model, of, a, modern, major, general]

Edges

Stack

[ROOT]

Buffer

[am, the, very, model, of, a, modern, major, general]

Edges

, I $\leftarrow$ am

Stack

[ROOT , am]

Buffer

[the, very, model, of, a, modern, major, general]

Edges

, I $\leftarrow$ am

Stack

[ROOT , am , the]

Buffer

[very, model, of, a, modern, major, general]

Edges

, I ← am

Stack

[ROOT , am , the , very]

Buffer

[model, of, a, modern, major, general]

Edges

, I $\leftarrow$ am

Stack

[ROOT , am , the]

Buffer

[model, of, a, modern, major, general]

Edges

, I $\leftarrow$ am

, very $\leftarrow$ model

Stack

[ROOT , am]

Buffer

[model, of, a, modern, major, general]

Edges

, I ← am

, very ← model

, the ← model

Stack

[ROOT , am , model]

Buffer

[of, a, modern, major, general]

Edges

, I $\leftarrow$ am

, very $\leftarrow$ model

, the $\leftarrow$ model

Stack

[ROOT , am , model , of]

Buffer

[a, modern, major, general]

Edges

, I $\leftarrow$ am
, very $\leftarrow$ model
, the $\leftarrow$ model
, model $\rightarrow$ of

Stack

[ROOT , am , model , of , a]

Buffer

[modern, major, general]

Edges

, I $\leftarrow$ am

, very $\leftarrow$ model

, the $\leftarrow$ model

, model $\rightarrow$ of

Stack

[ROOT , am , model , of , a , modern]

Buffer

[major, general]

Edges

, I $\leftarrow$ am
, very $\leftarrow$ model
, the $\leftarrow$ model
, model $\rightarrow$ of

Stack

[ROOT , am , model , of , a , modern , major]

Buffer

[general]

Edges

, I $\leftarrow$ am
, very $\leftarrow$ model
, the $\leftarrow$ model
, model $\rightarrow$ of

Stack

[ROOT , am , model , of , a , modern]

Buffer

[general]

Edges

, I $\leftarrow$ am
, very $\leftarrow$ model
, the $\leftarrow$ model
, model $\rightarrow$ of
, major $\leftarrow$ general

Stack

[ROOT , am , model , of , a]

Buffer

[general]

Edges

, I ← am
, very ← model
, the ← model
, model → of
, major ← general
, modern ← general

Stack

[ROOT , am , model , of]

Buffer

[general]

Edges

, I ← am
, very ← model
, the ← model
, model → of
, major ← general
, modern ← general
, a ← general

Stack

[ROOT , am , model]

Buffer

[of,]

Edges

, I $\leftarrow$ am
, very $\leftarrow$ model
, the $\leftarrow$ model
, model $\rightarrow$ of
, major $\leftarrow$ general
, modern $\leftarrow$ general
, a $\leftarrow$ general
, of $\rightarrow$ general

Stack

[ROOT , am]

Buffer

[model,]

Edges

, I $\leftarrow$ am
, very $\leftarrow$ model
, the $\leftarrow$ model
, model $\rightarrow$ of
, major $\leftarrow$ general
, modern $\leftarrow$ general
, a $\leftarrow$ general
, of $\rightarrow$ general
, model $\rightarrow$ of

Stack

[ROOT]

Buffer

[am]

Edges

, I $\leftarrow$ am
, very $\leftarrow$ model
, the $\leftarrow$ model
, model $\rightarrow$ of
, major $\leftarrow$ general
, modern $\leftarrow$ general
, a $\leftarrow$ general
, of $\rightarrow$ general
, model $\rightarrow$ of
, am $\rightarrow$ model

Stack

[ ]

Buffer

[ROOT]

Edges

, I ← am
, very ← model
, the ← model
, model → of
, major ← general
, modern ← general
, a ← general
, of → general
, model → of
, am → model
, ROOT → am

Stack

[ROOT]

Buffer

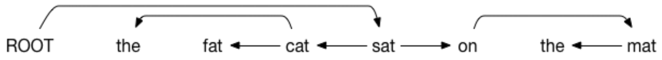
[]

Edges

, I $\leftarrow$ am
, very $\leftarrow$ model
, the $\leftarrow$ model
, model $\rightarrow$ of
, major $\leftarrow$ general
, modern $\leftarrow$ general
, a $\leftarrow$ general
, of $\rightarrow$ general
, model $\rightarrow$ of
, am $\rightarrow$ model
, ROOT $\rightarrow$ am

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*



Parse to Transition Sequence

Action	Head Index	Head Word	Dep Index	Dep Word
s				

Parse to Transition Sequence

Action	Head Index	Head Word	Dep Index	Dep Word
s				
s				

Parse to Transition Sequence

Action	Head Index	Head Word	Dep Index	Dep Word
s				
s				
l	3	cat	2	fat

Parse to Transition Sequence

Action	Head Index	Head Word	Dep Index	Dep Word
s				
s				
l	3	cat	2	fat
l	3	cat	1	the

Parse to Transition Sequence

Action	Head Index	Head Word	Dep Index	Dep Word
s				
s				
l	3	cat	2	fat
l	3	cat	1	the
s				

Parse to Transition Sequence

Action	Head Index	Head Word	Dep Index	Dep Word
s				
s				
l	3	cat	2	fat
l	3	cat	1	the
s				
l	4	sat	3	cat

Parse to Transition Sequence

Action	Head Index	Head Word	Dep Index	Dep Word
s				
s				
l	3	cat	2	fat
l	3	cat	1	the
s				
l	4	sat	3	cat
s				

Parse to Transition Sequence

Action	Head Index	Head Word	Dep Index	Dep Word
s				
s				
l	3	cat	2	fat
l	3	cat	1	the
s				
l	4	sat	3	cat
s				
s				

Parse to Transition Sequence

Action	Head Index	Head Word	Dep Index	Dep Word
s				
s				
l	3	cat	2	fat
l	3	cat	1	the
s				
l	4	sat	3	cat
s				
s				
s				

Parse to Transition Sequence

Action	Head Index	Head Word	Dep Index	Dep Word
s				
s				
l	3	cat	2	fat
l	3	cat	1	the
s				
l	4	sat	3	cat
s				
s				
s				
l	7	mat	6	the

Parse to Transition Sequence

Action	Head Index	Head Word	Dep Index	Dep Word
s				
s				
l	3	cat	2	fat
l	3	cat	1	the
s				
l	4	sat	3	cat
s				
s				
s				
l	7	mat	6	the
r	5	on	7	mat

Parse to Transition Sequence

Action	Head Index	Head Word	Dep Index	Dep Word
s				
s				
l	3	cat	2	fat
l	3	cat	1	the
s				
l	4	sat	3	cat
s				
s				
s				
l	7	mat	6	the
r	5	on	7	mat
r	4	sat	5	on

Parse to Transition Sequence

Action	Head Index	Head Word	Dep Index	Dep Word
s				
s				
l	3	cat	2	fat
l	3	cat	1	the
s				
l	4	sat	3	cat
s				
s				
s				
l	7	mat	6	the
r	5	on	7	mat
r	4	sat	5	on
r	0	None	4	sat

Parse to Transition Sequence

Action	Head Index	Head Word	Dep Index	Dep Word
s				
s				
l	3	cat	2	fat
l	3	cat	1	the
s				
l	4	sat	3	cat
s				
s				
s				
l	7	mat	6	the
r	5	on	7	mat
r	4	sat	5	on
r	0	None	4	sat
s				