Dependency Grammars and Parser

LING 571 — Deep Processing for NLP

October 17, 2018

Ryan Georgi

Announcements:

- Compiled `evalb` for MacOS on Canvas:
  - https://canvas.uw.edu/files/51379663/download?download_frd=1

PCKY Algorithm

function PROBABILISTIC-CKY-PARSE(*words*, *grammar*) **returns** most probable parse and its probability

for j ← **from** 1 to LENGTH(*words*) **do**
 for all { $A \mid A \rightarrow words[j] \in grammar$ }
  $table[\, j{-}1, j, A\,] \leftarrow P(A \rightarrow words[j])$
 for $i \leftarrow$ **from** $j{-}2$ **downto** 0 **do**
  for $k \leftarrow i + 1$ **to** $j{-}1$ **do**
  for all { $A \mid A \rightarrow B\ C \in grammar$,
   and $table[i, k, B] > 0$ **and** $table[\, k, j, C\,] > 0$ }
  if $(table[\, i, j, A\,] < P(A \rightarrow BC) \times table[\, i, k, B\,] \times table[\, k, j, C\,])$ **then**
   $table[\, i, j, A\,] \leftarrow P(A \rightarrow BC) \times table[i,k,B] \times table[k,j,C]$
   $back[\, i, j, A\,] \leftarrow \{\, k, B, C\,\}$
  return BUILD_TREE($back[\, 1,$ LENGTH($words$), $S\,]$), $table[\, 1,$LENGTH($words$), $S\,]$

Notes On HW #4

"the dog chased the cat on the mat"

0	Det → "the" [-0.693]	NP → $_0$Det$_1$N$_2$ [-1.897]$_{(0,0)}$			S → $_0$NP$_2$VP$_5$ [-4.528]$_{(0,0)}$		S → $_0$NP$_2$VP$_8$ [-8.610]$_{(0,0)}$
	1	N → "dog" [-0.916]					
		2	V → "chased" [-0.511]		VP → $_2$V$_3$NP$_5$ [-2.631]$_{(0,0)}$		VP → $_2$V$_3$NP$_8$ [-6.713]$_{(0,0)}$
			3	Det → "a" [-0.693]	NP → $_3$Det$_4$N$_5$ [-1.897]$_{(0,0)}$		NP → $_3$NP$_5$PP$_8$ [-5.979]$_{(0,0)}$
				4	N → "cat" [-0.916]		
					5	P → "on" [-0.105]	PP → $_5$P$_6$NP$_8$ [-2.696]$_{(0,0)}$
						6	Det → "the" [-0.693]	NP → $_6$Det$_7$N$_8$ [-2.590]$_{(0,0)}$
							7	N → "mat" [-1.609]
								8

S	→	*NP VP*	[1.0]
PP	→	*P NP*	[1.0]
NP	→	*Det N*	[0.75]
NP	→	*NP PP*	[0.25]
VP	→	*V NP*	[0.8]
VP	→	*VP PP*	[0.2]
Det	→	'a'	[0.5]
Det	→	'the'	[0.5]
N	→	'dog'	[0.4]
N	→	'cat'	[0.4]
N	→	'mat'	[0.2]
V	→	'chased'	[0.6]
V	→	'sat'	[0.4]
P	→	'on'	[0.9]
P	→	'in'	[0.1]

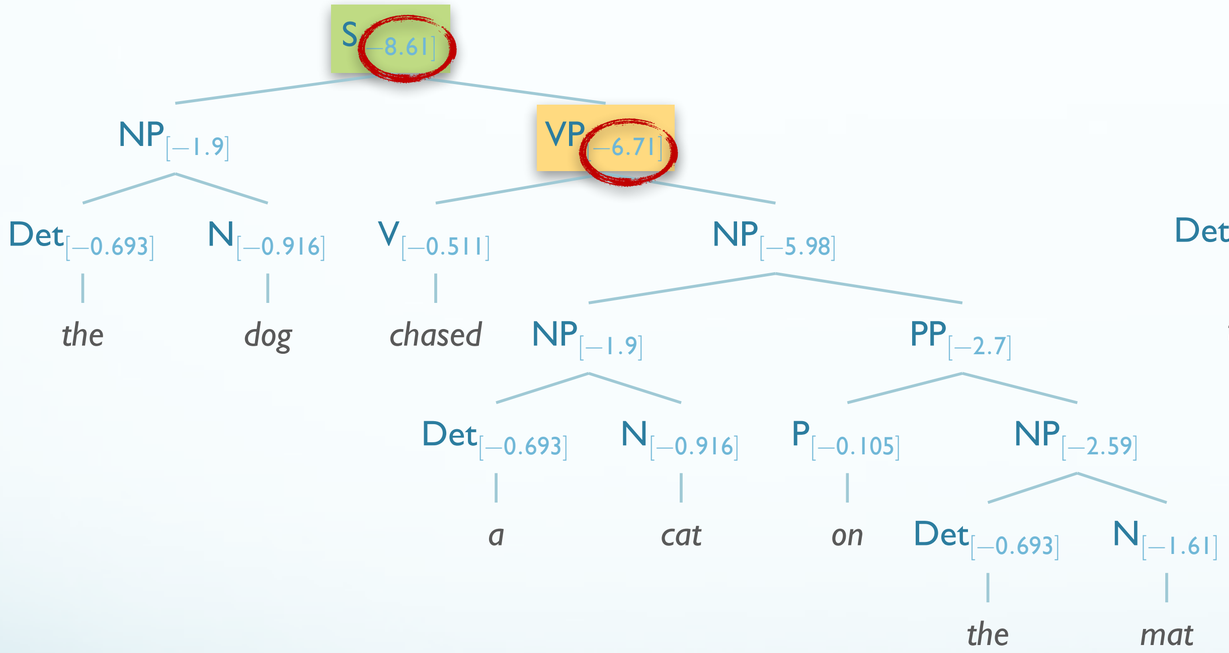
Notes On HW #4

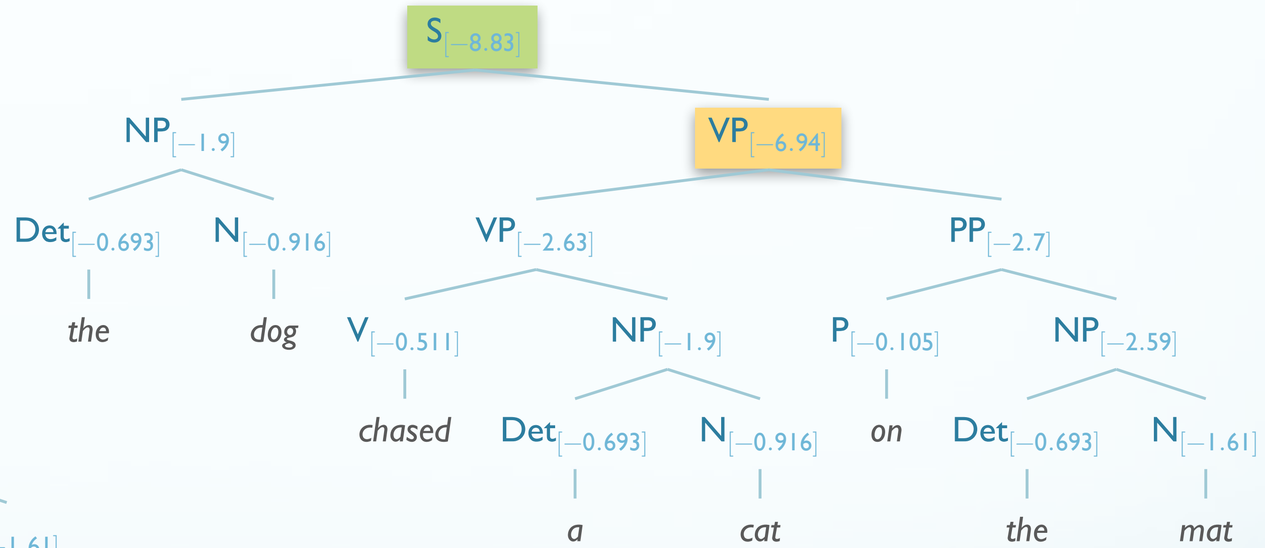
"the dog chased the cat on the mat"

0	Det → "the" [-0.693]	NP → $_0$Det$_1$N$_2$ [-1.897]$_{(0,0)}$			S → $_0$NP$_2$VP$_5$ [-4.528]$_{(0,0)}$		S → $_0$NP$_2$VP$_8$ [-8.610]$_{(0,0)}$ S → $_0$NP$_2$VP$_8$ [-8.833]$_{(0,1)}$
	1	N → "dog" [-0.916]					
		2	V → "chased" [-0.511]		VP → $_2$V$_3$NP$_5$ [-2.631]$_{(0,0)}$		VP → $_2$V$_3$NP$_8$ [-6.713]$_{(0,0)}$ VP → $_2$VP$_5$PP$_8$ [-6.936]$_{(0,0)}$
			3	Det → "a" [-0.693]	NP → $_3$Det$_4$N$_5$ [-1.897]$_{(0,0)}$		NP → $_3$NP$_5$PP$_8$ [-5.979]$_{(0,0)}$
				4	N → "cat" [-0.916]		
					5	P → "on" [-0.105]	PP → $_5$P$_6$NP$_8$ [-2.696]$_{(0,0)}$
						6	Det → "the" [-0.693]	NP → $_6$Det$_7$N$_8$ [-2.590]$_{(0,0)}$
							7	N → "mat" [-1.609]
								8

S	→	$NP\ VP$	[1.0]
PP	→	$P\ NP$	[1.0]
NP	→	$Det\ N$	[0.75]
NP	→	$NP\ PP$	[0.25]
VP	→	$V\ NP$	[0.8]
VP	→	$VP\ PP$	[0.2]
Det	→	'a'	[0.5]
Det	→	'the'	[0.5]
N	→	'dog'	[0.4]
N	→	'cat'	[0.4]
N	→	'mat'	[0.2]
V	→	'chased'	[0.6]
V	→	'sat'	[0.4]
P	→	'on'	[0.9]
P	→	'in'	[0.1]

Notes On HW #4





Roadmap

- Dependency Grammars
  - Definition
  - Motivation:
    - Limitations of Context-Free Grammars
- Dependency Parsing
  - By conversion to CFG
  - By Graph-based models
  - By transition-based parsing

Dependency Grammar

- **[P]CFGs**:
  - Phrase-Structure Grammars
  - Focus on modeling constituent structure
- **Dependency grammars**:
  - Syntactic structure described in terms of
    - Words
    - Syntactic/semantic relations between words

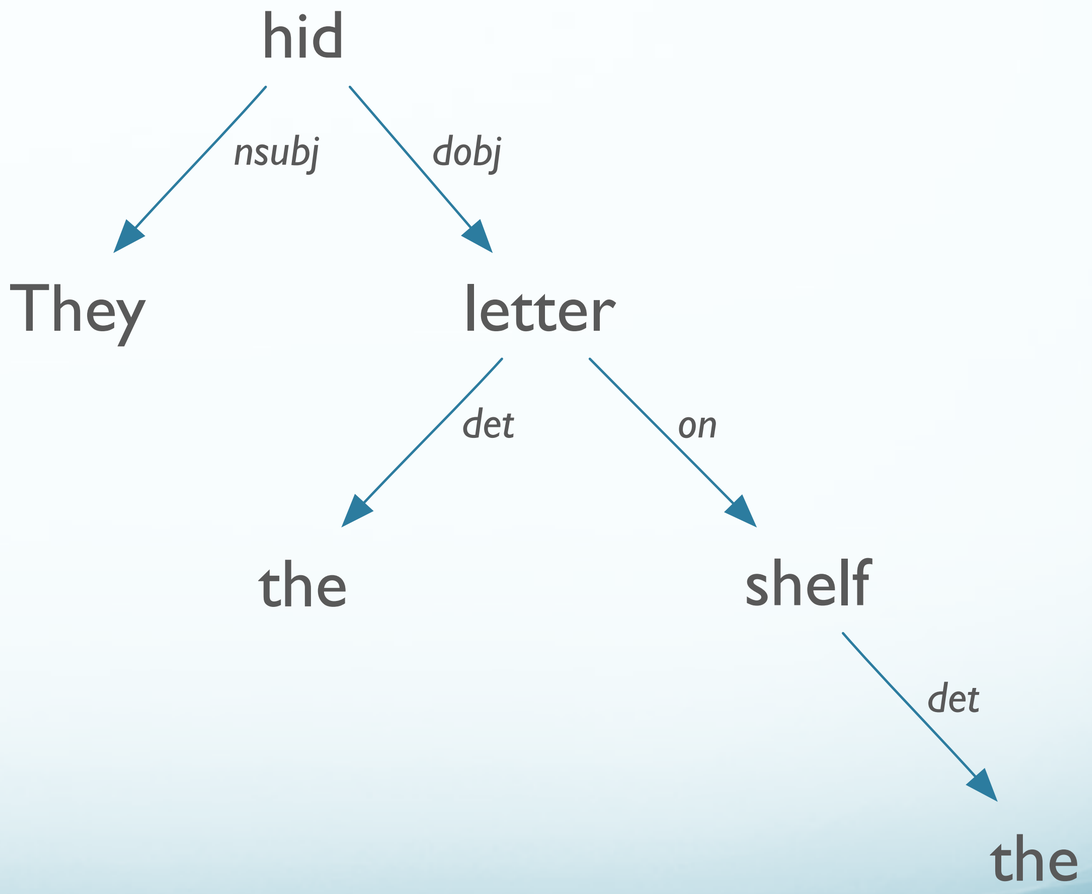
Dependency Parse

- A Dependency parse is a tree, where:
  - Nodes correspond to words in string
  - Edges between nodes represent dependency relations
    - Relations may or may not be labeled

Dependency Parse Example:

They hid the letter on the shelf

Argument Dependencies	
Abbreviation	**Description**
nsubj	nominal subject
csubj	clausal subject
dobj	direct object
iobj	indirect object
pobj	object of preposition
Modifier Dependencies	
Abbreviation	**Description**
tmod	temporal modifier
appos	appositional modifier
det	determiner
prep	prepositional modifier

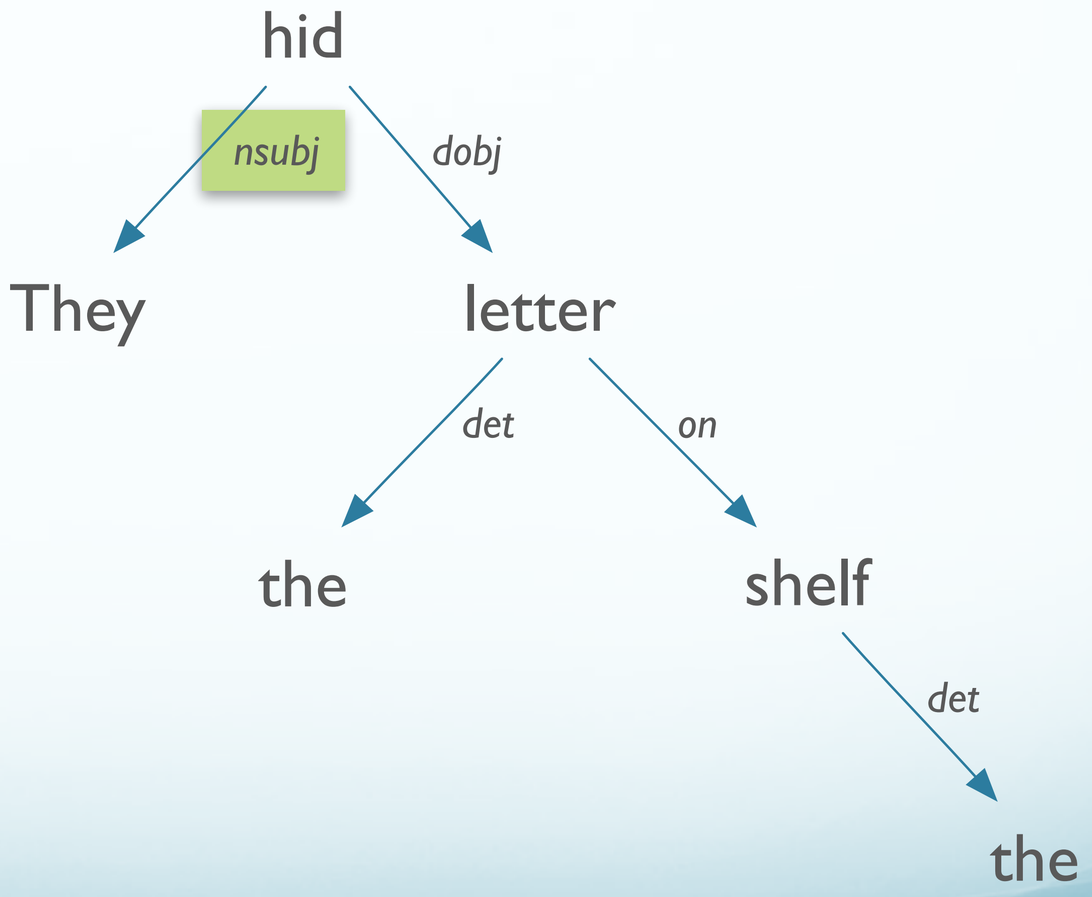


Dependency Parse Example:

They hid the letter on the shelf

Argument Dependencies	
Abbreviation	**Description**
nsubj	nominal subject
csubj	clausal subject
dobj	direct object
iobj	indirect object
pobj	object of preposition

Modifier Dependencies	
Abbreviation	**Description**
tmod	temporal modifier
appos	appositional modifier
det	determiner
prep	prepositional modifier

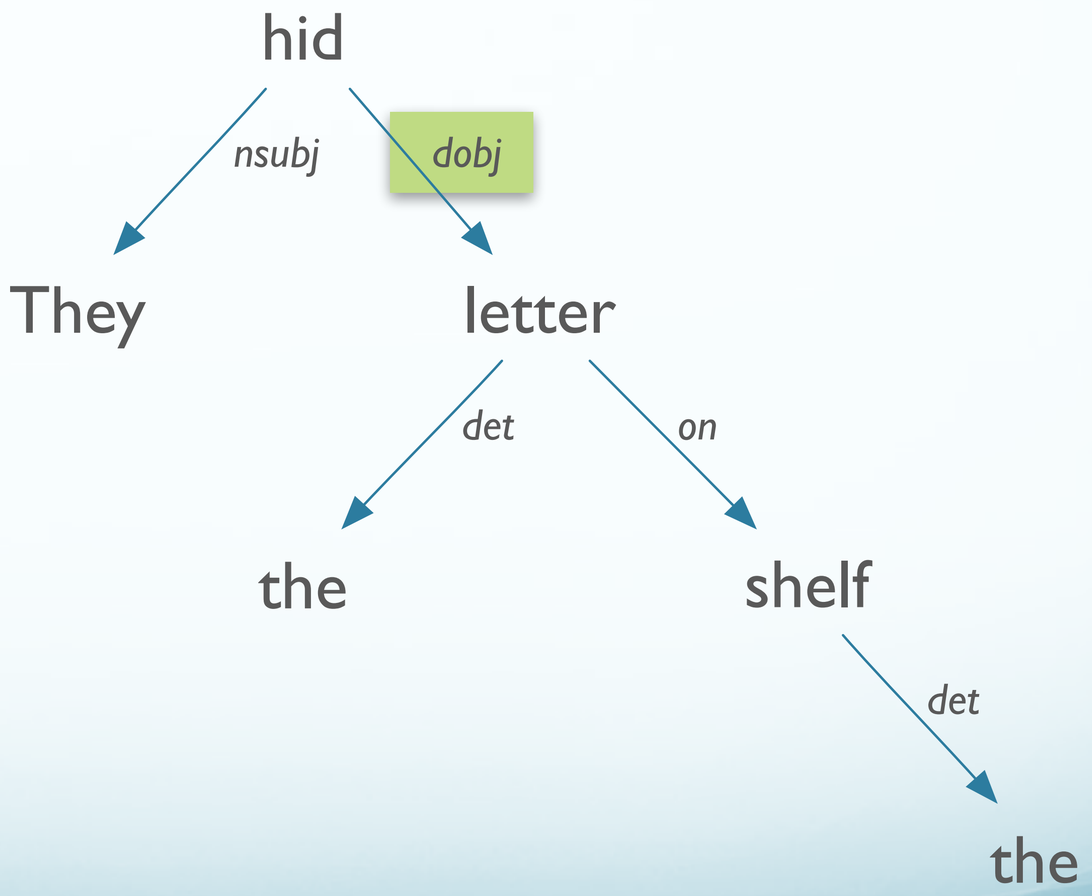


Dependency Parse Example:

They hid the letter on the shelf

Argument Dependencies	
Abbreviation	**Description**
nsubj	nominal subject
csubj	clausal subject
dobj	direct object
iobj	indirect object
pobj	object of preposition

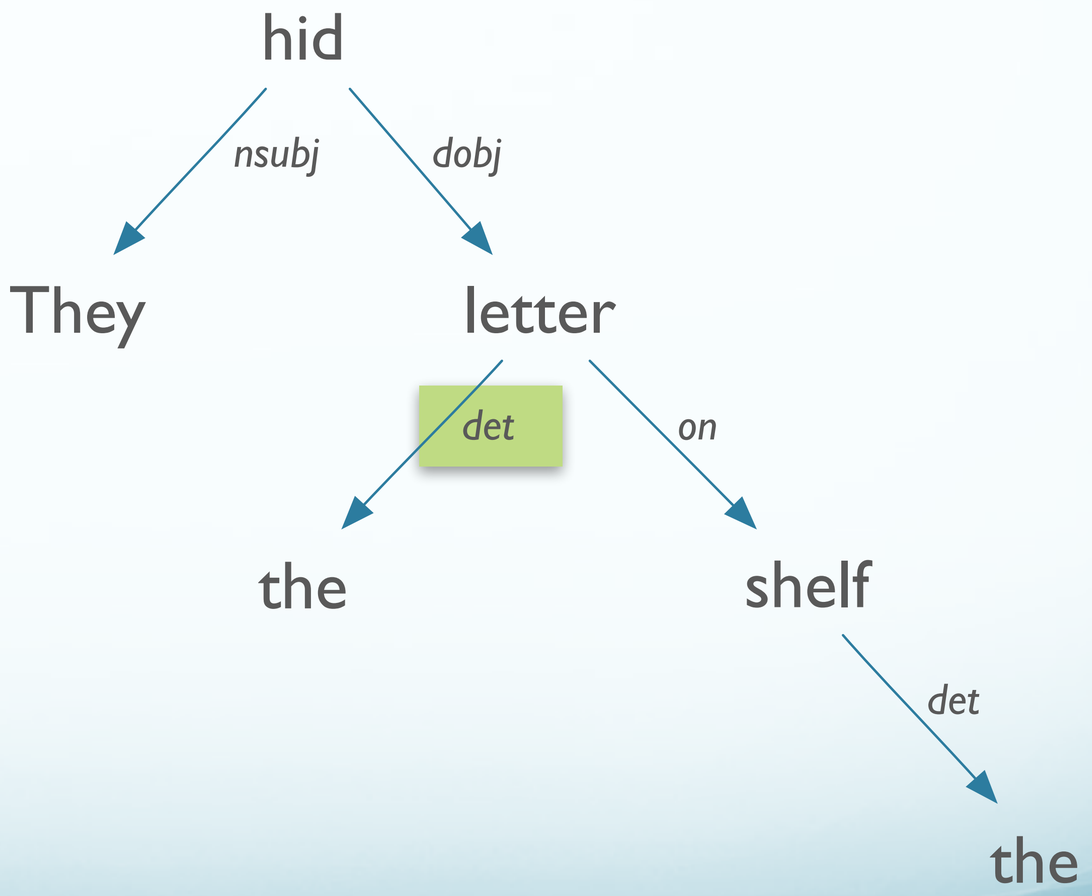
Modifier Dependencies	
Abbreviation	**Description**
tmod	temporal modifier
appos	appositional modifier
det	determiner
prep	prepositional modifier



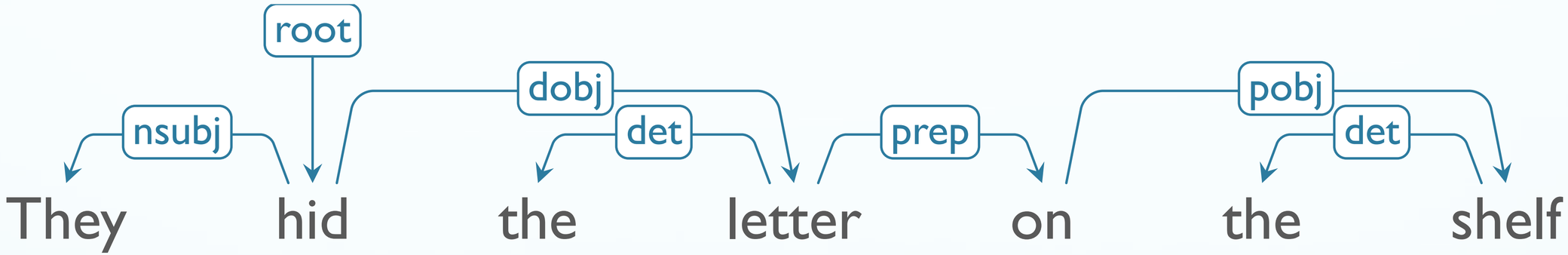
Dependency Parse Example:

They hid the letter on the shelf

Argument Dependencies	
Abbreviation	**Description**
nsubj	nominal subject
csubj	clausal subject
dobj	direct object
iobj	indirect object
pobj	object of preposition
Modifier Dependencies	
Abbreviation	**Description**
tmod	temporal modifier
appos	appositional modifier
det	determiner
prep	prepositional modifier



Alternative Representation

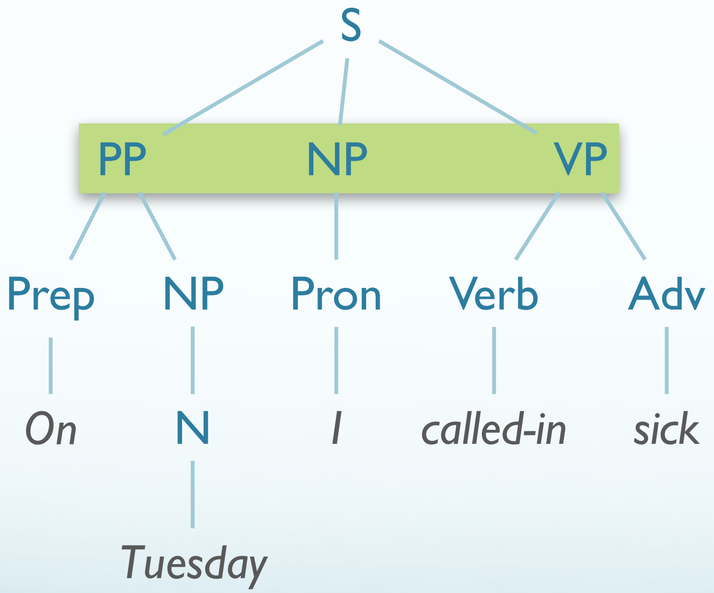


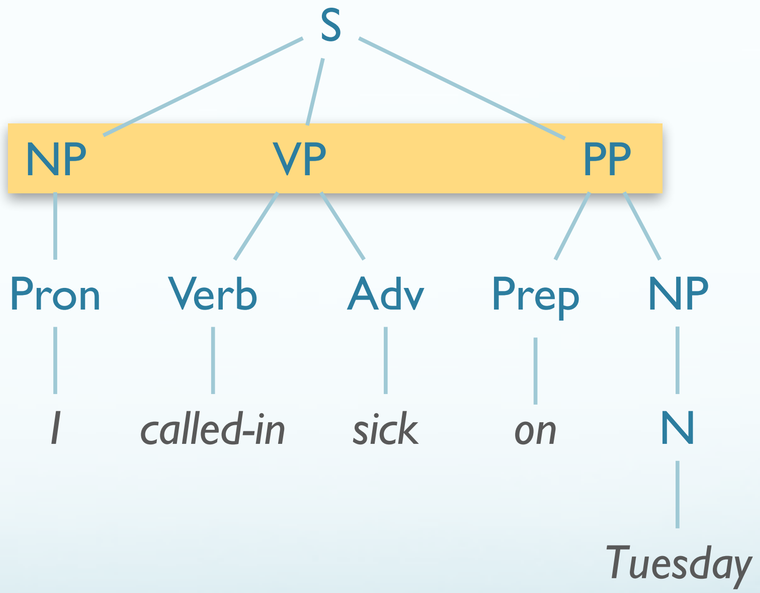
Why Dependency Grammar?

- More natural representation for many tasks
  - Clear encapsulation of predicate-argument structure
    - Phrase structure may obscure, e.g. *wh-movement*
- Good match for question-answering, relation extraction
  - *Who* did *what* to *whom*?
  - = (*Subject*) did (*theme*) to (*patient*)
  - Helps with parallel relations between roles in **questions**, and roles in **answers**

Why Dependency Grammar?

- Easier handling of flexible or free word order
- How does CFG handle variation in word order?



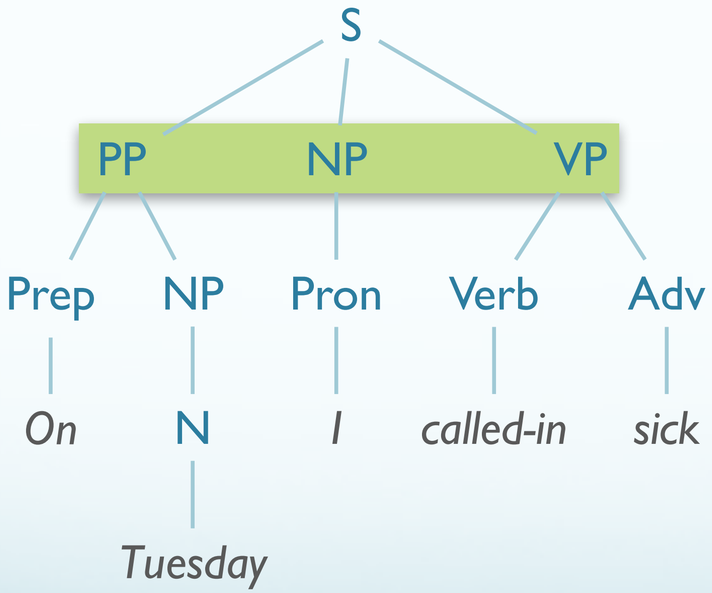


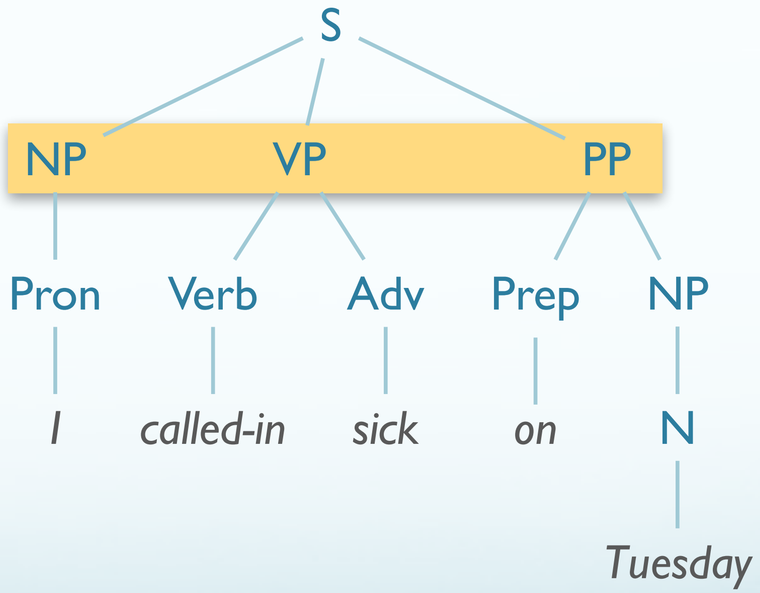
$S \rightarrow PP\ NP\ VP$

$S \rightarrow NP\ VP\ PP$

Why Dependency Grammar?

- English has relatively fixed word order
- Big problem for languages with freer word order



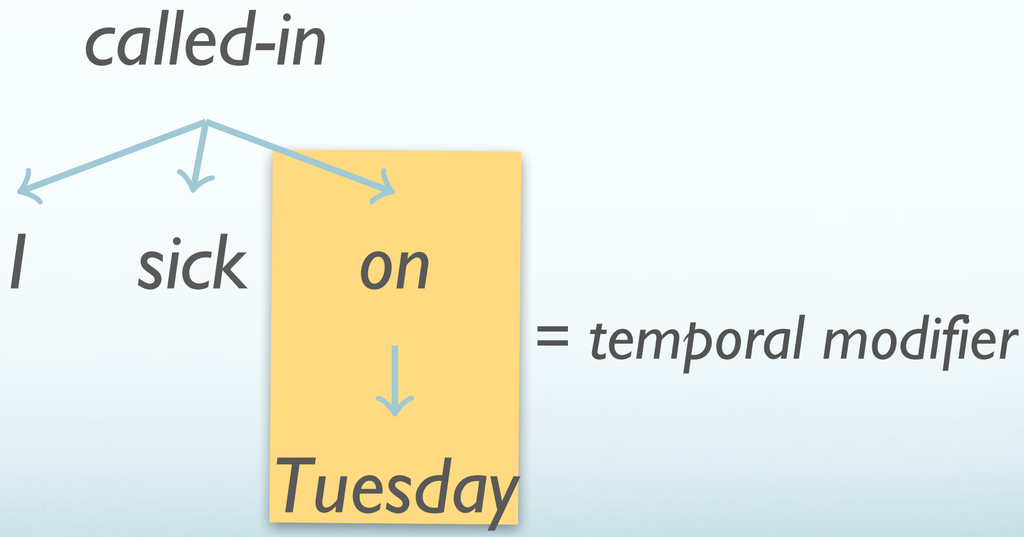


$S \rightarrow PP\ NP\ VP$

$S \rightarrow NP\ VP\ PP$

Why Dependency Grammar?

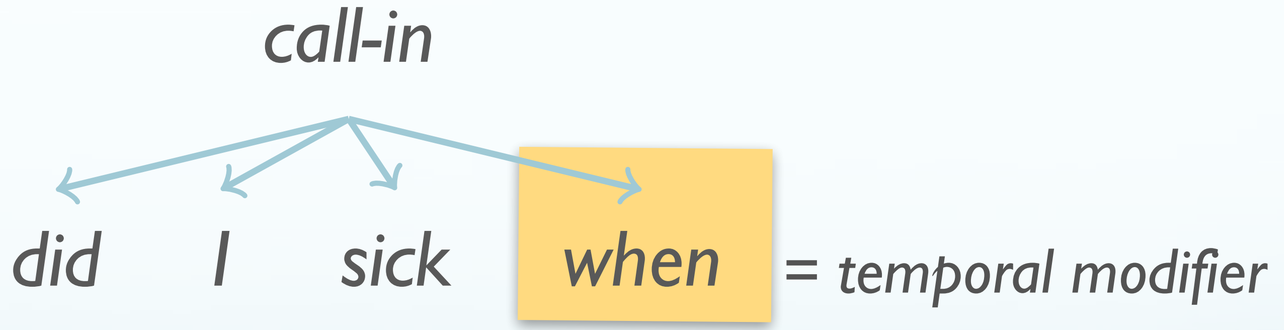
- How do dependency structures represent the difference?
  - Same structure
  - Relationships are between words, order insensitive



I called in sick on Tuesday

Why Dependency Grammar?

- How do dependency structures represent the difference?
  - Same structure
  - Relationships are between words, order insensitive



when did I call in sick?

Natural Efficiencies

- Phrase Structures:
  - Must derive full trees of many non-terminals
- Dependency Structures:
  - For each word, identify
    - Syntactic head, h
    - Dependency label, d
  - Inherently lexicalized
    - Strong constraints hold between pairs of words

Summary

- Dependency grammars balance complexity and expressiveness
  - Sufficiently expressive to capture predicate-argument structure
  - Sufficiently constrained to allow efficient parsing

- Still not perfect
  - "On Tuesday I called in sick" vs. "I called in sick on Tuesday"
  - …I would argue these feel pragmatically different, might want to represent differently.

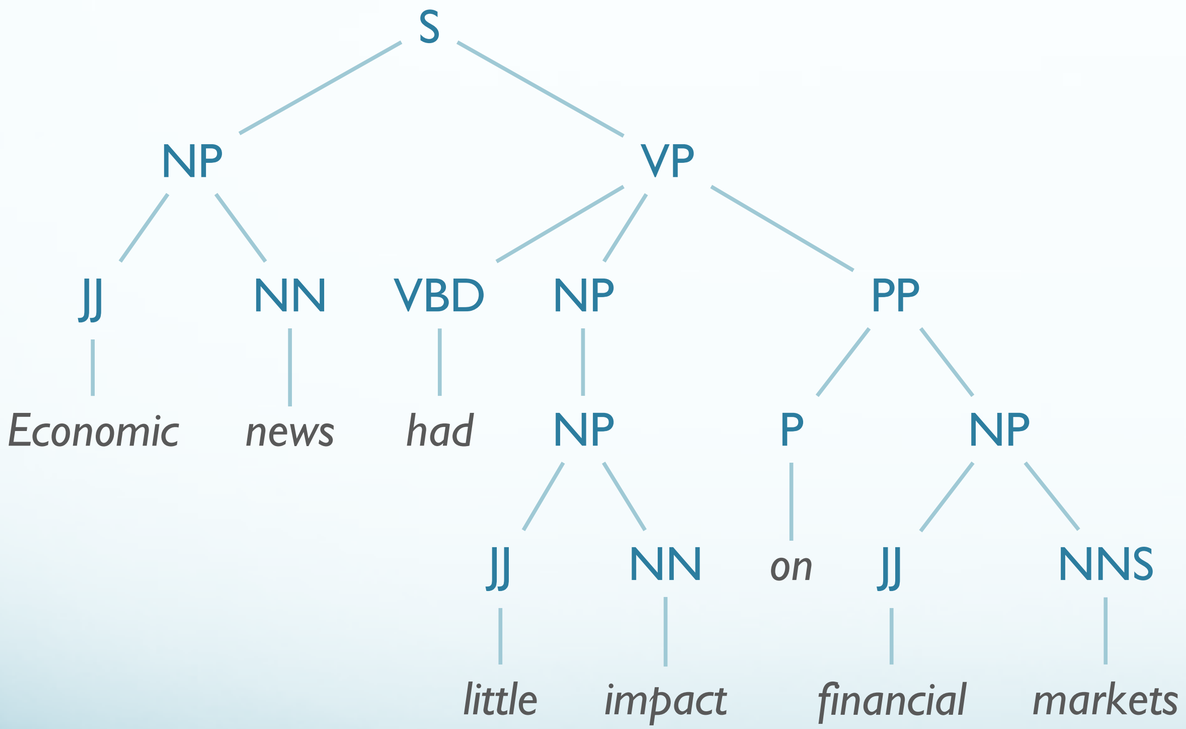
Roadmap

- Dependency Grammars
  - Definition
  - Motivation:
    - Limitations of Context-Free Grammars
- **Dependency Parsing**
  - **By conversion from CFG**
  - By Graph-based models
  - By transition-based parsing

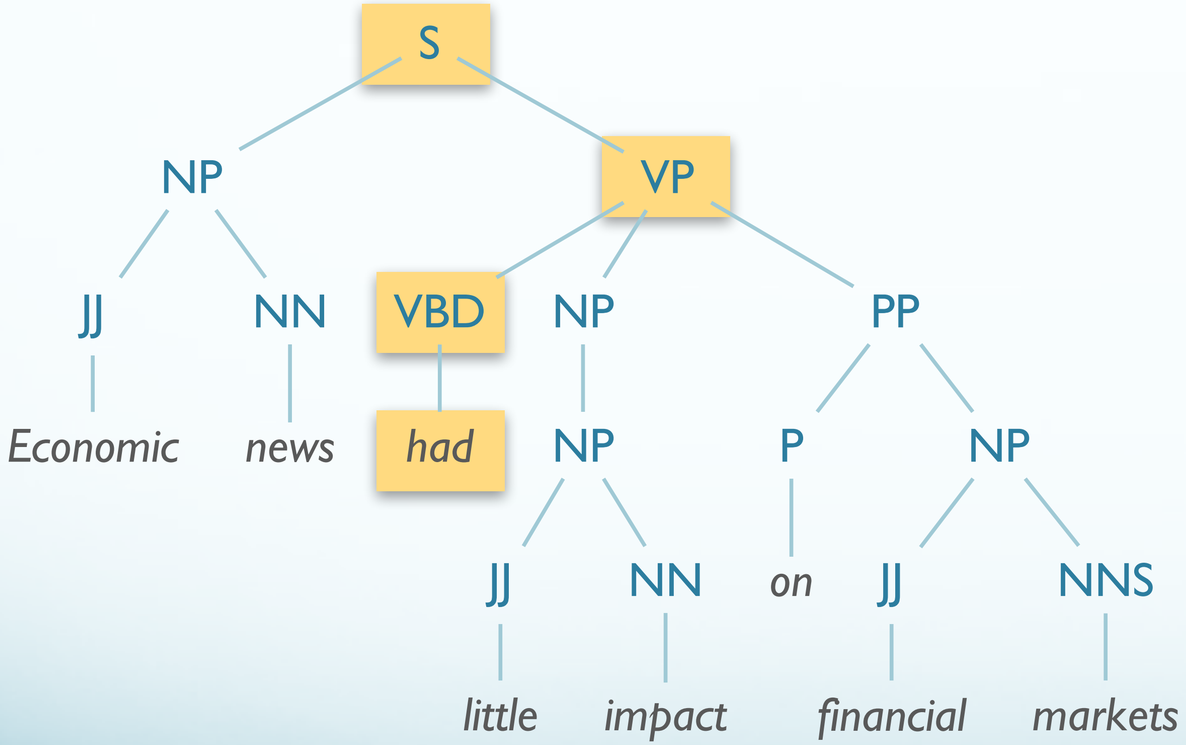
Conversion: PS → DS

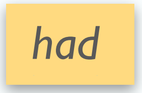
- Can convert Phrase Structure (PS) to Dependency Structure (DS)
  - …without the dependency labels (semantic roles)
- Algorithm:
  - Identify all head children in PS
  - Make head of each non-head-child depend on head of head-child
  - Use a *head percolation* table to determine headedness

Conversion: PS → DS

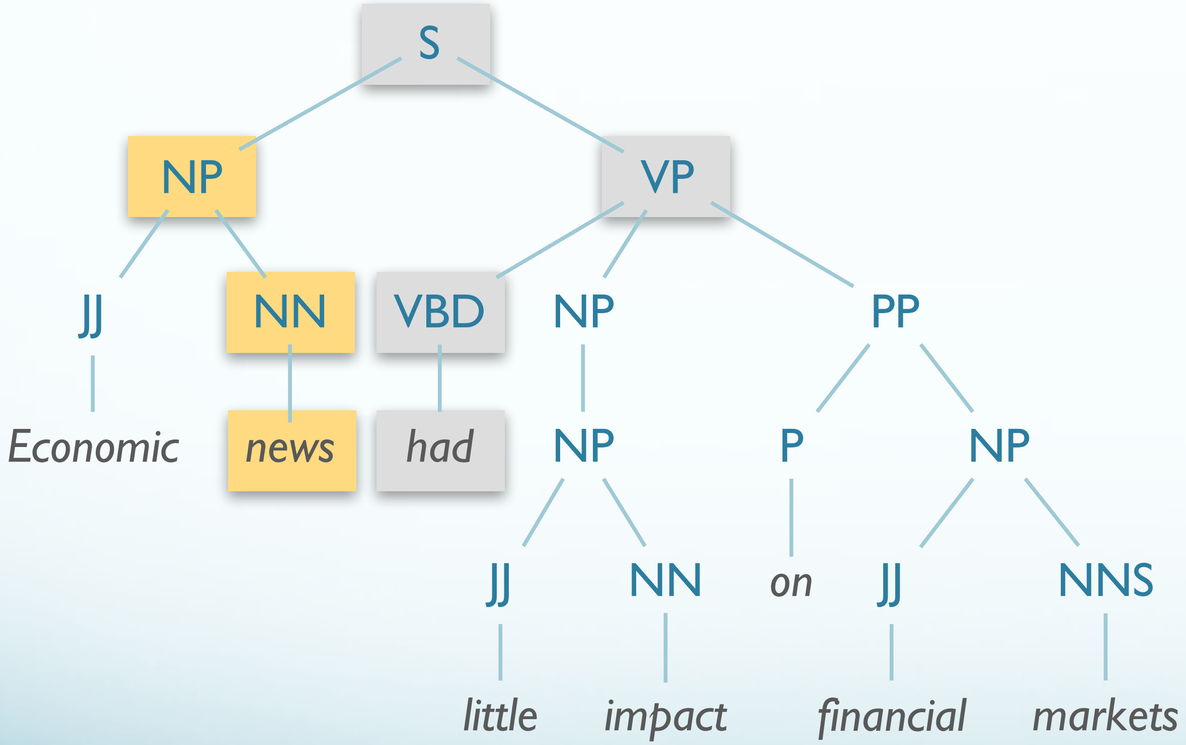


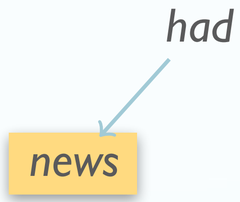
Conversion: PS → DS



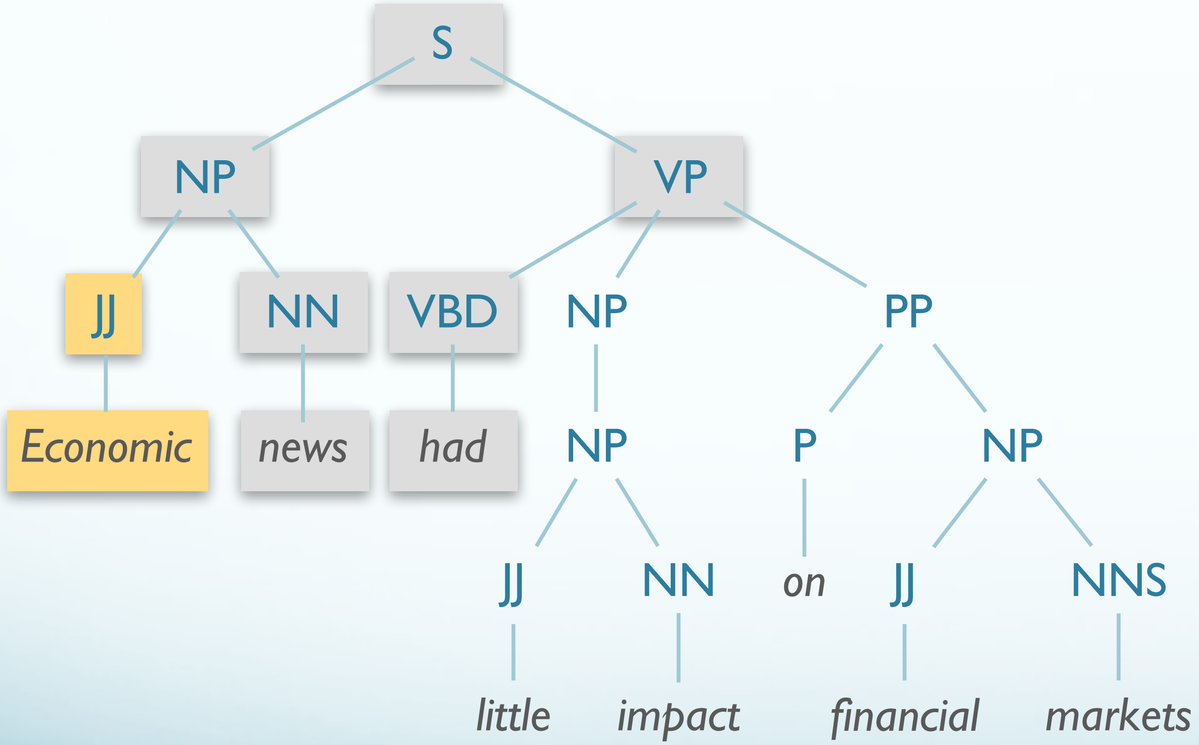


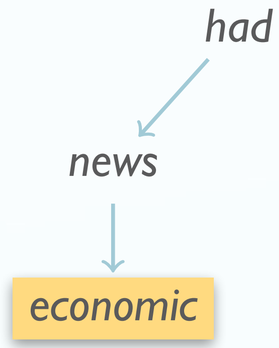
Conversion: PS → DS



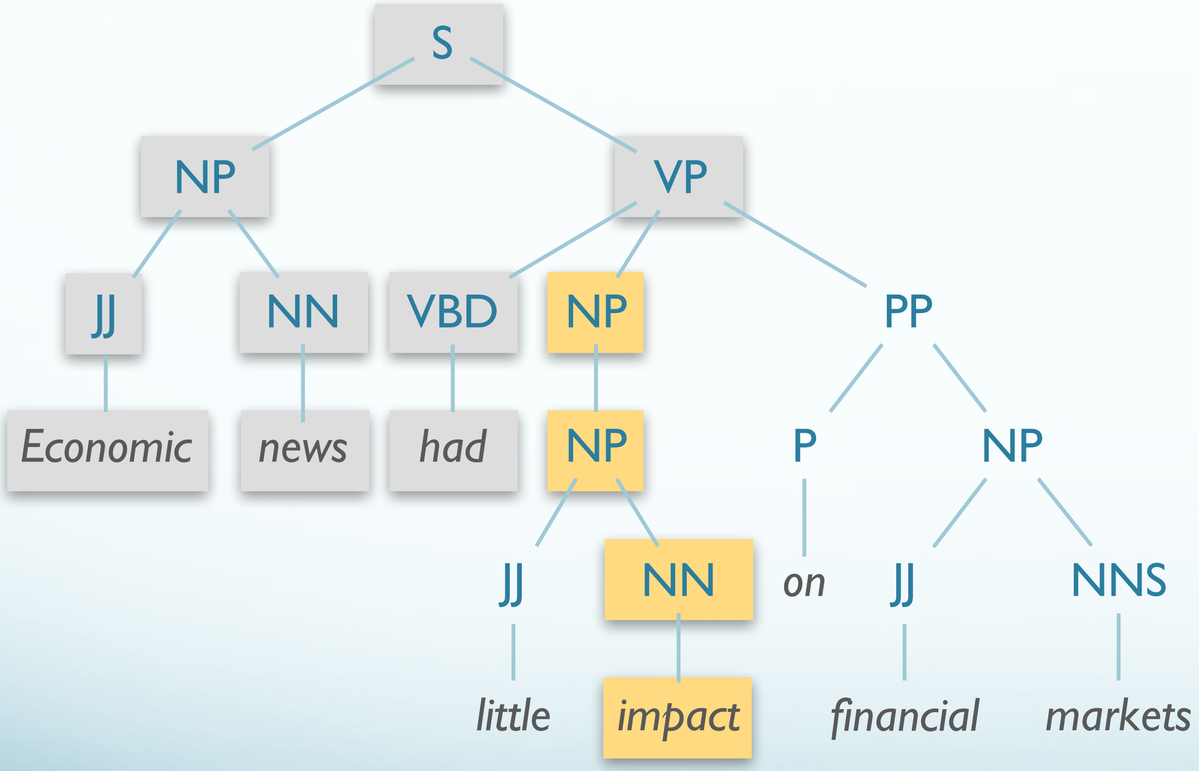


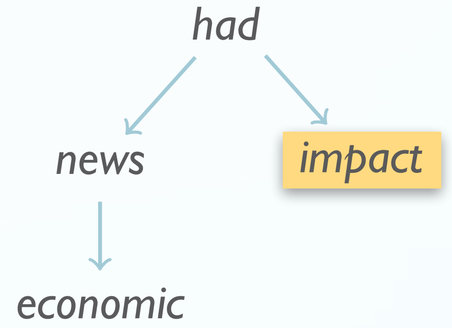
Conversion: PS → DS



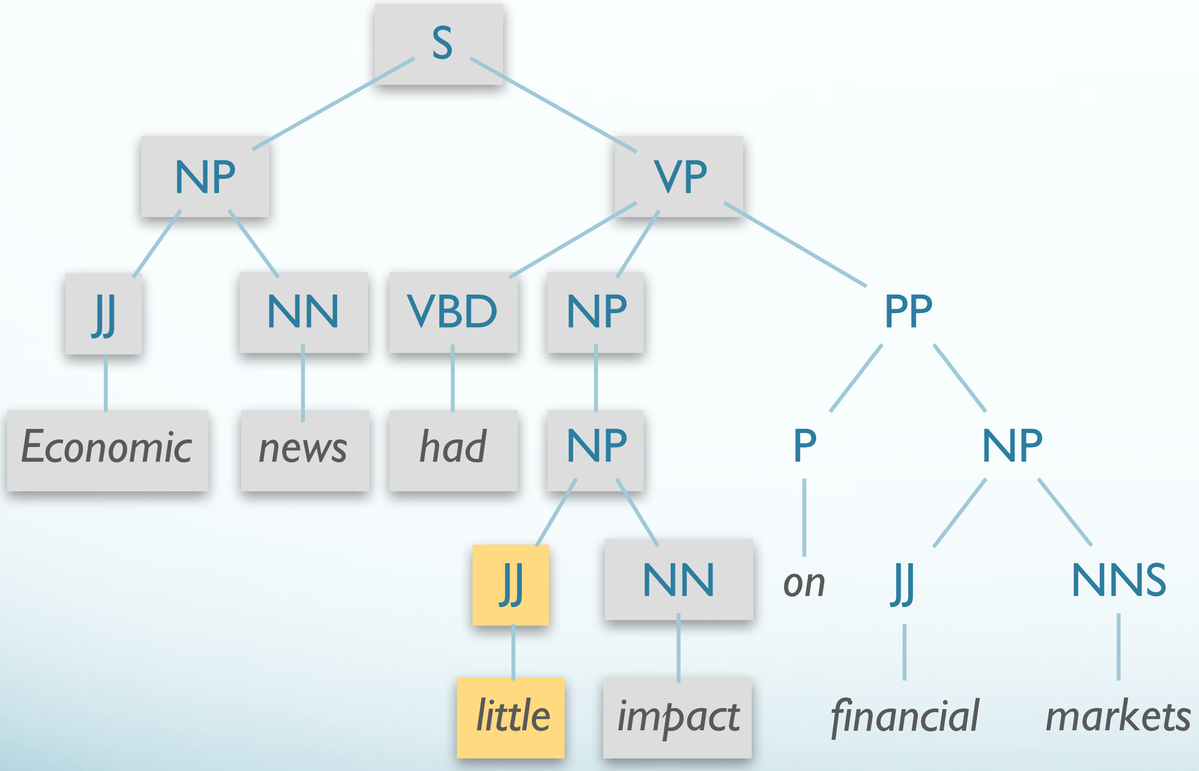


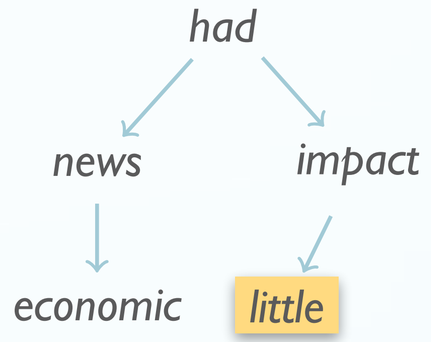
Conversion: PS → DS



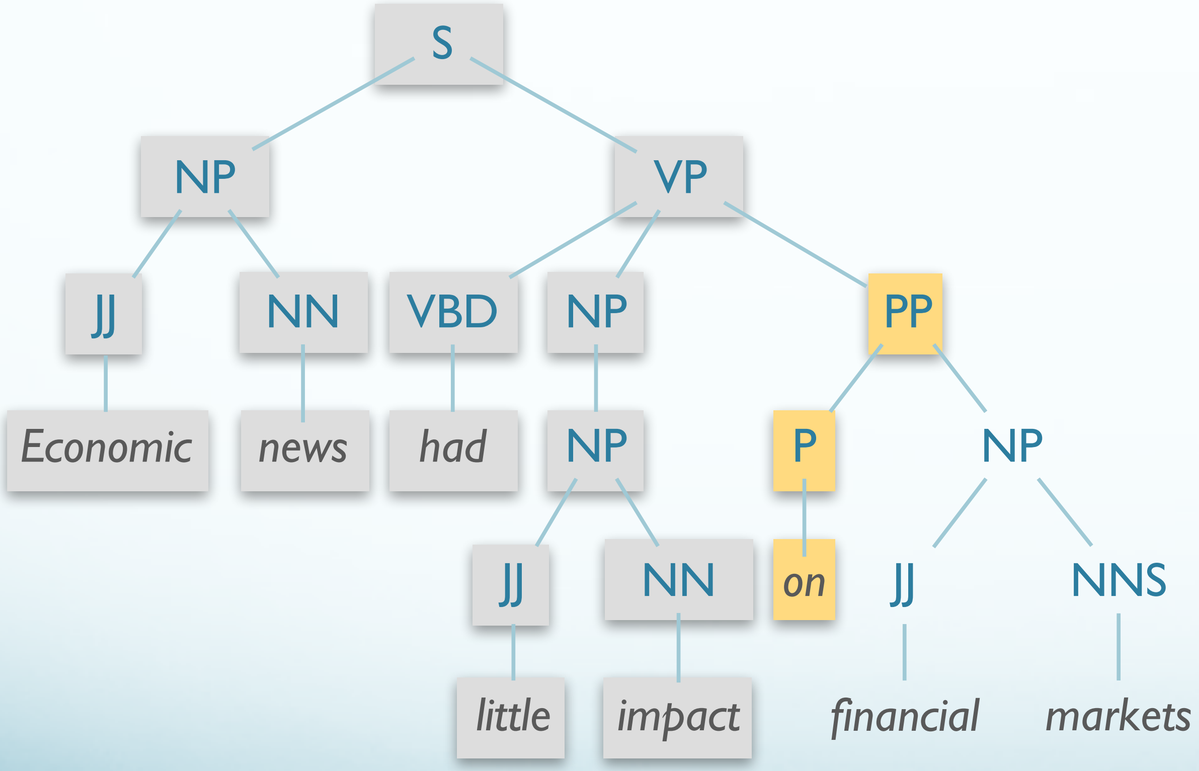


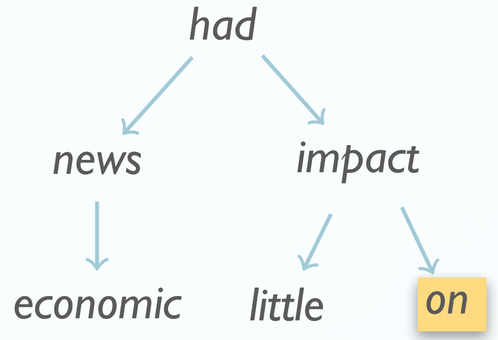
Conversion: PS → DS



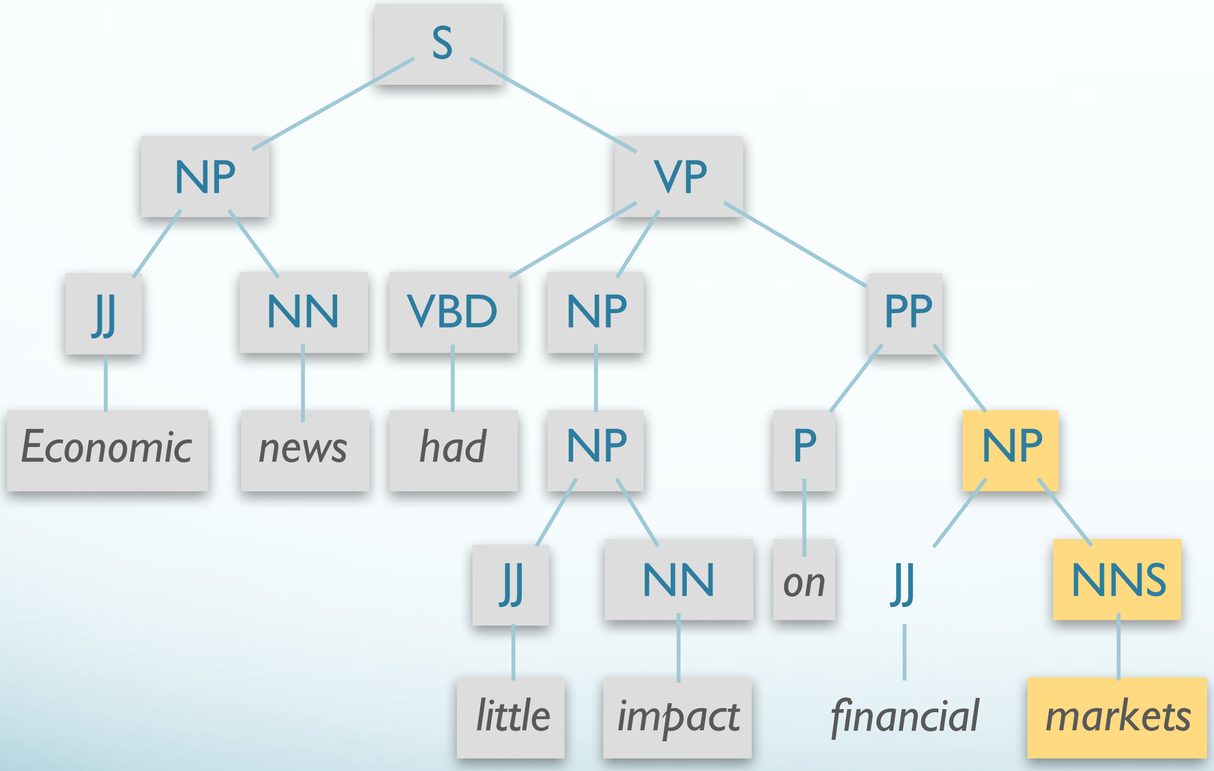


Conversion: PS → DS

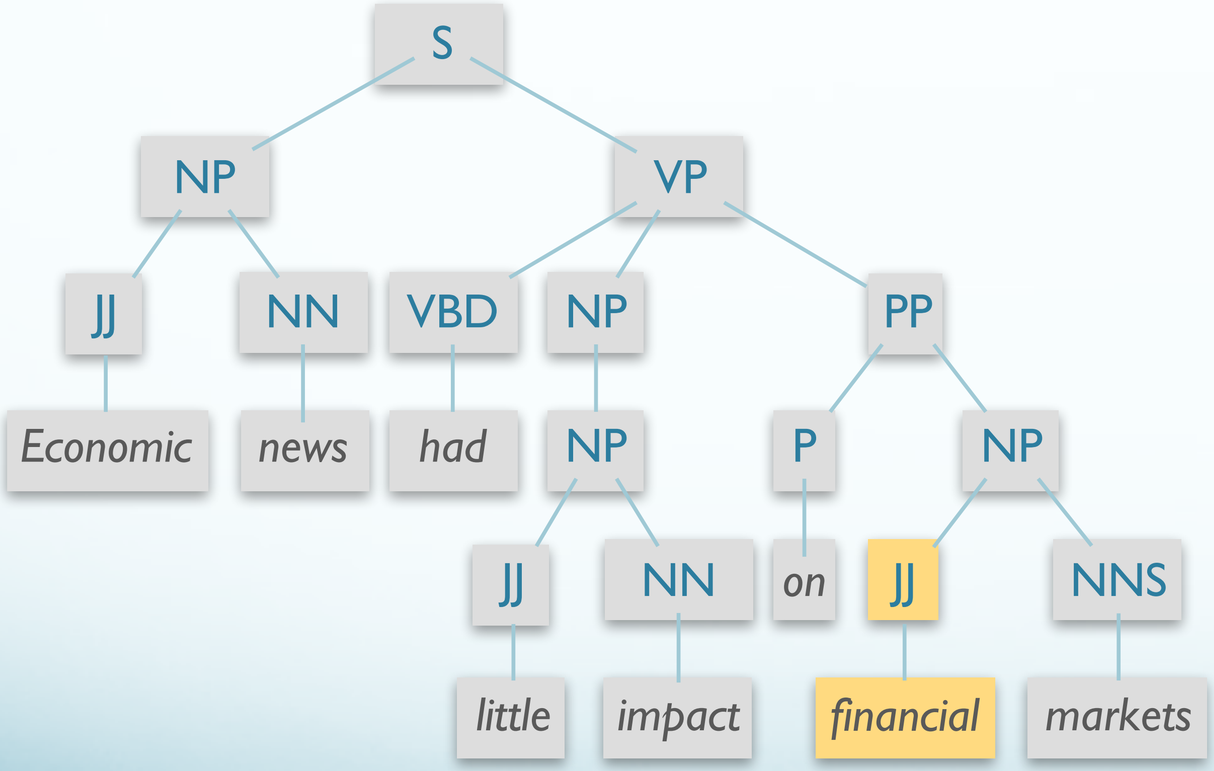




Conversion: PS → DS



Conversion: PS → DS



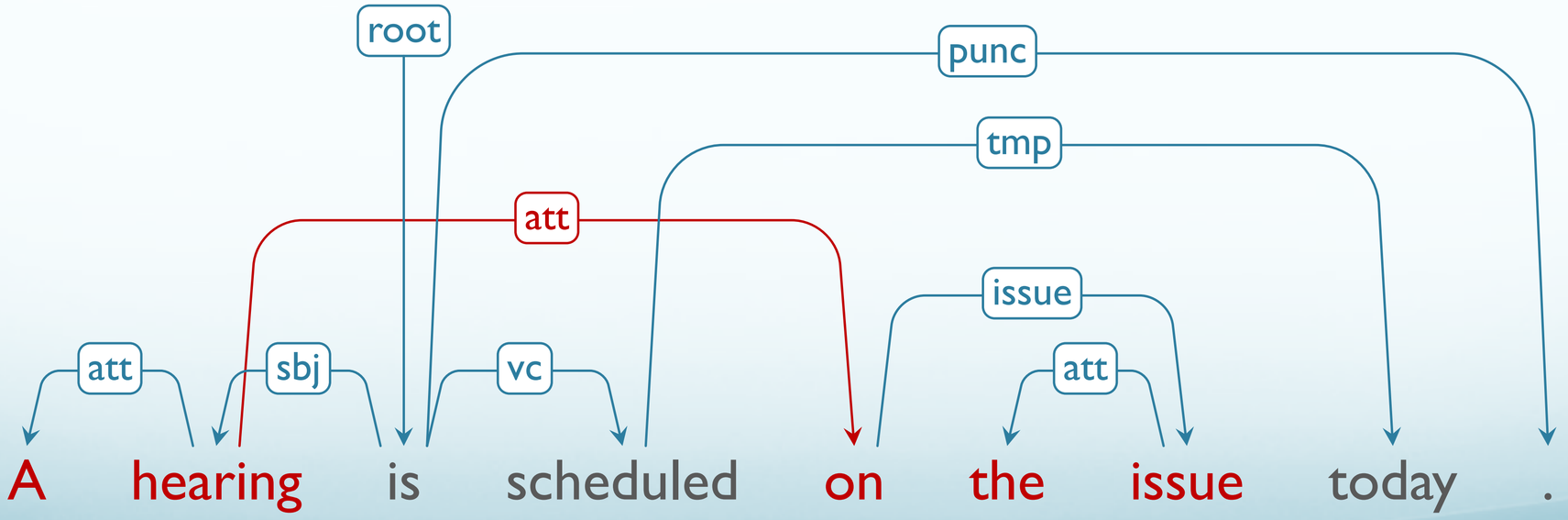
Head Percolation Table

- Finding the head of an NP:
  - If the rightmost word is preterminal, return
  - …else search Right→Left for first child which is *NN, NNP, NNPS…*
  - …else search Left→Right for first child which is *NP*
  - …else search Right→Left for first child which is *$, ADJP, PRN*
  - …else search Right→Left for first child which is *CD*
  - …else search Right→Left for first child which is *JJ, JJS, RB or QP*
  - …else return rightmost word.

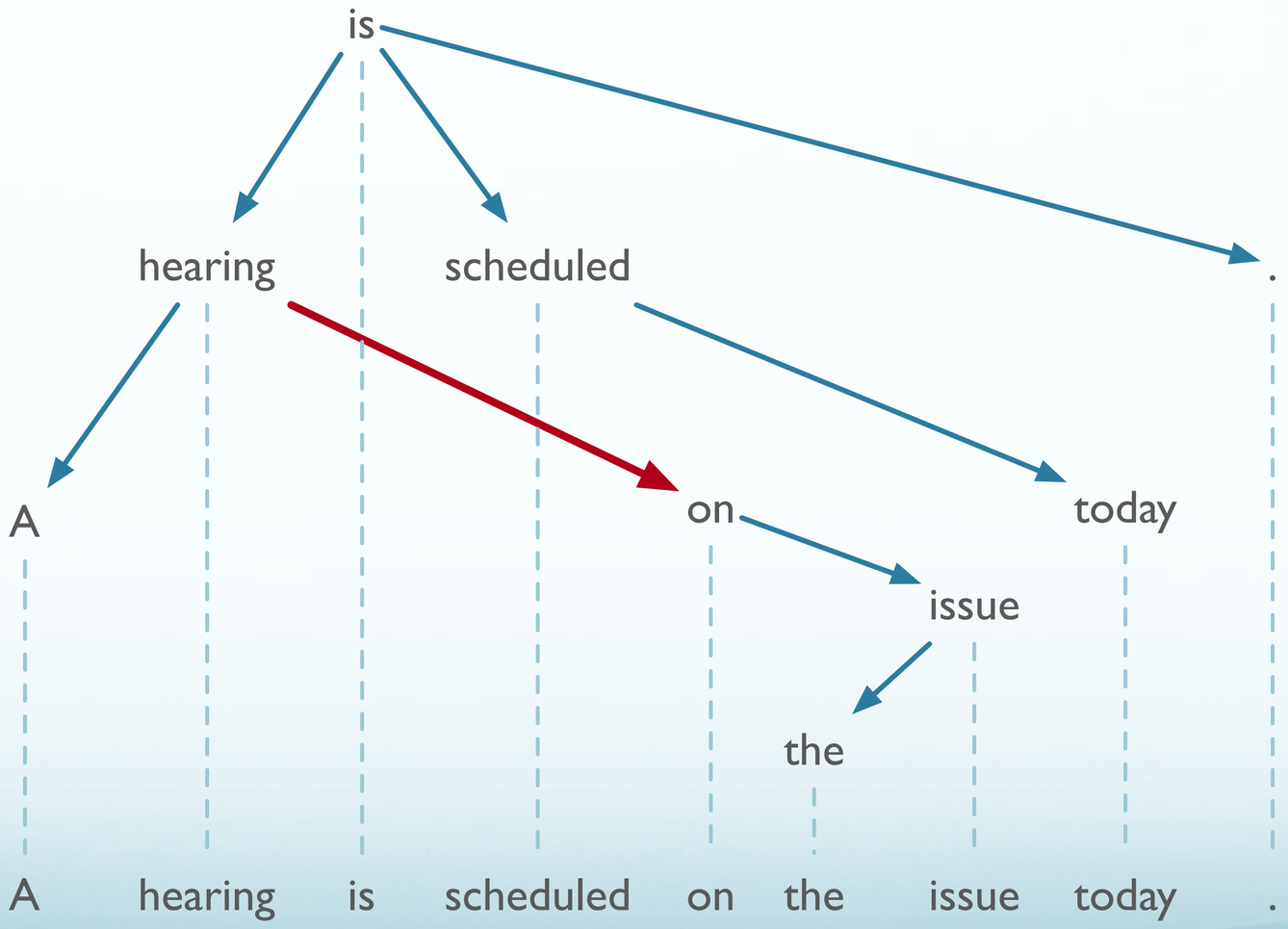
From J&M Page 411, via Collins (1999)

Conversion: DS → PS

- Can map any *projective* dependency tree to PS tree
- Projective:
  - Does not contain "crossing" dependencies w.r.t. word order

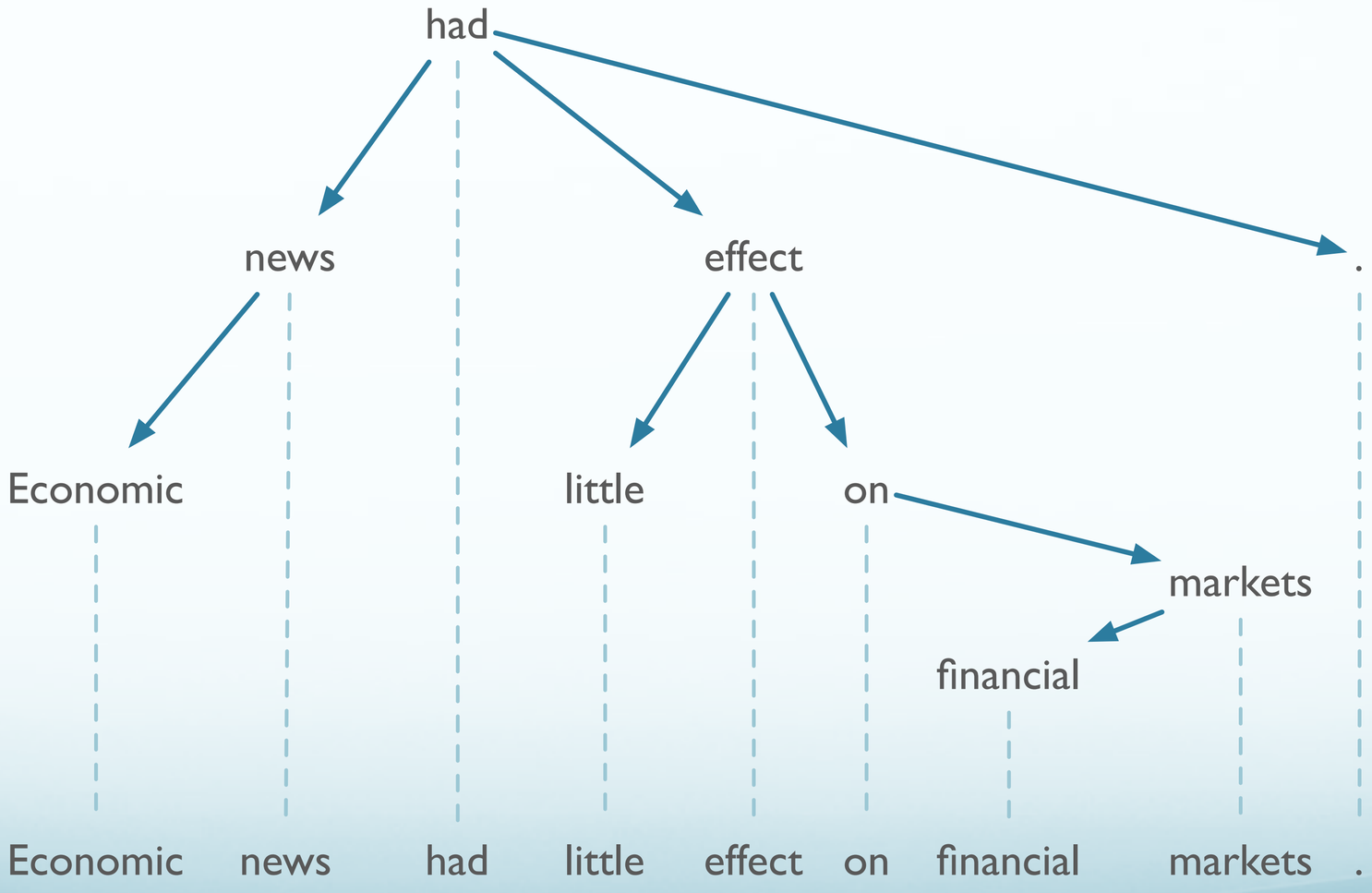


Non-Projective DS

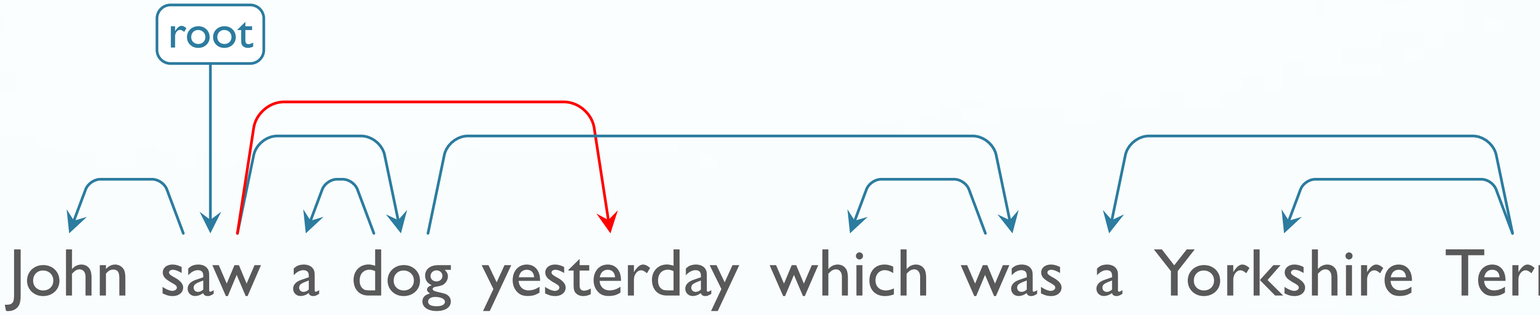


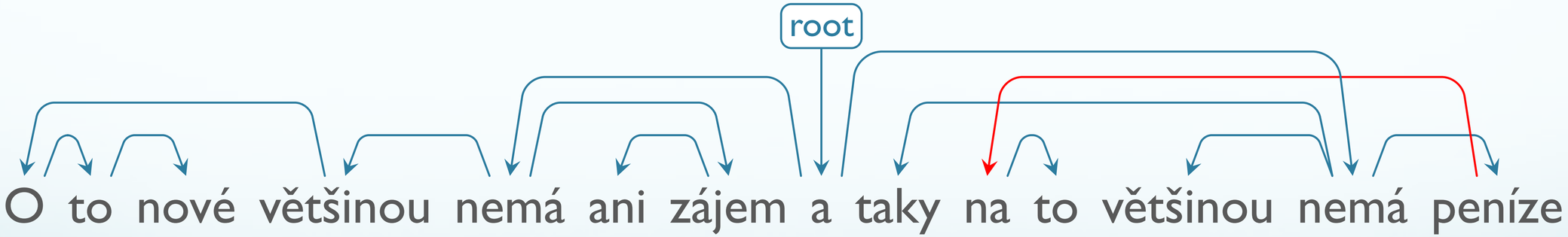
= *Projection*

Projective DS



More Non-Projective Parses





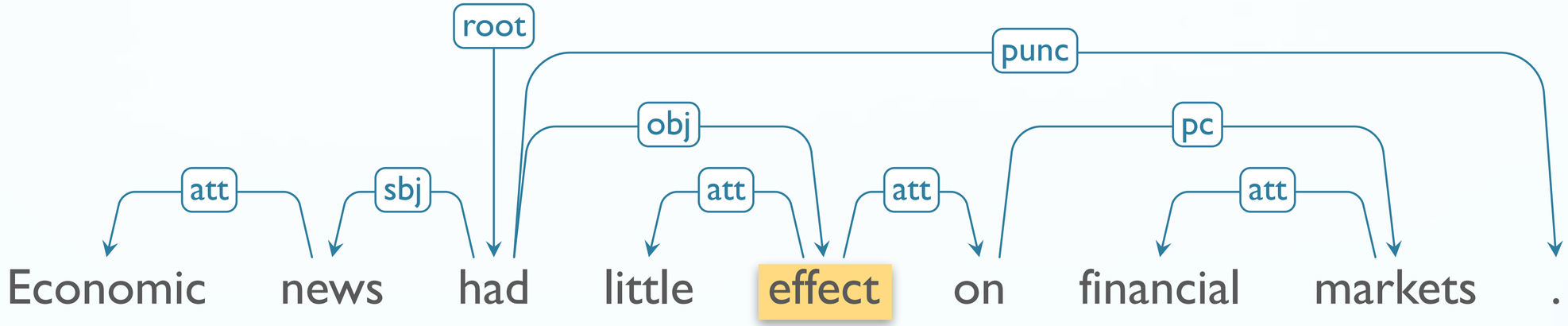
He is mostly not even interested in the new things and in most cases, he has no money for it either.

From McDonald et. al, 2005

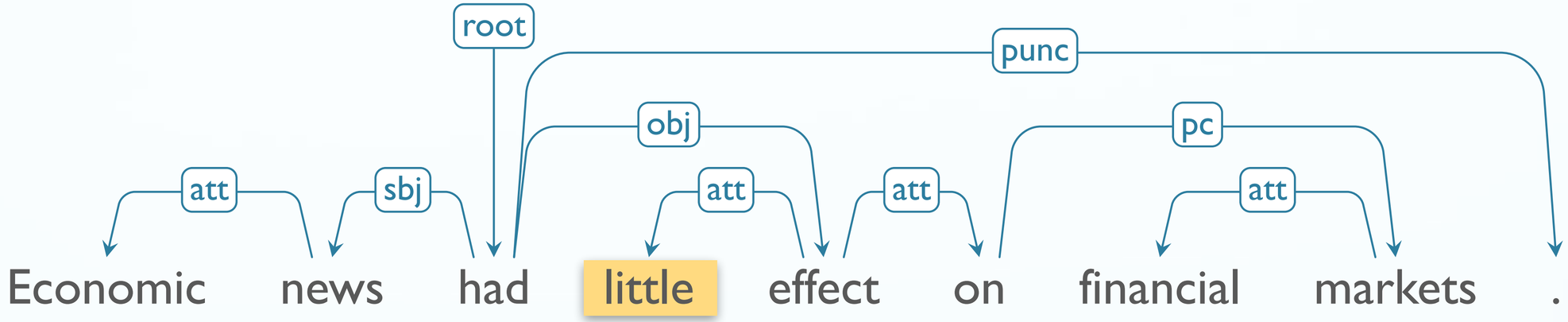
Conversion: DS → PS

- For each node w with outgoing arcs…
  - …convert the subtree w and its dependents $t_1,\ldots,t_n$ to a new subtree:
    - Nonterminal: X_w
    - Child: w
    - Subtrees $t_1,\ldots,t_n$ in original sentence order

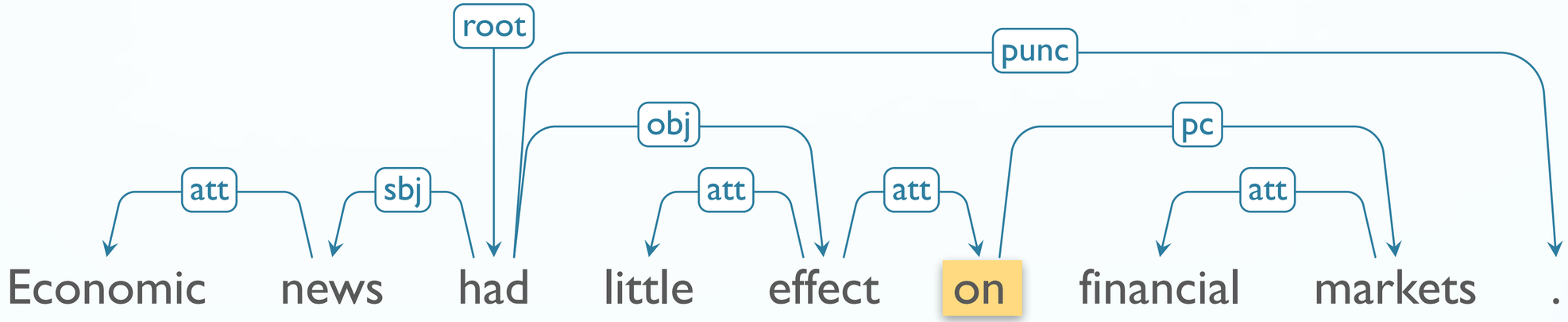
Conversion: DS → PS

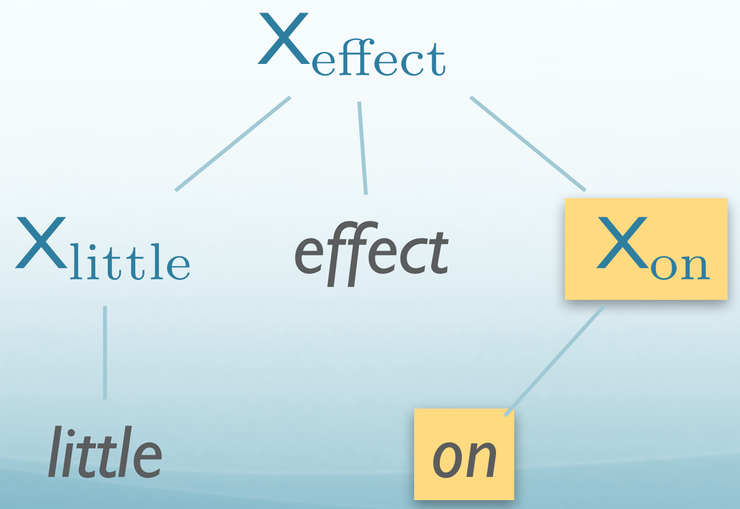


Conversion: DS → PS

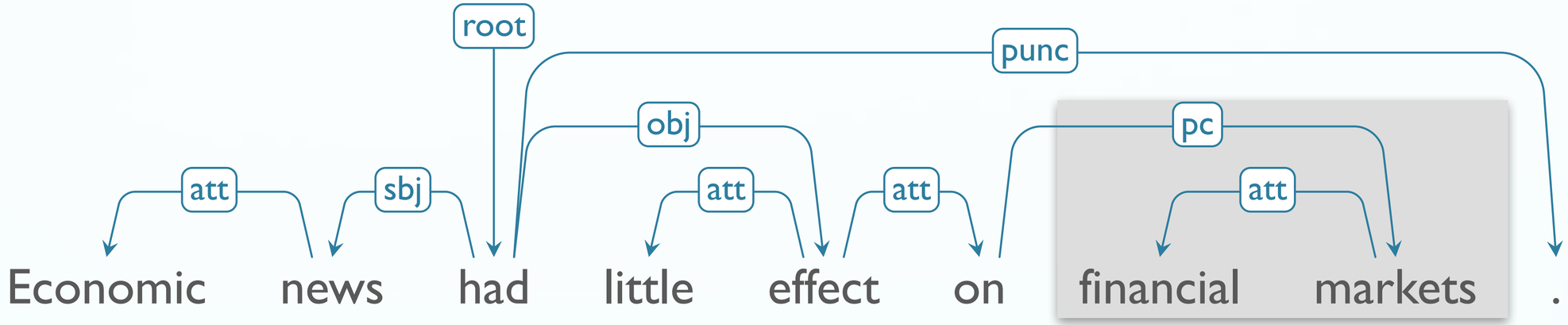


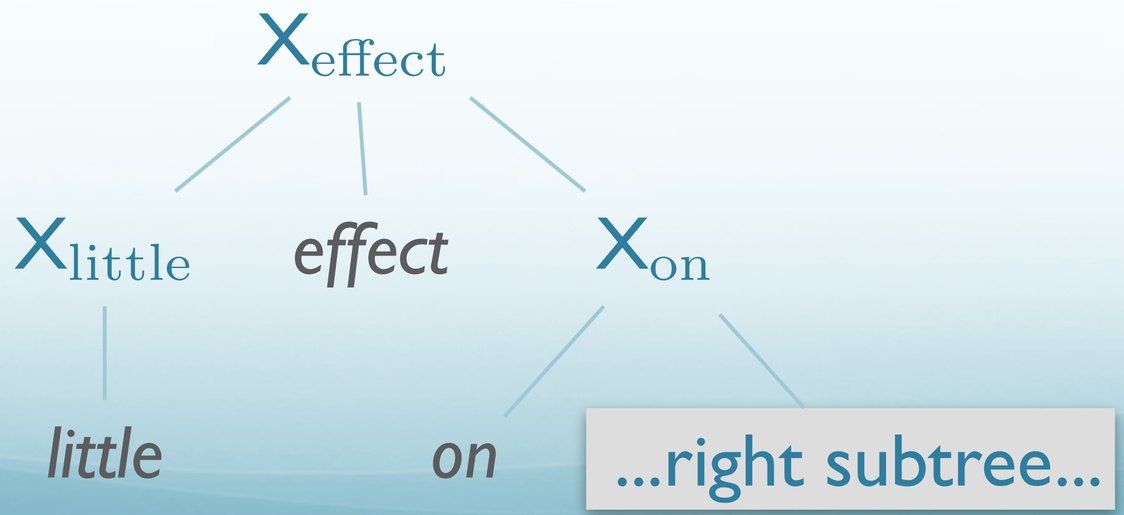
Conversion: DS → PS





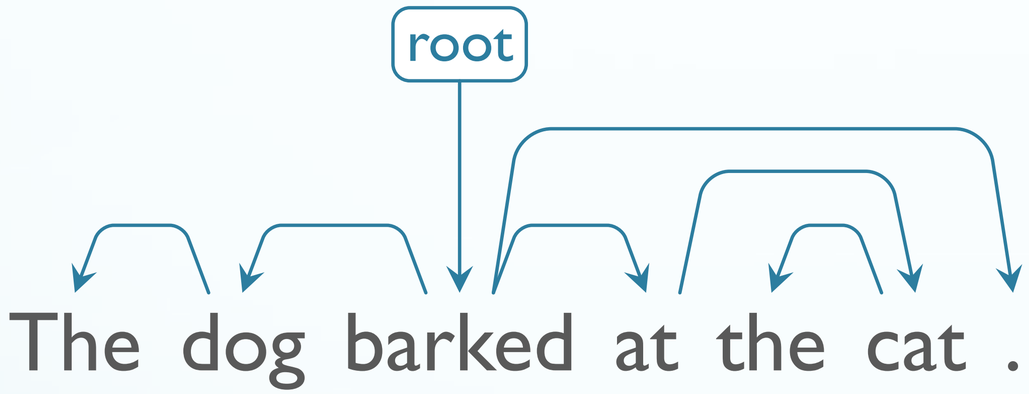
Conversion: DS → PS

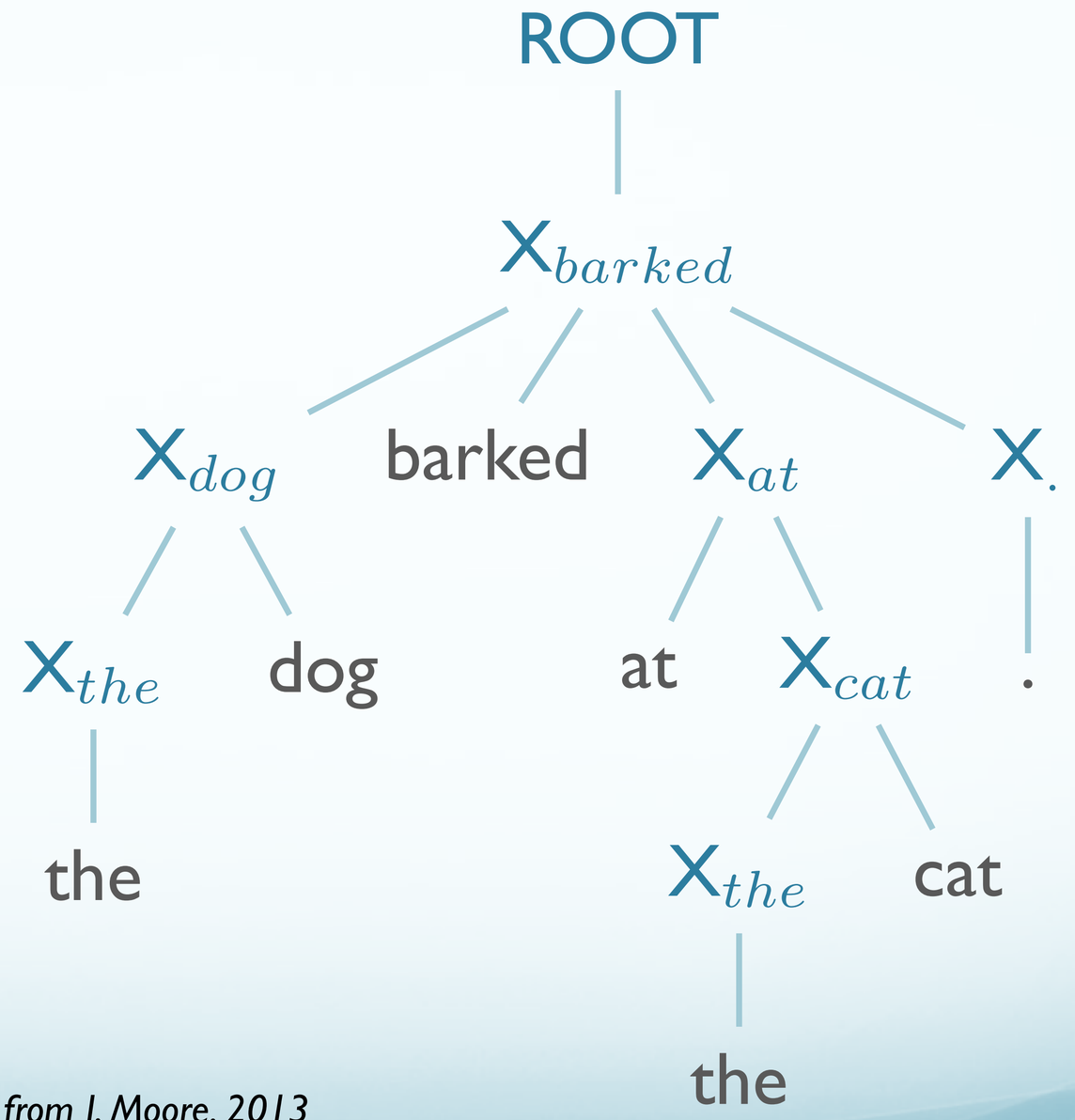




Conversion: DS → PS

- What about labeled dependencies?
  - Can attach labels to nonterminals associated with non-heads
  - e.g. $X_{little} \rightarrow X_{little:nmod}$
- Doesn't create typical PS trees
  - *Does* create fully lexicalized, labeled, context-free trees
- Can be parsed with any standard CFG parser





Example from J. Moore, 2013

Roadmap

- Dependency Grammars
  - Definition
  - Motivation:
    - Limitations of Context-Free Grammars
- **Dependency Parsing**
  - By conversion to CFG
  - **By Graph-based models**
  - By transition-based parsing

Graph-based Dependency Parsing

- Goal: Find the highest scoring dependency tree $\hat{T}$ for sentence S
  - If S is unambiguous, T is the correct parse
  - If S is ambiguous, T is the highest scoring parse
- Where do scores come from?
  - Weights on dependency edges by learning algorithm
  - Learned from dependency treebank
- Where are the grammar rules?
  - …there aren't any! All data-driven.

Graph-based Dependency Parsing

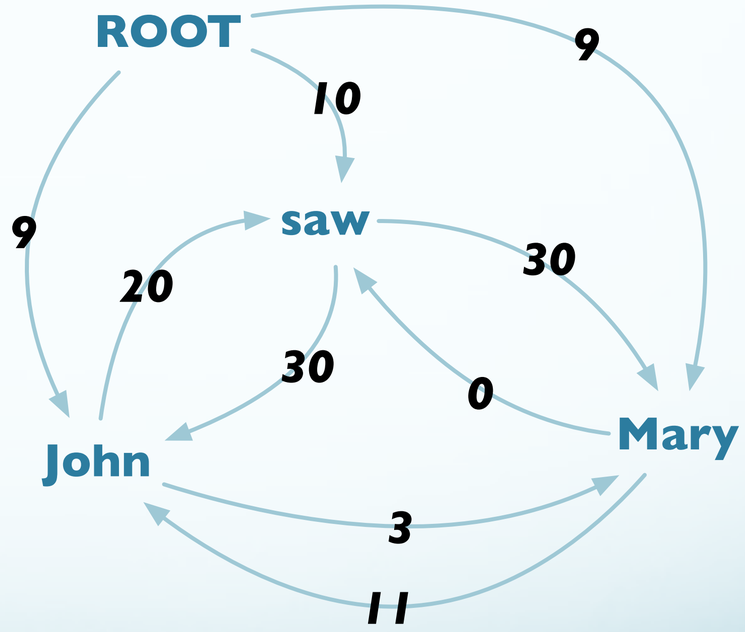
- Map dependency parsing to Maximum Spanning Tree (MST)
- Build fully connected initial graph:
  - Nodes: words in sentence to parse
  - Edges: directed edges between all words
    - + Edges from ROOT to all words
- Identify maximum spanning tree
  - Tree s.t. all nodes are connected
  - Select such tree with highest weight

Graph-based Dependency Parsing

- Arc-factored model:
  - Weights depend on end nodes & link
  - Weight of tree is sum of participating arcs

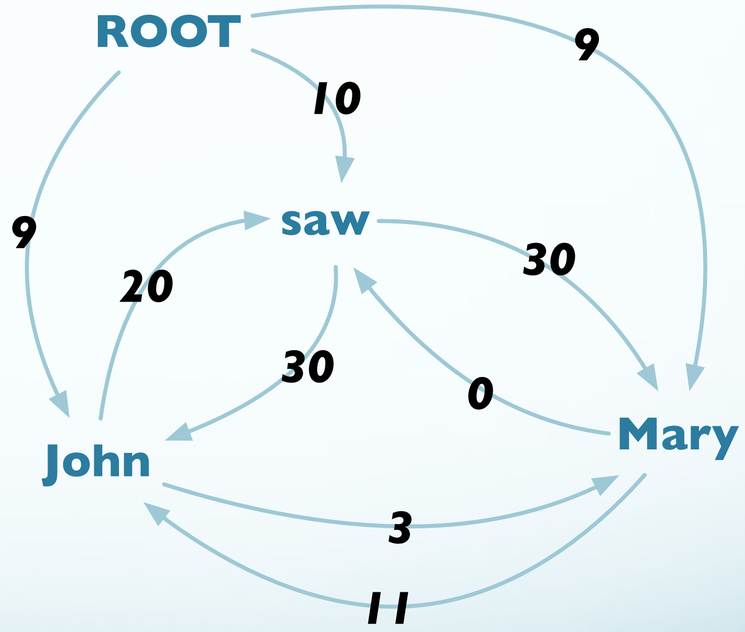
Initial Graph: *(McDonald et al, 2005b)*

- *John saw Mary*
  - All words connected: ROOT only has outgoing arcs
- Goal: Remove arcs to create a tree covering all words
  - Resulting tree is parse



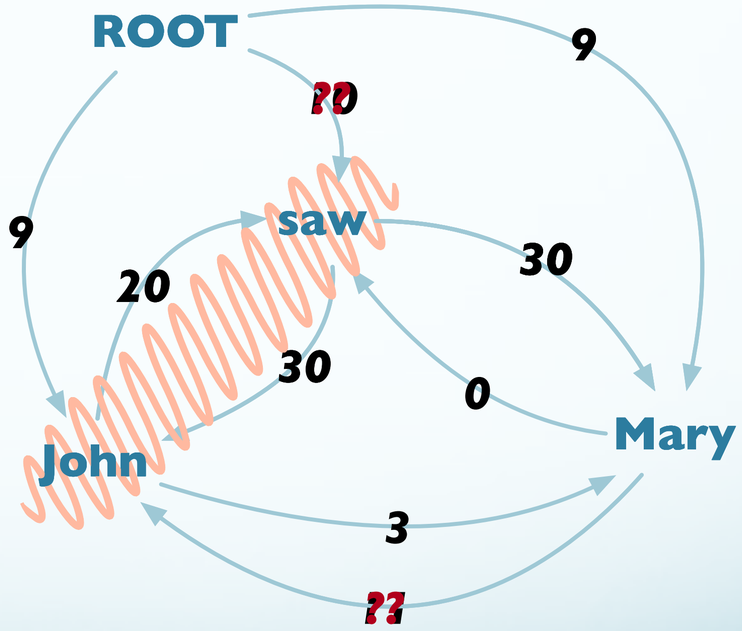
Maximum Spanning Tree

- McDonald et al, 2005 use variant of Chu-Liu-Edmonds algorithm for MST (CLE)
- Sketch of algorithm:
  - For each node, greedily select incoming arc with max weight
  - If the resulting set of arcs forms a tree, this is the MST.
  - If not, there must be a cycle.
    - "Contract" the cycle: Treat it as a single vertex
    - Recalculate weights into/out of the new vertex
    - Recursively do MST algorithm on resulting graph
- Running time: naïve: $O(n^3)$; Tarjan: $O(n^2)$
  - Applicable to non-projective graphs



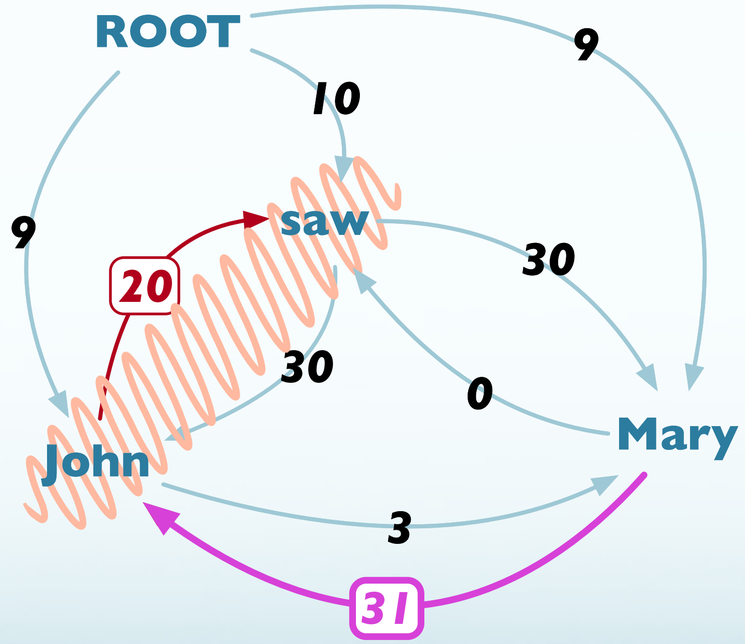
Step 1 & 2

- Find, for each word, the highest scoring incoming edge.
- Is it a tree?
  - No, there's a cycle.
- Collapse the cycle
- And re-examine the edges again



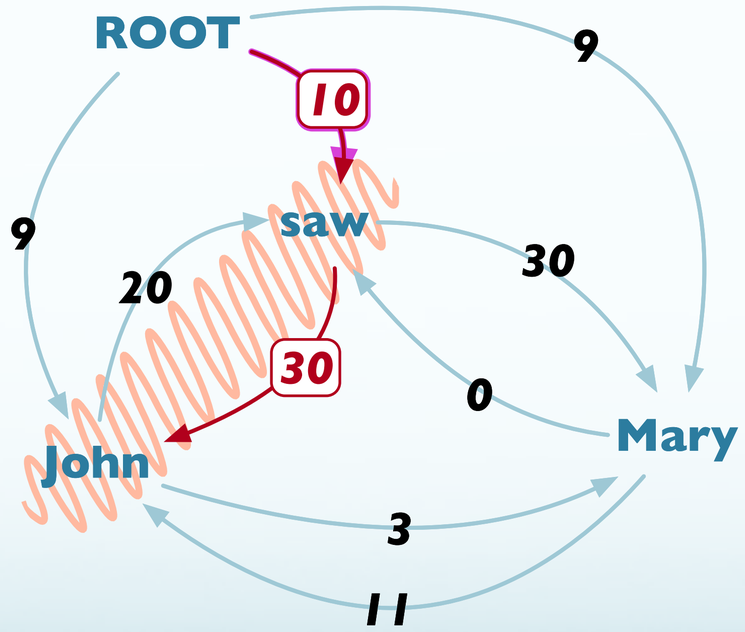
Calculating Weights for Collapsed Vertex

$$\mathrm{s}(\ \mathrm{Mary},\ C\ )\ 11 + 20 = 31$$



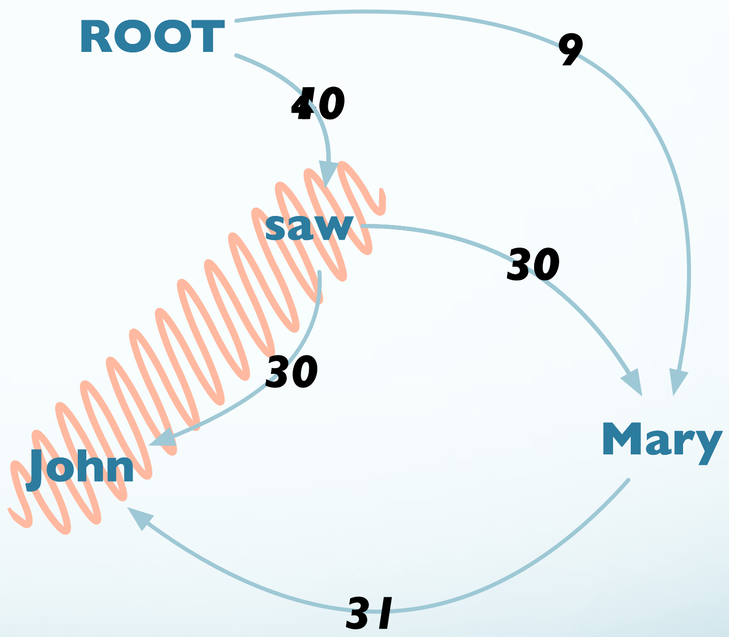
Calculating Weights for Collapsed Vertex

$$\mathbf{s}(\ \mathbf{ROOT},\ C\ )\ 10 + 30 = 40$$



Step 3

- With cycle collapsed, recurse on step 1:
- Keep highest weighted incoming edge for each edge
- Is it a tree?
  - **Yes!**
  - …but must recover collapsed portions.



Learning Weights

- Weights for arc-factored model learned from dependency treebank
  - Weights learned for tuple (w_i, w_j, l)
- McDonald et al, 2005a employed discriminative ML
  - MIRA (Crammer and Singer, 2003)
- Operates on vector of local features

Features for Learning Weights

- Simple categorical features for $(w_i,\ L,\ w_j)$ including:
  - Identity of w_i (or char 5-gram prefix), POS of w_i
  - Identity of w_j (or char 5-gram prefix), POS of w_j
  - Label of L, direction of L
  - Number of words between $w_i,\ w_j$
  - POS tag of w_{i-1}, POS tag of w_{i+1}
  - POS tag of w_{j-1}, POS tag of w_{j+1}

- Features conjoined with direction of attachment and distance between words

Dependency Parsing

- Dependency Grammars:
  - Compactly represent predicate–argument structure
  - Lexicalized, localized
  - Natural handling of flexible word order
- Dependency parsing:
  - Conversion to phrase structure trees
  - Graph-based parsing (MST), efficient non-proj $\mathbf{O}(n^2)$
  - Next time: *Transition-based parsing*

Further Reading

- Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online Large-Margin Training of Dependency Parsers. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 91–98. May. **[link]**
- Ryan McDonald, Fernando Pereira, K. Ribarov, and Jan Hajič. 2005b. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 523–530. Association for Computational Linguistics. **[link]**
- Sandra Kübler, Ryan McDonald, and Joakim Nivre. 2009. *Dependency Parsing*. Morgan & Claypool. **[link]**
- Jason M. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th Conference on Computational Linguistics*, pages 340–345. Association for Computational Linguistics. **[link]**
- Michael Collins. 1999. *Head-Driven Statistical Models For Natural Language Parsing*. **[link]**