

Natural Language Processing

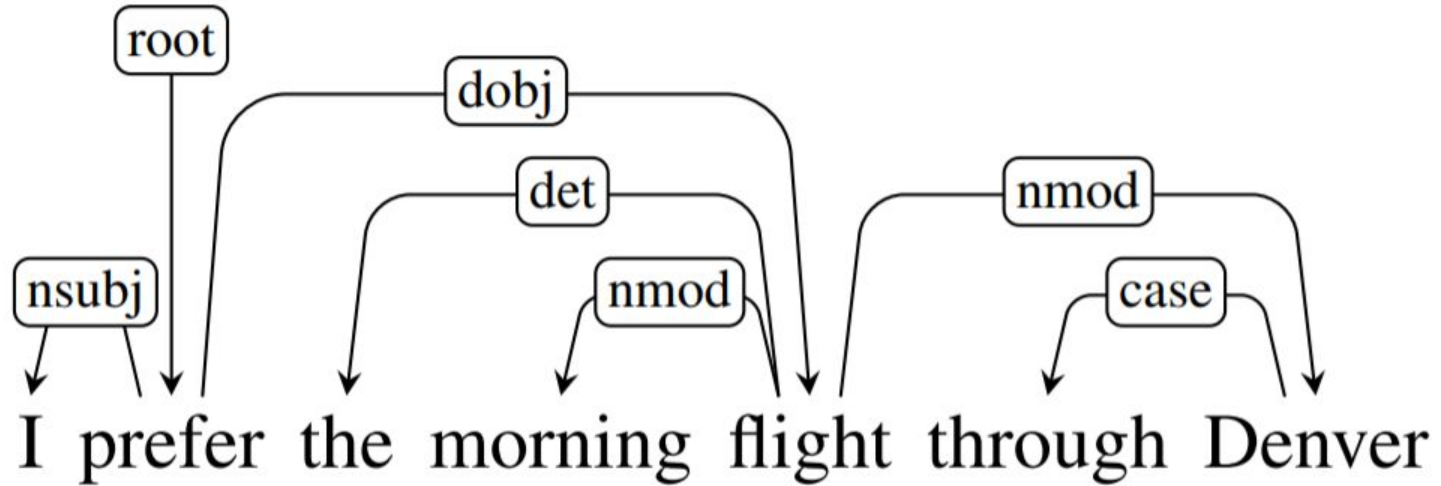
Syntactic parsing

Yulia Tsvetkov

yuliats@cs.washington.edu

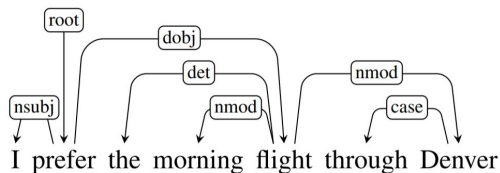


Dependency representation





Dependency representation



- A dependency structure can be defined as a directed graph G , consisting of
 - a set V of nodes – **vertices**, *words, punctuation, morphemes*
 - a set A of arcs – **directed edges**,
 - a linear precedence order $<$ on V (**word order**).
- **Labeled graphs**
 - nodes in V are labeled with word forms (and annotation).
 - arcs in A are labeled with dependency types
 - $L = \{l_1, \dots, l_{|L|}\}$ is the set of permissible arc labels;
 - Every arc in A is a triple (i, j, k) , representing a dependency from w_i to w_j with label l_k .



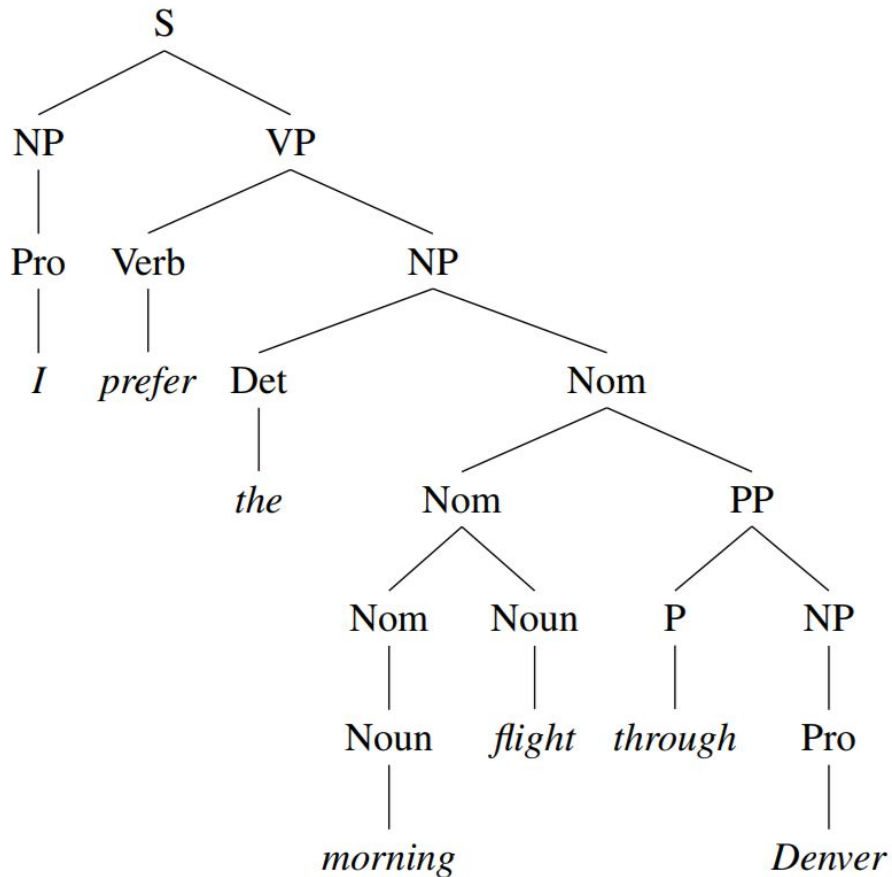
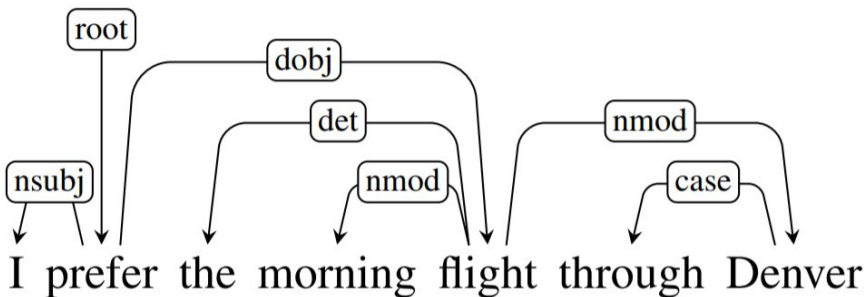
Dependency vs Constituency

- **Dependency structures explicitly represent**
 - head-dependent relations (directed arcs),
 - functional categories (arc labels)
 - possibly some structural categories (parts of speech)

- **Phrase (aka constituent) structures explicitly represent**
 - phrases (nonterminal nodes),
 - structural categories (nonterminal labels)



Dependency vs Constituency trees

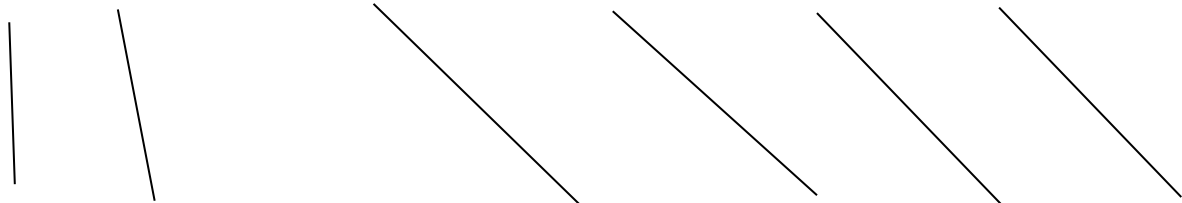




Parsing Languages with Flexible Word Order

I prefer the morning flight through Denver

Я предпочитаю утренний перелет через Денвер





Languages with free word order

I prefer the morning flight through Denver

Я предпочитаю утренний перелет через Денвер

Я предпочитаю через Денвер утренний перелет

Утренний перелет я предпочитаю через Денвер

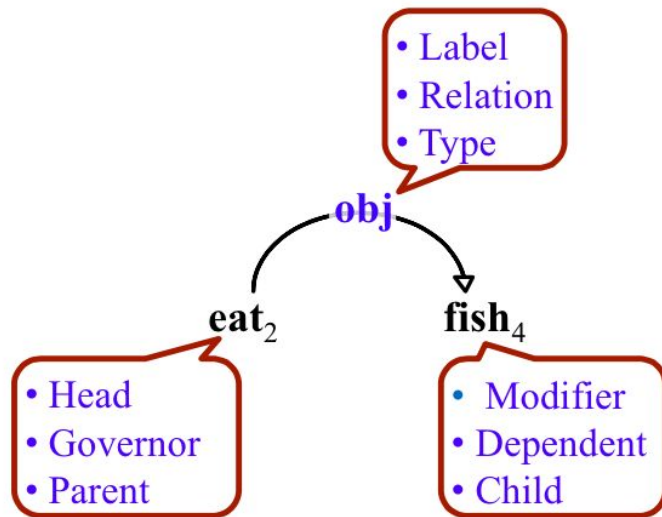
Перелет утренний я предпочитаю через Денвер

Через Денвер я предпочитаю утренний перелет

Я через Денвер предпочитаю утренний перелет

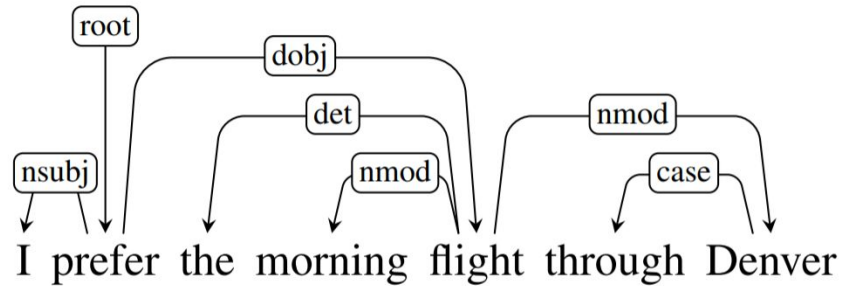


Dependency relations





Types of relationships



- The clausal relations NSUBJ and DOBJ identify the **arguments**: the subject and direct object of the predicate *cancel*
- The NMOD, DET, and CASE relations denote **modifiers** of the nouns *flights* and *Houston*.



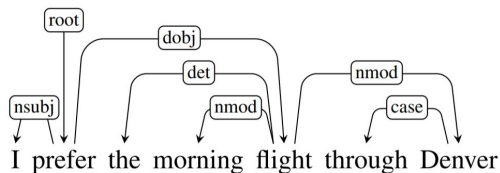
Grammatical functions

Clausal Argument Relations	Description
NSUBJ	Nominal subject
DOBJ	Direct object
IOBJ	Indirect object
CCOMP	Clausal complement
XCOMP	Open clausal complement
Nominal Modifier Relations	Description
NMOD	Nominal modifier
AMOD	Adjectival modifier
NUMMOD	Numeric modifier
APPOS	Appositional modifier
DET	Determiner
CASE	Prepositions, postpositions and other case markers
Other Notable Relations	Description
CONJ	Conjunct
CC	Coordinating conjunction

Figure 13.2 Selected dependency relations from the Universal Dependency set. (de Marneffe et al., 2014)



Dependency Constraints



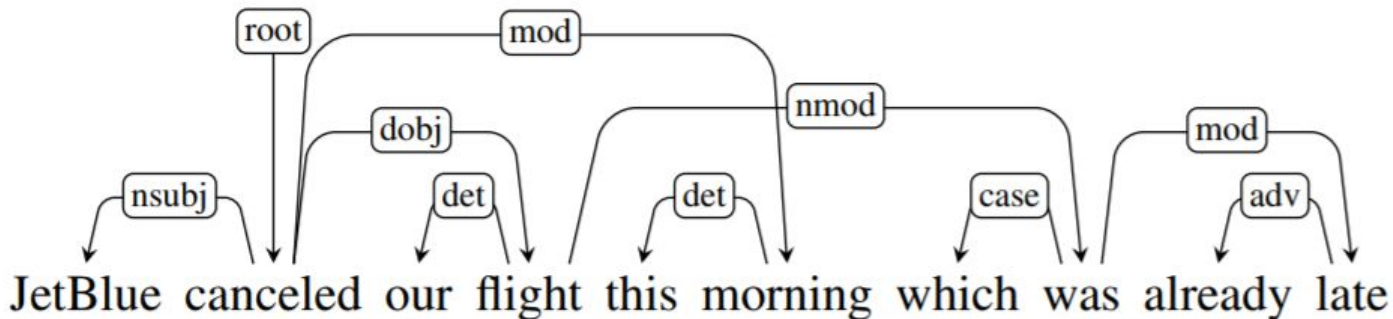
- Syntactic structure is complete (**connectedness**)
 - connectedness can be enforced by adding a special root node
- Syntactic structure is hierarchical (**acyclicity**)
 - there is a unique pass from the root to each vertex
- Every word has at most one syntactic head (**single-head constraint**)
 - except root that does not have incoming arcs

This makes the dependencies a tree



Projectivity

- Projective parse
 - arcs don't cross each other
 - mostly true for English
- Non-projective structures are needed to account for
 - long-distance dependencies
 - flexible word order





Projectivity

- Dependency grammars do not normally assume that all dependency-trees are projective, because some linguistic phenomena can only be achieved using non-projective trees.
- But a lot of parsers assume that the output trees are projective
- Reasons
 - conversion from constituency to dependency
 - the most widely used families of parsing algorithms impose projectivity



Non-Projective Statistics

Arabic: 11.2 %
Bulgarian: 5.4 %
Chinese: 0.0 %
Czech: 23.2 %
Danish: 15.6 %
Dutch: 36.4 %
German: 27.8 %
Japanese: 5.3 %
Polish: 18.9 %
Slovene: 22.2 %
Spanish 1.7 %
Swedish: 9.8 %
Turkish: 11.6 %
English: 0.0% (SD: 0.1%)

Percentage of non-projective trees for some treebanks of the CoNLL-X Shared Task and English.



Parsing problem

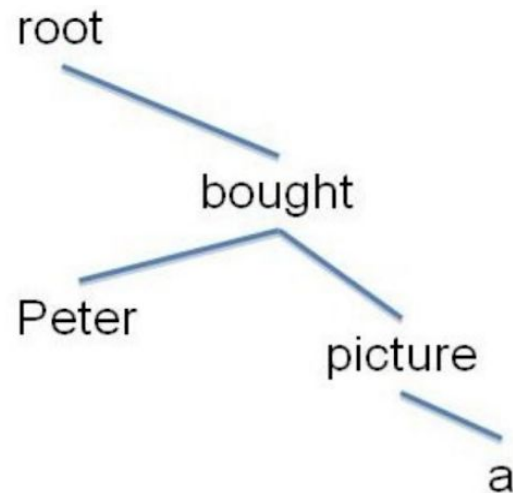
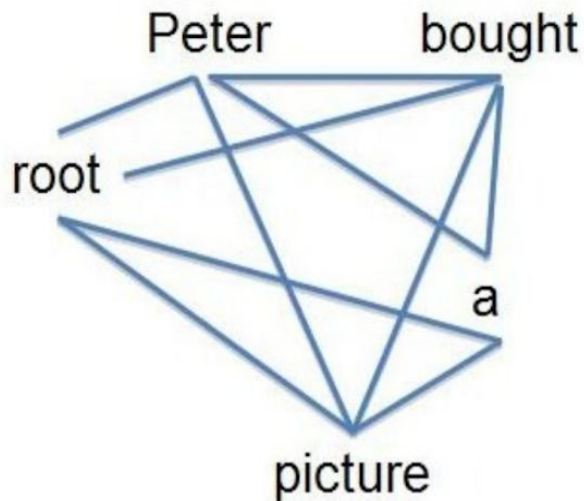
The parsing problem for a dependency parser is to find the optimal dependency tree y given an input sentence x

This amounts to assigning a syntactic head i and a label l to every node j corresponding to a word x_j in such a way that the resulting graph is a tree rooted at the node 0



Parsing problem

- This is equivalent to finding a spanning tree in the complete graph containing all possible arcs





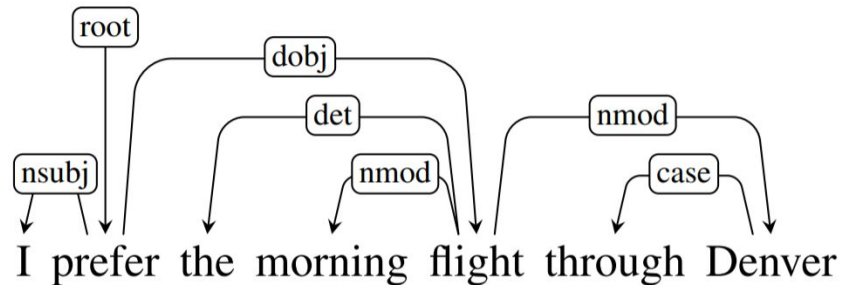
Parsing algorithms

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



Transition Based Parsing

- **greedy discriminative** dependency parser
- motivated by a stack-based approach called **shift-reduce parsing** originally developed for analyzing programming languages (Aho & Ullman, 1972).
- Nivre 2003





Configuration

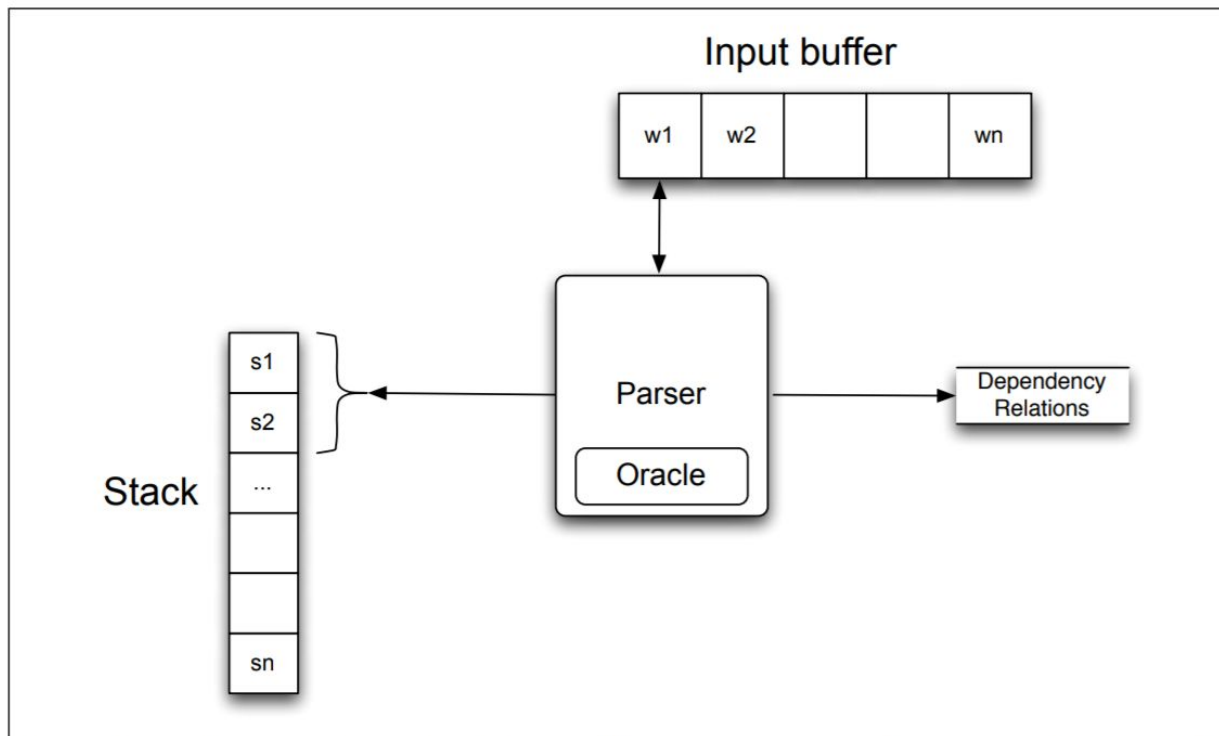
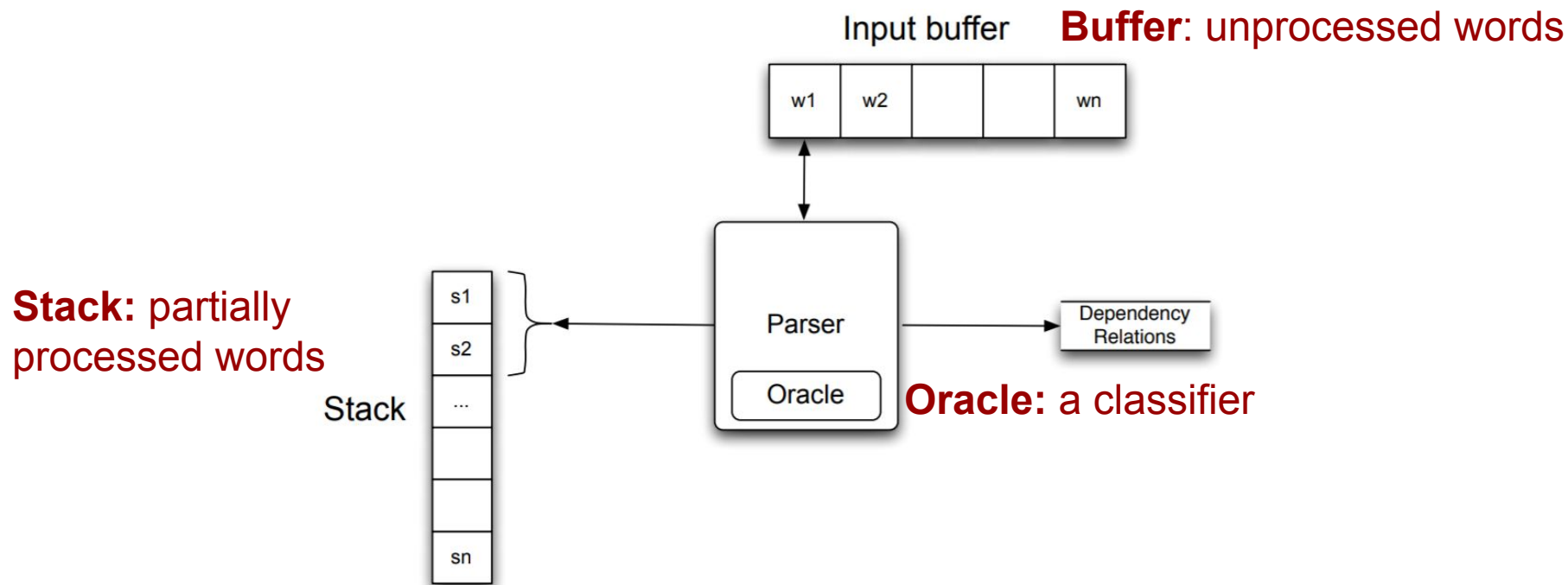


Figure 13.5 Basic transition-based parser. The parser examines the top two elements of the stack and selects an action based on consulting an oracle that examines the current configuration.

$$C = (\sigma, \beta, A)$$



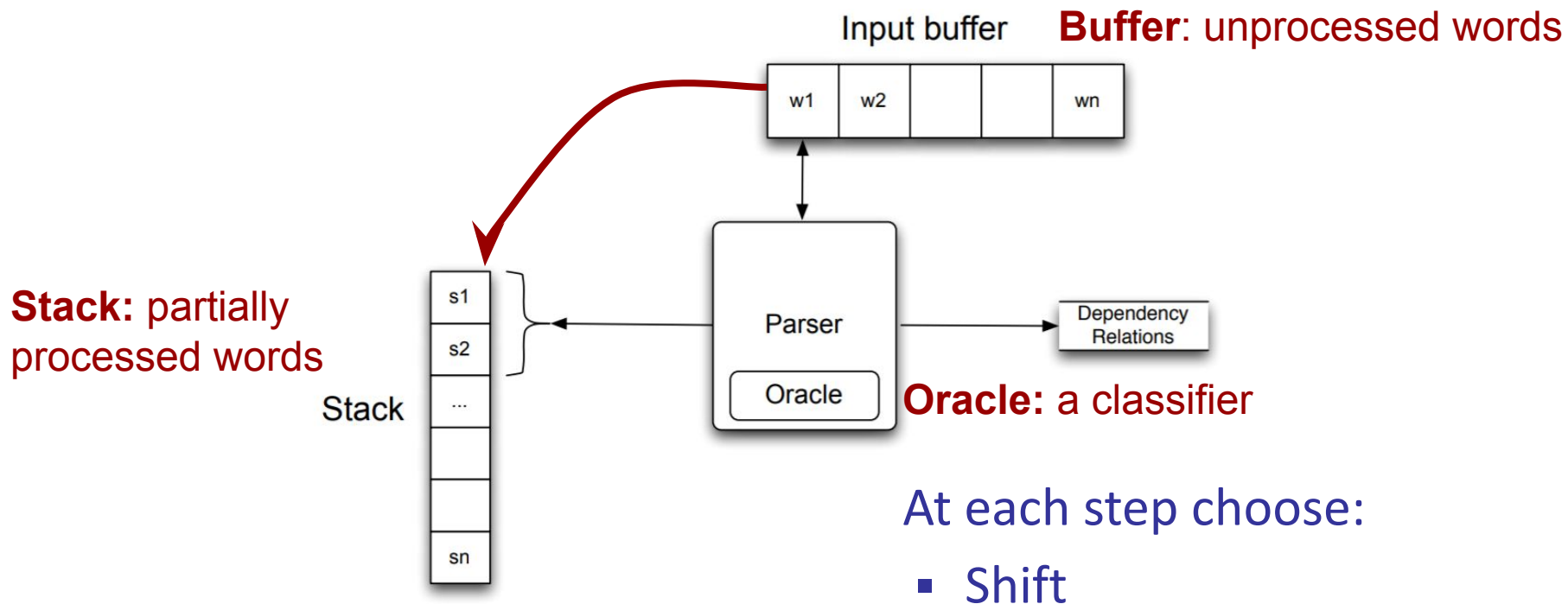
Configuration



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

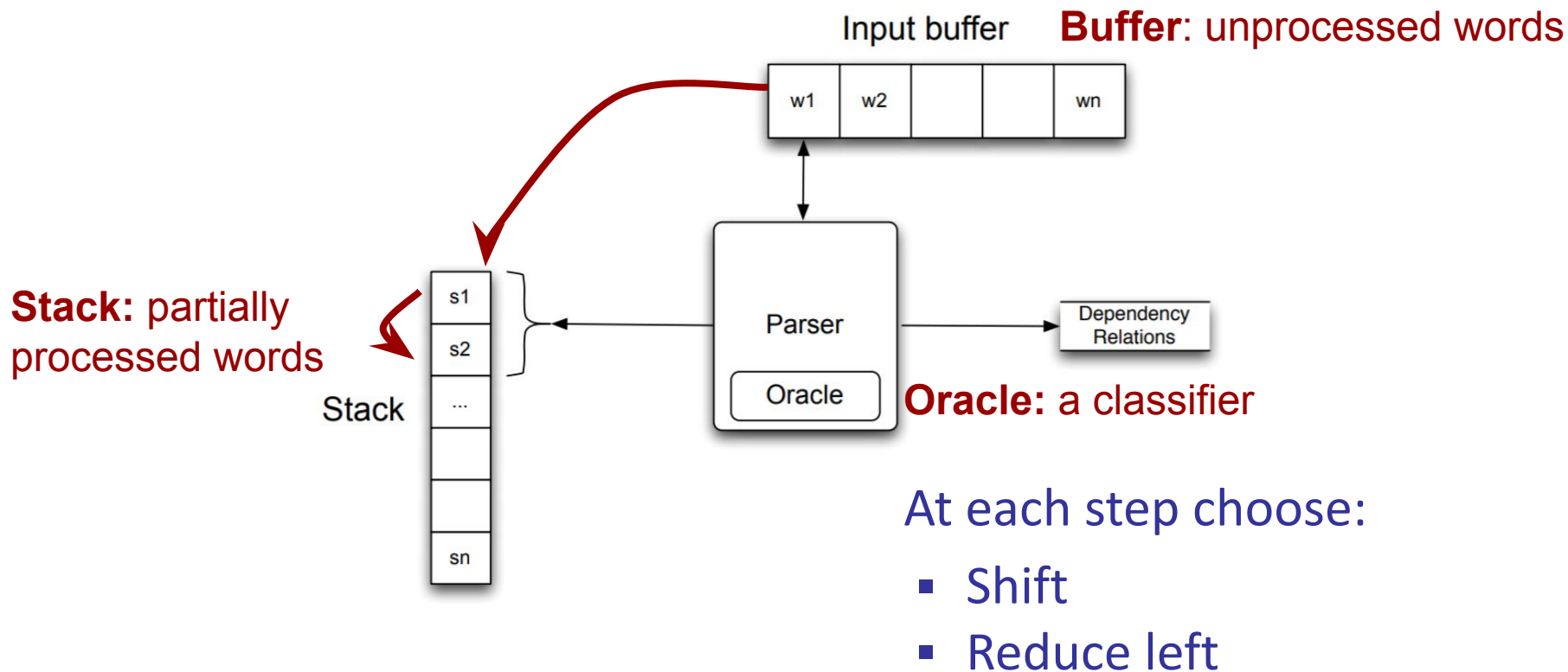


Operations



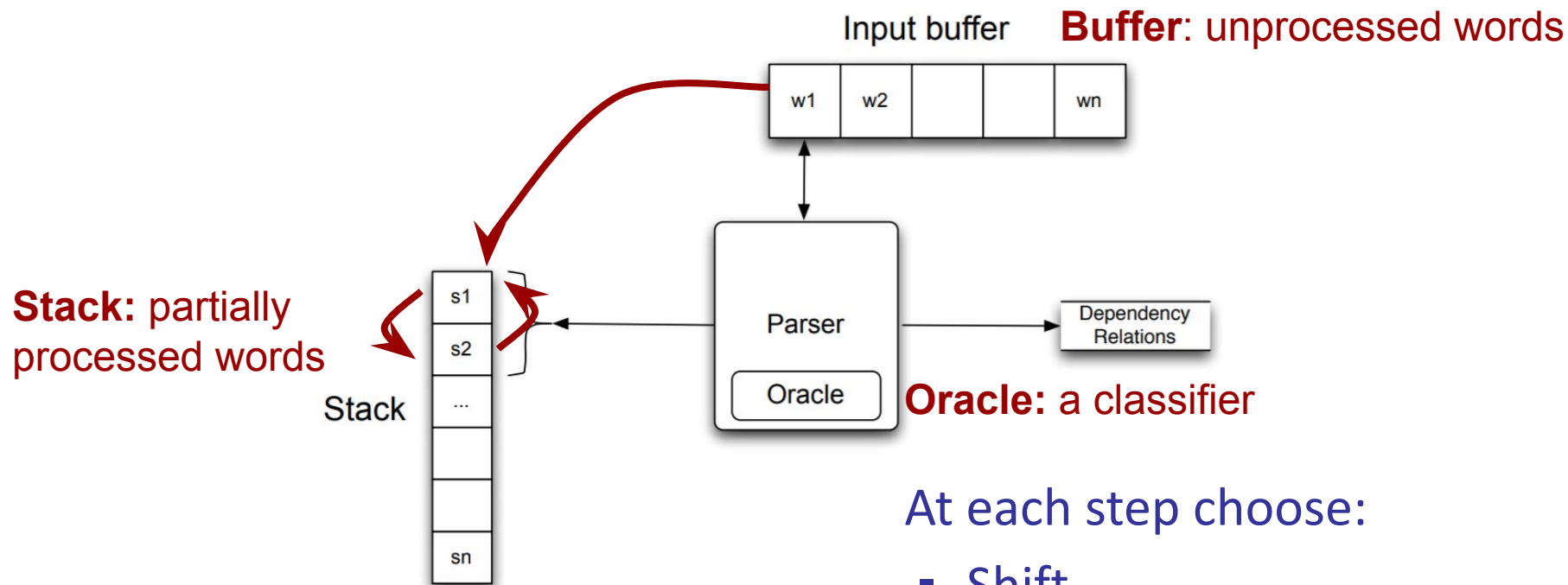


Operations





Operations



At each step choose:

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

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



Shift-Reduce Parsing

Configuration:

- Stack, Buffer, Oracle, Set of dependency relations

Operations by a classifier at each step:

- **Shift**
 - remove w_1 from the buffer, add it to the top of the stack as s_1
- **LeftArc or Reduce left**
 - assert a head-dependent relation between s_1 and s_2 ($s_1 \rightarrow s_2$)
 - remove s_2 from the stack
- **RightArc or Reduce right**
 - assert a head-dependent relation between s_2 and s_1 ($s_2 \rightarrow s_1$)
 - remove s_1 from the stack



Shift-Reduce Parsing (Arc-standard)

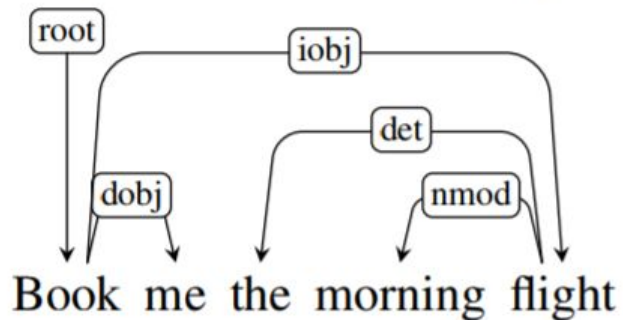
$$C_{\text{initial}} = ([\text{ROOT}], \mathbf{w}, \emptyset)$$

Book me the morning flight

Step	Stack	Word List	Action	Relation Added
------	-------	-----------	--------	----------------



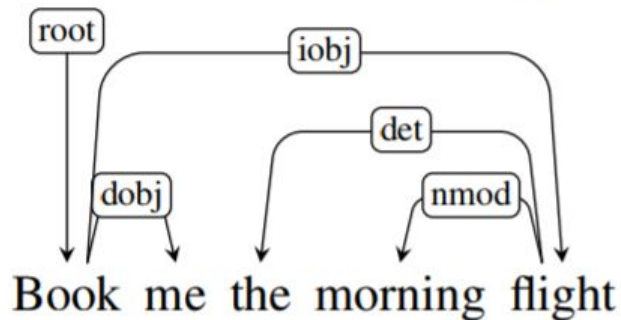
Shift-Reduce Parsing



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



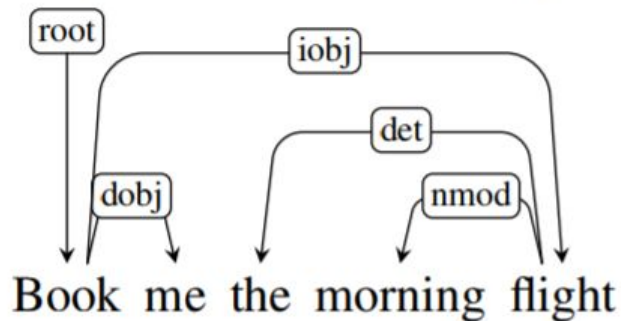
Shift-Reduce Parsing



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



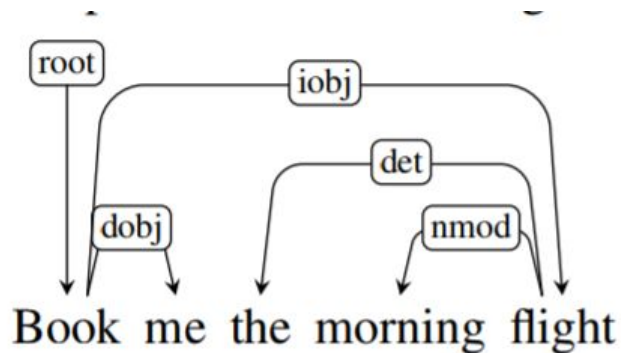
Shift-Reduce Parsing



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



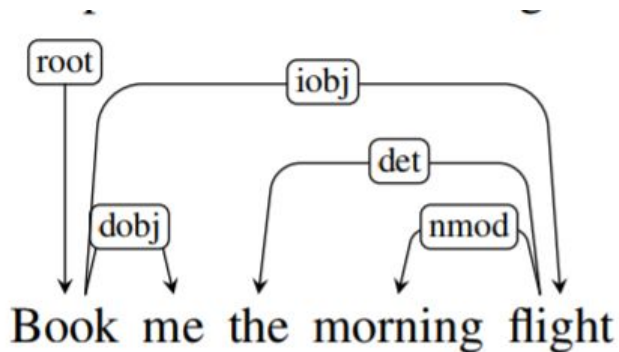
Shift-Reduce Parsing



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



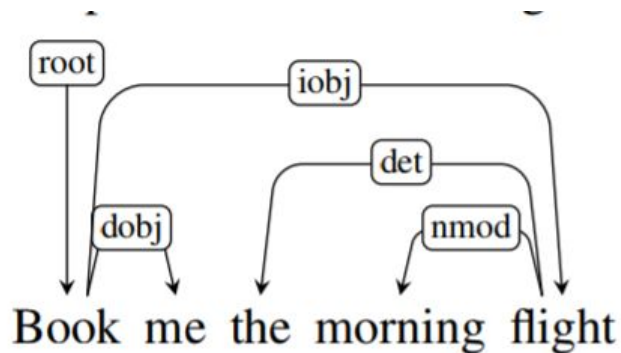
Shift-Reduce Parsing



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]		



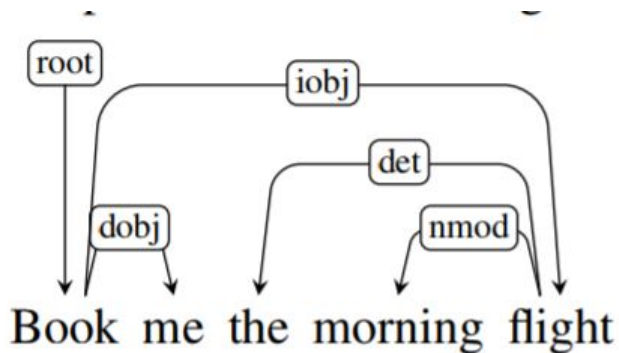
Shift-Reduce Parsing



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	



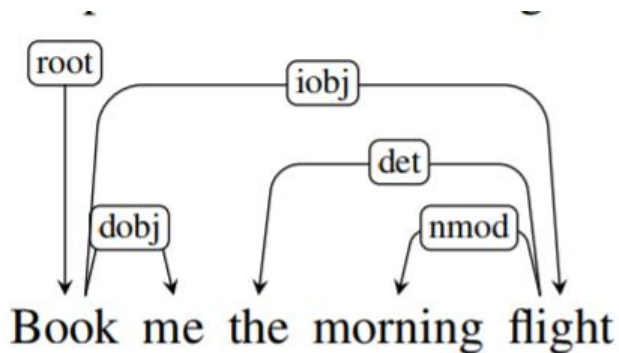
Shift-Reduce Parsing



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)



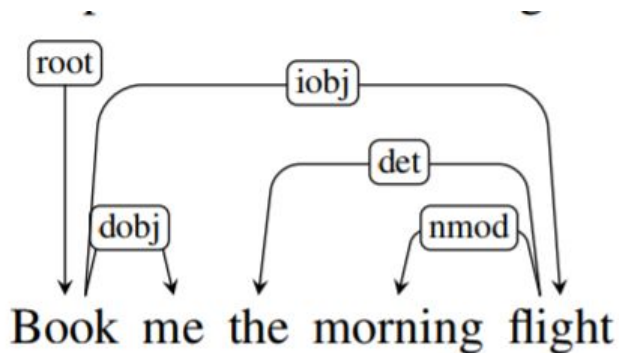
Shift-Reduce Parsing



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]		



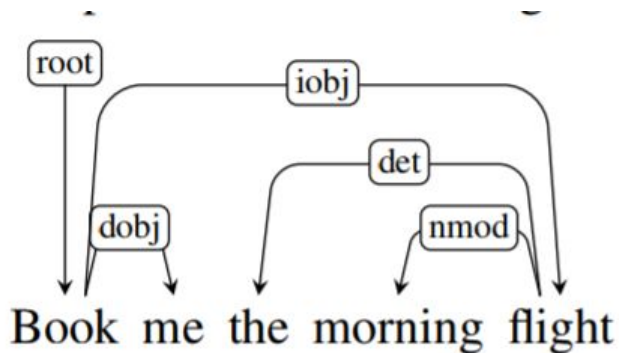
Shift-Reduce Parsing



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	



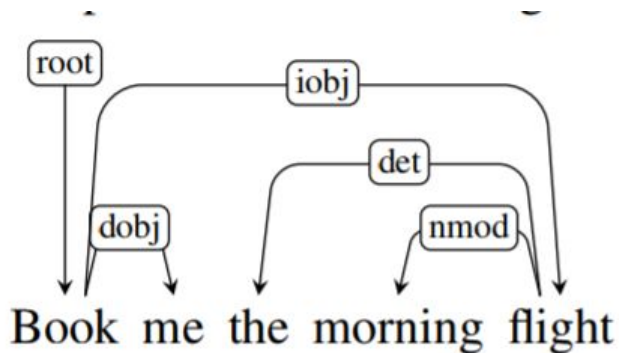
Shift-Reduce Parsing



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]		



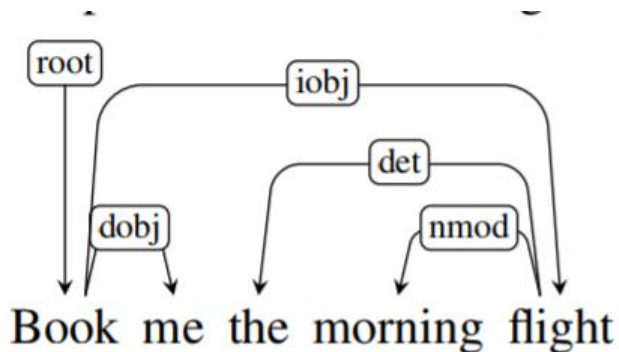
Shift-Reduce Parsing



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	



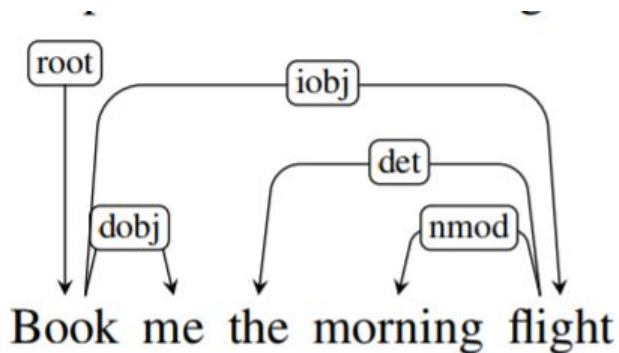
Shift-Reduce Parsing



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]		



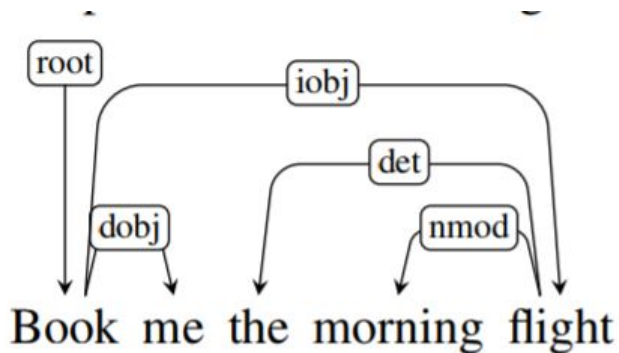
Shift-Reduce Parsing



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	



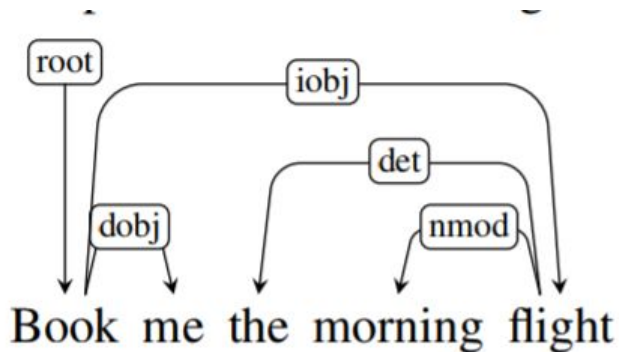
Shift-Reduce Parsing



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]	[]		



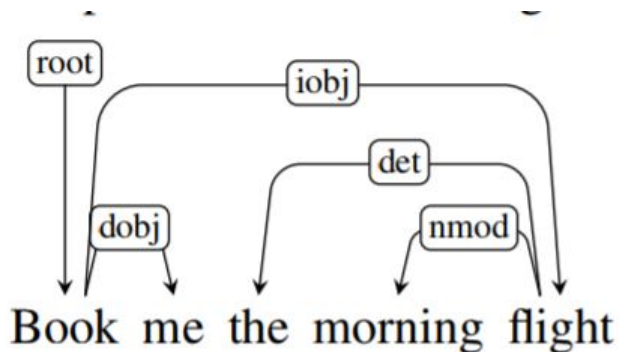
Shift-Reduce Parsing



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)



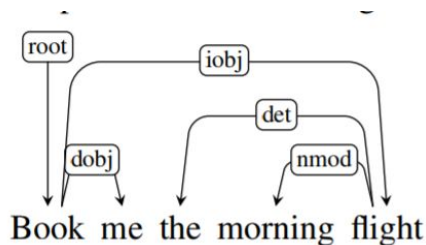
Shift-Reduce Parsing



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)



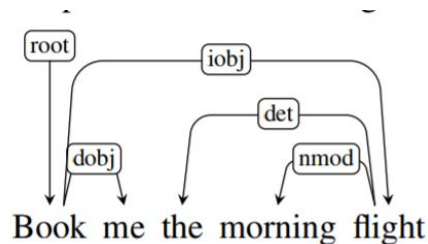
Shift-Reduce Parsing



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)



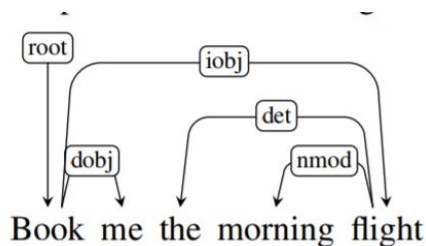
Shift-Reduce Parsing



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



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

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	



Shift-Reduce Parsing

Configuration:

- Stack, Buffer, Oracle, Set of dependency relations

Operations by a classifier at each step:

Complexity?

- Shift

- remove w_1 from the buffer, add it to the top of the stack as s_1

- LeftArc or Reduce left

- assert a head-dependent relation between
 - remove s_2 from the stack

Oracle decisions can correspond to unlabeled or labeled arcs

- RightArc or Reduce right

- assert a head-dependent relation between s_2 and s_1
 - remove s_1 from the stack



Training an Oracle

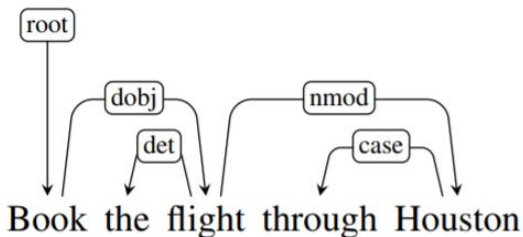
- Oracle is a supervised classifier that learns a function from the configuration to the next operation
- How to extract the training set?



Training an Oracle

- How to extract the training set?

- if LeftArc \rightarrow LeftArc
- if RightArc
 - if s1 dependents have been processed \rightarrow RightArc
- else \rightarrow Shift

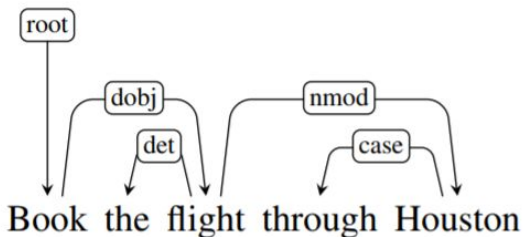


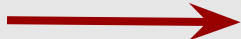


Training an Oracle

How to extract the training set?

- if LeftArc \rightarrow LeftArc
- if RightArc
 - if s1 dependents have been processed \rightarrow RightArc
- else \rightarrow 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



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?
 - if LeftArc \rightarrow LeftArc
 - if RightArc
 - if s1 dependents have been processed \rightarrow RightArc
 - else \rightarrow 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	$s_1.w \circ s_2.w$	$s_1.t \circ s_2.t$	$s_1.t \circ b_1.w$
	$s_1.t \circ s_2.wt$	$s_1.w \circ s_2.w \circ s_2.t$	$s_1.w \circ s_1.t \circ s_2.t$
	$s_1.w \circ s_1.t \circ s_2.t$	$s_1.w \circ s_1.t$	

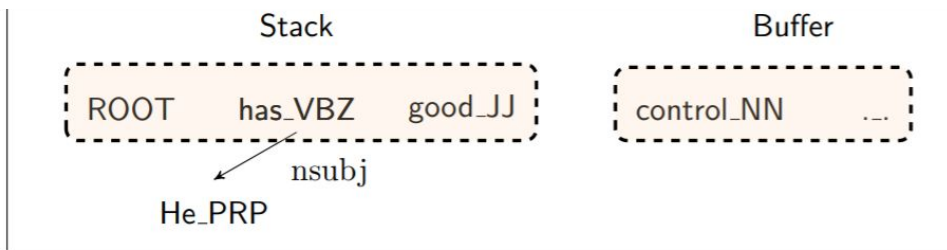


Learning

- Before 2014: SVMs,
- After 2014: Neural Nets



Chen & Manning 2014



binary, sparse
dim = $10^6 \sim 10^7$



Indicator
features

$$s_2.w = \text{has} \wedge s_2.t = \text{VBZ}$$

$$s_1.w = \text{good} \wedge s_1.t = \text{JJ} \wedge b_1.w = \text{control}$$

$$lc(s_2).t = \text{PRP} \wedge s_2.t = \text{VBZ} \wedge s_1.t = \text{JJ}$$

$$lc(s_2).w = \text{He} \wedge lc(s_2).l = \text{nsubj} \wedge s_2.w = \text{has}$$



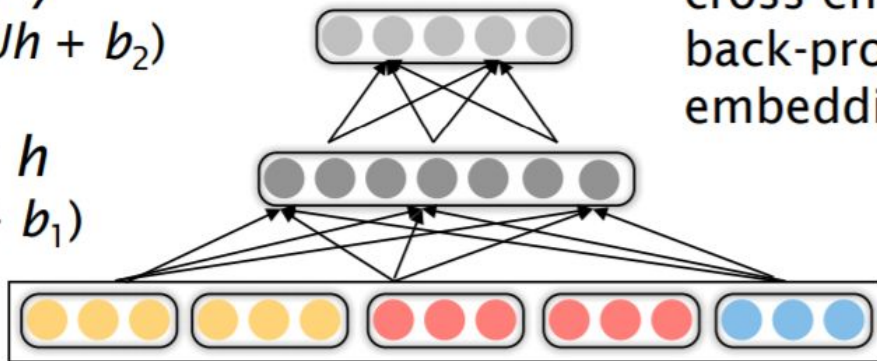
Chen & Manning 2014

Softmax probabilities

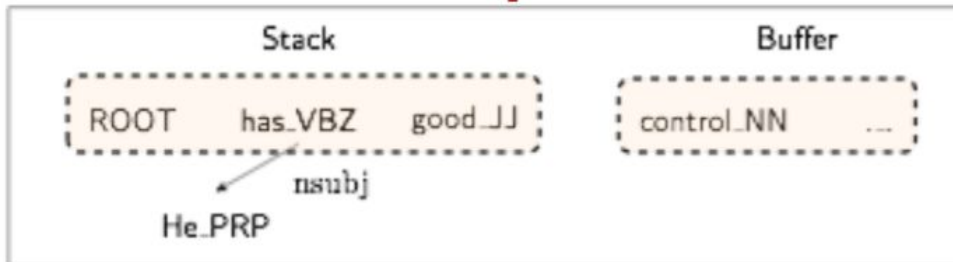
Output layer y
 $y = \text{softmax}(Uh + b_2)$

Hidden layer h
 $h = \text{ReLU}(Wx + b_1)$

Input layer x
lookup + concat



cross-entropy error will be back-propagated to the embeddings.

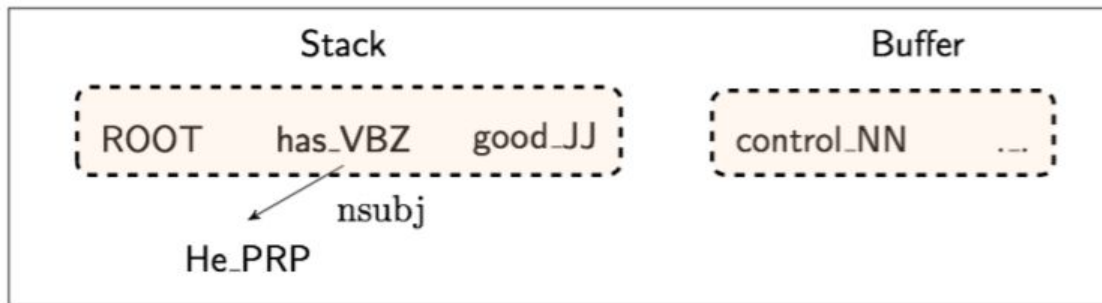




Chen & Manning 2014

■ 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.
s1	good	JJ	∅
s2	has	VBZ	∅
b1	control	NN	∅
lc(s1)	∅	+	∅
rc(s1)	∅	+	∅
lc(s2)	He	PRP	nsubj
rc(s2)	∅	∅	∅



Evaluation of Dependency Parsers

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

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



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)

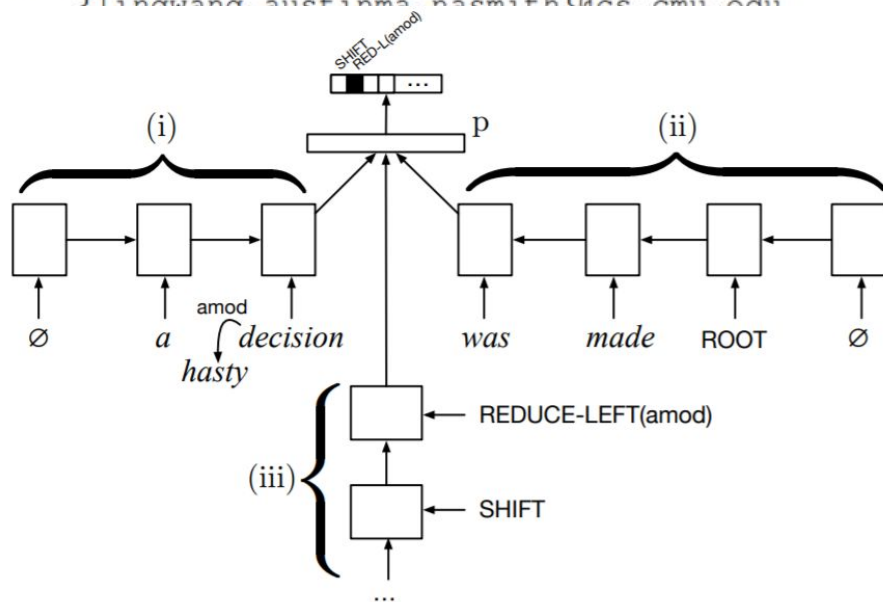
Transition-Based Dependency Parsing with Stack Long Short-Term Memory

Chris Dyer^{♣♣} Miguel Ballesteros^{◇♣} Wang Ling[♣] Austin Matthews[♣] Noah A. Smith[♣]

^{♣♣}Marianas Labs [◇]NLP Group, Pompeu Fabra University [♣]Carnegie Mellon University

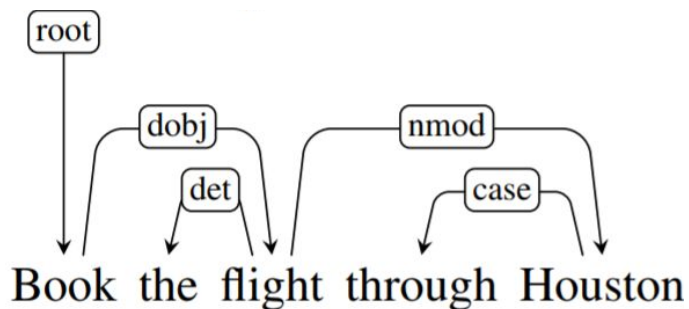
chris@marianaslabs.com, miguel.ballesteros@upf.edu,

flingwang, austinma, nasmith@cs.cmu.edu





Arc-Eager version



- LEFTARC: Assert a head-dependent relation between s_1 and b_1 ; pop the stack.
- RIGHTARC: Assert a head-dependent relation between s_1 and b_1 ; shift b_1 to be s_1 .
- SHIFT: Remove b_1 and push it to be s_1 .
- 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