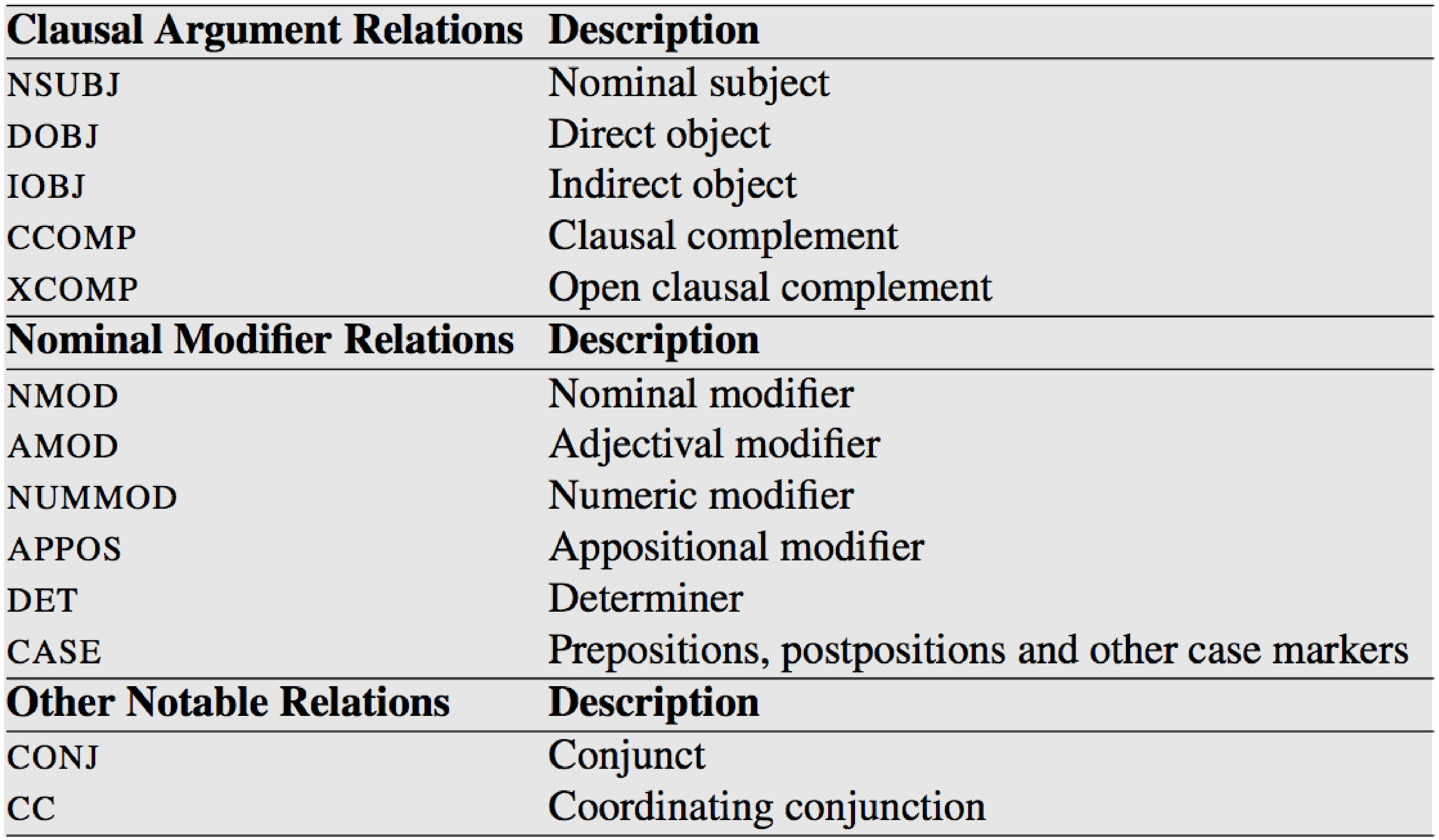
Dependency Parsing

Sudeshna Sarkar

21 AUG 2019

Dependency Relations
(A sample from UDEP)



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

Relation	Examples with *head* and **dependent**
NSUBJ	**United** *canceled* the flight.
DOBJ	United *diverted* the **flight** to Reno.
	We *booked* her the first **flight** to Miami.
IOBJ	We *booked* **her** the flight to Miami.
NMOD	We took the **morning** *flight*.
AMOD	Book the **cheapest** *flight*.
NUMMOD	Before the storm JetBlue canceled **1000** *flights*.
APPOS	*United*, a **unit** of UAL, matched the fares.
DET	**The** *flight* was canceled.
	Which *flight* was delayed?
CONJ	We *flew* to Denver and **drove** to Steamboat.
CC	We flew to Denver **and** *drove* to Steamboat.
CASE	Book the flight **through** *Houston*.

Figure 14.3 Examples of core Universal Dependency relations.

Dependency Parsing

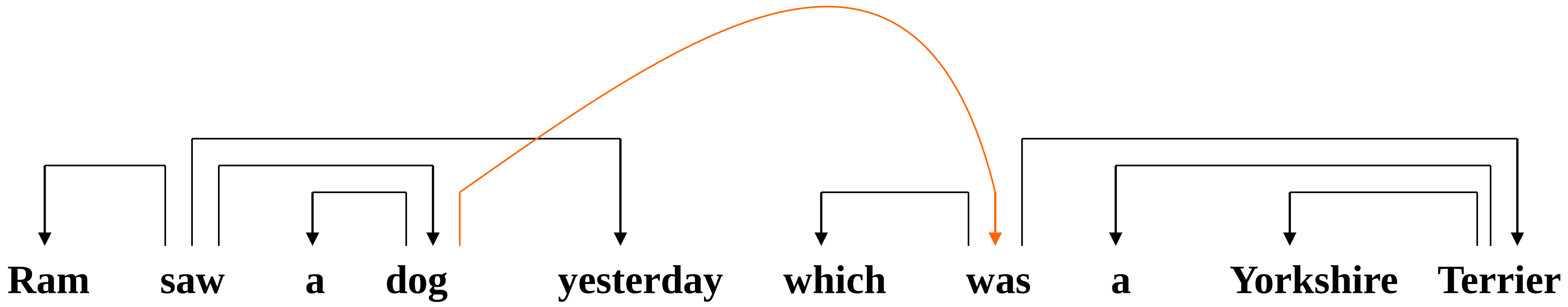
- Given an input sentence, draw edges between pairs of words, and label them.
  - Result should be a tree.
  - The edge-labels between word pairs should convey the correct syntactic relation.

1. Could construct a tree one edge at a time.
   - Transition parsing
2. Could construct a fully connected tree, and prune it.
   - Graph-based methods

Definition: dependency graph

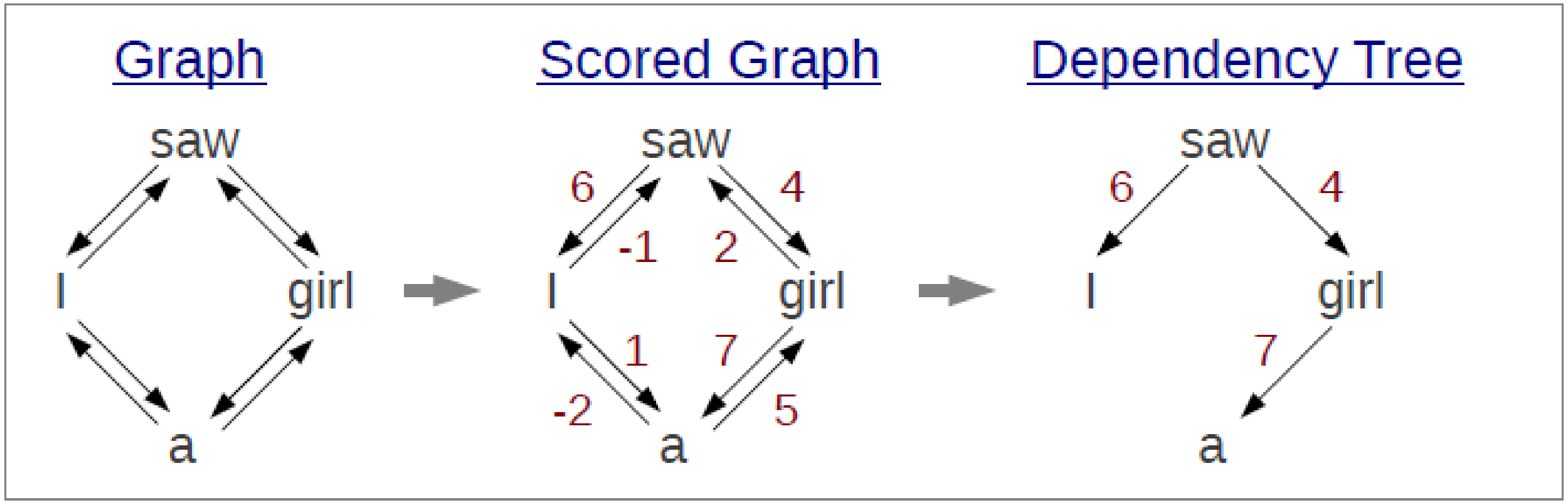
- A dependency graph is well-formed iff
  - **Single head**: Each word has only one head
  - **Acyclic**: The graph should be acyclic
  - **Connected**: The graph should be a single tree with all the words in the sentence
  - Projective: If word A depends on word B, then all words between A and B are also subordinate to B (i.e. dominated by B)

Non-projective dependencies



Maximum Spanning Tree

- Each dependency is an edge in a directed graph
- Assign each edge a score (with machine learning)
- Keep the tree with the highest score



(Chu-Liu-Edmonds Algorithm)

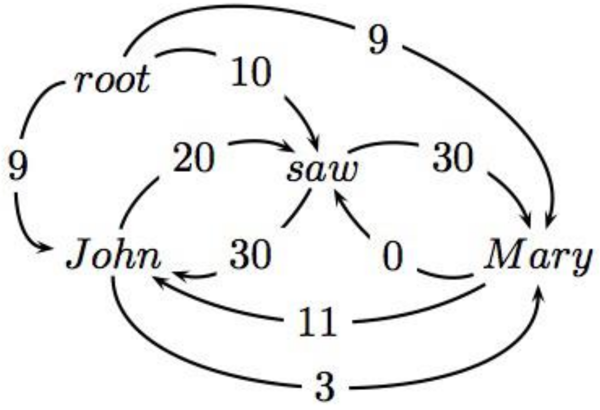
Graph-based parsing

- Assume there is a scoring function:

$$s : V \times V \times L \rightarrow R$$

- The score of a graph is

$$s(G=(V,E)) = \sum_{(i,j)\in E} s(i,j)$$



- Parsing for input string *x* is

$$G^* = \text{argmax}_{G\in D(G_x)} \sum_{(i,j)\in E} s(i,j)$$

All
dependenc
y graphs

MST algorithm (McDonald, 2006)

- Scores are based on features, independent of other dependencies

$$s(i,j) = w \times f(i,j)$$

- Features can be
  - Head and dependent word and POS separately
  - Head and dependent word and POS bigram features
  - Words between head and dependent
  - Length and direction of dependency
- Parsing can be formulated as **maximum spanning tree** problem
- Use Chu-Liu-Edmonds (CLE) algorithm for MST
- Uses online learning for determining weight vector *w*

Neural Network Graph Parser

- Biaffine Attention Model (Dozat&Manning)

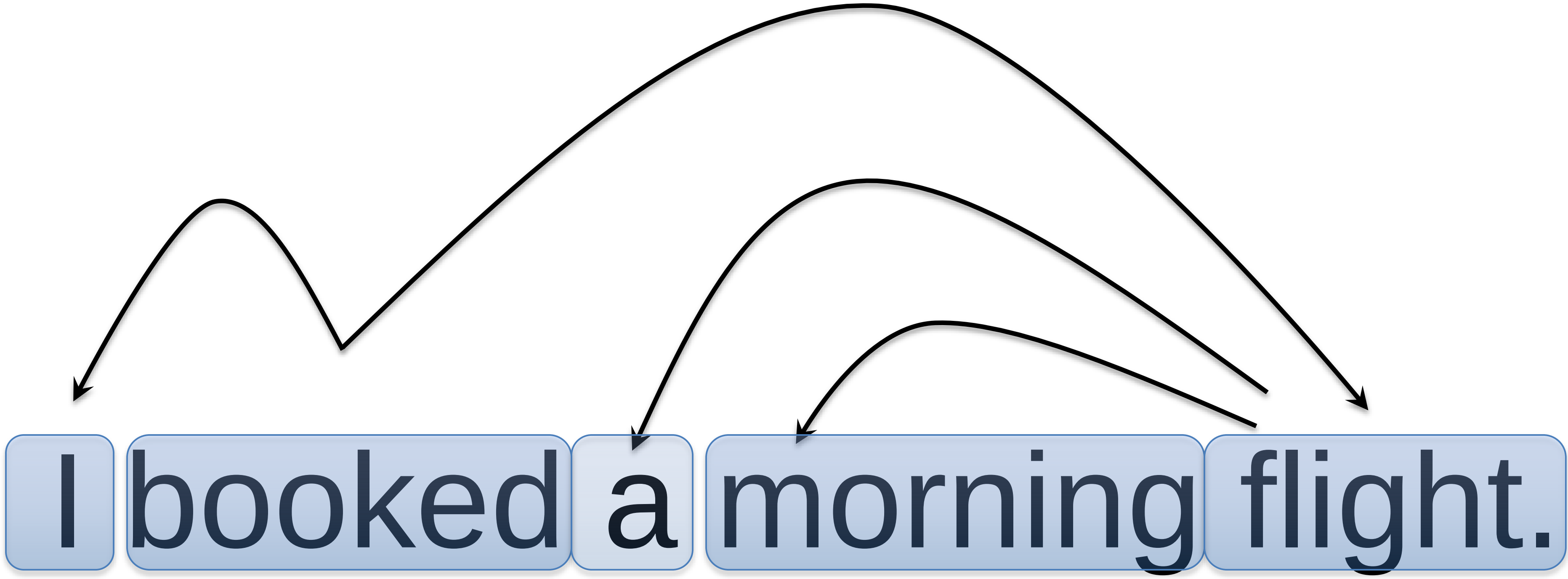
  http://aclweb.org/anthology/K18-2016

- Revived graph-based dependency parsing in a neural world
  - Design a biaffine scoring model for neural dependency parsing
  - Uses a neural sequence model
- Great results!
  - But slower than simple neural transition-based parsers
  - There are n^2 possible dependencies in a sentence of length n

Transition-Based Parsing

- Transition-based parsing is a greedy word-by-word approach to parsing
  - A single dependency tree is built up an arc at a time as we move left to right through a sentence
  - No backtracking
  - ML-based classifiers are used to make decisions as we move through the sentence

Dependency Parse



Transition-Based Parsing

- A state consists of three elements
  - A stack representing partially processed words
  - A buffer/list containing the remaining words to be processed
  - A set/arcs containing the relations discovered so far

- Makes arc decisions about entries in the top of the stack and buffer.
- Keeps shifting words from the buffer until all words are consumed.

Transition-based dependency parsing

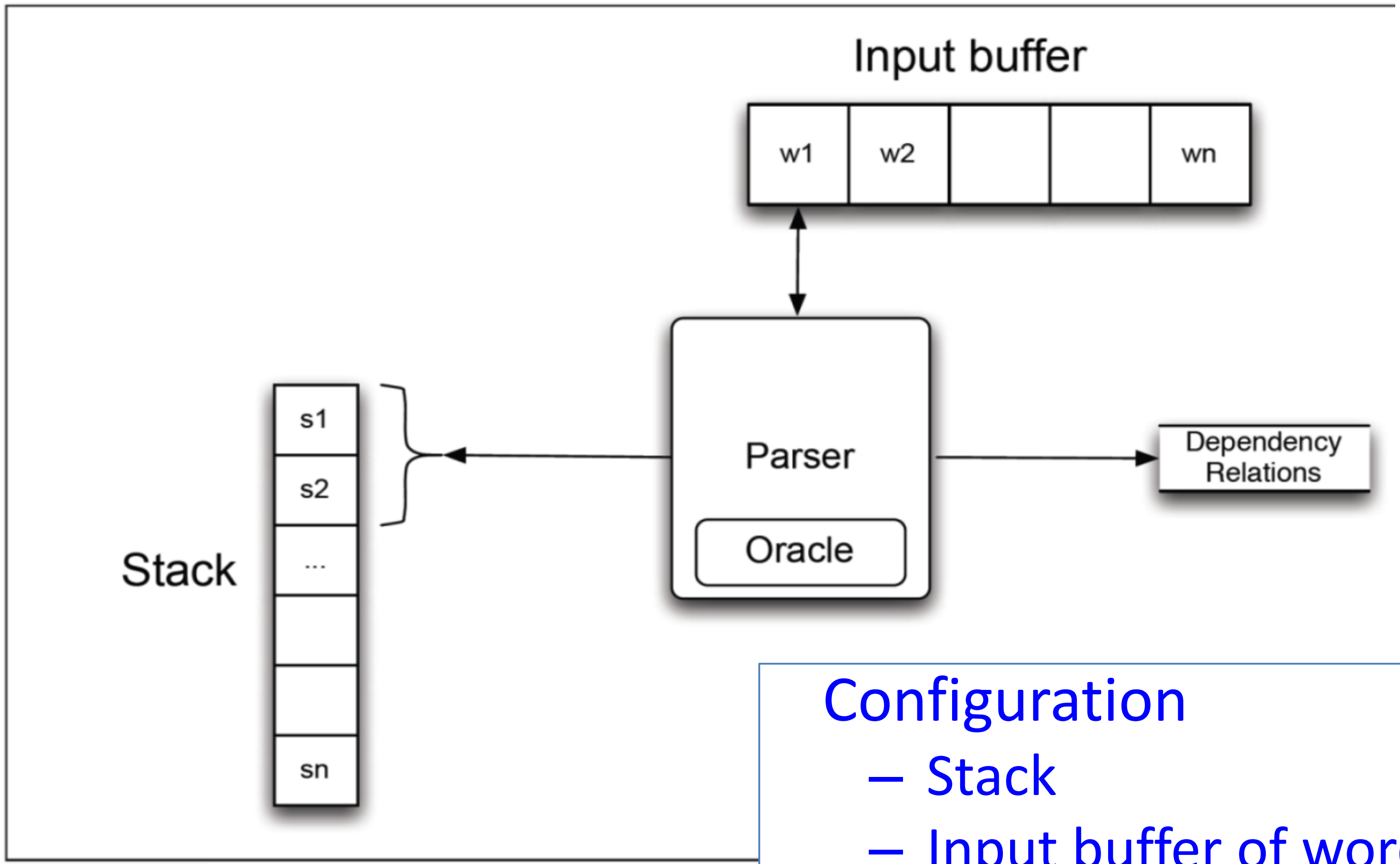


Figure 14.5 Basic transition-based parser. Th

- Configuration
  - Stack
  - Input buffer of words
  - Set of dependency relations

Arc Standard Transition System

Defines 3 transition operators

- LEFT-ARC
  - create head-dependent rel. between word at top of stack and 2ndword (under top)
  - remove 2ndword from stack
- RIGHT-ARC:
  - Create head-dependent rel. between word on 2nd word on stack and word on top
  - Remove word at top of stack
- SHIFT
  - Remove word at head of input buffer
  - Push it on the stack

Transition Based Dependency Parser

Start: $\sigma = [\text{ROOT}]$, $\beta = w_1, \ldots, w_n$, $A = \emptyset$

1. Shift $\sigma, w_i|\beta, A \rightarrow \sigma|w_i, \beta, A$
2. Left-Arc$_r$ $\sigma|w_i|w_j, \beta, A \rightarrow \sigma|w_j, \beta, A \cup \{r(w_j,w_i)\}$
3. Right-Arc$_r$ $\sigma|w_i|w_j, \beta, A \rightarrow \sigma|w_i, \beta, A \cup \{r(w_i,w_j)\}$

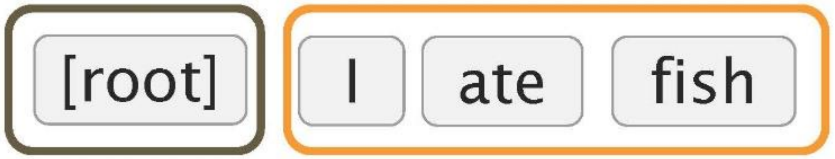
Finish: $\sigma = [w]$, $\beta = \emptyset$

Arc-standard transition-based parser

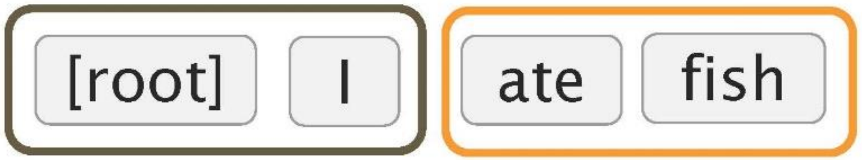
(there are other transition schemes …)

Analysis of "I ate fish"

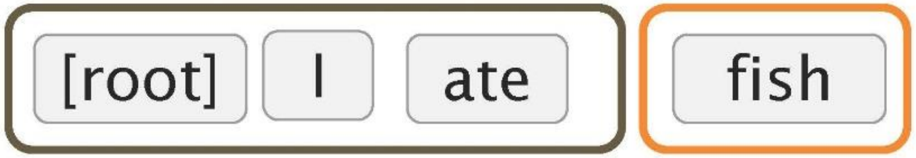
Start



Shift



Shift



Start: σ = [ROOT], $\beta = w_1, \ldots, w_n$, $A = \emptyset$

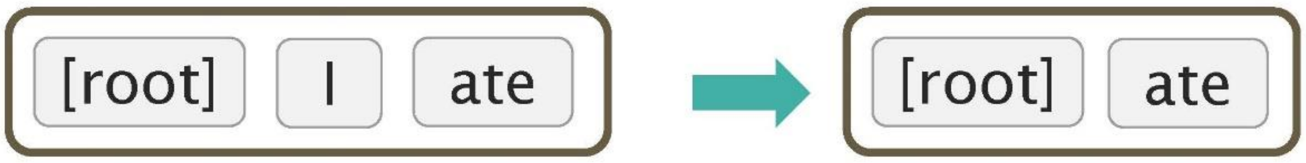
1. Shift $\quad \sigma, w_i|\beta, A \rightarrow \sigma|w_i, \beta, A$
2. Left-Arc$_r$ $\quad \sigma|w_i|w_j, \beta, A \rightarrow \sigma|w_j, \beta, A \cup \{r(w_j, w_i)\}$
3. Right-Arc$_r$ $\quad \sigma|w_i|w_j, \beta, A \rightarrow \sigma|w_i, \beta, A \cup \{r(w_i, w_j)\}$

Finish: $\beta = \emptyset$

Arc-standard transition-based parser

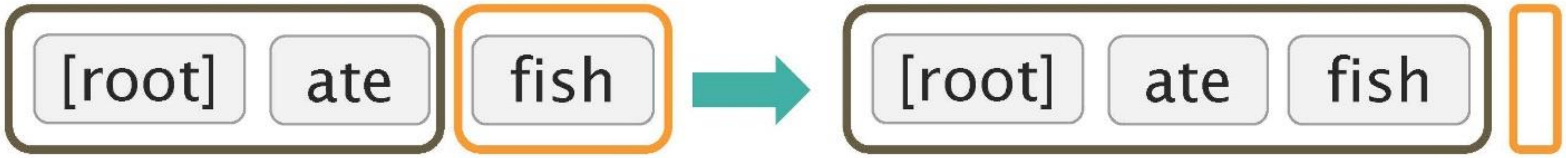
Analysis of “I ate fish”

Left Arc

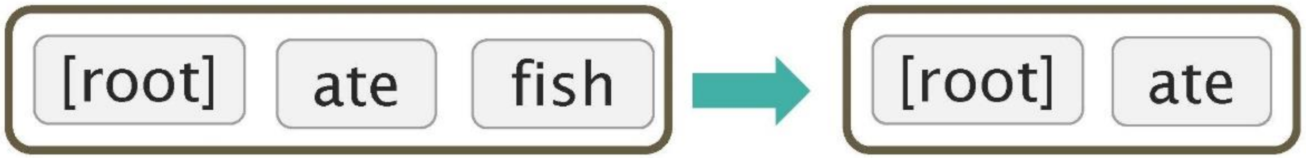


A += nsubj(ate → I)

Shift

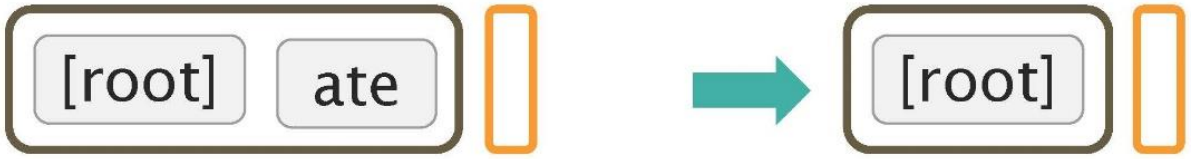


Right Arc



A += obj(ate → fish)

Right Arc



A += root([root] → ate)

Finish