Natural Language Processing
Dependency parsing

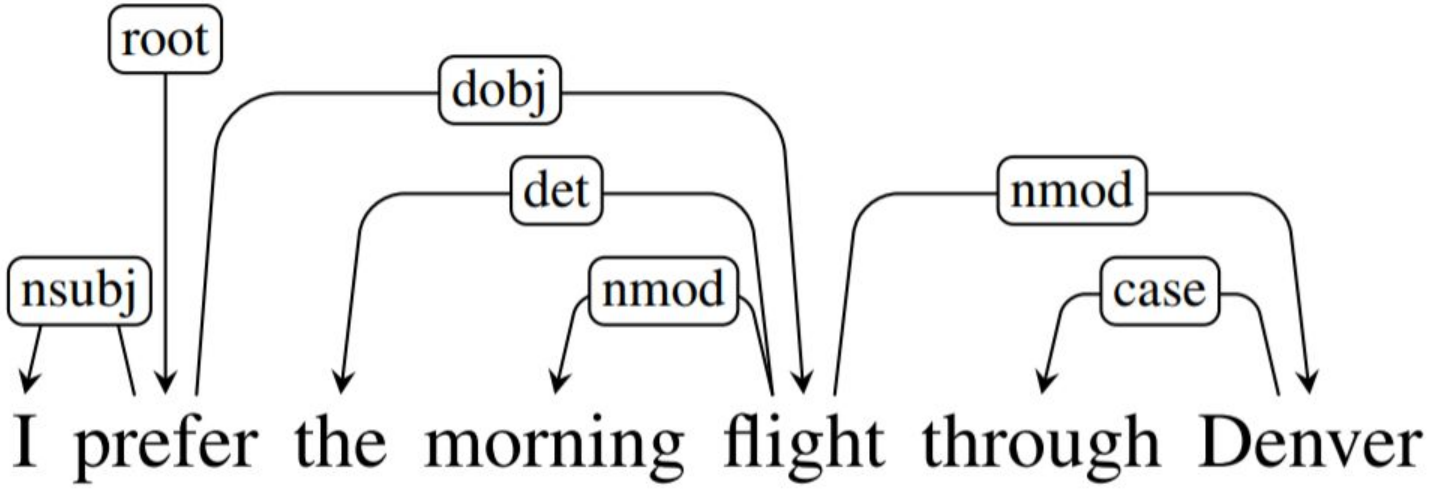
Sofia Serrano
sofias6@cs.washington.edu

Credit to Yulia Tsvetkov for slides

Announcement(s)

- A3 is out!
- Quiz 7 goes out on Canvas on Wednesday 3/1 at the end of lecture
  - Available until Friday 3/3 at 2:20pm; you'll have 15 minutes to complete it once you start.
  - Remember that you can use your notes during the quiz
  - Will cover material from last Wednesday's lecture through the end of today's lecture (so, parsing)
- For **this Wednesday** and **this Friday's** lectures: you have a choice of whether to join live over zoom or come to our usual classroom!
  - Our 3/1 and 3/3 guest speakers will be joining virtually to give their talks, so I'll be sending the zoom link that they'll be using to all of you this afternoon via Ed
  - But if you'd rather come to the usual classroom, I'll be here and projecting their lectures on the big screen!

Dependency representation

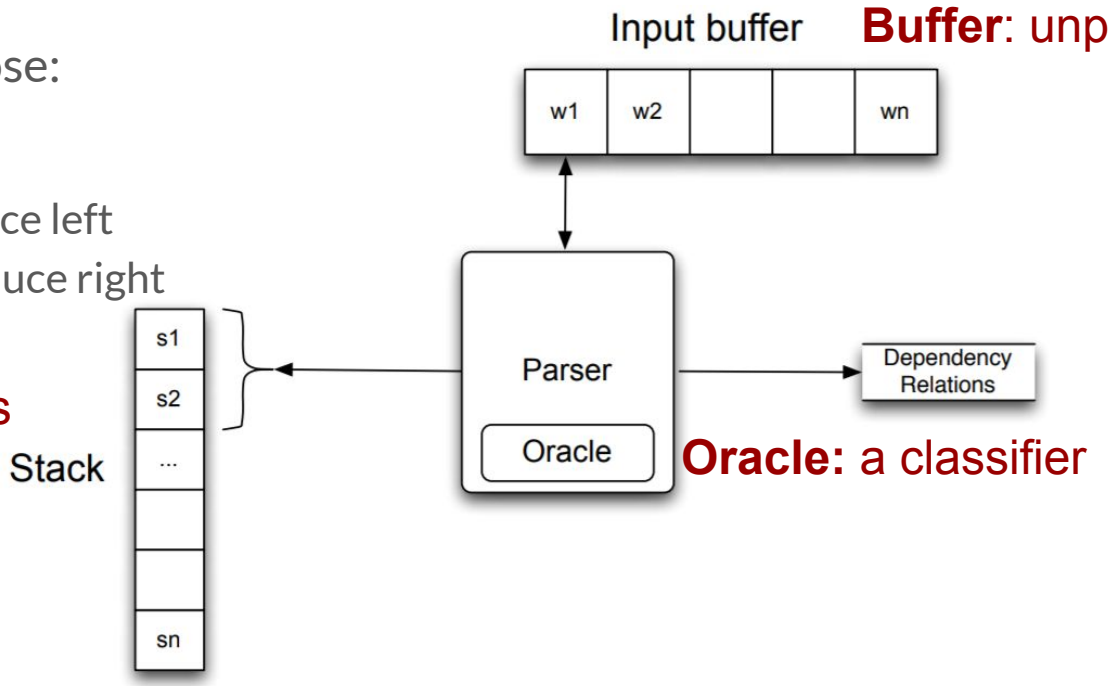


Operations

At each step choose:

- Shift
- LeftArc or Reduce left
- RightArc or Reduce right

Stack: partially processed words



$$C_{\text{accept}} = ([\text{ROOT}], \varnothing, A)$$

Shift-Reduce Parsing

Configuration:

- Stack, Buffer, Oracle, Set of dependency relations

Operations by a classifier at each step:

- Shift
  - remove w1 from the buffer, push it onto the stack as s1
- LeftArc or Reduce left
  - assert a head-dependent relation between s1 and s2 (s1 → s2)
  - pop s1 from the stack; pop s2 from the stack; then push (s2← s1) onto the stack
- RightArc or Reduce right
  - assert a head-dependent relation between s2 and s1 (s2 → s1)
  - pop s1 from the stack; pop s2 from the stack; then push (s2 → s1) onto the stack

Or as A3/Eisenstein do it...

Operations by a classifier at each step:

- Shift
  - remove w1 from the buffer, push it onto the stack as s1
- LeftArc or Reduce left
  - assert a head-dependent relation between s1 and s2 (s1 → s2)
  - pop s1 from the stack; pop s2 from the stack; then push (s2 ← s1) onto the stack
- RightArc or Reduce right
  - assert a head-dependent relation between s2 and s1 (s2 → s1)
  - pop s1 from the stack; pop s2 from the stack; then push (s2 → s1) onto the stack

Operations by a classifier at each step:

- Shift
  - remove w1 from the buffer, push it onto the stack as s1
- LeftArc or Reduce left
  - assert a head-dependent relation between b1 and s1 (b1 → s1)
  - pop s1 from the stack; pop b1 from the buffer; then push (s1 ← b1) onto the buffer
- RightArc or Reduce right
  - assert a head-dependent relation between s1 and b1 (s1 → b1)
  - pop s1 from the stack; pop b1 from the buffer; then push (s1 → b1) onto the buffer

Want to see an(other) example of transition-based parsing in action?

Slides 30-44 of this slide deck by Noah Smith do a really nice job of walking through the full transition-based assembly of a sentence's parse visually.

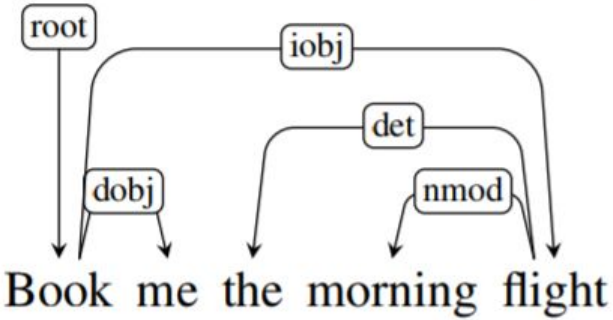
Shift-Reduce Parsing (Arc-standard)

$$C_{\text{initial}} = ([\text{ROOT}], \boldsymbol{w}, \varnothing)$$

Book me the morning flight

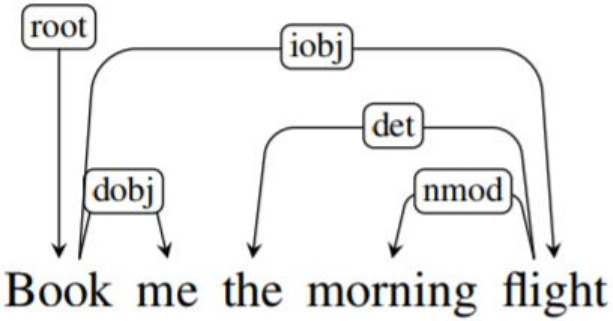
Step	Stack	Word List	Action	Relation Added

Shift-Reduce Parsing



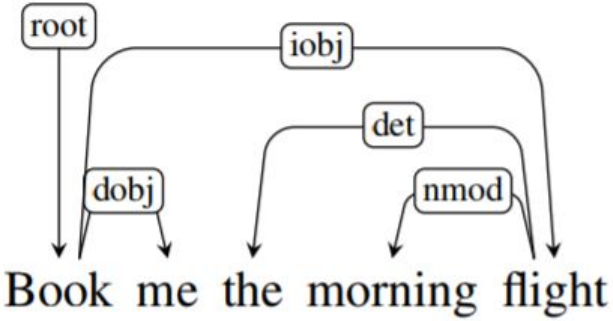
Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]		

Shift-Reduce Parsing



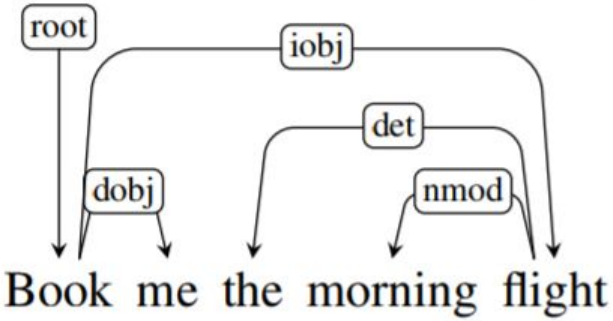
Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]		

Shift-Reduce Parsing



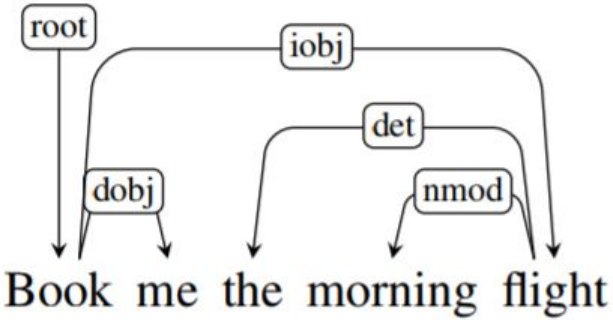
Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]		

Shift-Reduce Parsing



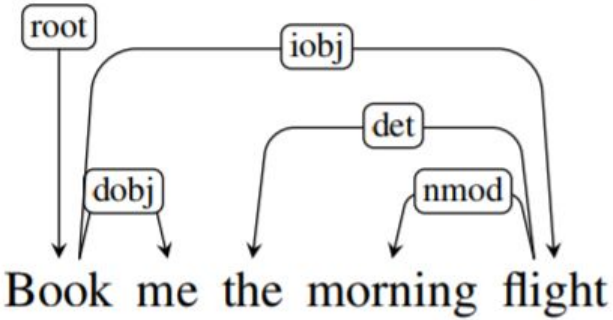
Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	

Shift-Reduce Parsing



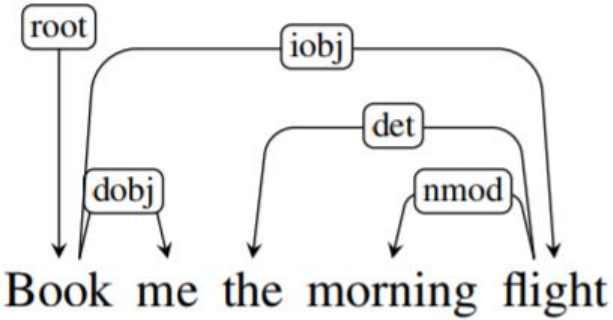
Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]		

Shift-Reduce Parsing



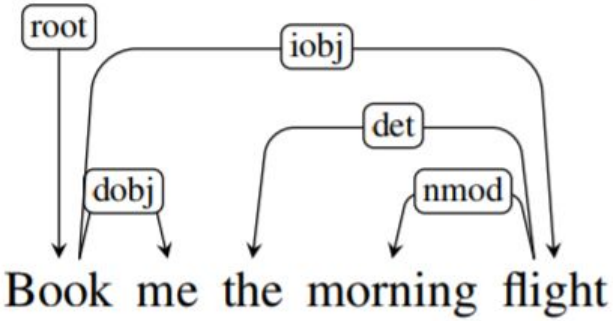
Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	

Shift-Reduce Parsing



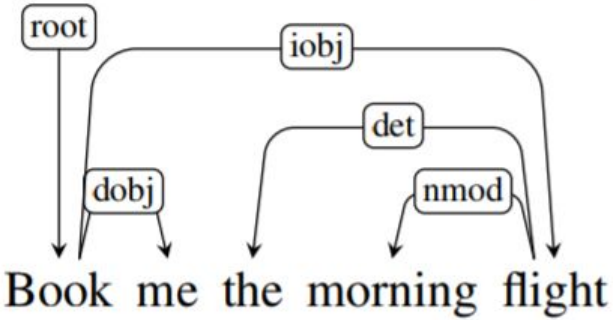
Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book → me)

Shift-Reduce Parsing



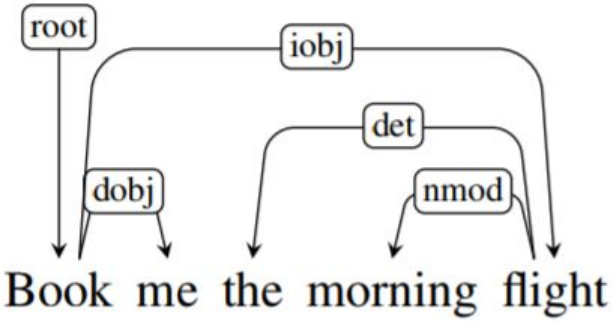
Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book → me)
3	[root, book]	[the, morning, flight]		

Shift-Reduce Parsing



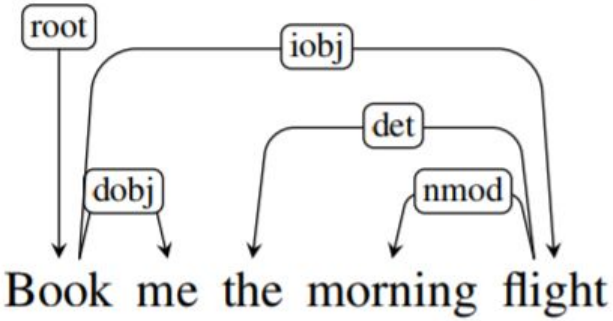
Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book → me)
3	[root, book]	[the, morning, flight]	SHIFT	

Shift-Reduce Parsing



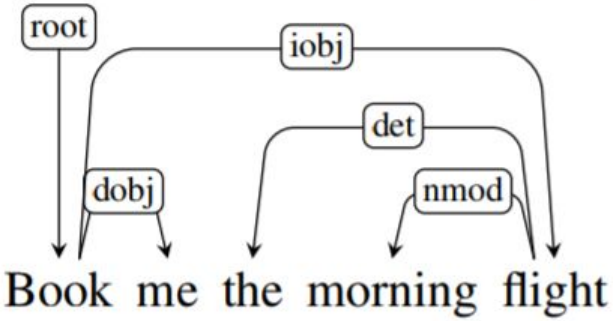
Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book → me)
3	[root, book]	[the, morning, flight]	SHIFT	
4	[root, book, the]	[morning, flight]		

Shift-Reduce Parsing



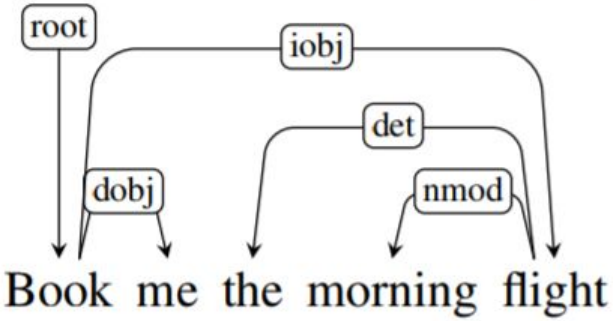
Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book → me)
3	[root, book]	[the, morning, flight]	SHIFT	
4	[root, book, the]	[morning, flight]	SHIFT	

Shift-Reduce Parsing



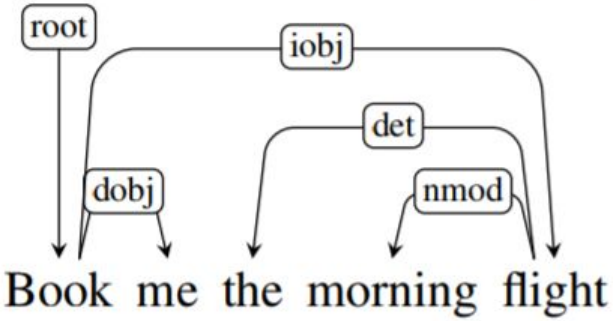
Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book → me)
3	[root, book]	[the, morning, flight]	SHIFT	
4	[root, book, the]	[morning, flight]	SHIFT	
5	[root, book, the, morning]	[flight]		

Shift-Reduce Parsing



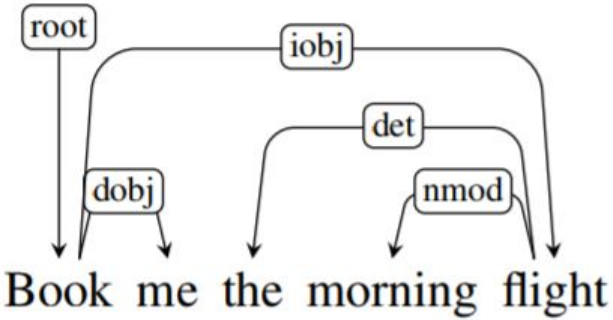
Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book → me)
3	[root, book]	[the, morning, flight]	SHIFT	
4	[root, book, the]	[morning, flight]	SHIFT	
5	[root, book, the, morning]	[flight]	SHIFT	

Shift-Reduce Parsing



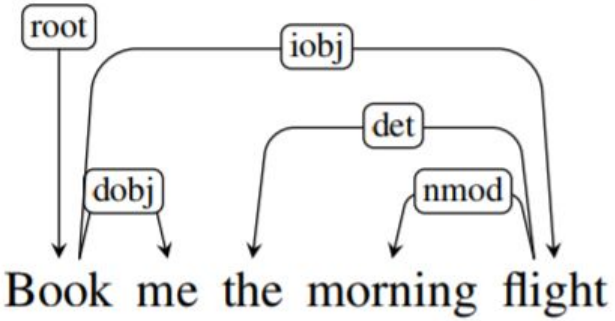
Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book → me)
3	[root, book]	[the, morning, flight]	SHIFT	
4	[root, book, the]	[morning, flight]	SHIFT	
5	[root, book, the, morning]	[flight]	SHIFT	
6	[root, book, the, morning, flight]	[]		

Shift-Reduce Parsing



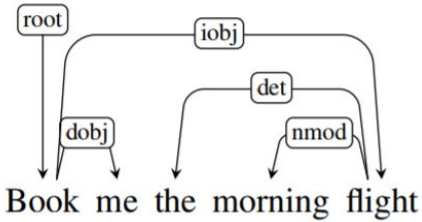
Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book → me)
3	[root, book]	[the, morning, flight]	SHIFT	
4	[root, book, the]	[morning, flight]	SHIFT	
5	[root, book, the, morning]	[flight]	SHIFT	
6	[root, book, the, morning, flight]	[]	LEFTARC	(morning ← flight)

Shift-Reduce Parsing



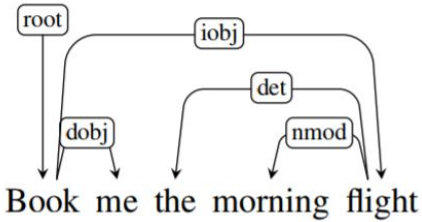
Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book → me)
3	[root, book]	[the, morning, flight]	SHIFT	
4	[root, book, the]	[morning, flight]	SHIFT	
5	[root, book, the, morning]	[flight]	SHIFT	
6	[root, book, the, morning, flight]	[]	LEFTARC	(morning ← flight)
7	[root, book, the, flight]	[]	LEFTARC	(the ← flight)

Shift-Reduce Parsing



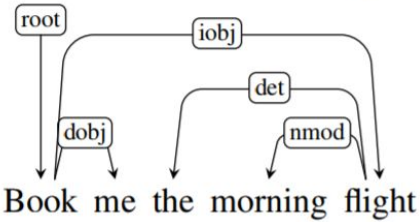
Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book → me)
3	[root, book]	[the, morning, flight]	SHIFT	
4	[root, book, the]	[morning, flight]	SHIFT	
5	[root, book, the, morning]	[flight]	SHIFT	
6	[root, book, the, morning, flight]	[]	LEFTARC	(morning ← flight)
7	[root, book, the, flight]	[]	LEFTARC	(the ← flight)
8	[root, book, flight]	[]	RIGHTARC	(book → flight)

Shift-Reduce Parsing



Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book → me)
3	[root, book]	[the, morning, flight]	SHIFT	
4	[root, book, the]	[morning, flight]	SHIFT	
5	[root, book, the, morning]	[flight]	SHIFT	
6	[root, book, the, morning, flight]	[]	LEFTARC	(morning ← flight)
7	[root, book, the, flight]	[]	LEFTARC	(the ← flight)
8	[root, book, flight]	[]	RIGHTARC	(book → flight)
9	[root, book]	[]	RIGHTARC	(root → book)

Shift-Reduce Parsing



$$C_{\text{accept}} = ([\text{ROOT}], \varnothing, A)$$

Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book → me)
3	[root, book]	[the, morning, flight]	SHIFT	
4	[root, book, the]	[morning, flight]	SHIFT	
5	[root, book, the, morning]	[flight]	SHIFT	
6	[root, book, the, morning, flight]	[]	LEFTARC	(morning ← flight)
7	[root, book, the, flight]	[]	LEFTARC	(the ← flight)
8	[root, book, flight]	[]	RIGHTARC	(book → flight)
9	[root, book]	[]	RIGHTARC	(root → book)
10	[root]	[]	Done	

Or as A3/Eisenstein do it…

Note: for A3, the buffer shouldn't actually be empty, it should consist of just <EOS>

$$C_{\text{accept}} = ([\text{ROOT}], \varnothing, A)$$

$$C_{\text{accept}} = ([\text{ROOT}], \varnothing, A)$$

(now just has to end with a SHIFT action)

Shift-Reduce Parsing

Configuration:

- Stack, Buffer, Oracle, Set of dependency relations

Complexity?

Operations by a classifier at each step:

Oracle decisions can correspond to unlabeled or labeled arcs

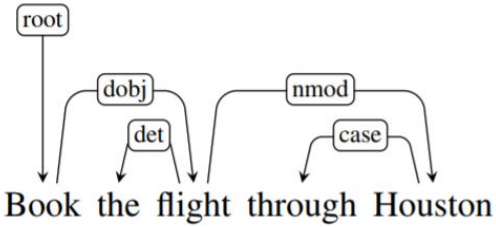
- Shift
  - remove w1 from the buffer, push it onto the stack as s1
- LeftArc or Reduce left
  - assert a head-dependent relation between s1 and s2 (s1 → s2)
  - pop s1 from the stack; pop s2 from the stack; then push (s2← s1) onto the stack
- RightArc or Reduce right
  - assert a head-dependent relation between s2 and s1 (s2 → s1)
  - pop s1 from the stack; pop s2 from the stack; then push (s2 → s1) onto the stack

Training an Oracle

- Oracle is a supervised classifier that learns a function from the configuration to the next operation
- How to extract the training set?

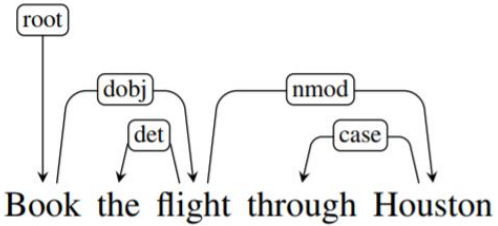
Training an Oracle

- Given a gold tree, how to extract the gold set of steps for training?
  - if LeftArc → LeftArc
  - if RightArc
    - if s1 dependents have been processed → RightArc
  - else → Shift



Training an Oracle

- Given a gold tree, how to extract the gold set of steps for training?
  - if LeftArc → LeftArc
  - if RightArc
    - if s1 dependents have been processed → RightArc
  - else → Shift



Step	Stack	Word List	Predicted Action
0	[root]	[book, the, flight, through, houston]	SHIFT
1	[root, book]	[the, flight, through, houston]	SHIFT
2	[root, book, the]	[flight, through, houston]	SHIFT
3	[root, book, the, flight]	[through, houston]	LEFTARC
4	[root, book, flight]	[through, houston]	SHIFT
5	[root, book, flight, through]	[houston]	SHIFT
6	[root, book, flight, through, houston]	[]	LEFTARC
7	[root, book, flight, houston]	[]	RIGHTARC
8	[root, book, flight]	[]	RIGHTARC
9	[root, book]	[]	RIGHTARC
10	[root]	[]	Done

Figure 13.8 Generating training items consisting of configuration/predicted action pairs by simulating a parse with a given reference parse.

Training an Oracle

- Oracle is a supervised classifier that learns a function from the configuration to the next operation
- Given a gold tree, how to extract the gold set of steps for training?
  - if LeftArc → LeftArc
  - if RightArc
    - if s1 dependents have been processed → RightArc
  - else → Shift
- What features to use?

Features

- POS, word-forms, lemmas on the stack/buffer
- morphological features for some languages
- previous relations
- conjunction features (e.g. Zhang&Clark'08; Huang&Sagae'10; Zhang&Nivre'11)

$$\langle s_1.w = flights, op = shift \rangle$$
$$\langle s_2.w = canceled, op = shift \rangle$$
$$\langle s_1.t = NNS, op = shift \rangle$$
$$\langle s_2.t = VBD, op = shift \rangle$$
$$\langle b_1.w = to, op = shift \rangle$$
$$\langle b_1.t = TO, op = shift \rangle$$
$$\langle s_1.wt = flightsNNS, op = shift \rangle$$
$$\langle s_1.t \circ s_2.t = NNSVBD, op = shift \rangle$$

Source	Feature templates		
One word	$s_1.w$	$s_1.t$	$s_1.wt$
	$s_2.w$	$s_2.t$	$s_2.wt$
	$b_1.w$	$b_1.w$	$b_0.wt$
Two word	$s_1.w \circ s_2.w$	$s_1.t \circ s_2.t$	$s_1.t \circ b_1.w$
	$s_1.t \circ s_2.wt$	$s_1.w \circ s_2.w \circ s_2.t$	$s_1.w \circ s_1.t \circ s_2.t$
	$s_1.w \circ s_1.t \circ s_2.t$	$s_1.w \circ s_1.t$	

In other words, features are usually some combination of...

- Information about the word(s) left in the buffer (often just the first)
- Information about the top elements in the stack (often just the top two)

Or as A3/Eisenstein do it…

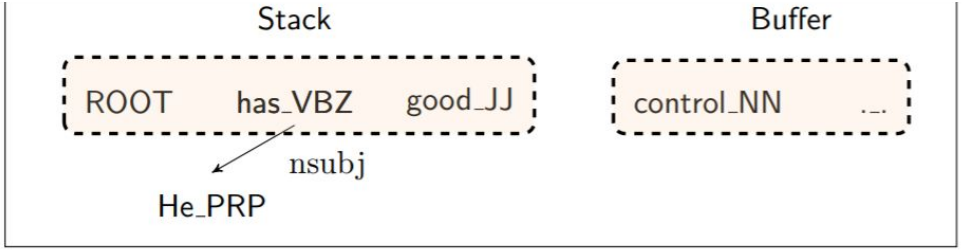
- Information about the word(s) left in the buffer (often just the first)
- Information about the top elements in the stack (often just the top two)

- Information about the word(s) left in the buffer (often just the first two)
- Information about the top elements in the stack (often just the top one)

Learning

- Before 2014: Support Vector Machines (SVMs),
- After 2014: Neural Nets

Chen & Manning 2014



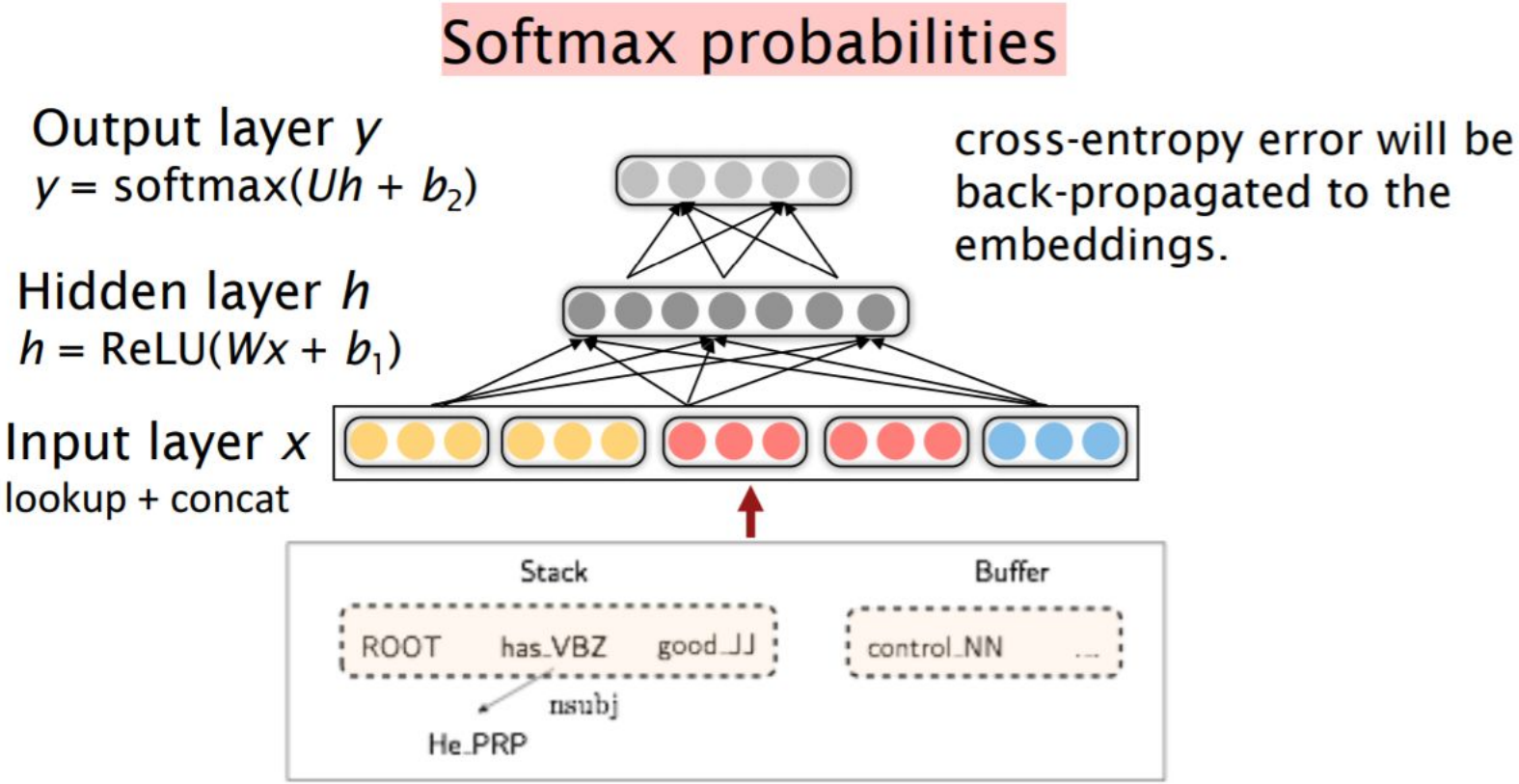
binary, sparse
dim = 10^6 ~ 10^7

Indicator features

$$s_2.w = \text{has} \wedge s_2.t = \text{VBZ}$$
$$s_1.w = \text{good} \wedge s_1.t = \text{JJ} \wedge b_1.w = \text{control}$$
$$lc(s_2).t = \text{PRP} \wedge s_2.t = \text{VBZ} \wedge s_1.t = \text{JJ}$$
$$lc(s_2).w = \text{He} \wedge lc(s_2).l = \text{nsubj} \wedge s_2.w = \text{has}$$

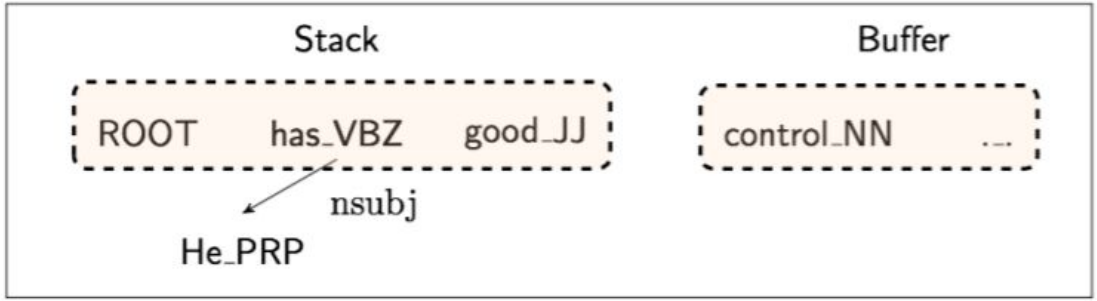
Slides by Danqi Chen & Chris Manning

Chen & Manning 2014



Chen & Manning 2014

- Features
  - s1, s2, s3, b1, b2, b3
  - leftmost/rightmost children of s1 and s2
  - leftmost/rightmost grandchildren of s1 and s2
  - POS tags for the above
  - arc labels for children/grandchildren

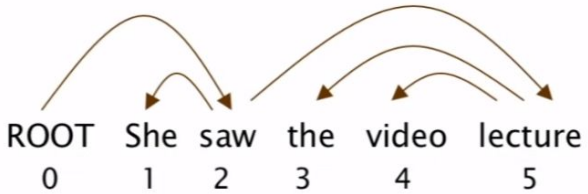


	word	POS	dep.
s_1	good	JJ	∅
s_2	has	VBZ	∅
b_1	control	NN	∅
$lc(s_1)$	∅	∅	∅
$rc(s_1)$	∅	∅	∅
$lc(s_2)$	He	PRP	nsubj
$rc(s_2)$	∅	∅	∅

Evaluation of Dependency Parsers

- LAS - labeled attachment score
- UAS - unlabeled attachment score

$$\frac{\#correct\ dependencies}{\#of\ dependencies}$$



Gold			
1	2	She	nsubj
2	0	saw	root
3	5	the	det
4	5	video	nn
5	2	lecture	obj

Parsed			
1	2	She	nsubj
2	0	saw	root
3	4	the	det
4	5	video	nsubj
5	2	lecture	ccomp

Chen & Manning 2014

Parser	UAS	LAS	sent. / s
MaltParser	89.8	87.2	469
MSTParser	91.4	88.1	10
TurboParser	**92.3***	89.6*	8
C & M 2014	92.0	**89.7**	**654**

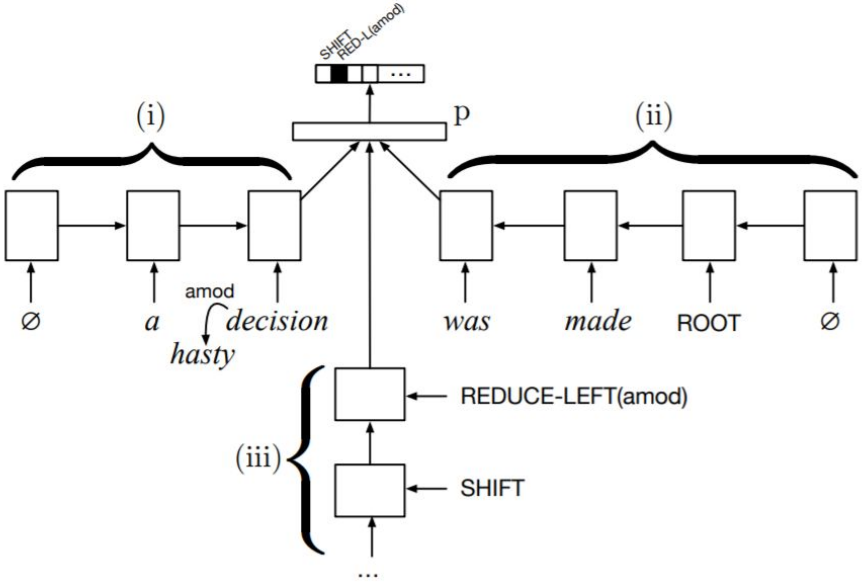
Follow-up

Method	UAS	LAS (PTB WSJ SD 3.3
Chen & Manning 2014	92.0	89.7
Weiss et al. 2015	93.99	92.05
Andor et al. 2016	94.61	92.79

Stack LSTMs (Dyer et al. 2015)

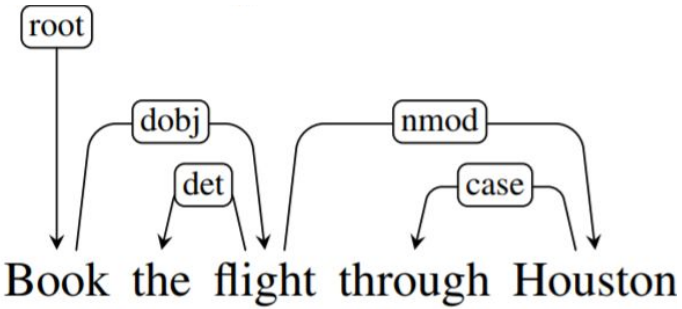
Transition-Based Dependency Parsing with Stack Long Short-Term Memory

Chris Dyer[♣♠] **Miguel Ballesteros**[◇♠] **Wang Ling**[♠] **Austin Matthews**[♠] **Noah A. Smith**[♠]
[♣]Marianas Labs [◇]NLP Group, Pompeu Fabra University [♠]Carnegie Mellon University
chris@marianaslabs.com, miguel.ballesteros@upf.edu,
{lingwang,austinma,nasmith}@cs.cmu.edu

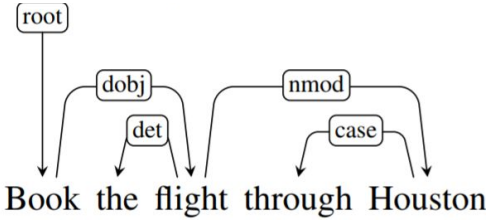


Arc-Eager version (in A3/Eisenstein speak)

- LEFTARC: Assert a head-dependent relation from b1 to s1; pop the stack.
- RIGHTARC: Assert a head-dependent relation from s1 to b1; shift b1 to be s1 instead of dropping it from the stack/buffer.
- SHIFT: Remove b1 and push it to be the new s1.
- REDUCE: Pop the stack.



Arc-Eager



Step	Stack	Word List	Action	Relation Added
0	[root]	[book, the, flight, through, houston]	RIGHTARC	(root → book)
1	[root, book]	[the, flight, through, houston]	SHIFT	
2	[root, book, the]	[flight, through, houston]	LEFTARC	(the ← flight)
3	[root, book]	[flight, through, houston]	RIGHTARC	(book → flight)
4	[root, book, flight]	[through, houston]	SHIFT	
5	[root, book, flight, through]	[houston]	LEFTARC	(through ← houston)
6	[root, book, flight]	[houston]	RIGHTARC	(flight → houston)
7	[root, book, flight, houston]	[]	REDUCE	
8	[root, book, flight]	[]	REDUCE	
9	[root, book]	[]	REDUCE	
10	[root]	[]	Done	

Parsing algorithms

- Transition based
  - greedy choice of local transitions guided by a good classifier
  - deterministic
  - MaltParser (Nivre et al. 2008), Stack LSTM (Dyer et al. 2015)
- Graph based
  - Minimum Spanning Tree for a sentence
  - non-projective
  - globally optimized
  - McDonald et al.'s (2005) MSTParser
  - Martins et al.'s (2009) Turbo Parser

Summary

- Transition-based
  - + Fast
  - + Rich features of context
  - - Greedy decoding
- Graph-based
  - + Exact or close to exact decoding
  - - Weaker features

Well-engineered versions of the approaches achieve comparable accuracy (on English), but make different errors

→ combining the strategies results in a substantial boost in performance

Previewing the last few lectures of the quarter

Remaining lecture topics

- Wednesday 3/1: Saadia Gabriel on commonsense reasoning, factuality, also touches on toxic language detection **[virtual;** your choice whether to join Zoom meeting or come to classroom]
- Friday 3/3: Alane Suhr on multimodal NLP and grounding for NLP **[virtual;** your choice whether to join Zoom meeting or come to classroom]
- Monday 3/6: Sewon Min on prompting and in-context learning using large language models [back to in-person]
- Wednesday 3/8: Akari Asai on multilingual NLP [in-person]
- Friday 3/10: I'll be giving the wrap-up lecture and leaving time for Q&A/chatting about NLP more broadly. Have questions about NLP? Bring 'em!! :)

Preview of remaining announcements for the quarter

- Remember: this Wednesday's quiz is on parsing!
- A2 grades should be out sometime late tomorrow or on Wednesday; regrade requests can be submitted (via private Ed post) up to a week after A2 grades are posted on Canvas
- We'll be holding extra office hours next week (3/6-3/10) leading up to the A3 deadline (which is 11:59pm on Friday 3/10, the last day of class)
    - Mind your late days! The number of late days you have available to use on A3 is min(3, 5 - your total late days used so far this quarter). Send us a message to ask what your total late day usage so far has been if you're not sure!
- The last quiz of the quarter, Wednesday of next week (3/8), will be on a combination of topics from Saadia, Alane, and Sewon's lectures