Introduction to NLP

255.

Dependency Grammars

Dependency structure

blue house

- blue
  - modifier, dependent, child, subordinate
- house
  - head, governor, parent, regent

Dependency Structure

Unionized workers are usually better paid than their non-union counterparts.

1 2 3 4 5 6 7 8 9 10

Dependency Structure

Unionized workers are usually better paid than their non-union counterparts.

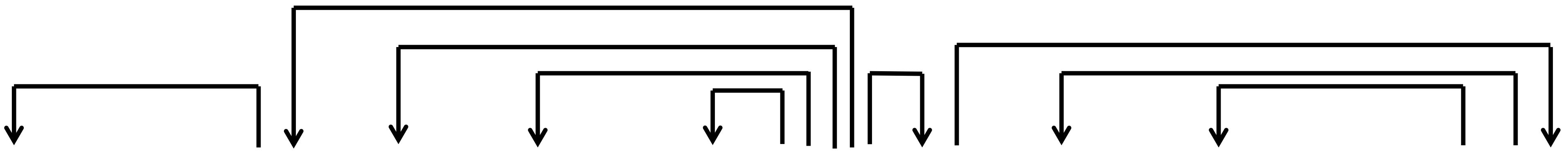
1 2 3 4 5 6 7 8 9 10

Dependency Structure

Unionized workers are usually better paid than their non-union counterparts.

1 2 3 4 5 6 7 8 9 10

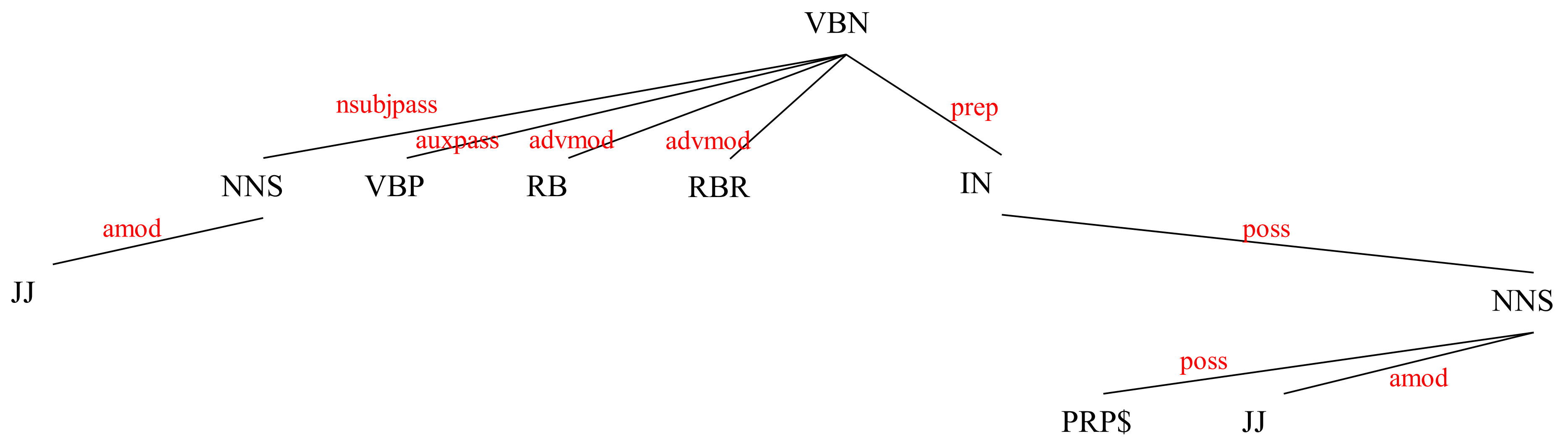
Dependency Structure



Unionized workers are usually better paid than their non-union counterparts.

1 2 3 4 5 6 7 8 9 10

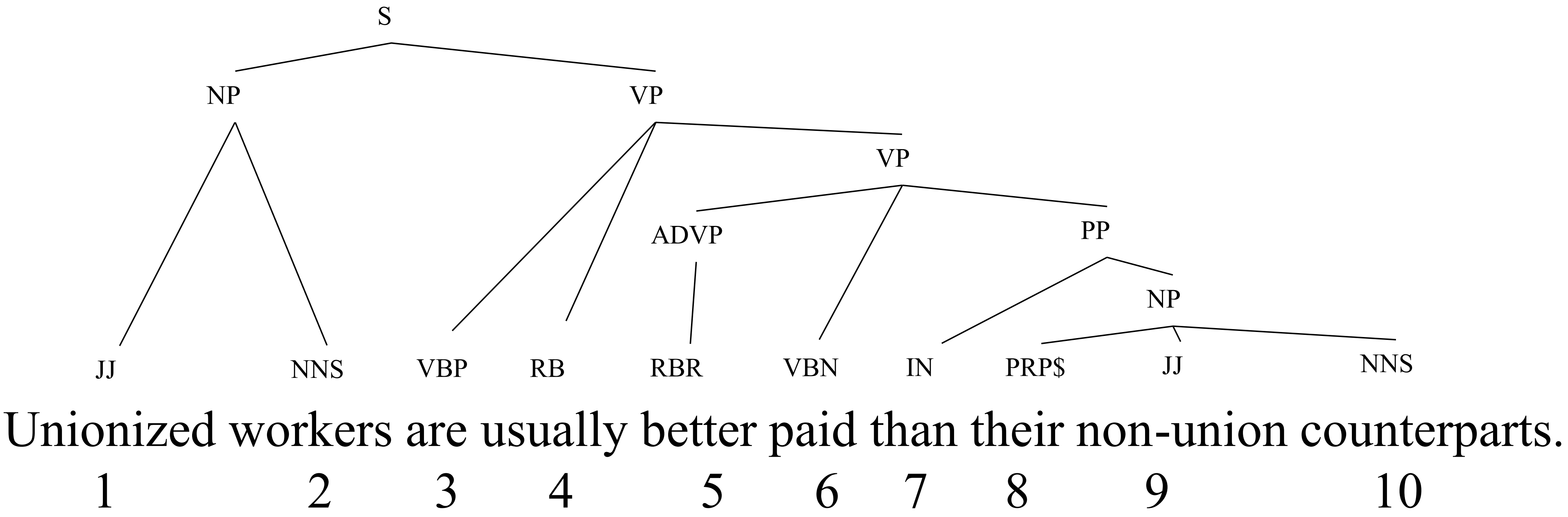
Other notations



Unionized workers are usually better paid than their non-union counterparts.

1 2 3 4 5 6 7 8 9 10

Phrase Structure

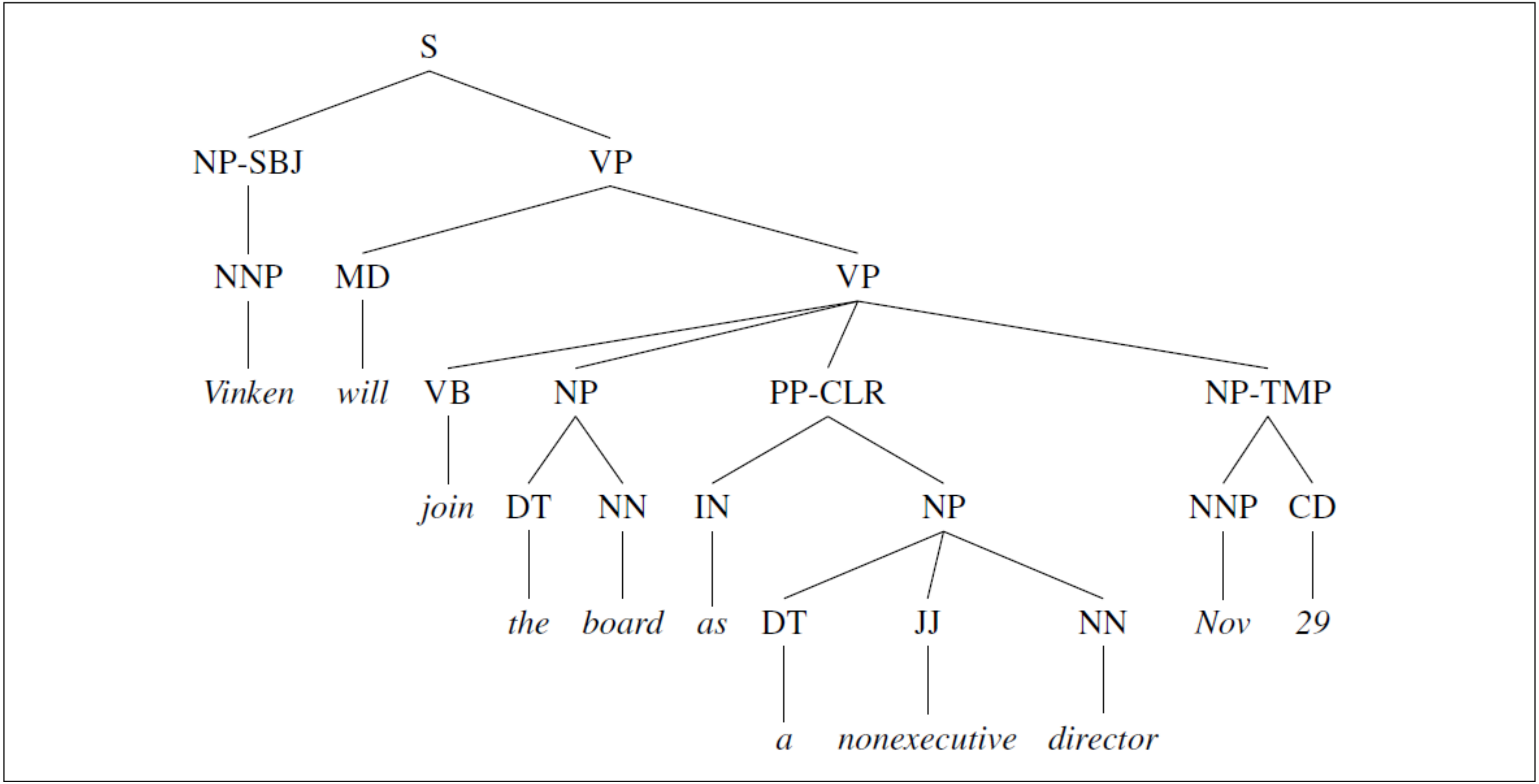


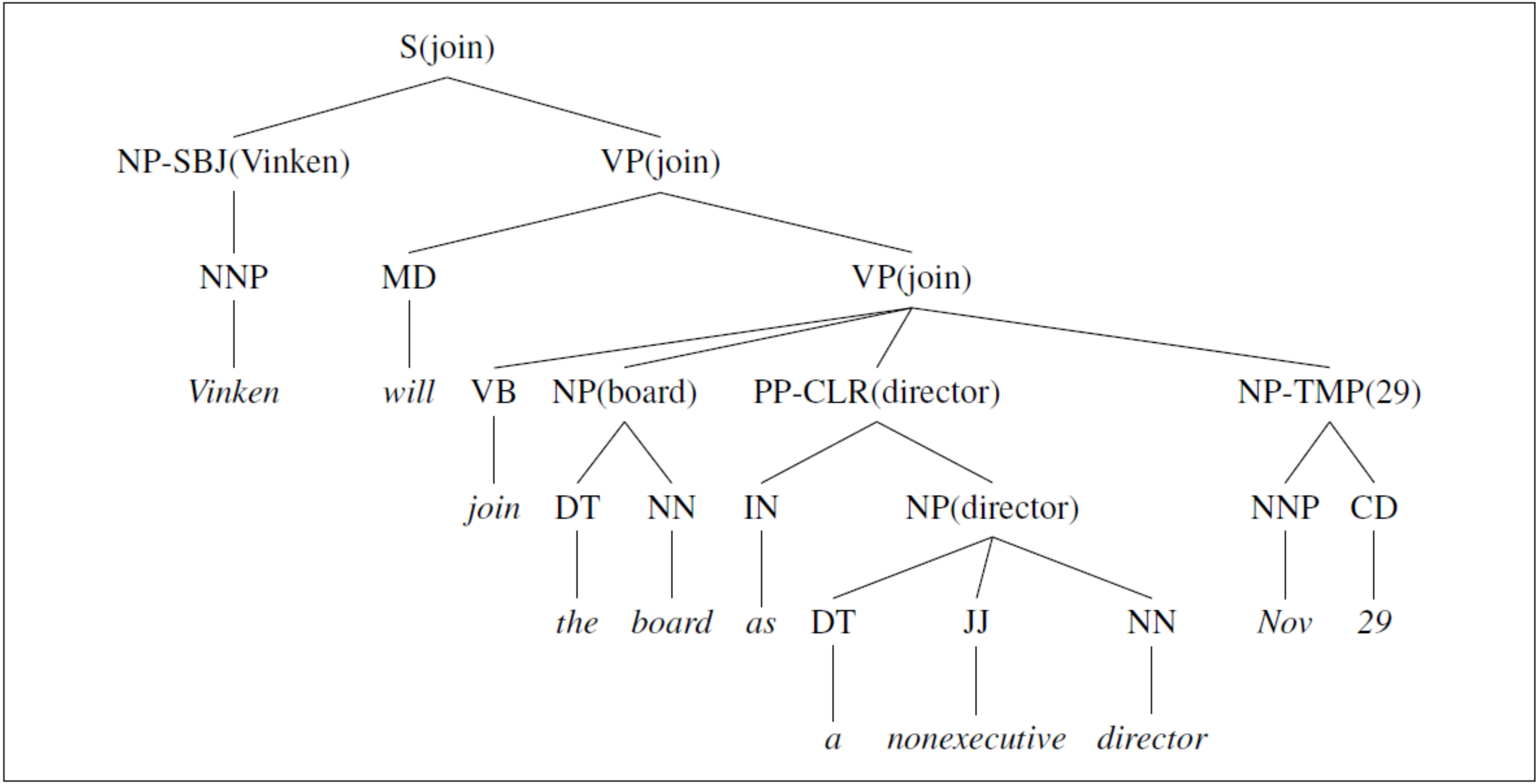
Dependency grammars

- Characteristics
  - Lexical/syntactic dependencies between words
  - The top-level predicate of a sentence is the root
  - Simpler to parse than context-free grammars
  - Particularly useful for free word order languages
  - Older idea compared to constituent grammars (as far back as Pāṇini (5th century BCE)

How to identify the heads

- H=head, M=modifier
  - H determines the syntactic category of the construct
  - H determines the semantic category of the construct
  - H is required; M may be skipped
  - Fixed linear position of M with respect to H


S
NP-SBJ
VP
NNP
MD
VP
Vinken
will
VB
NP
PP-CLR
NP-TMP
join
DT
NN
IN
NP
NNP
CD
the
board
as
DT
JJ
NN
Nov
29
a
nonexecutive
director


S(join)
NP-SBJ(Vinken)
VP(join)
NNP
MD
VP(join)
Vinken
will
VB
NP(board)
PP-CLR(director)
NP-TMP(29)
join
DT
NN
IN
NP(director)
NNP
CD
the
board
as
DT
JJ
NN
Nov
29
a
nonexecutive
director

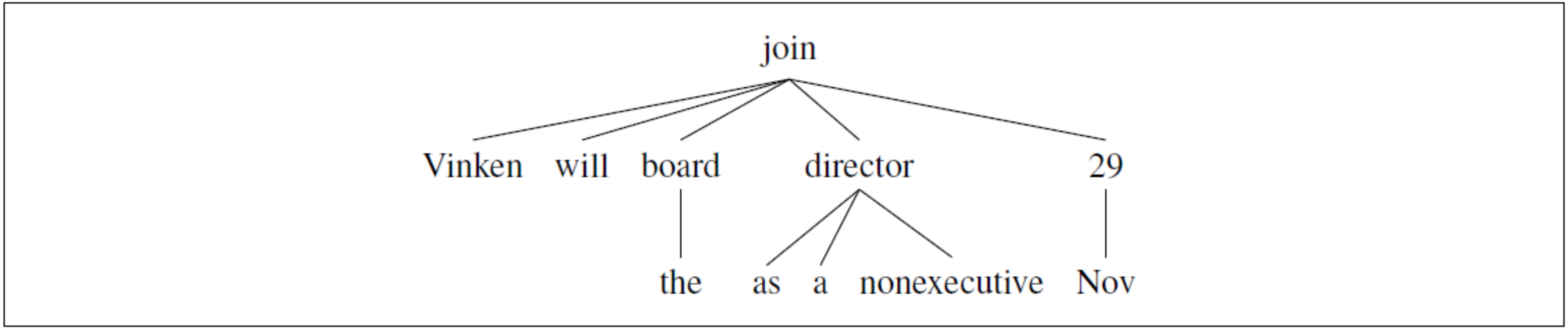
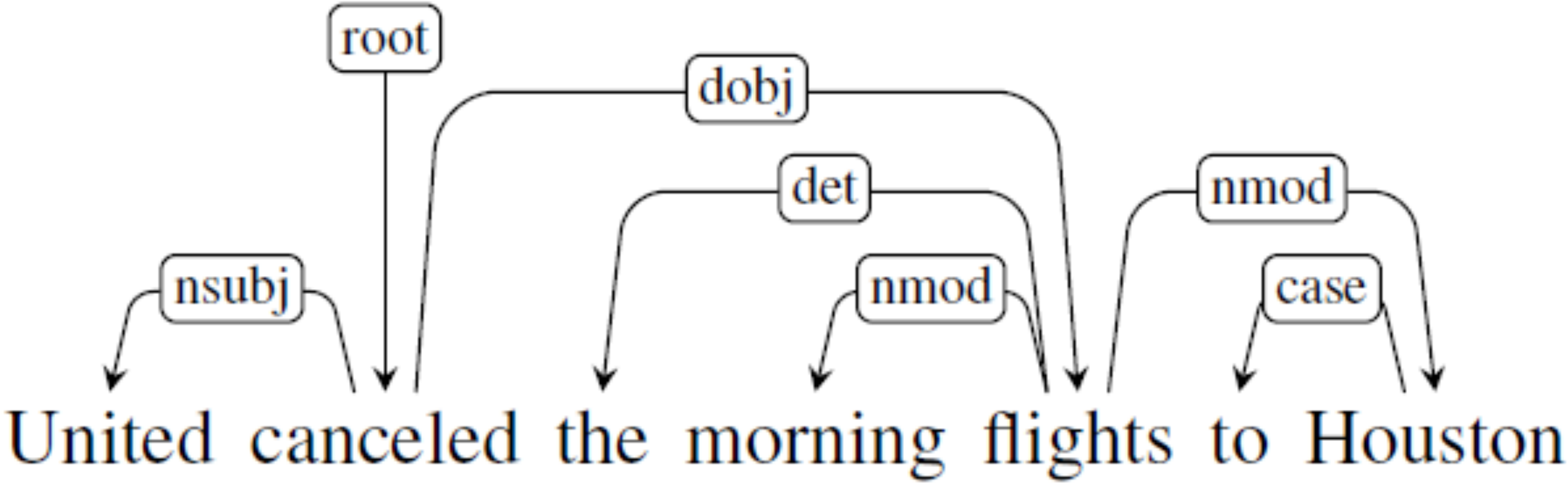


Figure 14.4 A phrase-structure tree from the *Wall Street Journal* component of the Penn Treebank 3.

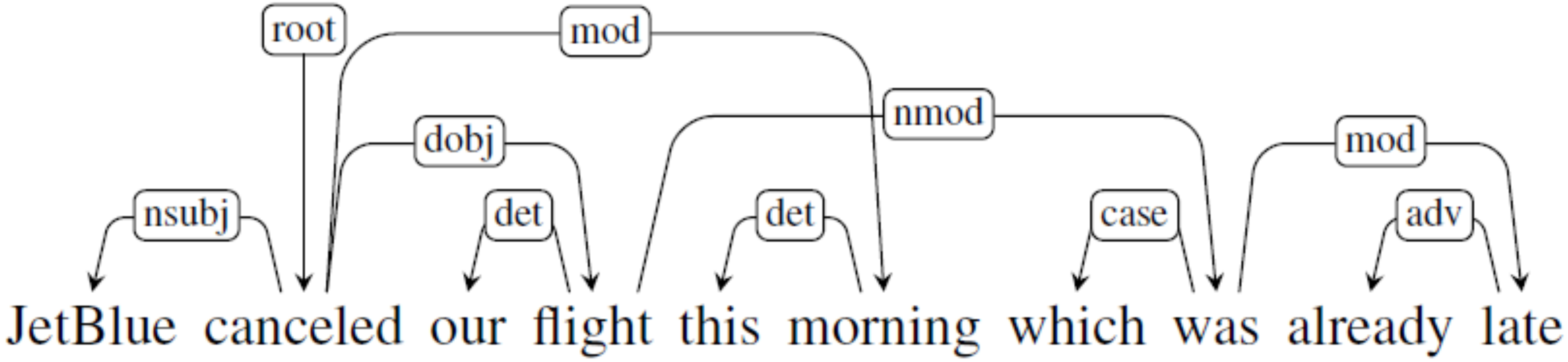
Clausal Argument Relations	Description
NSUBJ	Nominal subject
DOBJ	Direct object
IOBJ	Indirect object
CCOMP	Clausal complement
XCOMP	Open clausal complement
Nominal Modifier Relations	**Description**
NMOD	Nominal modifier
AMOD	Adjectival modifier
NUMMOD	Numeric modifier
APPOS	Appositional modifier
DET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

Figure 14.2 Selected dependency relations from the Universal Dependency set. (de Marneffe et al., 2014)


root
dobj
det
nmod
nsubj
nmod
case
United canceled the morning flights to Houston

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.

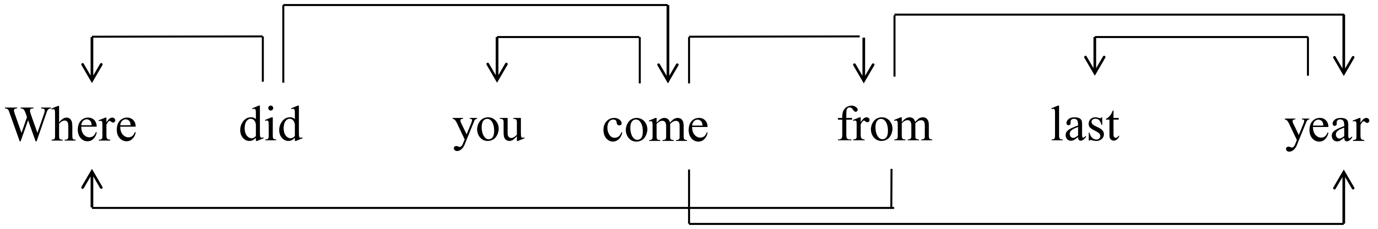

root
mod
nmod
dobj
mod
nsubj
det
det
case
adv
JetBlue canceled our flight this morning which was already late

Non-Projectivity

- Rare in English
- Topicalization
  - Cats, I like a lot.
- Extraposition
  - The pizza is ready with pepperoni.

Non-projectivity

Output of (the non-projective) MSTParser



Output of Stanford parser

advmod(come-4, Where-1)
aux(come-4, did-2)
nsubj(come-4, you-3)
root(ROOT-0, come-4)
prep(come-4, from-5)
amod(year-7, last-6)
pobj(from-5, year-7)

1	Where	Where	WRB	WRB	-	2	SBJ	-	-
2	did	did	VBD	VBD	-	0	ROOT	-	-
3	you	you	PRP	PRP	-	4	SBJ	-	-
4	come	come	VBP	VBP	-	2	VC	-	-
5	from	from	IN	IN	-	4	DIR	-	-
6	last	last	JJ	JJ	-	7	NMOD	-	-
7	year	year	NN	NN	-	5	PMOD	-	-
8	?	?	.	.	-	2	P	-	-

Notes

- How to extend a projective method for non-projective parses
  - Use a SWAP operator (Nivre 2009)
- Not clear what to do with conjunctions
  - “cats, dogs, and hamsters”
  - Options: “cats” or “and”

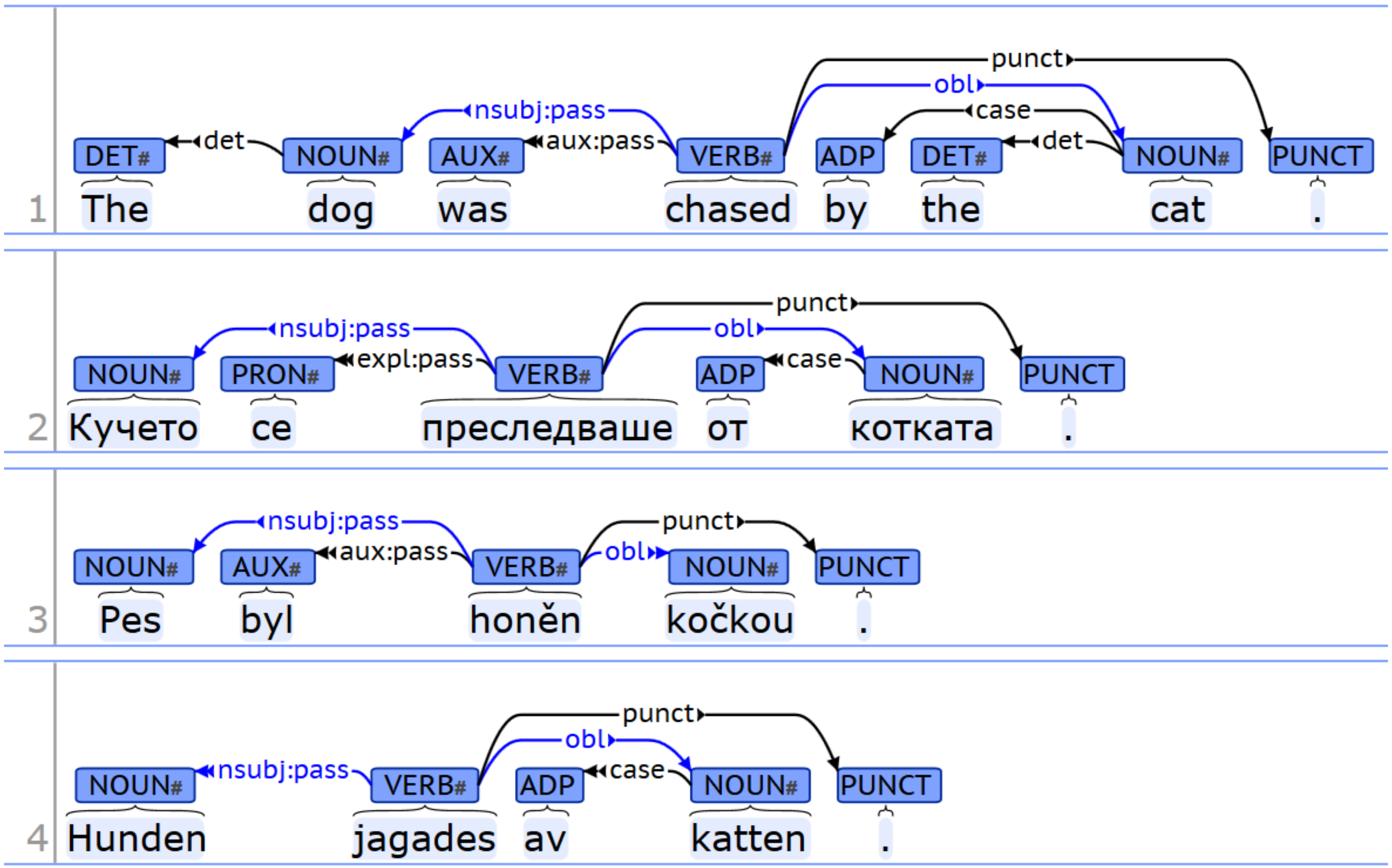
Rate of Non-Projectivity

	#T	#S	#T/#S	%NST	%NPR	%NPS	IR
Arabic	54	1.5	37.2	8.8	0.4	11.2	Yes
Bulgarian	190	12.8	14.8	14.4	0.4	5.4	No
Chinese	337	57	5.9	0.8	0.0	0.0	No
Czech	1249	72.7	17.2	14.9	1.9	23.2	Yes
Danish	94	5.2	18.2	13.9	1.0	15.6	No
Dutch	195	13.3	14.6	11.3	5.4	36.4	No
German	700	39.2	17.8	11.5	2.3	27.8	No
Japanese	151	17	8.9	11.6	1.1	5.3	No
Portuguese	207	9.1	22.8	14.2	1.3	18.9	Yes
Slovene	29	1.5	18.7	17.3	1.9	22.2	Yes
Spanish	89	3.3	27	12.6	0.1	1.7	No
Swedish	191	11	17.3	11.0	1.0	9.8	No
Turkish	58	5	11.5	33.1	1.5	11.6	No

Table 1: Treebank information; #T = number of tokens * 1000, #S = number of sentences * 1000, #T/#S = tokens per sentence, %NST = % of non-scoring tokens, %NPR = % of non-projective relations, %NPS = % of non-projective sentences, IR = has informative root labels

[CoNLL-X data: Hall and Nilsson 2006]

https://universaldependencies.org/



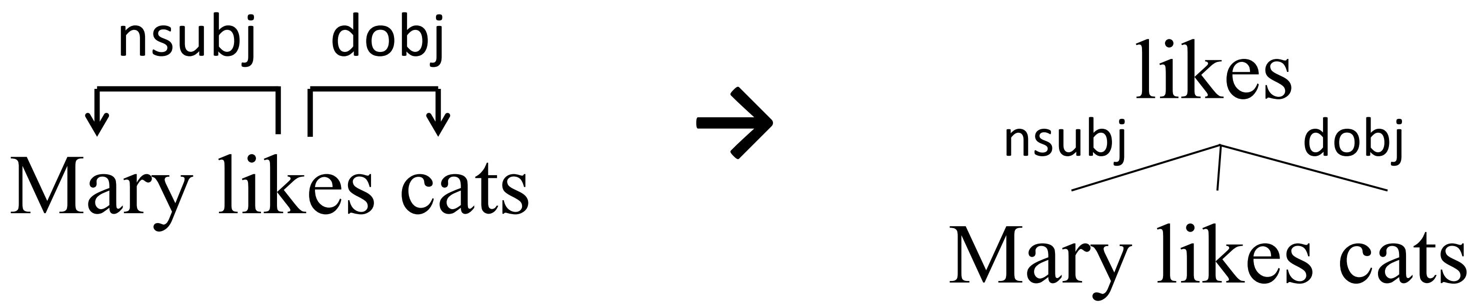
Introduction to NLP

256.

Dependency Parsing

Classic Techniques

- Dynamic programming
  - CKY – similar to lexicalized PCFG, cubic complexity (Eisner 96)

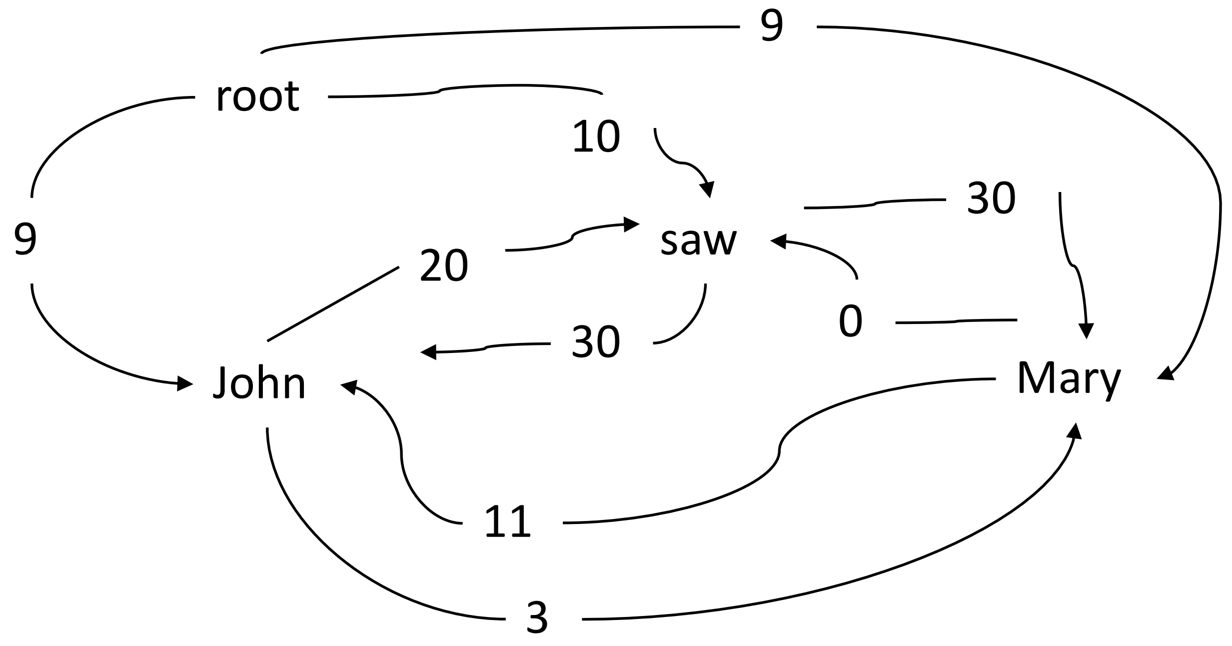


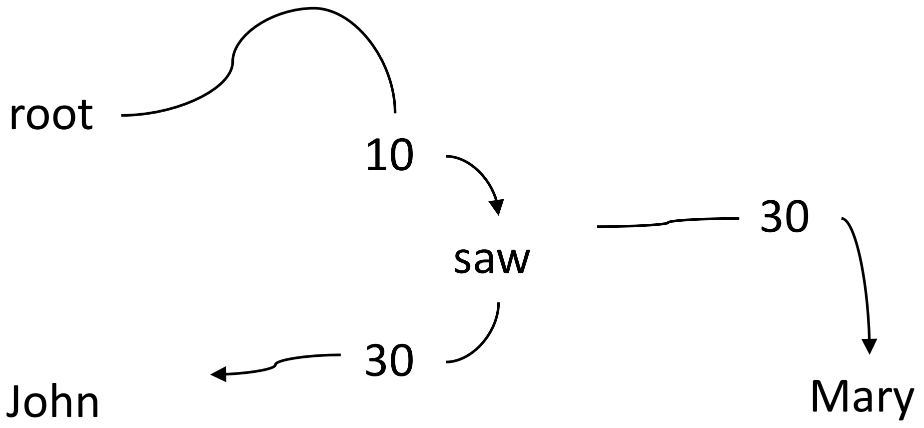
Graph-based Dependency Parsing

- McDonald et al. 2005
- Dependency parsing is equivalent to search for a maximum spanning tree (MST) in a directed graph.
- Efficient algorithm for finding MST for directed graphs
  - Chu and Liu (1965) and Edmonds (1967) give an.

MST Parser example

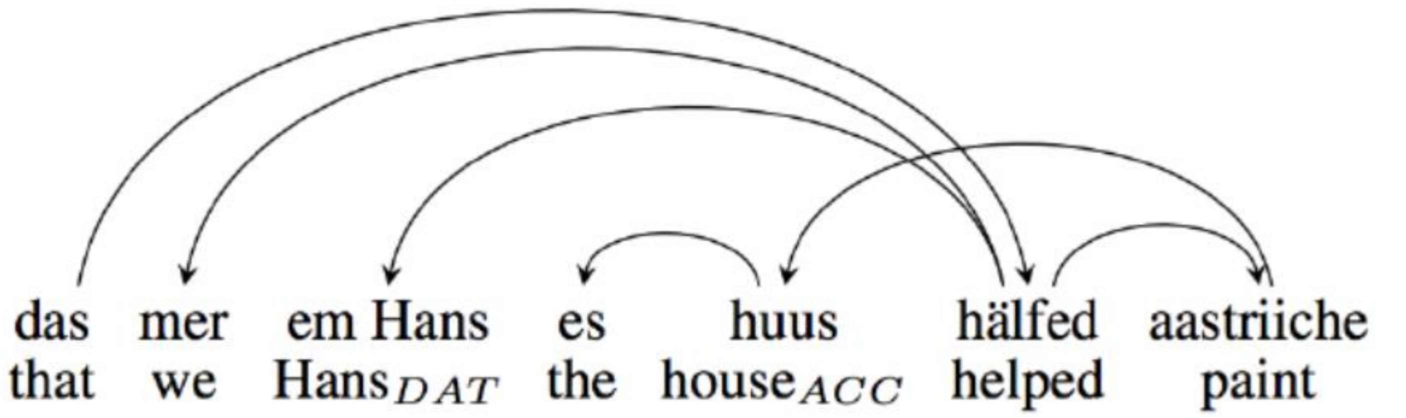
- Consider the sentence “John saw Mary”
- Recursively remove cycle
- The Chu-Liu-Edmonds algorithm gives the MST on the right hand side (right). This is in general a non-projective tree.





Notes

- Complexity
  - Interestingly, MST is $O(n^2)$, compared with $O(n^3)$ for Eisner, even though MST is non-projective.
- Example of a highly non-projective language
  - Swiss German



[example from Pitler 2013]

Introduction to NLP

257.

Transition-based Dependency Parsing

Transition-Based Parsing

- Similar to shift-reduce
- Produces a single (projective) tree
- Data structures
  - Stack of partially processed (unattached) words
  - Input buffer
  - Set of dependency arcs
    - Attach the word on the top of the stack to the word at the current position in the buffer (or in the other direction)

Transition-Based Parsing

- Initial configuration
  - Stack (including the root token w0)
  - Buffer (sentence)
  - Arcs (empty)
- Goal configuration
  - Stack (empty)
  - Buffer (empty)
  - Arcs (complete tree)

Example

- “Book me the morning flight”

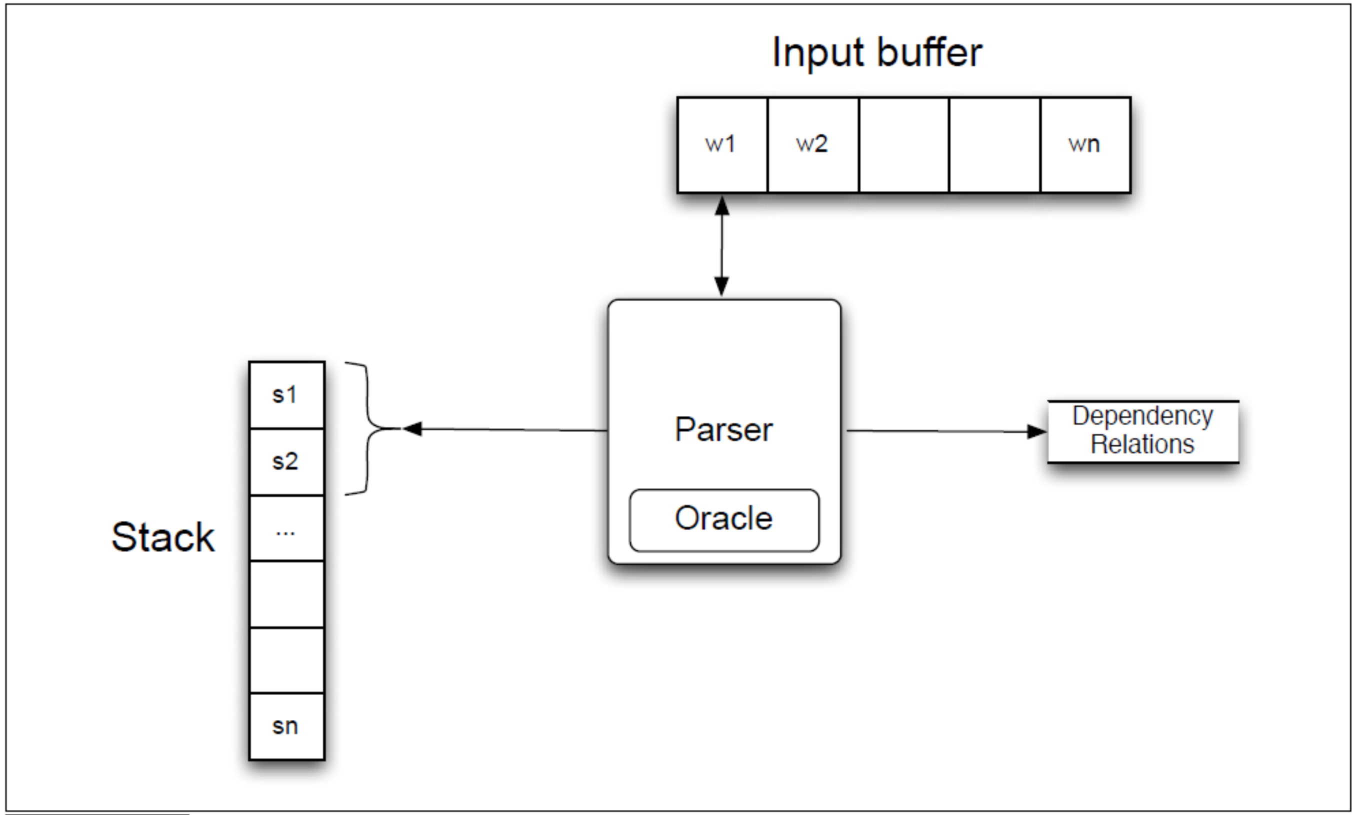


Figure 14.5 Basic transition-based parser. The parser examines the top two elements of the stack and selects an action based on consulting an oracle that examines the current configuration.

[Example from Jurafsky and Martin]

MaltParser (Nivre 2008)

- The reduce operations combine an element from the stack and one from the buffer
- Arc-standard parser
  - The actions are shift, left-arc, right-arc
- Arc-eager parser
  - The actions are shift, reduce, left-arc, right-arc

(Arc-Eager) MaltParser Actions

$$\text{Shift} \quad \frac{[\ldots]_S \quad [w_i, \ldots]_Q}{[\ldots, w_i]_S \quad [\ldots]_Q}$$

$$\text{Reduce} \quad \frac{[\ldots, w_i]_S \quad [\ldots]_Q \quad \exists w_k : w_k \rightarrow w_i}{[\ldots]_S \quad [\ldots]_Q}$$

$$\text{Left-Arc}_r \quad \frac{[\ldots, w_i]_S \quad [w_j, \ldots]_Q \quad \neg\exists w_k : w_k \rightarrow w_i}{[\ldots]_S \quad [w_j, \ldots]_Q \quad w_i \xleftarrow{r} w_j}$$

$$\text{Right-Arc}_r \quad \frac{[\ldots, w_i]_S \quad [w_j, \ldots]_Q \quad \neg\exists w_k : w_k \rightarrow w_j}{[\ldots, w_i, w_j]_S \quad [\ldots]_Q \quad w_i \xrightarrow{r} w_j}$$

[Example from Nivre and Kuebler]

Transition	Configuration		
	([ROOT],	[Economic, . . . , .],	$\emptyset$)
SH $\Rightarrow$	([ROOT, Economic],	[news, . . . , .],	$\emptyset$)
$\text{LA}_{\text{ATT}} \Rightarrow$	([ROOT],	[news, . . . , .],	$A_1 = \{(\text{news, ATT, Economic})\})$
SH $\Rightarrow$	([ROOT, news],	[had, . . . , .],	A_1)
$\text{LA}_{\text{SBJ}} \Rightarrow$	([ROOT],	[had, . . . , .],	$A_2 = A_1 \cup \{(\text{had, SBJ, news})\})$
$\text{RA}_{\text{PRED}} \Rightarrow$	([ROOT, had],	[little, . . . , .],	$A_3 = A_2! \cup \{(\text{ROOT, PRED, had})\})$
SH $\Rightarrow$	([ROOT, had, little],	[effect, . . . , .],	A_3)
$\text{LA}_{\text{ATT}} \Rightarrow$	([ROOT, had],	[effect, . . . , .],	$A_4 = A_3 \cup \{(\text{effect, ATT, little})\})$
$\text{RA}_{\text{OBJ}} \Rightarrow$	([ROOT, had, effect],	[on, . . . , .],	$A_5 = A_4 \cup \{(\text{had, OBJ, effect})\})$
$\text{RA}_{\text{ATT}} \Rightarrow$	([ROOT, . . . on],	[financial, markets, .],	$A_6 = A_5 \cup \{(\text{effect, ATT, on})\})$
SH $\Rightarrow$	([ROOT, . . . , financial],	[markets, .],	A_6)
$\text{LA}_{\text{ATT}} \Rightarrow$	([ROOT, . . . on],	[markets, .],	$A_7 = A_6 \cup \{(\text{markets, ATT, financial})\})$
$\text{RA}_{\text{PC}} \Rightarrow$	([ROOT, . . . , markets],	[.],	$A_8 = A_7 \cup \{(\text{on, PC, markets})\})$
RE $\Rightarrow$	([ROOT, . . . , on],	[.],	A_8)
RE $\Rightarrow$	([ROOT, . . . , effect],	[.],	A_8)
RE $\Rightarrow$	([ROOT, had],	[.],	A_8)
$\text{RA}_{\text{PU}} \Rightarrow$	([ROOT, . . . , .],	[ ],	$A_9 = A_8 \cup \{(\text{had, PU, .})\})$

Figure 3.7: Arc-eager transition sequence for the English sentence in figure 1.1 (LA_r = LEFT-ARC$_r$, RA_r = RIGHT-ARC$_r$, RE = REDUCE, SH = SHIFT).

[Example from Kuebler, McDonald, Nivre]

Example

- Example: "People want to be free"

	[ROOT]	[People, want, to, be, free]	Ø
Shift	[ROOT, People]	[want, to, be, free]	
LA_{nsubj}	[ROOT]	[want, to, be, free]	A_1 = {nsubj(want, people)}
RA_{root}	[ROOT, want]	[to, be, free]	$A_2 = A_1 \cup$ {root(ROOT, want)}

- Characteristics
  - Approximate the oracle with a classifier: $o(c) = \text{argmax}_t\ \mathbf{w}.\mathbf{f}(c,t)$
  - There is no search in the greedy version (although beam search also works)
  - The final list of arcs is returned as the dependency tree
  - Trained on a dependency treebank
- Very fast method

Feature Model

Table 3.1: Feature model for transition-based parsing.

f_i	Address	Attribute
1	STK[0]	FORM
2	BUF[0]	FORM
3	BUF[1]	FORM
4	LDEP(STK[0])	DEPREL
5	RDEP(STK[0])	DEPREL
6	LDEP(BUF[0])	DEPREL
7	RDEP(BUF[0])	DEPREL

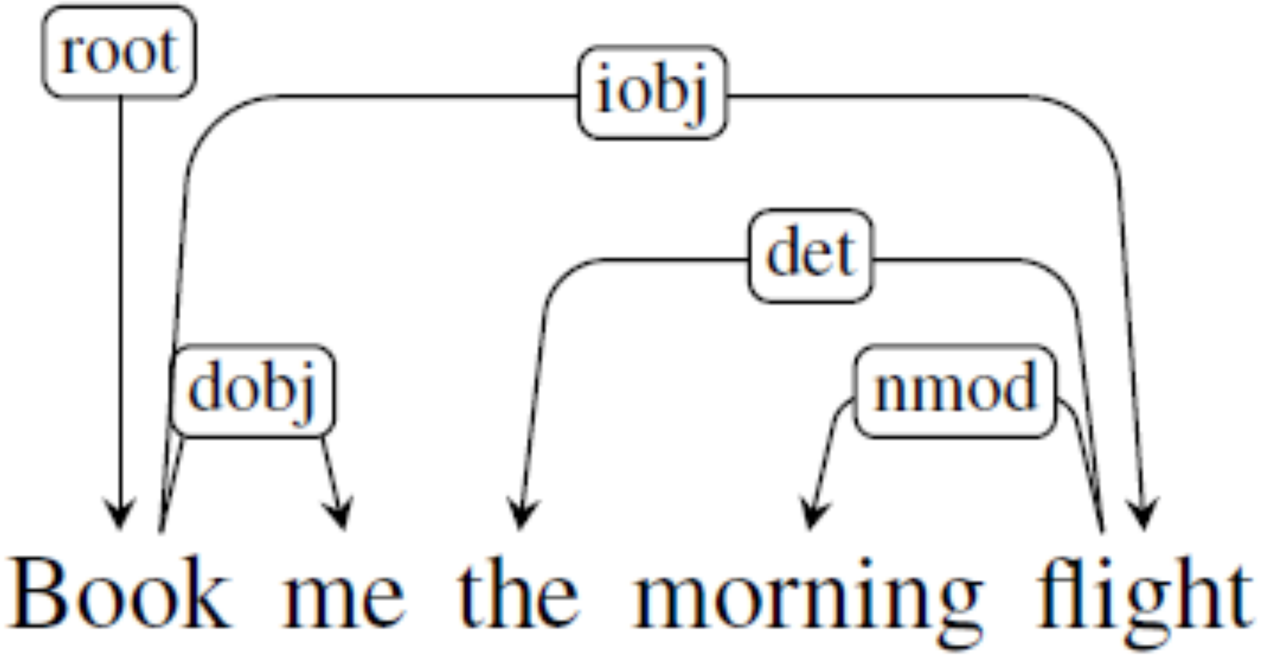
[Example from Kuebler, McDonald, Nivre]

Feature Vectors

$\mathbf{f}(c_0)$	=	(ROOT	Economic	news	NULL	NULL	NULL	NULL)
$\mathbf{f}(c_1)$	=	(Economic	news	had	NULL	NULL	NULL	NULL)
$\mathbf{f}(c_2)$	=	(ROOT	news	had	NULL	NULL	ATT	NULL)
$\mathbf{f}(c_3)$	=	(news	had	little	ATT	NULL	NULL	NULL)
$\mathbf{f}(c_4)$	=	(ROOT	had	little	NULL	NULL	SBJ	NULL)
$\mathbf{f}(c_5)$	=	(had	little	effect	SBJ	NULL	NULL	NULL)
$\mathbf{f}(c_6)$	=	(little	effect	on	NULL	NULL	NULL	NULL)
$\mathbf{f}(c_7)$	=	(had	effect	on	SBJ	NULL	ATT	NULL)
$\mathbf{f}(c_8)$	=	(effect	on	financial	ATT	NULL	NULL	NULL)
$\mathbf{f}(c_9)$	=	(on	financial	markets	NULL	NULL	NULL	NULL)
$\mathbf{f}(c_{10})$	=	(financial	markets	.	NULL	NULL	NULL	NULL)
$\mathbf{f}(c_{11})$	=	(on	markets	.	NULL	NULL	ATT	NULL)
$\mathbf{f}(c_{12})$	=	(effect	on	.	ATT	NULL	NULL	ATT)
$\mathbf{f}(c_{13})$	=	(had	effect	.	SBJ	NULL	ATT	ATT)
$\mathbf{f}(c_{14})$	=	(ROOT	had	.	NULL	NULL	SBJ	OBJ)
$\mathbf{f}(c_{15})$	=	(had	.	NULL	SBJ	OBJ	NULL	NULL)
$\mathbf{f}(c_{16})$	=	(ROOT	had	NULL	NULL	NULL	SBJ	PU)
$\mathbf{f}(c_{17})$	=	(NULL	ROOT	NULL	NULL	NULL	NULL	PRED)
$\mathbf{f}(c_{18})$	=	(ROOT	NULL	NULL	NULL	PRED	NULL	NULL)

Figure 3.5: Feature vectors for the configurations in figure 3.2.

[Example from Kuebler, McDonald, Nivre]


root
iobj
det
dobj
nmod
Book me the morning flight

Introduction to NLP

258.

Evaluation of Dependency Parsing

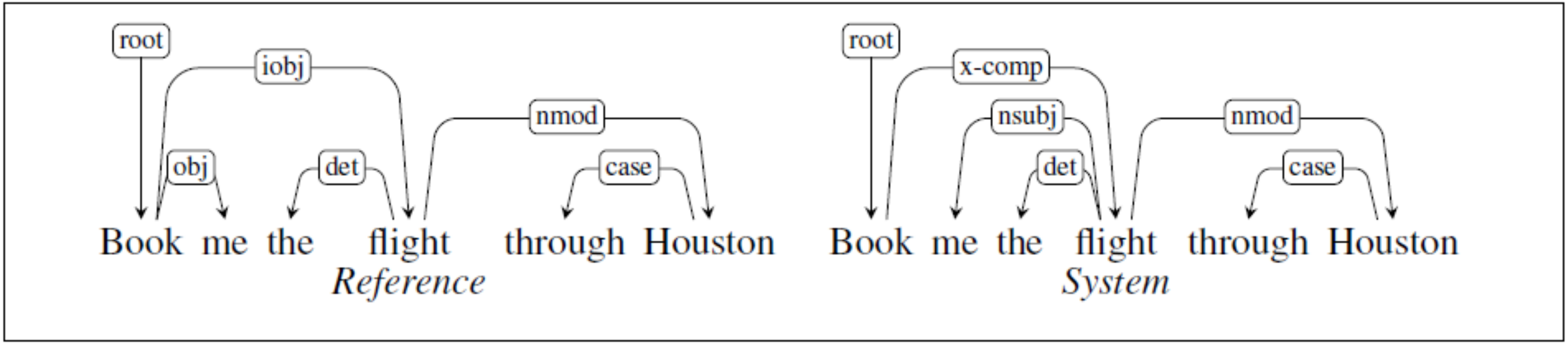


Figure 14.15 Reference and system parses for *Book me the flight through Houston*, resulting in an LAS of 3/6 and an UAS of 4/6.

Evaluation of Dependency Parsing

- Attachment Score (Buchholz & Marsi 2006)
  - # correct deps/# deps (attached to the right head)
  - Unlabeled dependency accuracy (UAS)
  - Labeled dependency accuracy (LAS)

1	Unionized	Unionized	VBN	VBN	-	2	NMOD	-	-
2	workers	workers	NNS	NNS	-	3	SBJ	-	-
3	are	are	VBP	VBP	-	0	ROOT	-	-
4	usually	usually	RB	RB	-	3	TMP	-	-
5	better	better	RBR	RBR	-	4	ADV	-	-
6	paid	paid	VBN	VBN	-	5	AMOD	-	-
7	than	than	IN	IN	-	5	AMOD	-	-
8	their	their	PRP$	PRP$	-	10	NMOD	-	-
9	non-union	non-union	JJ	JJ	-	10	NMOD	-	-
10	counterparts	counterparts	NNS	NNS	-	7	PMOD	-	-

External Links

- http://ilk.uvt.nl/conll/
  - CONLL-X Shared task
- http://ufal.mff.cuni.cz/pdt2.0/
  - Prague Dependency Treebank
- http://nextens.uvt.nl/depparse-wiki/SharedTaskWebsite
- http://nextens.uvt.nl/depparse-wiki/DataOverview
- http://maltparser.org/
  - Joakim Nivre's Maltparser
- http://www.cs.ualberta.ca/~lindek/minipar.htm
  - Dekang Lin's Minipar
- http://www.link.cs.cmu.edu/link/
  - Daniel Sleator and Davy Temperley's Link parser

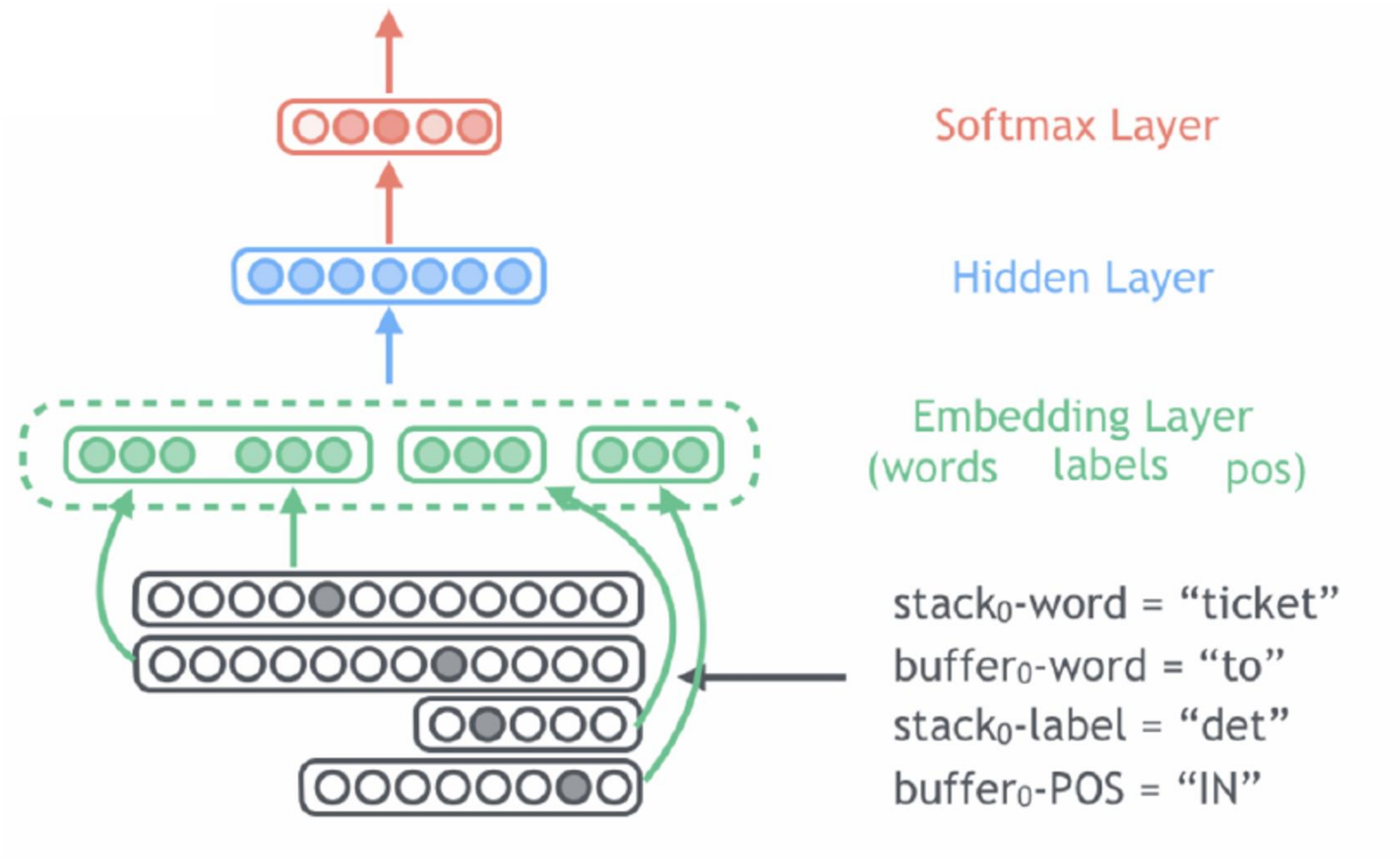
Notes

- The original versions of MSTParser and MaltParser from 2007 achieve about 81% accuracy
  - Highest in Japanese (91-92%)
  - Lowest in Arabic and Turkish (63-67%)
- Non-projective parsing is harder than projective parsing

Introduction to NLP

Neural Dependency Parsing

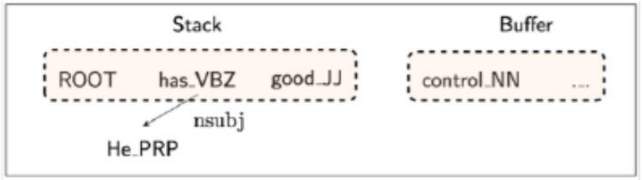
Neural dependency parsing



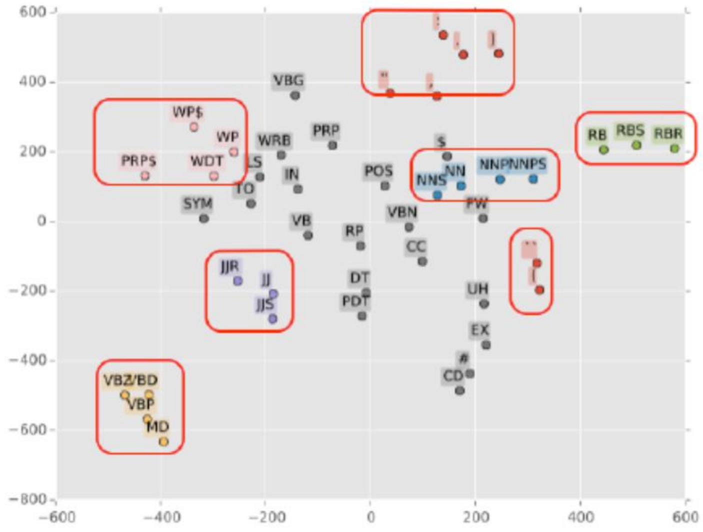
[Chen and Manning 2014]

Neural dependency parsing

- Used pre-trained word embeddings
- Part-of-speech tags and dependency labels are also represented as vectors
- No feature template any more!



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



- A simple feedforward NN: what is left is backpropagation!

(Chen and Manning, 2014): A Fast and Accurate Dependency Parser using Neural Networks

[Danqi Chen]