CS5740: Natural Language Processing
Spring 2018

Dependency Parsing

Instructor: Yoav Artzi

Slides adapted from Dan Klein, Luke Zettlemoyer, Chris Manning, and Dan Jurafsky, and David Weiss

Overview

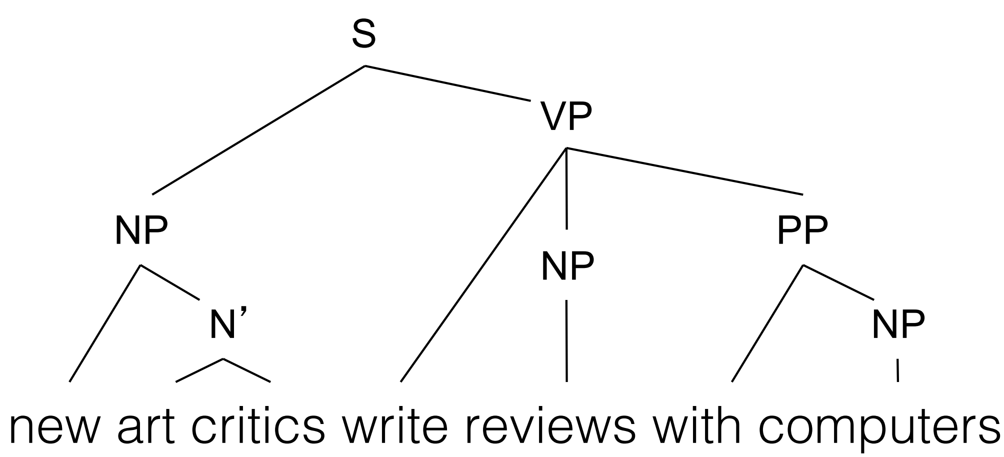
- The parsing problem
- Methods
  - Transition-based parsing
- Evaluation
- Projectivity

Parse Trees

- Part-of-speech Tagging:
  - Word classes
- Parsing:
  - From words to phrases to sentences
  - Relations between words
- Two views
  - Constituency
  - Dependency

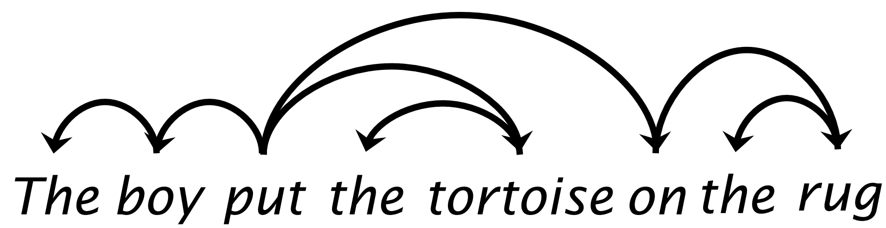
Constituency (Phrase Structure) Parsing

- Phrase structure organizes words into nested constituents
- Linguists can, and do, argue about details
- Lots of ambiguity



Dependency Parsing

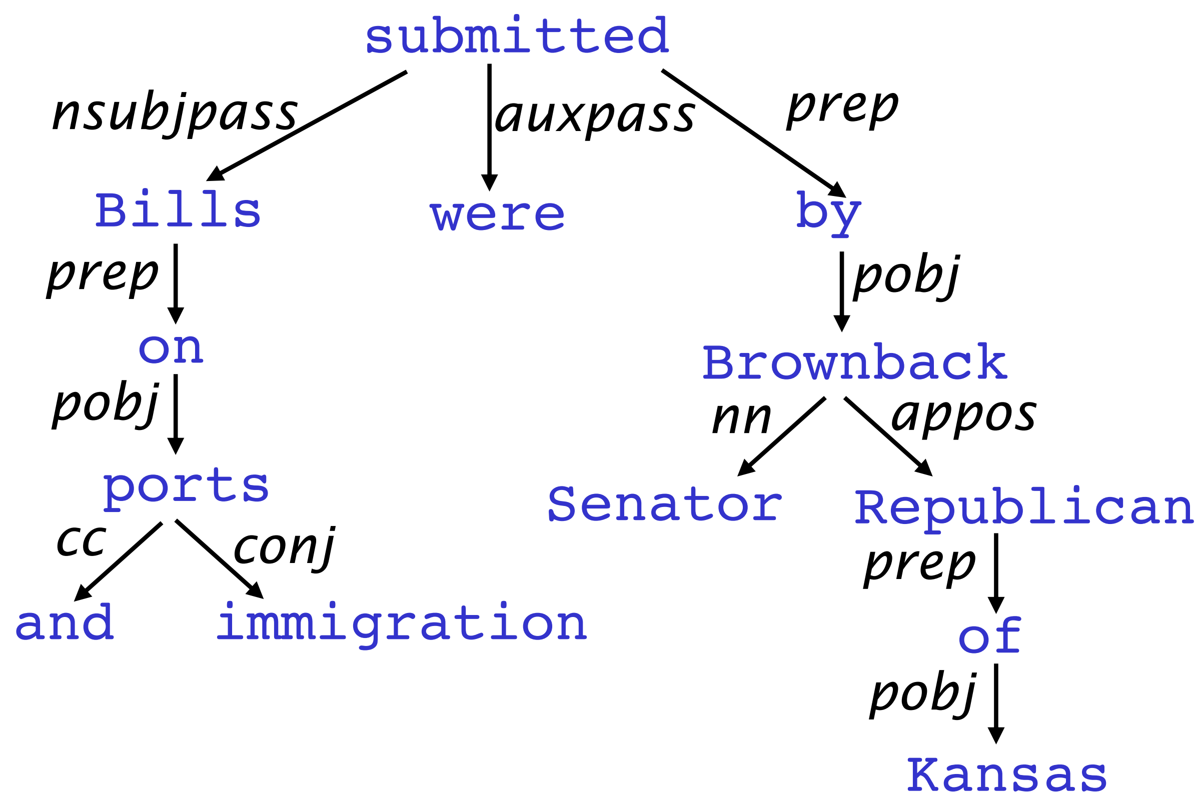
- Dependency structure shows which words depend on (modify or are arguments of) which other words.



Dependency Structure

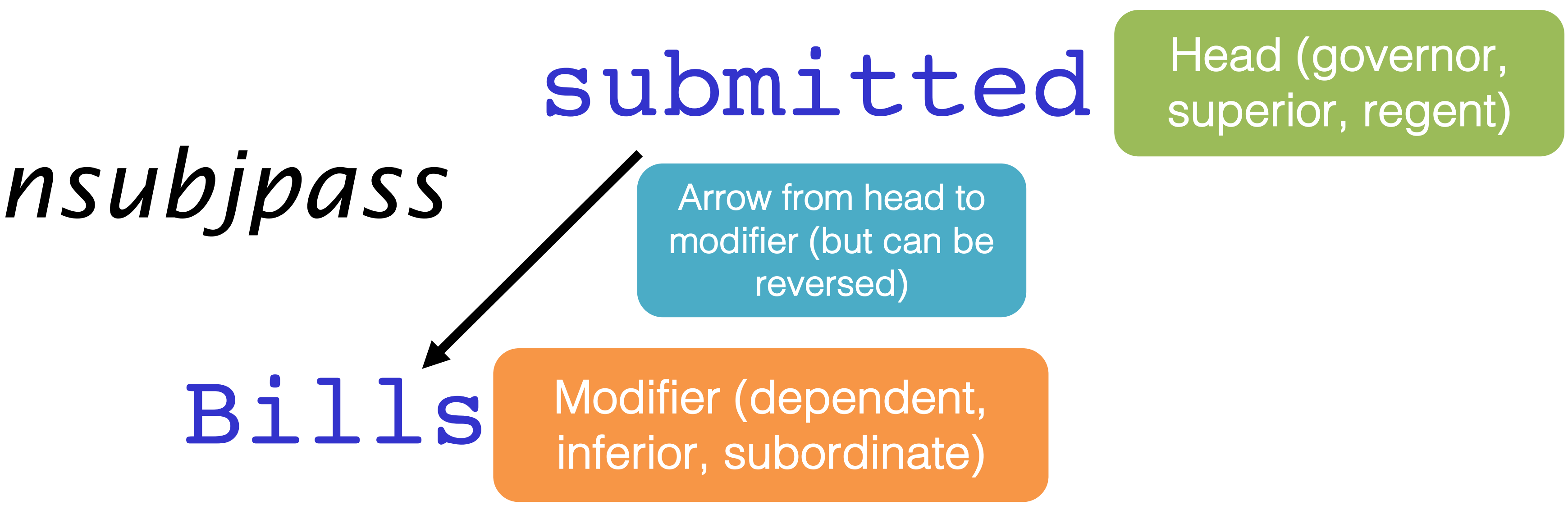
- Syntactic structure consists of:
  - Lexical items
  - Binary asymmetric relations →dependencies

Dependencies are typed with name of grammatical relation



Dependency Structure

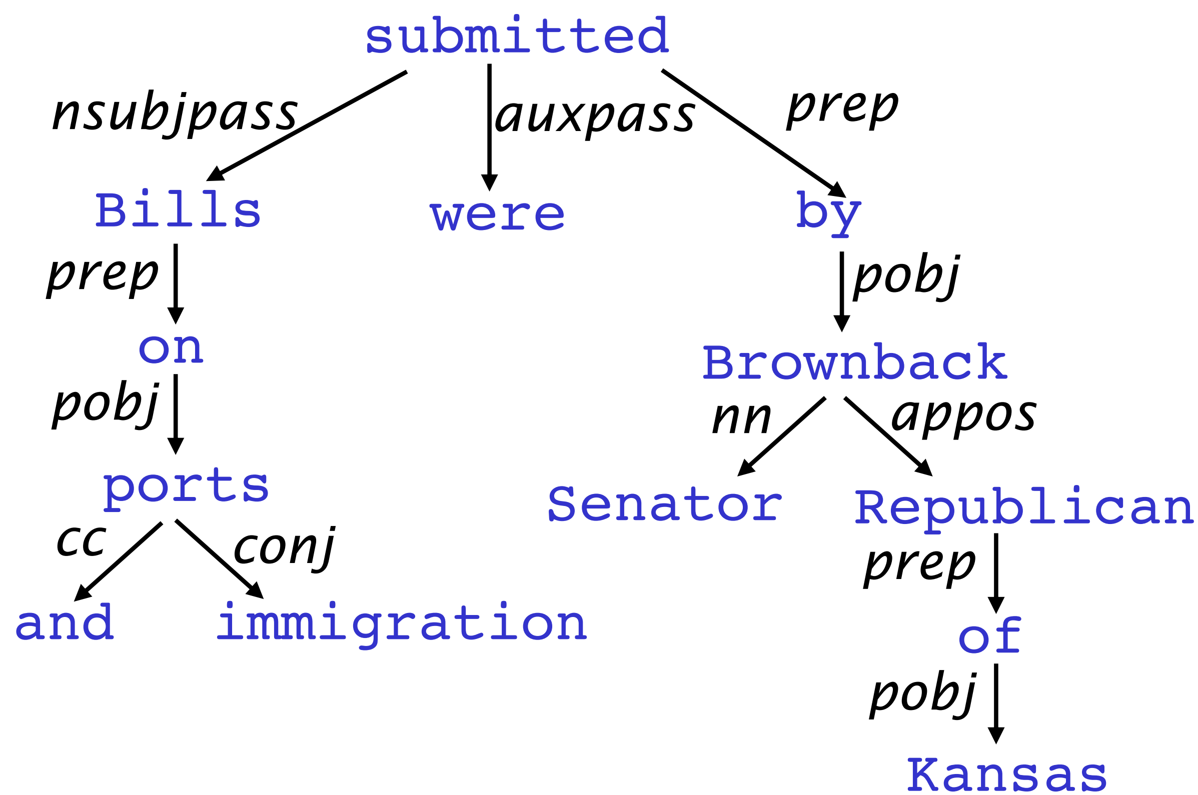
- Syntactic structure consists of:
  - Lexical items
  - Binary asymmetric relations →dependencies



Dependency Structure

- Syntactic structure consists of:
  - Lexical items
  - Binary asymmetric relations →dependencies

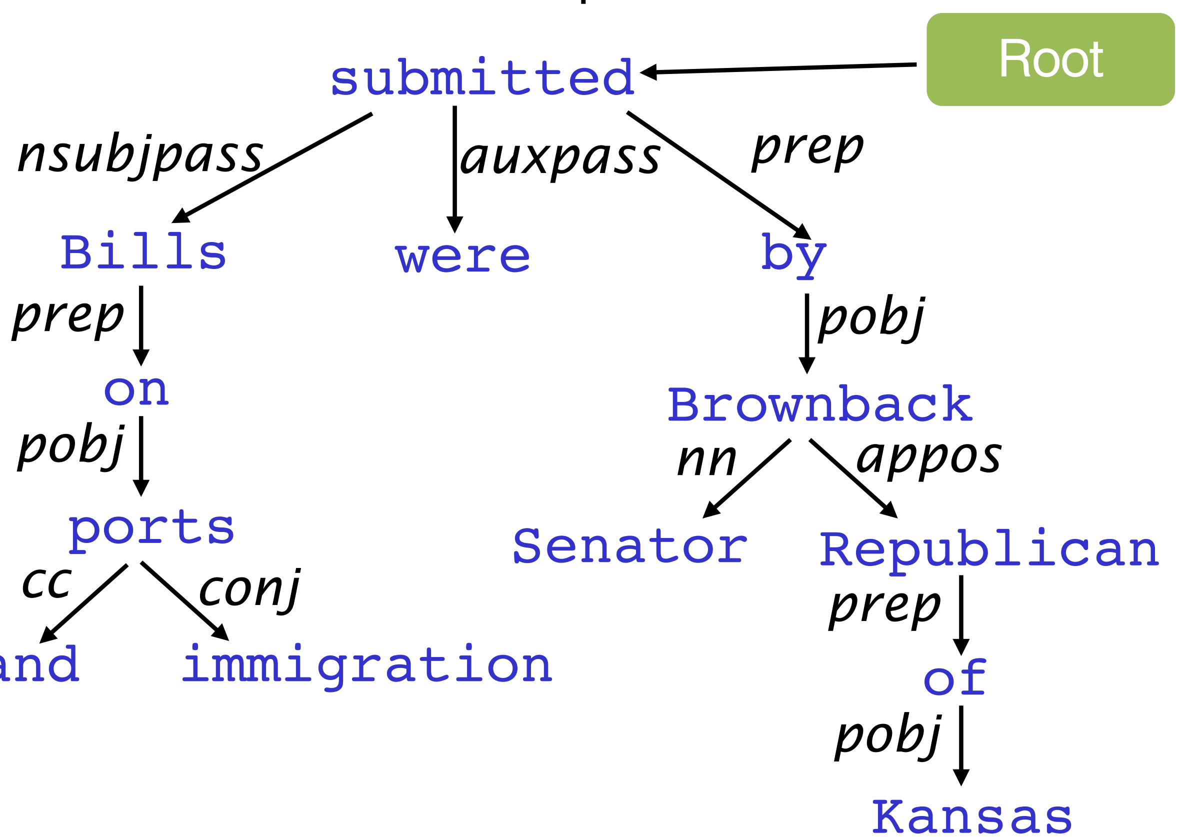
Dependencies form a tree



Dependency Structure

- Syntactic structure consists of:
  - Lexical items
  - Binary asymmetric relations →dependencies

Dependencies form a tree



Let’s Parse

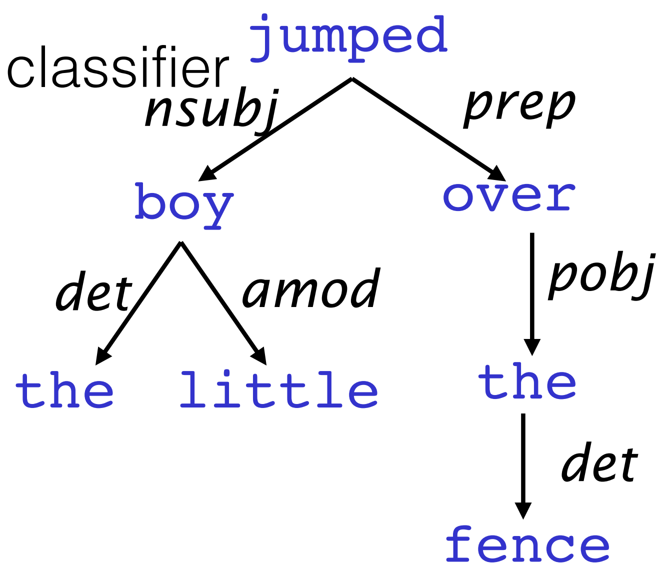
Start with main verb, and draw dependencies. Don’t worry about labels. Just try the modifiers right.

John saw Mary

He said that the boy who was wearing the blue shirt with the white pockets has left the building

Methods for Dependency Parsing

- Dynamic programming
  - Eisner (1996): $O(n^3)$
- Graph algorithms
  - McDonald et al. (2005): score edges independently using classifier and use maximum spanning tree
- Constraint satisfaction
  - Start with all edges, eliminate based on hard constraints
- "Deterministic parsing"
  - Left-to-right, each choice is done with a classifier



Making Decisions

What are the sources of information for dependency parsing?

1. Bilexical affinities
   – [issues → the] is plausible
2. Dependency distance
   – mostly with nearby words
3. Intervening material
   – Dependencies rarely span intervening verbs or punctuation
4. Valency of heads
   – How many dependents on which side are usual for a head?

MaltParse (Nivre et al. 2008)

- Greedy transition-based parser
- Each decision: how to attach each word as we encounter it
  - If you are familiar: like shift-reduce parser
- Select each action with a classifier
- The parser has:
  - a stack σ, written with the top to the right
    - which starts with the ROOT symbol
  - a buffer β, written with the top to the left
    - which starts with the input sentence
  - a set of dependency arcs A
    - which starts off empty
  - a set of actions

Arc-standard Dependency Parsing

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

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

Finish: $\beta = \emptyset$

ROOT Joe likes Marry

Arc-standard Dependency Parsing

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

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

Finish: $\beta = \emptyset$

ROOT Joe likes Marry

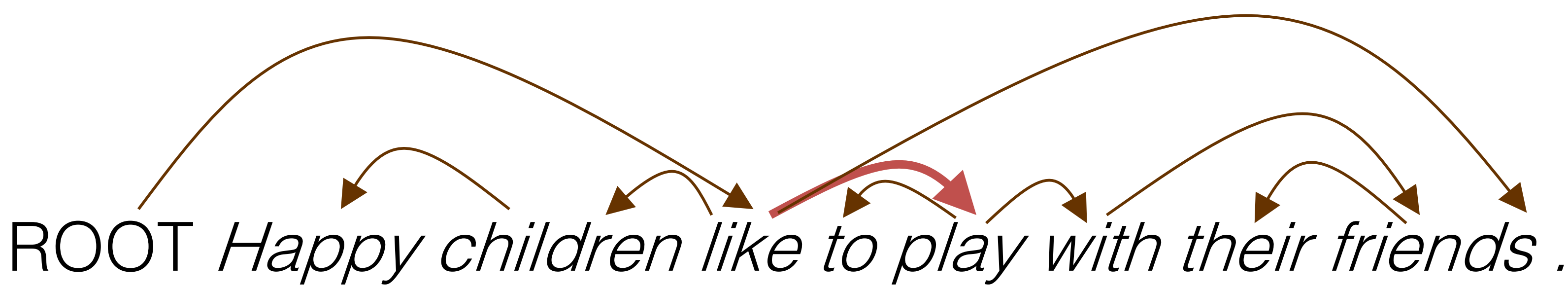
	[ROOT]	[Joe, likes, marry]	$\emptyset$
Shift	[ROOT, Joe]	[likes, marry]	$\emptyset$
Left-Arc	[ROOT]	[likes, marry]	{(likes,Joe)} = A_1
Shift	[ROOT, likes]	[marry]	A_1
Right-Arc	[ROOT]	[likes]	$A_1 \cup$ {(likes,Marry)} = A_2
Right-Arc	[]	[ROOT]	$A_2 \cup$ {(ROOT, likes)} = A_3
Shift	[ROOT]	[]	A_3

Arc-standard Dependency Parsing

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

- Shift $\quad \sigma, w_i|\beta, A \rightarrow \sigma|w_i, \beta, A$
- Left-Arc$_r$ $\quad \sigma|w_i, w_j|\beta, A \rightarrow \sigma, w_j|\beta, A \cup \{r(w_j,w_i)\}$
- 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-eager Dependency Parsing

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

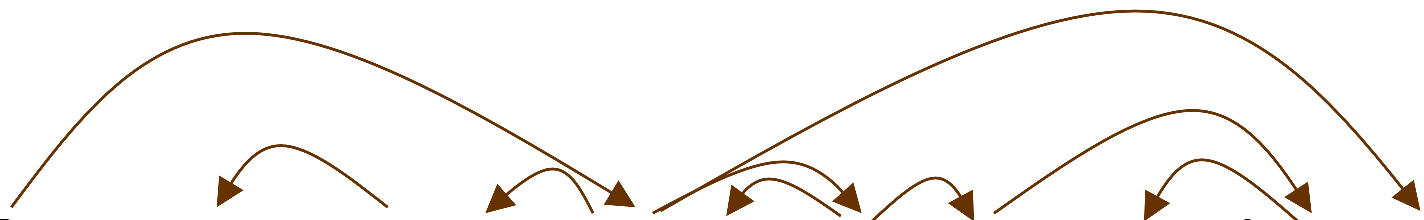
- Left-Arc$_r$ $\quad \sigma|w_i, w_j|\beta, A \rightarrow \sigma, w_j|\beta, A \cup \{r(w_j,w_i)\}$
  - Precondition: $r'(w_k, w_i) \notin A$, $w_i \neq \text{ROOT}$
- Right-Arc$_r$ $\quad \sigma|w_i, w_j|\beta, A \rightarrow \sigma|w_i|w_j, \beta, A \cup \{r(w_i,w_j)\}$
- Reduce $\quad \sigma|w_i, \beta, A \rightarrow \sigma, \beta, A$
  - Precondition: $r'(w_k, w_i) \in A$
- Shift $\quad \sigma, w_i|\beta, A \rightarrow \sigma|w_i, \beta, A$

Finish: $\beta = \emptyset$

This is the common "arc-eager" variant: a head can immediately take a right dependent, before *its* dependents are found

Arc-eager

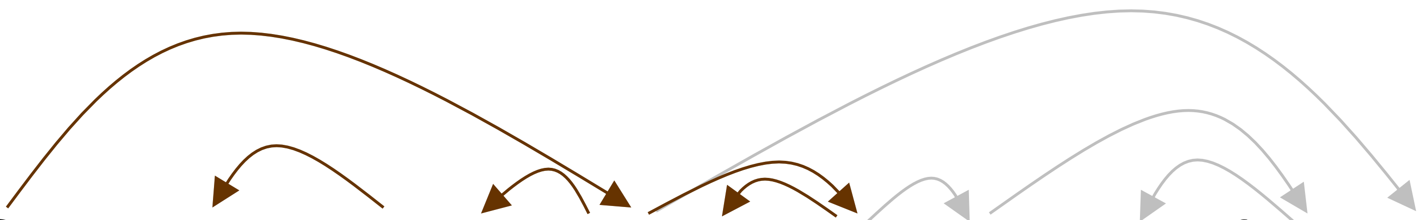
1. Left-Arc$_r$ $\sigma|w_i,\ w_j|\beta,\ A \Rightarrow \sigma,\ w_j|\beta,\ A\cup\{r(w_j, w_i)\}$
   Precondition: $r'(w_k, w_i) \notin A,\ w_i \neq \mathrm{ROOT}$
2. Right-Arc$_r$ $\sigma|w_i,\ w_j|\beta,\ A \Rightarrow \sigma|w_i|w_j,\ \beta,\ A\cup\{r(w_i, w_j)\}$
3. Reduce $\sigma|w_i,\ \beta,\ A \Rightarrow \sigma,\ \beta,\ A$
   Precondition: $r'(w_k, w_i) \in A$
4. Shift $\sigma,\ w_i|\beta,\ A \Rightarrow \sigma|w_i,\ \beta,\ A$



ROOT *Happy children like to play with their friends .*

Arc-eager

1. Left-Arc$_r$ $\sigma|w_i,\ w_j|\beta,\ A \rightarrow \sigma,\ w_j|\beta,\ A\cup\{r(w_j, w_i)\}$
   Precondition: $r'(w_k,\ w_i) \notin A,\ w_i \neq$ ROOT
2. Right-Arc$_r$ $\sigma|wi,\ w_j|\beta,\ A \rightarrow \sigma|w_i|w_j,\ \beta,\ A\cup\{r(w_i, w_j)\}$
3. Reduce $\sigma|w_i,\ \beta,\ A \rightarrow \sigma,\ \beta,\ A$
   Precondition: $r'(w_k,\ w_i) \in A$
4. Shift $\sigma,\ w_i|\beta,\ A \rightarrow \sigma|w_i,\ \beta,\ A$



ROOT *Happy children like to play with their friends .*

	[ROOT]	[Happy, children, …]	Ø
Shift	[ROOT, Happy]	[children, like, …]	Ø
LA$_{amod}$	[ROOT]	[children, like, …]	{amod(children, happy)} = A_1
Shift	[ROOT, children]	[like, to, …]	A_1
LA$_{nsubj}$	[ROOT]	[like, to, …]	$A_1 \cup$ {nsubj(like, children)} = A_2
RA$_{root}$	[ROOT, like]	[to, play, …]	$A_2 \cup${root(ROOT, like) = A_3
Shift	[ROOT, like, to]	[play, with, …]	A_3
LA$_{aux}$	[ROOT, like]	[play, with, …]	$A_3\cup${aux(play, to) = A_4
RA$_{xcomp}$	[ROOT, like, play]	[with their, …]	$A_4\cup${xcomp(like, play) = A_5

Arc-eager

1. Left-Arc$_r$ $\sigma|w_i,\ w_j|\beta,\ A \Rightarrow \sigma,\ w_j|\beta,\ A\cup\{r(w_j, w_i)\}$
   Precondition: $r'(w_k, w_i) \notin A,\ w_i \neq \text{ROOT}$
2. Right-Arc$_r$ $\sigma|wi,\ w_j|\beta,\ A \Rightarrow \sigma|w_i|w_j,\ \beta,\ A\cup\{r(w_i, w_j)\}$
3. Reduce $\sigma|w_i,\ \beta,\ A \Rightarrow \sigma,\ \beta,\ A$
   Precondition: $r'(w_k, w_i) \in A$
4. Shift $\sigma,\ w_i|\beta,\ A \Rightarrow \sigma|w_i,\ \beta,\ A$

ROOT *Happy children like to play with their friends .*

RA$_{xcomp}$	[ROOT, like, play]	[with their, …]	$A_4\cup$\{xcomp(like, play) = A_5
RA$_{prep}$	[ROOT, like, play, with]	[their, friends, …]	$A_5\cup$\{prep(play, with) = A_6
Shift	[ROOT, like, play, with, their]	[friends, .]	A_6
LA$_{poss}$	[ROOT, like, play, with]	[friends, .]	$A_6\cup$\{poss(friends, their) = A_7
RA$_{pobj}$	[ROOT, like, play, with, friends]	[.]	$A_7\cup$\{pobj(with, friends) = A_8
Reduce	[ROOT, like, play, with]	[.]	A_8
Reduce	[ROOT, like, play]	[.]	A_8
Reduce	[ROOT, like]	[.]	A_8
RA$_{punc}$	[ROOT, like, .]	[]	$A_8\cup$\{punc(like, .) = A_9

You terminate as soon as the buffer is empty. Dependencies = A_9

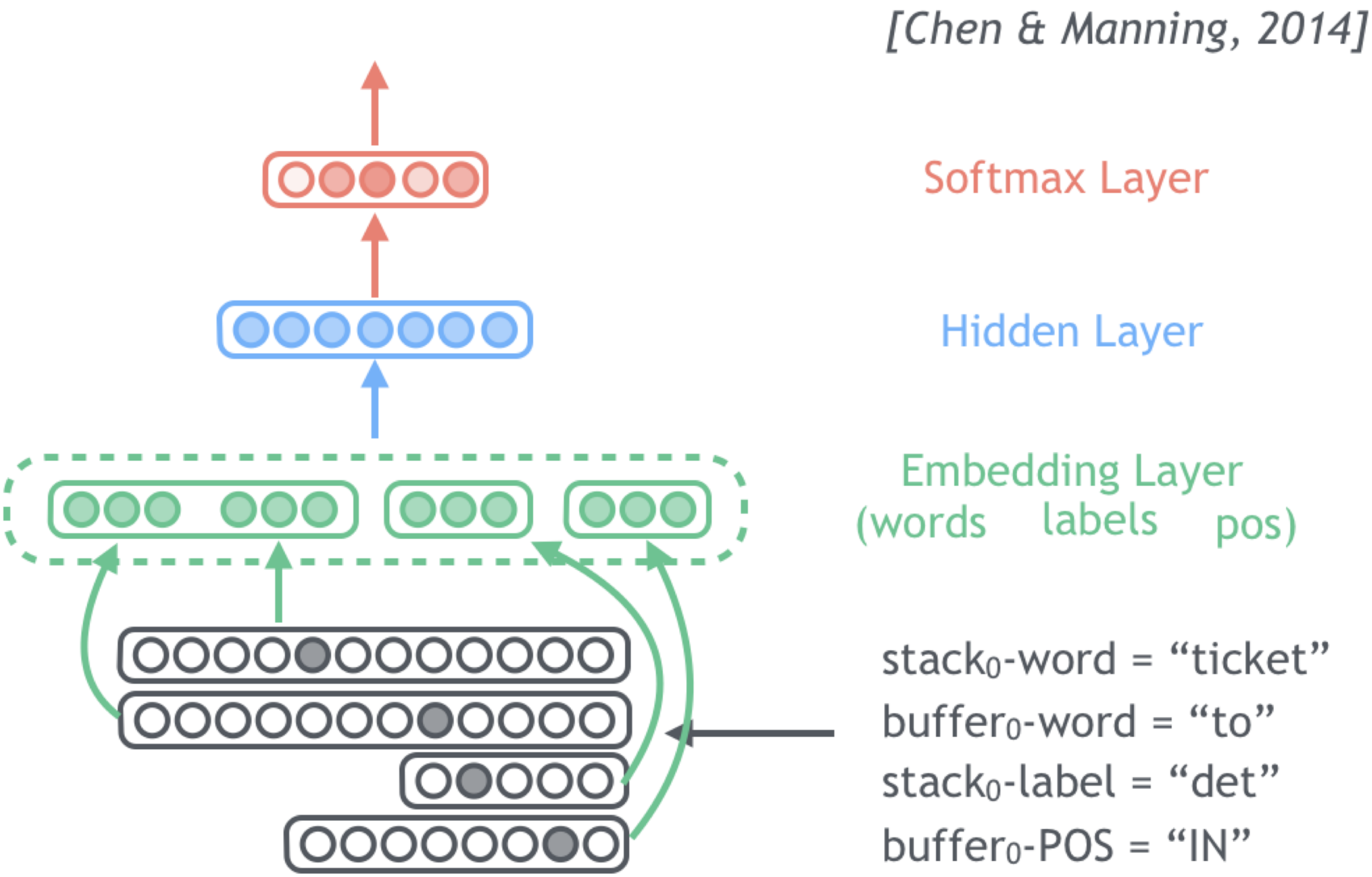
MaltParser (Nivre et al. 2008)

- Selecting the next action:
  - Discriminative classifier (SVM, MaxEnt, etc.)
  - Untyped choices: 4
  - Typed choices: |R| * 2 + 2
- Features: POS tags, word in stack, word in buffer, etc.
- Greedy → no search
  - But can easily do beam search
- Close to state of the art
- Linear time parser → very fast!

Parsing with Neural Networks Chen and Manning (2014)

- Arc-standard Transitions
  - Shift
  - Left-Arc$_r$
  - Right-Arc$_r$
- Selecting the next actions:
  - Untyped choices: 3
  - Typed choices: |R| * 2 + 1
  - Neural network classifier
- With a few model improvements and very careful hyper-parameter tuning gives SOTA results

Parsing with Neural Networks
Chen and Manning (2014)



Hyperparameters?

- Regularization
- Loss function

Slide from David Weiss

Hyperparameters?

- Regularization
- Loss function

- Dimensions
- Activation function
- Initialization
- Adagrad
- Dropout

Slide from David Weiss

Hyperparameters?

- Regularization
- Loss function

- Dimensions
- Activation function

- Initialization
- Adagrad
- Dropout

- Mini-batch size
- Initial learning rate
- Learning rate schedule
- Momentum

- Stopping time
- Parameter averaging

Slide from David Weiss

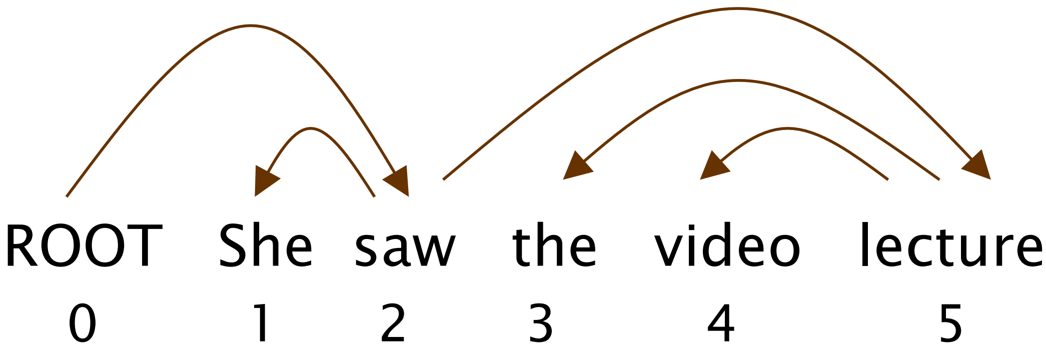
Hyperparameters?

Slide from David Weiss

Hyperparameters?

Slide from David Weiss

Evaluation



$$\text{Acc} = \frac{\#\ \text{correct deps}}{\#\ \text{of deps}}$$

UAS = 4 / 5 = 80%
LAS = 2 / 5 = 40%

Gold

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

Parsed

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

Projectivity

- Dependencies from CFG trees with head rules must be projective
  - Crossing arcs are not allowed
- But: theory allows to account for displaced constituents → non-projective structures

Projectivity

- Arc-eager transition system:
  - Can't handle non-projectivity
- Possible directions:
  - Give up!
  - Post-processing
  - Add new transition types
  - Switch to a different algorithm
    - Graph-based parsers (e.g., MSTParser)