Statistical Parsing

A Context-Free Grammar

S	$\Rightarrow$	NP	VP
VP	$\Rightarrow$	Vi	
VP	$\Rightarrow$	Vt	NP
VP	$\Rightarrow$	VP	PP
NP	$\Rightarrow$	DT	NN
NP	$\Rightarrow$	NP	PP
PP	$\Rightarrow$	P	NP

Vi	$\Rightarrow$	sleeps
Vt	$\Rightarrow$	saw
NN	$\Rightarrow$	man
NN	$\Rightarrow$	dog
NN	$\Rightarrow$	telescope
DT	$\Rightarrow$	the
IN	$\Rightarrow$	with
IN	$\Rightarrow$	in

Ambiguity

- A sentence of reasonable length can easily have 10s, 100s, or 1000s of possible analyses, most of which are very implausible

- Examples of sources of ambiguity:

  – Part-of-Speech ambiguity

    NNS $\rightarrow$ walks

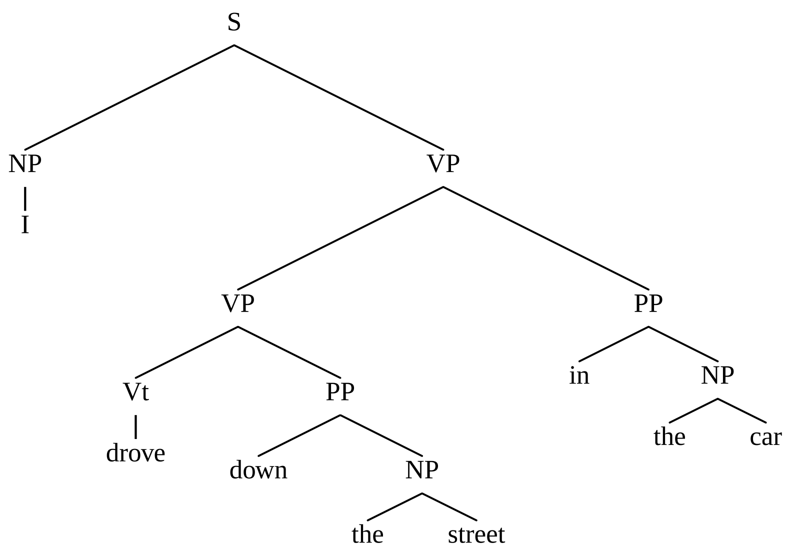
    Vi $\rightarrow$ walks

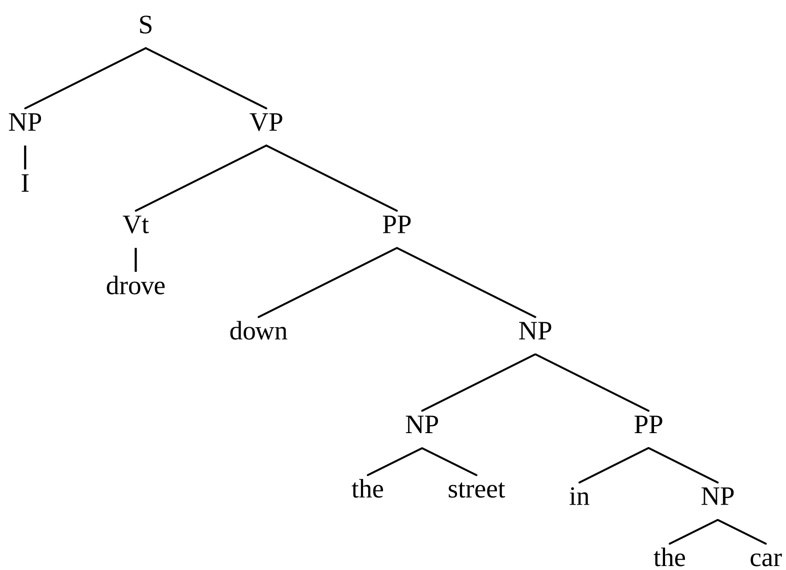
  – Prepositional phrase attachment

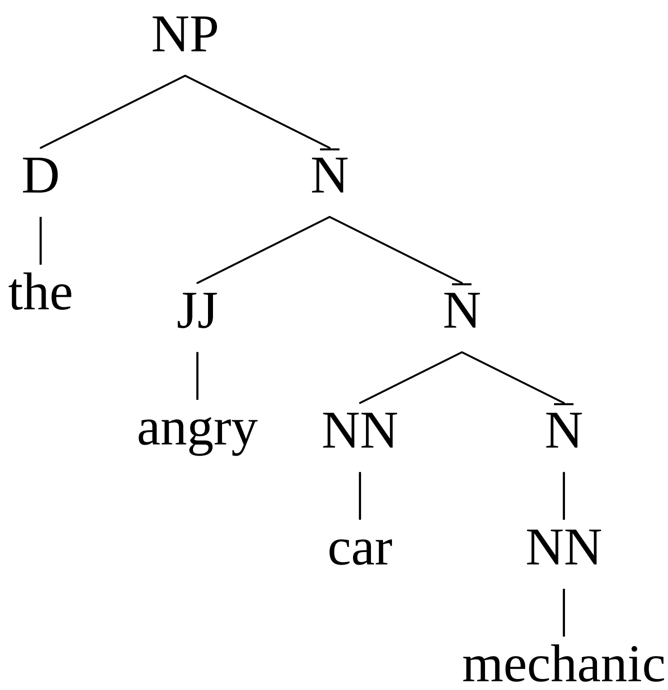
    I drove down the street in the car

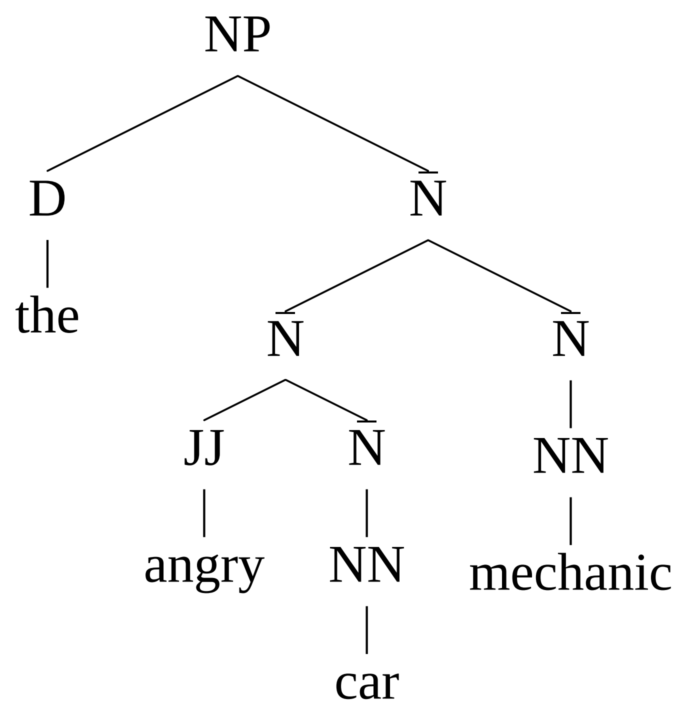
  – Pre-nominal modifiers

    the angry car mechanic


S
NP
VP
I
VP
PP
Vt
PP
in
NP
drove
down
NP
the
car
the
street


S
NP
VP
I
Vt
PP
drove
down
NP
NP
PP
the
street
in
NP
the
car


NP
D
N̄
the
JJ
N̄
angry
NN
N̄
car
NN
mechanic


NP
D
N̄
the
N̄
N̄
JJ
N̄
NN
angry
NN
mechanic
car

A program to promote safety in trucks and vans

There are at least 14 analyses for this noun phrase...

A Probabilistic Context-Free Grammar

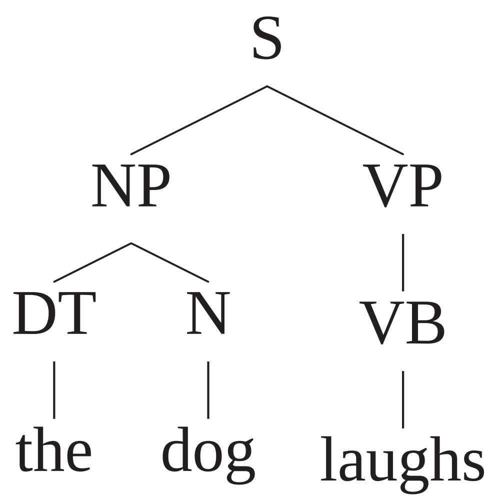
S	$\Rightarrow$	NP	VP	1.0
VP	$\Rightarrow$	Vi		0.4
VP	$\Rightarrow$	Vt	NP	0.4
VP	$\Rightarrow$	VP	PP	0.2
NP	$\Rightarrow$	DT	NN	0.3
NP	$\Rightarrow$	NP	PP	0.7
PP	$\Rightarrow$	P	NP	1.0

Vi	$\Rightarrow$	sleeps	1.0
Vt	$\Rightarrow$	saw	1.0
NN	$\Rightarrow$	man	0.7
NN	$\Rightarrow$	dog	0.2
NN	$\Rightarrow$	telescope	0.1
DT	$\Rightarrow$	the	1.0
IN	$\Rightarrow$	with	0.5
IN	$\Rightarrow$	in	0.5

- Probability of a tree with rules $\alpha_i \rightarrow \beta_i$ is $\prod_i P(\alpha_i \rightarrow \beta_i | \alpha_i)$

DERIVATION	RULES USED	PROBABILITY
S	S $\longrightarrow$ NP VP	1.0
NP VP	NP $\longrightarrow$ DT N	0.3
DT N VP	DT $\longrightarrow$ the	1.0
the N VP	N $\longrightarrow$ dog	0.1
the dog VP	VP $\longrightarrow$ VB	0.4
the dog VB	VB $\longrightarrow$ laughs	0.5
the dog laughs		

PROBABILITY = $1.0 \times 0.3 \times 1.0 \times 0.1 \times 0.4 \times 0.5$



Properties of PCFGs

- Say we have a sentence S, set of derivations for that sentence is $\mathcal{T}(S)$. Then a PCFG assigns a probability to each member of $\mathcal{T}(S)$. i.e., *we now have a ranking in order of probability*.

- Given a PCFG and a sentence S, we can find

$$\arg\max_{T \in \mathcal{T}(S)} P(T, S)$$

using dynamic programming (e.g., a variant of the CKY algorithm)

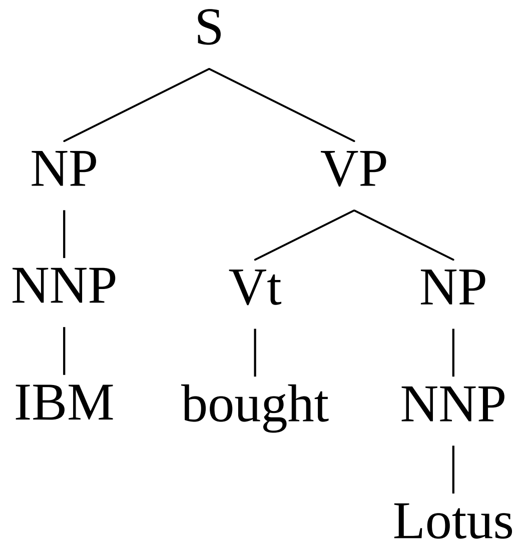
Overview

- Weaknesses of PCFGs
- Heads in context-free rules
- Dependency representations of parse trees
- Two models making use of dependencies

Weaknesses of PCFGs

- Lack of sensitivity to lexical information

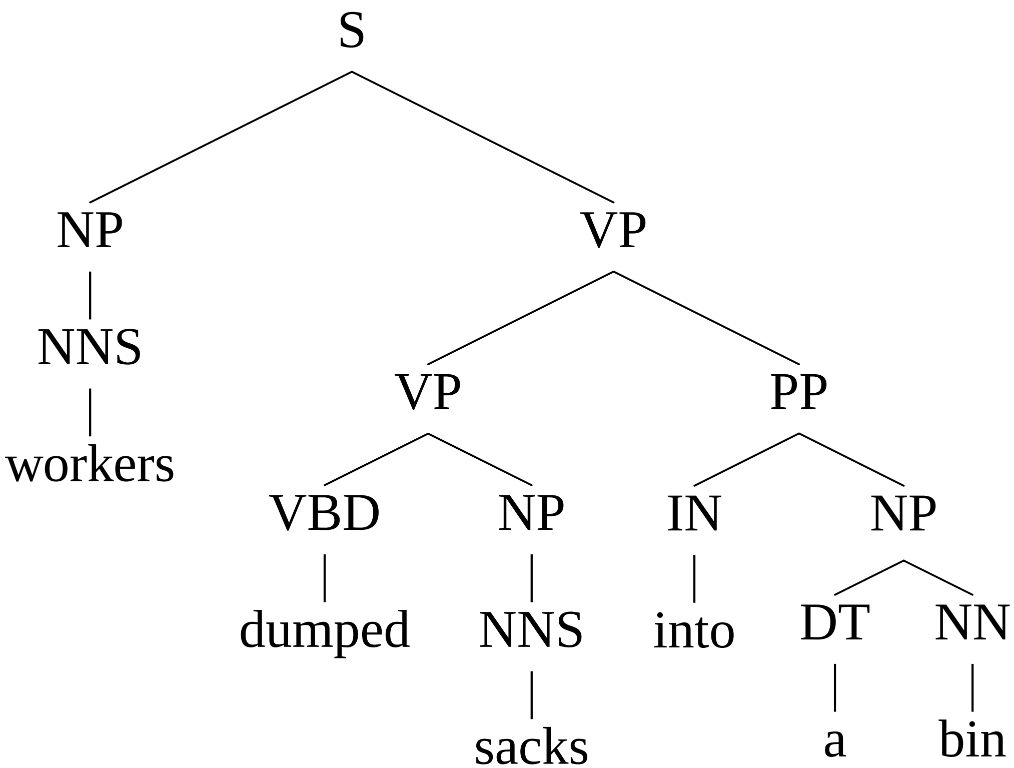
- Lack of sensitivity to structural frequencies



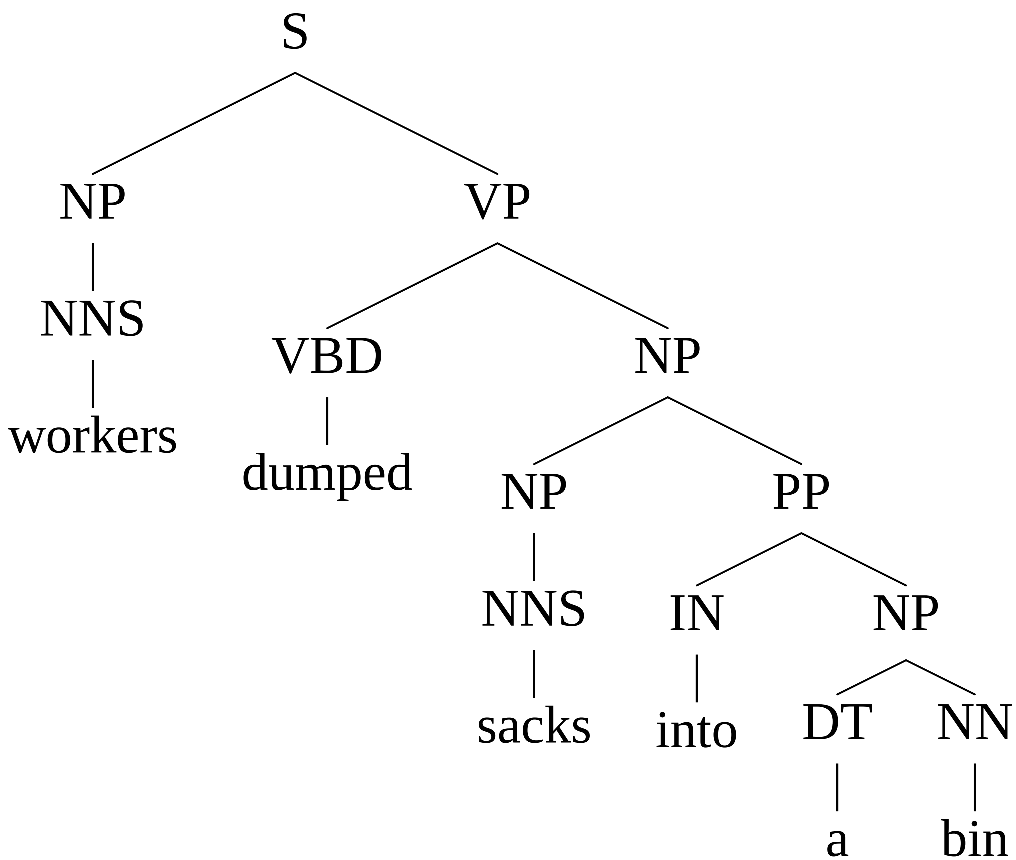
$$
\begin{aligned}
\text{PROB} = \quad & P(\text{S} \rightarrow \text{NP VP} \mid \text{S}) & \times P(\text{NNP} \rightarrow IBM \mid \text{NNP}) \\
& \times P(\text{VP} \rightarrow \text{V NP} \mid \text{VP}) & \times P(\text{Vt} \rightarrow bought \mid \text{Vt}) \\
& \times P(\text{NP} \rightarrow \text{NNP} \mid \text{NP}) & \times P(\text{NNP} \rightarrow Lotus \mid \text{NNP}) \\
& \times P(\text{NP} \rightarrow \text{NNP} \mid \text{NP}) &
\end{aligned}
$$

A Case of PP Attachment Ambiguity

(a)



(b)


S
NP
VP
NNS
workers
VBD
dumped
NP
NP
PP
NNS
sacks
IN
into
NP
DT
a
NN
bin

(a)

Rules
S $\rightarrow$ NP VP
NP $\rightarrow$ NNS
VP $\rightarrow$ VP PP
VP $\rightarrow$ VBD NP
NP $\rightarrow$ NNS
PP $\rightarrow$ IN NP
NP $\rightarrow$ DT NN
NNS $\rightarrow$ workers
VBD $\rightarrow$ dumped
NNS $\rightarrow$ sacks
IN $\rightarrow$ into
DT $\rightarrow$ a
NN $\rightarrow$ bin

(b)

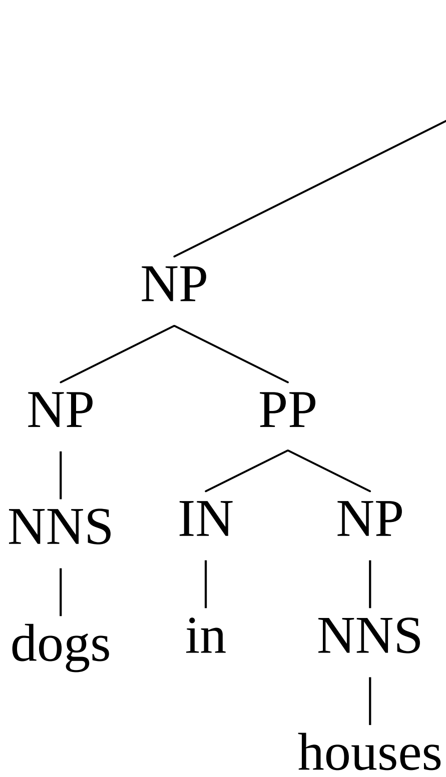
Rules
S $\rightarrow$ NP VP
NP $\rightarrow$ NNS
NP $\rightarrow$ NP PP
VP $\rightarrow$ VBD NP
NP $\rightarrow$ NNS
PP $\rightarrow$ IN NP
NP $\rightarrow$ DT NN
NNS $\rightarrow$ workers
VBD $\rightarrow$ dumped
NNS $\rightarrow$ sacks
IN $\rightarrow$ into
DT $\rightarrow$ a
NN $\rightarrow$ bin

If $P(\text{NP} \rightarrow \text{NP PP} \mid \text{NP}) > P(\text{VP} \rightarrow \text{VP PP} \mid \text{VP})$ then (b) is more probable, else (a) is more probable.

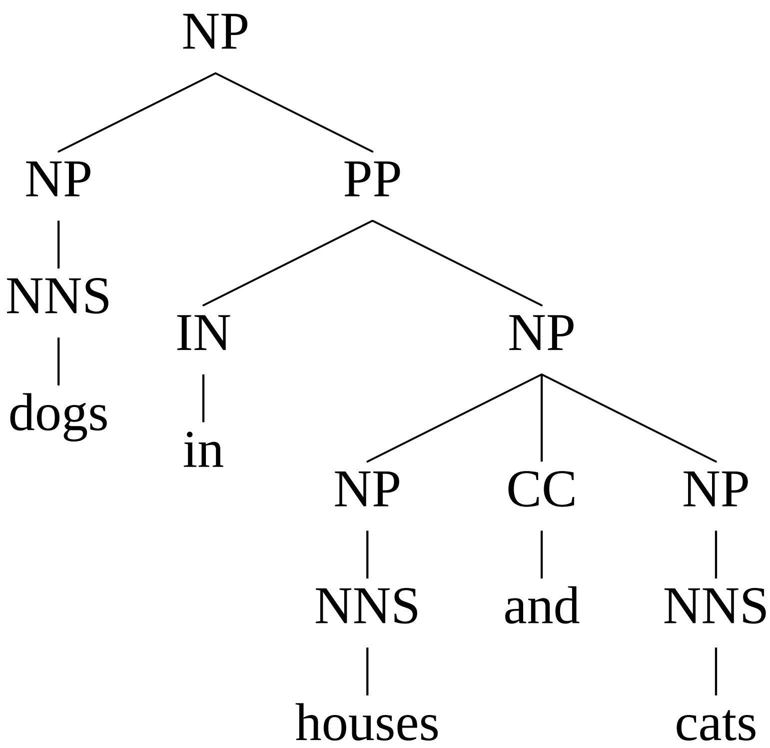
Attachment decision is completely independent of the words

A Case of Coordination Ambiguity

(a)



(b)



(a)

Rules
NP $\longrightarrow$ NP CC NP
NP $\longrightarrow$ NP PP
NP $\longrightarrow$ NNS
PP $\longrightarrow$ IN NP
NP $\longrightarrow$ NNS
NP $\longrightarrow$ NNS
NNS $\longrightarrow$ dogs
IN $\longrightarrow$ in
NNS $\longrightarrow$ houses
CC $\longrightarrow$ and
NNS $\longrightarrow$ cats

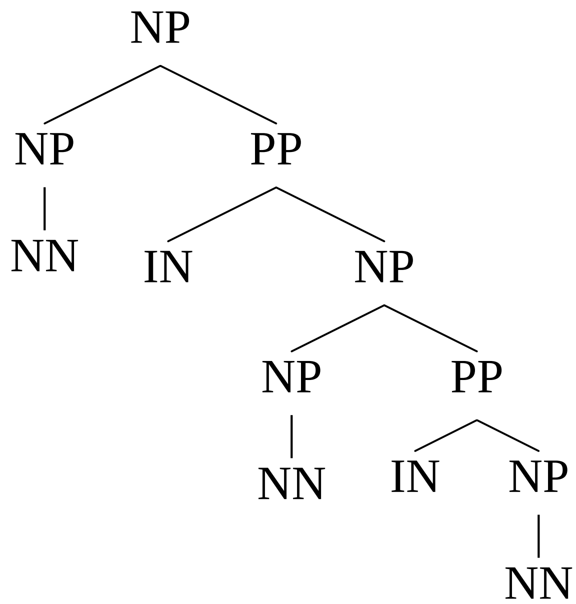
(b)

Rules
NP $\longrightarrow$ NP CC NP
NP $\longrightarrow$ NP PP
NP $\longrightarrow$ NNS
PP $\longrightarrow$ IN NP
NP $\longrightarrow$ NNS
NP $\longrightarrow$ NNS
NNS $\longrightarrow$ dogs
IN $\longrightarrow$ in
NNS $\longrightarrow$ houses
CC $\longrightarrow$ and
NNS $\longrightarrow$ cats

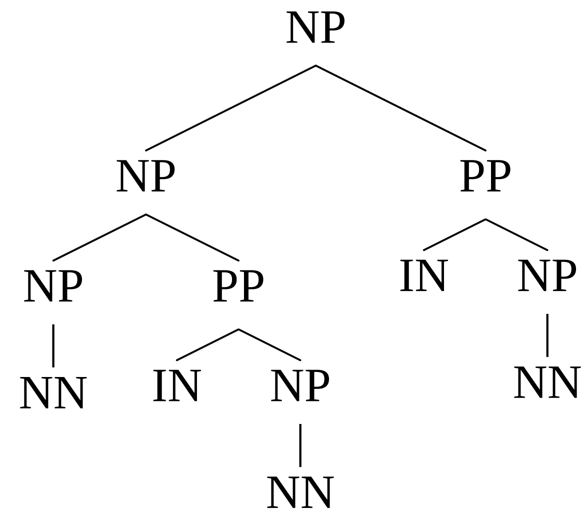
Here the two parses have identical rules, and therefore have identical probability under any assignment of PCFG rule probabilities

Structural Preferences: Close Attachment

(a)



(b)



- Example: president of a company in Africa
- Both parses have the same rules, therefore receive same probability under a PCFG
- "Close attachment" (structure (a)) is twice as likely in Wall Street Journal text.

<u>Structural Preferences: Close Attachment</u>

John was believed to have been shot by Bill

Here the low attachment analysis (Bill does the *shooting*) contains same rules as the high attachment analysis (Bill does the *believing*), so the two analyses receive same probability.

Heads in Context-Free Rules

Add annotations specifying the "head" of each rule:

S	$\Rightarrow$	NP	VP
VP	$\Rightarrow$	Vi	
VP	$\Rightarrow$	Vt	NP
VP	$\Rightarrow$	VP	PP
NP	$\Rightarrow$	DT	NN
NP	$\Rightarrow$	NP	PP
PP	$\Rightarrow$	IN	NP

Vi	$\Rightarrow$	sleeps
Vt	$\Rightarrow$	saw
NN	$\Rightarrow$	man
NN	$\Rightarrow$	woman
NN	$\Rightarrow$	telescope
DT	$\Rightarrow$	the
IN	$\Rightarrow$	with
IN	$\Rightarrow$	in

Note: S=sentence, VP=verb phrase, NP=noun phrase, PP=prepositional phrase, DT=determiner, Vi=intransitive verb, Vt=transitive verb, NN=noun, IN=preposition

Rules which Recover Heads: An Example of rules for NPs

If the rule contains NN, NNS, or NNP:
Choose the rightmost NN, NNS, or NNP

Else If the rule contains an NP: Choose the leftmost NP

Else If the rule contains a JJ: Choose the rightmost JJ

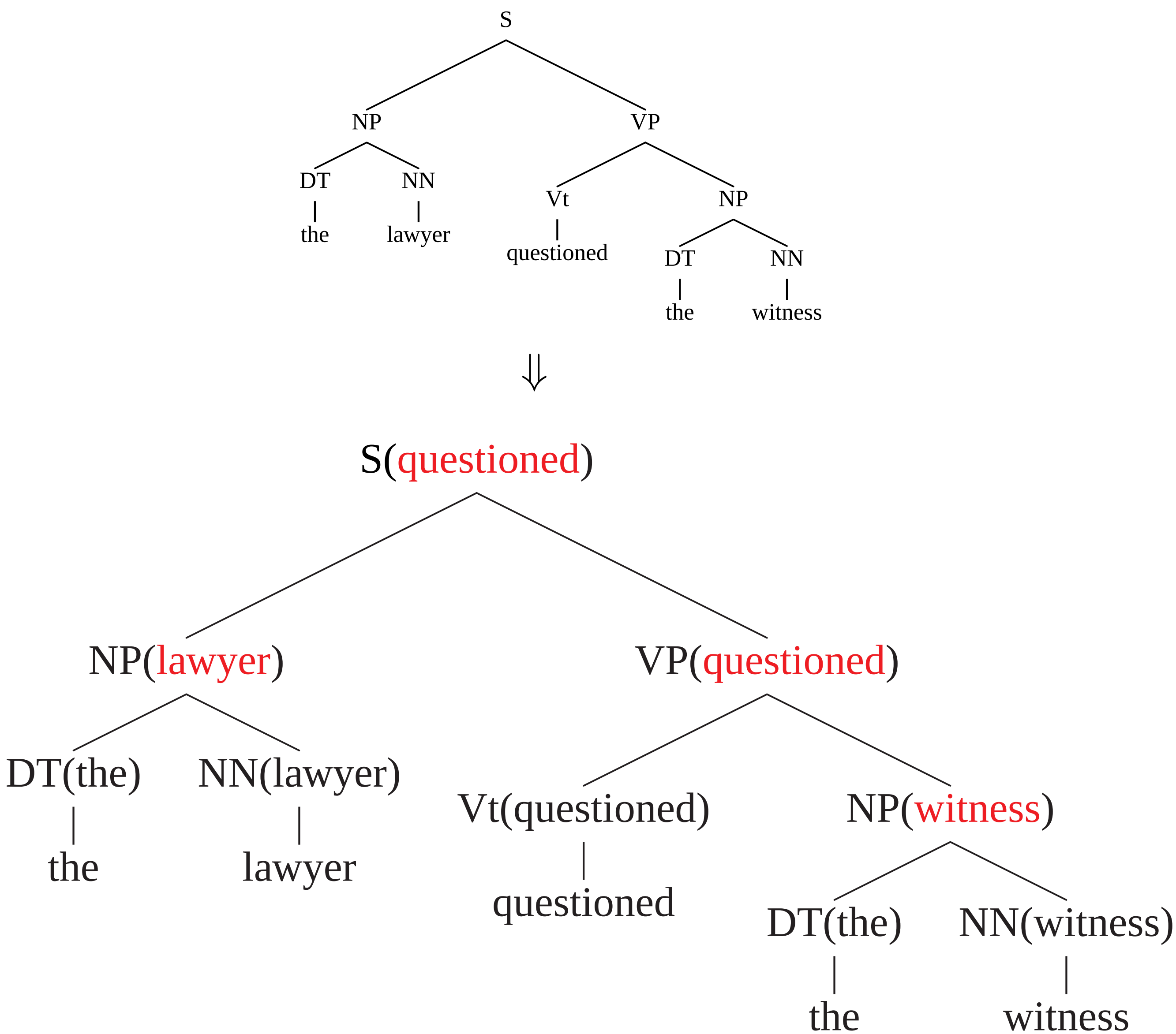
Else If the rule contains a CD: Choose the rightmost CD

Else Choose the rightmost child

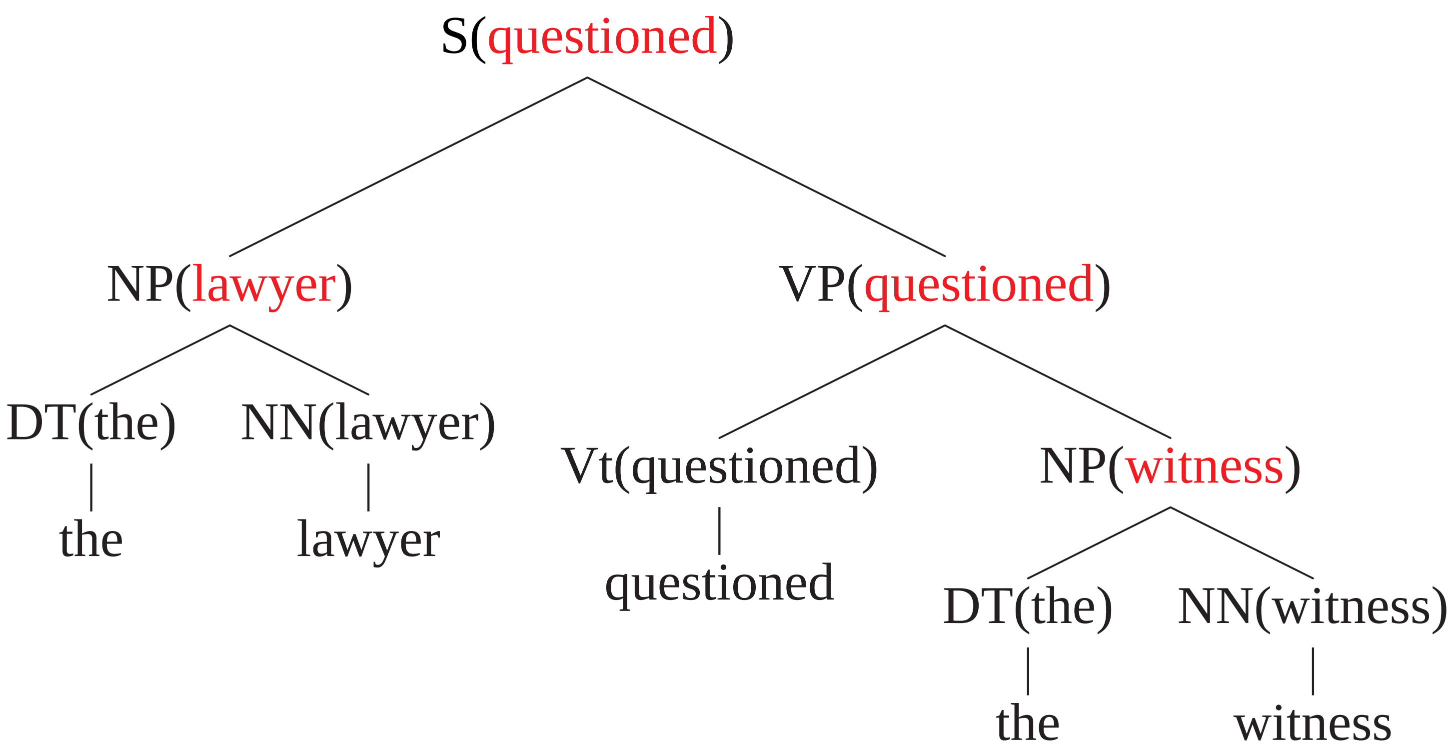
e.g.,

NP	$\Rightarrow$	DT	NNP	NN
NP	$\Rightarrow$	DT	NN	NNP
NP	$\Rightarrow$	NP	PP	
NP	$\Rightarrow$	DT	JJ	
NP	$\Rightarrow$	DT		

Adding Headwords to Trees



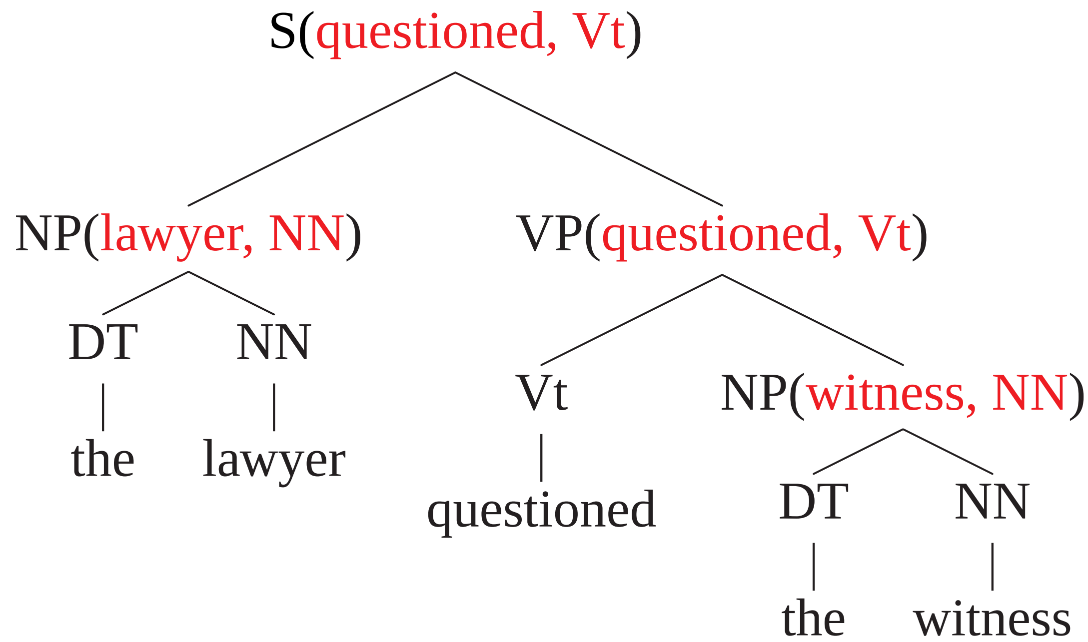
Adding Headwords to Trees



- A constituent receives its headword from its **head child**.

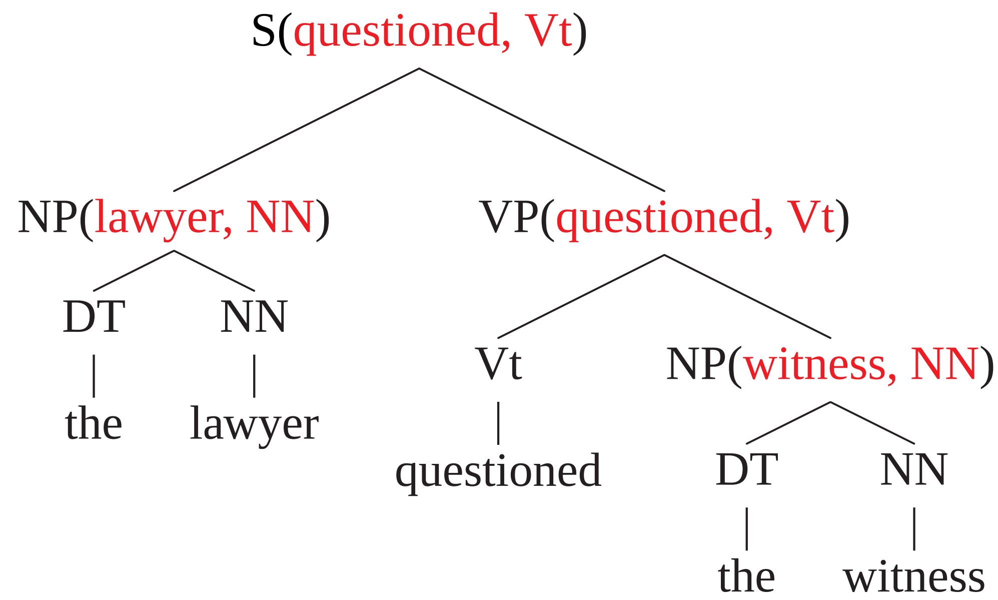
S	$\Rightarrow$	NP	VP		(S receives headword from VP)
VP	$\Rightarrow$	Vt	NP		(VP receives headword from Vt)
NP	$\Rightarrow$	DT		NN	(NP receives headword from NN)

Adding Headtags to Trees



- Also propagate **part-of-speech tags** up the trees
  (We'll see soon why this is useful!)

Lexicalized PCFGs



- In PCFGs we had rules such as `S -> NP VP`, with probabilities such as

$$P(\texttt{NP VP}|\texttt{S})$$

- In lexicalized PCFGs we have rules such as

  `S(questioned,Vt) -> NP(lawyer,NN) VP(questioned,Vt)`

  with probabilities such as

  $P(\texttt{NP(lawyer,NN) VP(questioned,Vt)}|\texttt{S(questioned,Vt)})$

A Model from Charniak (1997)

S(questioned,Vt)

$\Downarrow$ $\quad P(\text{NP}(\_\_,\text{NN})\ \text{VP} \mid \text{S(questioned,Vt)})$

S(questioned,Vt)

NP(__,NN) VP(questioned,Vt)

$\Downarrow$ $\quad P(\text{lawyer} \mid \text{S,VP,NP,NN, questioned,Vt})$

S(questioned,Vt)

NP(lawyer,NN) VP(questioned,Vt)

Smoothed Estimation

$$P(\text{NP(\_\_,NN) VP} \mid \text{S(questioned,Vt)}) =$$

$$\lambda_1 \times \frac{Count(\text{S(questioned,Vt)} \rightarrow \text{NP(\_\_,NN) VP})}{Count(\text{S(questioned,Vt)})}$$

$$+\lambda_2 \times \frac{Count(\text{S(\_\_,Vt)} \rightarrow \text{NP(\_\_,NN) VP})}{Count(\text{S(\_\_,Vt)})}$$

- Where $0 \leq \lambda_1, \lambda_2 \leq 1$, and $\lambda_1 + \lambda_2 = 1$

Smoothed Estimation

$$P(\text{lawyer} \mid \text{S,VP,NP,NN,questioned,Vt}) =$$

$$\lambda_1 \times \frac{Count(\text{lawyer} \mid \text{S,VP,NP,NN,questioned,Vt})}{Count(\text{S,VP,NP,NN,questioned,Vt})}$$

$$+\lambda_2 \times \frac{Count(\text{lawyer} \mid \text{S,VP,NP,NN,Vt})}{Count(\text{S,VP,NP,NN,Vt})}$$

$$+\lambda_3 \times \frac{Count(\text{lawyer} \mid \text{NN})}{Count(\text{NN})}$$

- Where $0 \leq \lambda_1, \lambda_2, \lambda_3 \leq 1$, and $\lambda_1 + \lambda_2 + \lambda_3 = 1$

$$P(\text{NP(lawyer,NN) VP} \mid \text{S(questioned,Vt)}) =$$

$$\Big(\lambda_1 \times \frac{Count(\text{S(questioned,Vt)}\rightarrow\text{NP(\_\_,NN) VP})}{Count(\text{S(questioned,Vt)})}$$

$$+\lambda_2 \times \frac{Count(\text{S(\_\_,Vt)}\rightarrow\text{NP(\_\_,NN) VP})}{Count(\text{S(\_\_,Vt)})}\ \Big)$$

$$\times\ \Big(\ \lambda_1 \times \frac{Count(\text{lawyer} \mid \text{S,VP,NP,NN,questioned,Vt})}{Count(\text{S,VP,NP,NN,questioned,Vt})}$$

$$+\lambda_2 \times \frac{Count(\text{lawyer} \mid \text{S,VP,NP,NN,Vt})}{Count(\text{S,VP,NP,NN,Vt})}$$

$$+\lambda_3 \times \frac{Count(\text{lawyer} \mid \text{NN})}{Count(\text{NN})}\ \Big)$$

Motivation for Breaking Down Rules

- First step of decomposition of (Charniak 1997):

S(questioned,Vt)

$$\Downarrow \qquad P(\text{NP(\_\_,NN) VP} \mid \text{S(questioned,Vt)})$$

S(questioned,Vt)

NP(__,NN) VP(questioned,Vt)

- Relies on counts of entire rules
- These counts are *sparse*:
  - 40,000 sentences from Penn treebank have 12,409 rules.
  - 15% of all test data sentences contain a rule never seen in training

Motivation for Breaking Down Rules

Rule Count	No. of Rules by Type	Percentage by Type	No. of Rules by token	Percentage by token
1	6765	54.52	6765	0.72
2	1688	13.60	3376	0.36
3	695	5.60	2085	0.22
4	457	3.68	1828	0.19
5	329	2.65	1645	0.18
6 ... 10	835	6.73	6430	0.68
11 ... 20	496	4.00	7219	0.77
21 ... 50	501	4.04	15931	1.70
51 ... 100	204	1.64	14507	1.54
$>$ 100	439	3.54	879596	93.64

Statistics for rules taken from sections 2-21 of the treebank
(Table taken from my PhD thesis).

Modeling Rule Productions as Markov Processes

- Step 1: generate category of head child

S(told,V[6])

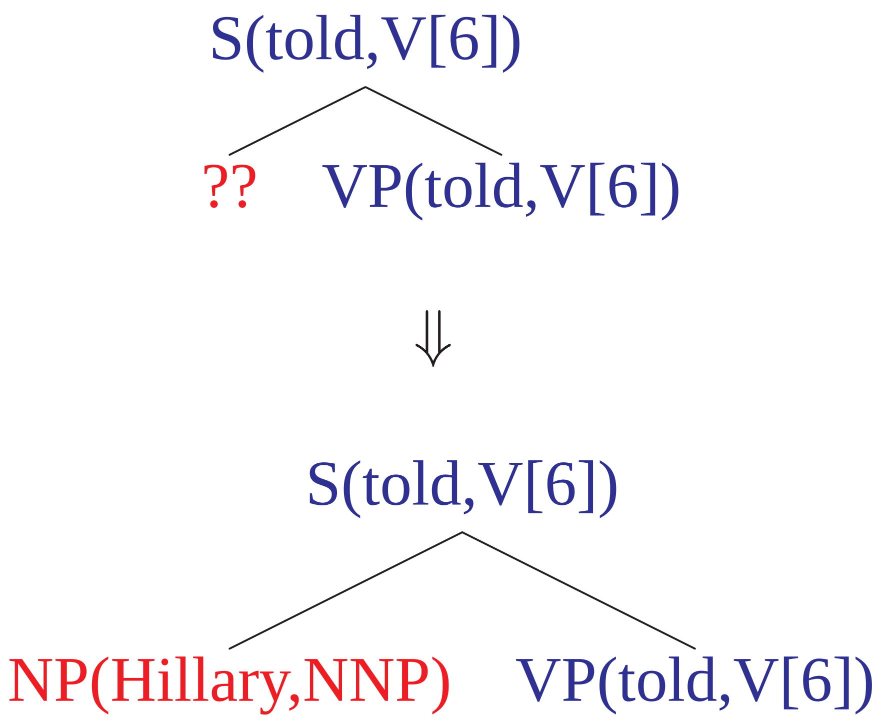
$\Downarrow$

S(told,V[6])
|
VP(told,V[6])

$P_h(\text{VP} \mid \text{S, told, V[6]})$

Modeling Rule Productions as Markov Processes

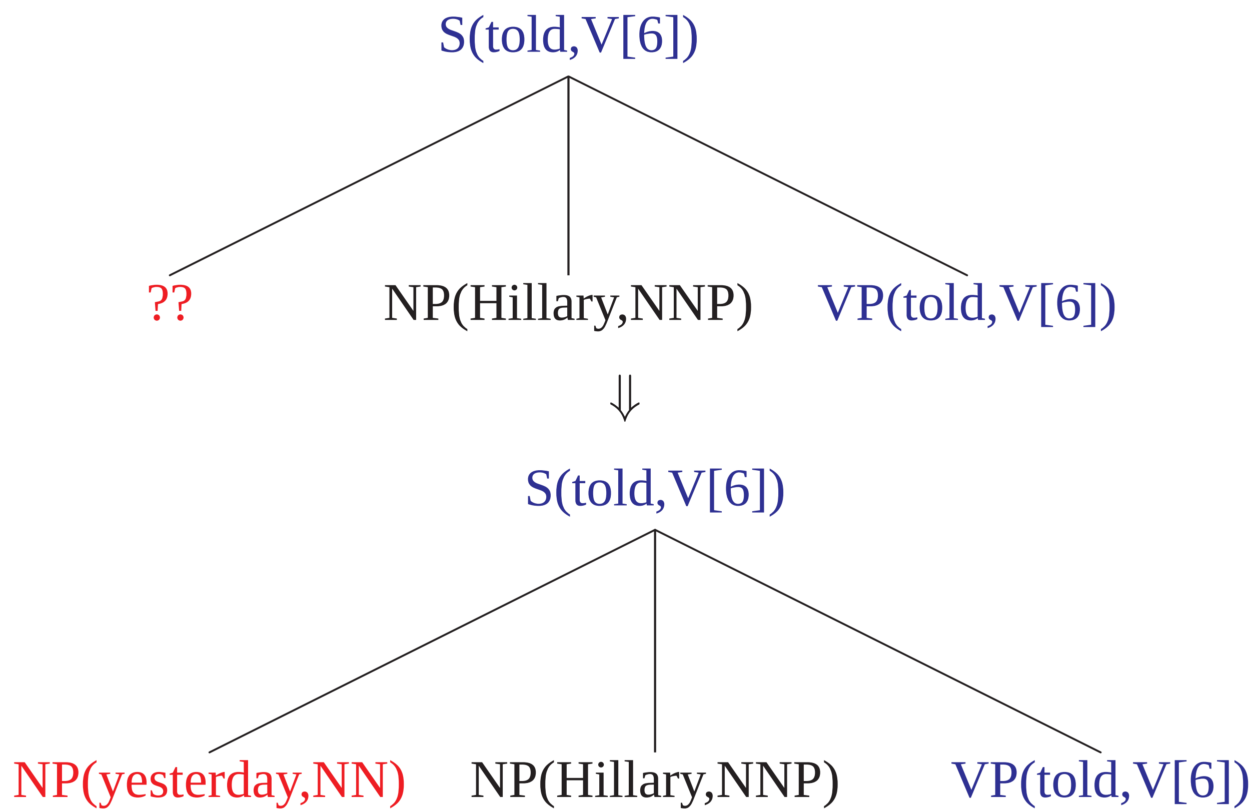
- Step 2: generate left modifiers in a Markov chain



$$P_h(\text{VP} \mid \text{S, told, V[6]}) \times P_d(\text{NP(Hillary,NNP)} \mid \text{S,VP,told,V[6],LEFT})$$

Modeling Rule Productions as Markov Processes

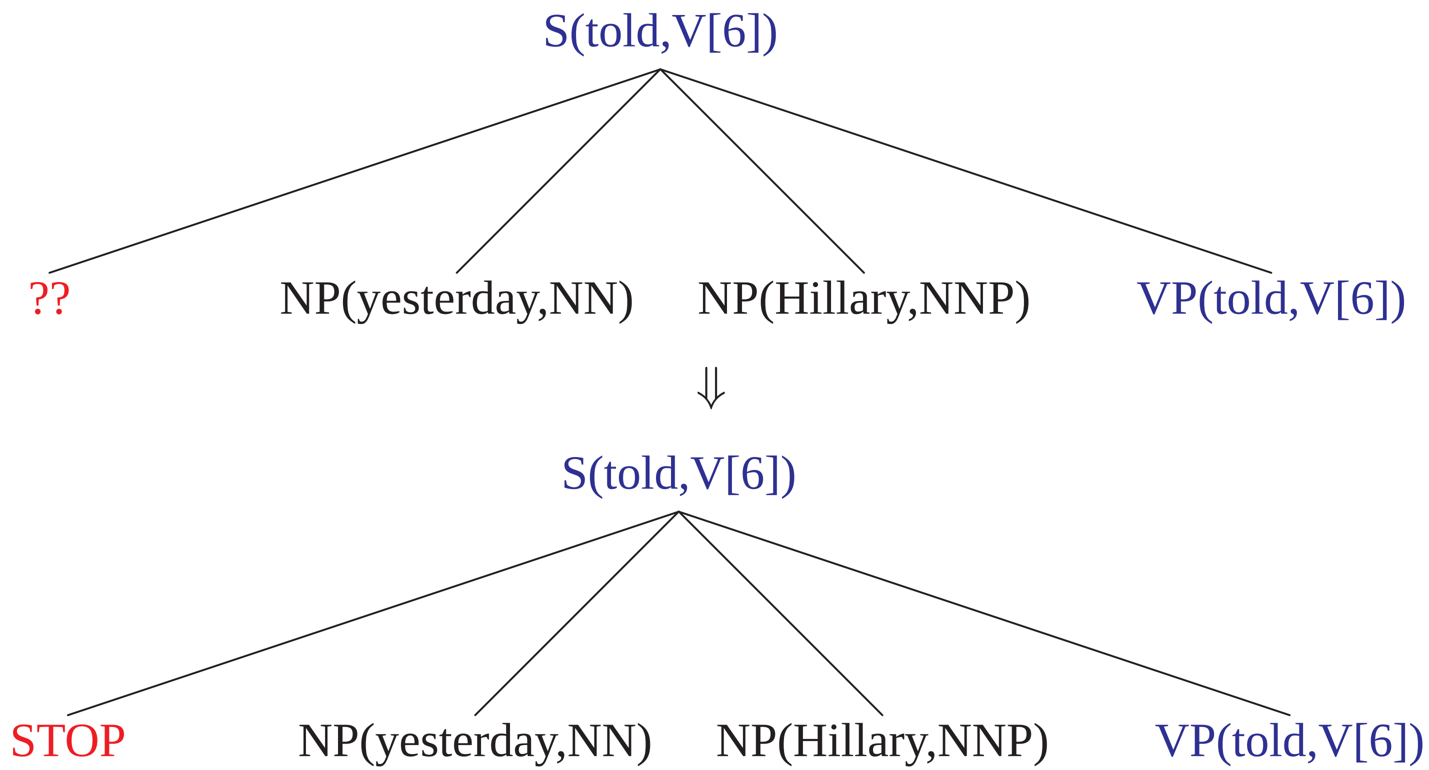
- Step 2: generate left modifiers in a Markov chain



$P_h(\text{VP} \mid \text{S, told, V[6]}) \times P_d(\text{NP(Hillary,NNP)} \mid \text{S,VP,told,V[6],LEFT}) \times$
$P_d(\text{NP(yesterday,NN)} \mid \text{S,VP,told,V[6],LEFT})$

Modeling Rule Productions as Markov Processes

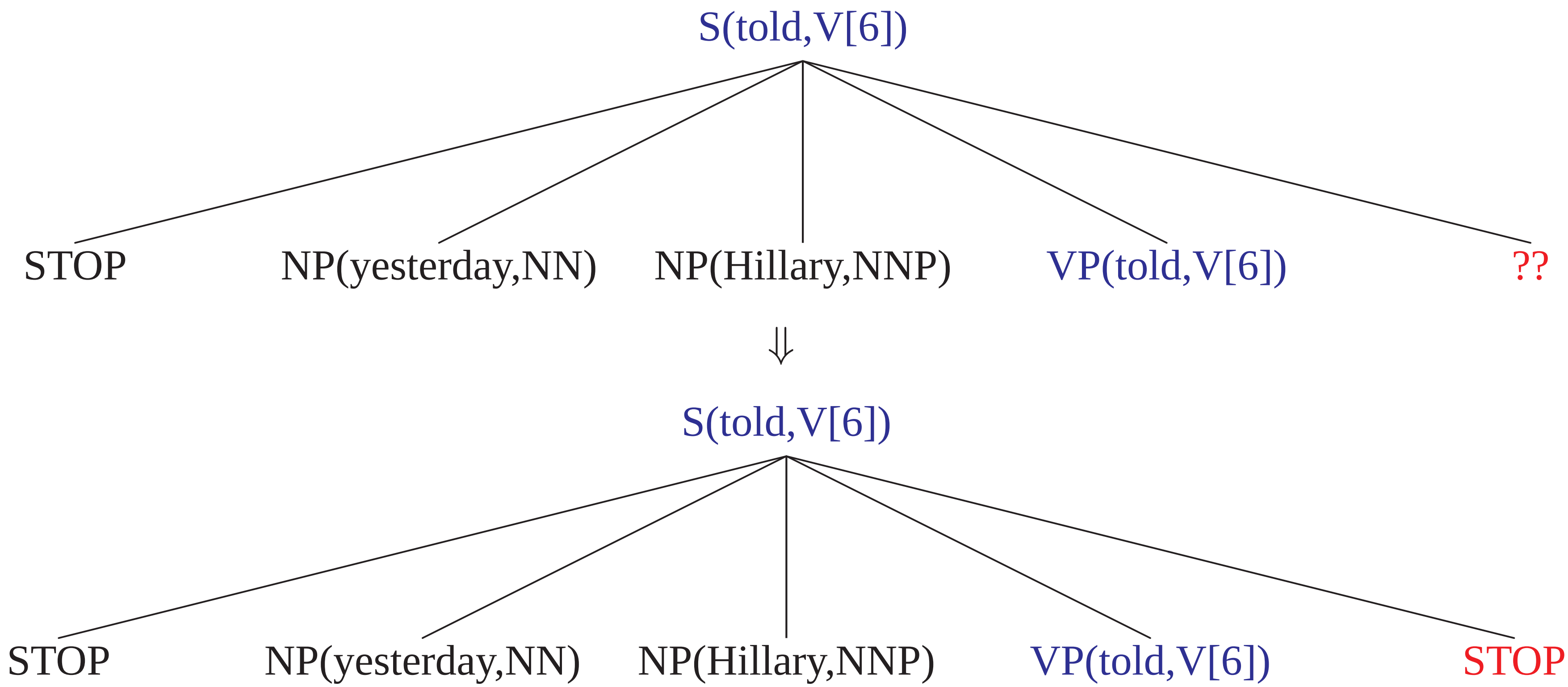
- Step 2: generate left modifiers in a Markov chain



$$P_h(\text{VP} \mid \text{S, told, V[6]}) \times P_d(\text{NP(Hillary,NNP)} \mid \text{S,VP,told,V[6],LEFT}) \times$$
$$P_d(\text{NP(yesterday,NN)} \mid \text{S,VP,told,V[6],LEFT}) \times P_d(\text{STOP} \mid \text{S,VP,told,V[6],LEFT})$$

Modeling Rule Productions as Markov Processes

- Step 3: generate right modifiers in a Markov chain



$P_h(\text{VP} \mid \text{S, told, V[6]}) \times P_d(\text{NP(Hillary,NNP)} \mid \text{S,VP,told,V[6],LEFT}) \times$
$P_d(\text{NP(yesterday,NN)} \mid \text{S,VP,told,V[6],LEFT}) \times P_d(\text{STOP} \mid \text{S,VP,told,V[6],LEFT}) \times$
$P_d(\text{STOP} \mid \text{S,VP,told,V[6],RIGHT})$

A Refinement: Adding a Distance Variable

- $\Delta = 1$ if position is adjacent to the head.

S(told,V[6])

?? VP(told,V[6])

$\Downarrow$

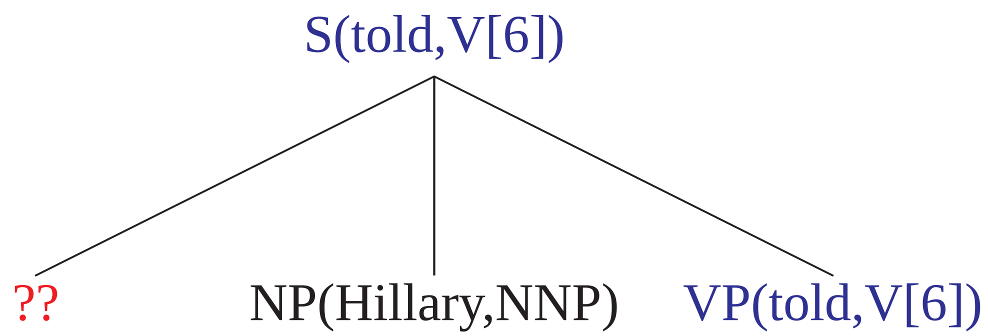
S(told,V[6])

NP(Hillary,NNP) VP(told,V[6])

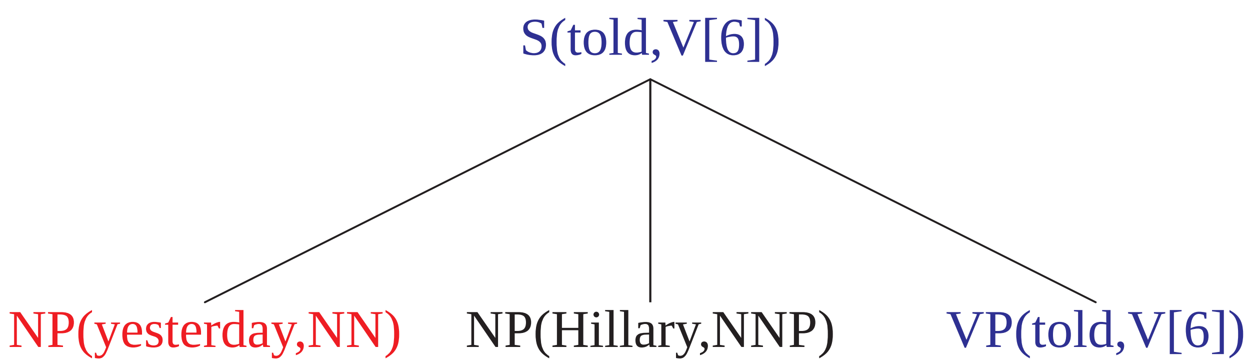
$P_h(\text{VP} \mid \text{S, told, V[6]}) \times$
$P_d(\text{NP(Hillary,NNP)} \mid \text{S,VP,told,V[6],LEFT,}\Delta = 1)$

A Refinement: Adding a Distance Variable

- $\Delta = 1$ if position is adjacent to the head.

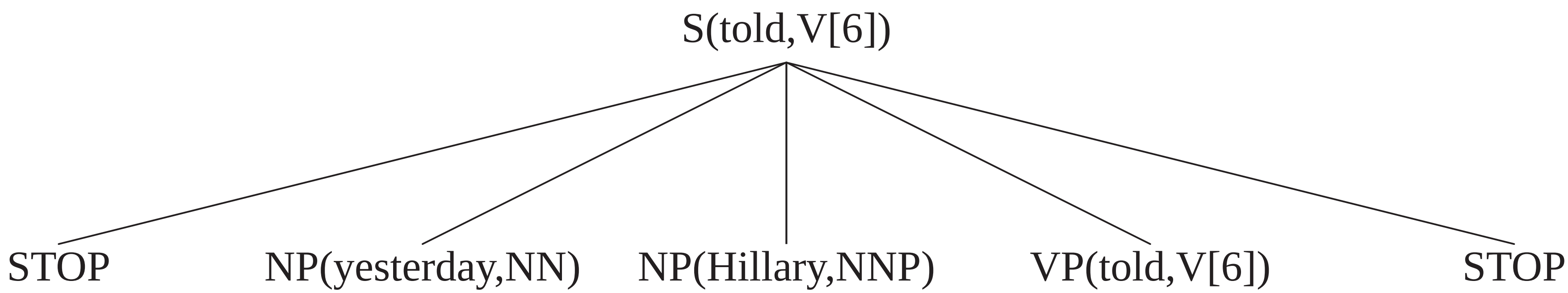


$$\Downarrow$$



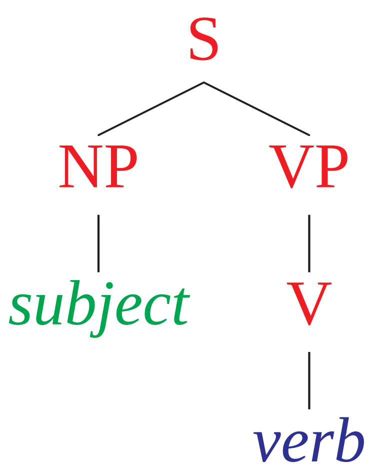
$$P_h(\text{VP} \mid \text{S, told, V[6]}) \times P_d(\text{NP(Hillary,NNP)} \mid \text{S,VP,told,V[6],LEFT}) \times$$
$$P_d(\text{NP(yesterday,NN)} \mid \text{S,VP,told,V[6],LEFT,}\Delta = 0)$$

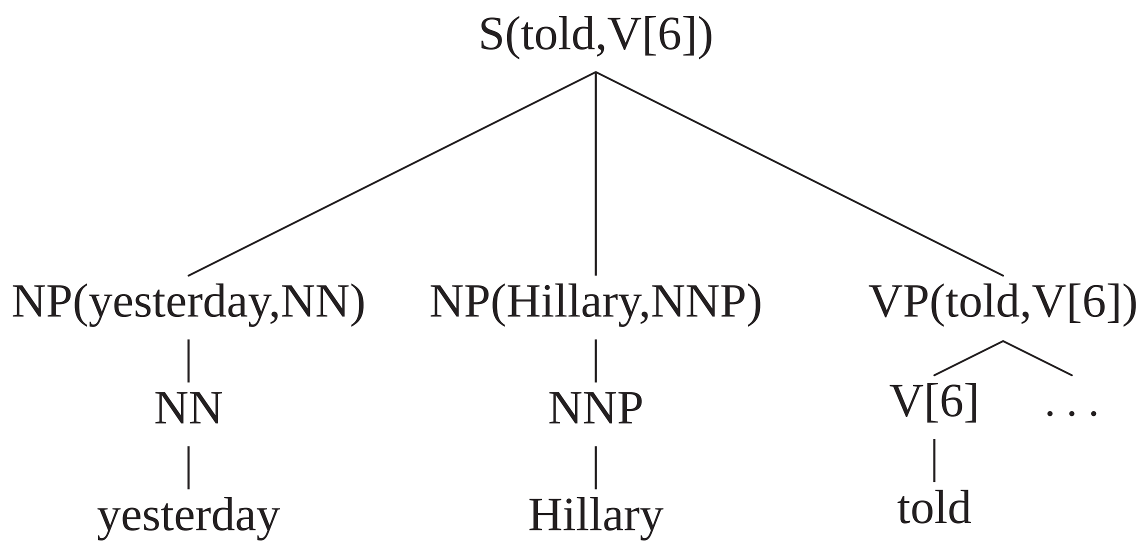
The Final Probabilities



$$
\begin{aligned}
&P_h(\text{VP} \mid \text{S, told, V[6]})\times\\
&P_d(\text{NP(Hillary,NNP)} \mid \text{S,VP,told,V[6],LEFT},\Delta = 1)\times\\
&P_d(\text{NP(yesterday,NN)} \mid \text{S,VP,told,V[6],LEFT},\Delta = 0)\times\\
&P_d(\text{STOP} \mid \text{S,VP,told,V[6],LEFT},\Delta = 0)\times\\
&P_d(\text{STOP} \mid \text{S,VP,told,V[6],RIGHT},\Delta = 1)
\end{aligned}
$$

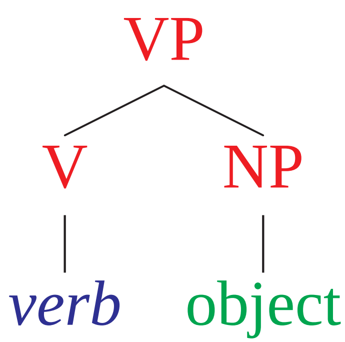
Adding the Complement/Adjunct Distinction

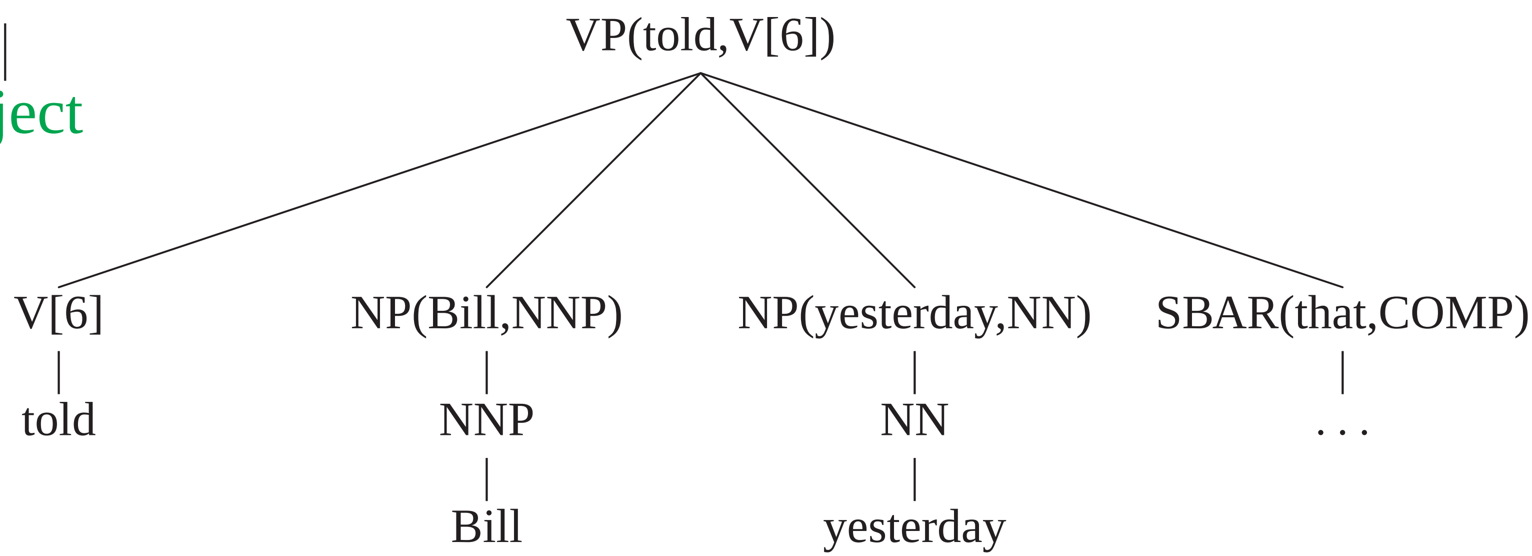




- *Hillary* is the subject
- *yesterday* is a temporal modifier
- **But nothing to distinguish them.**

Adding the Complement/Adjunct Distinction





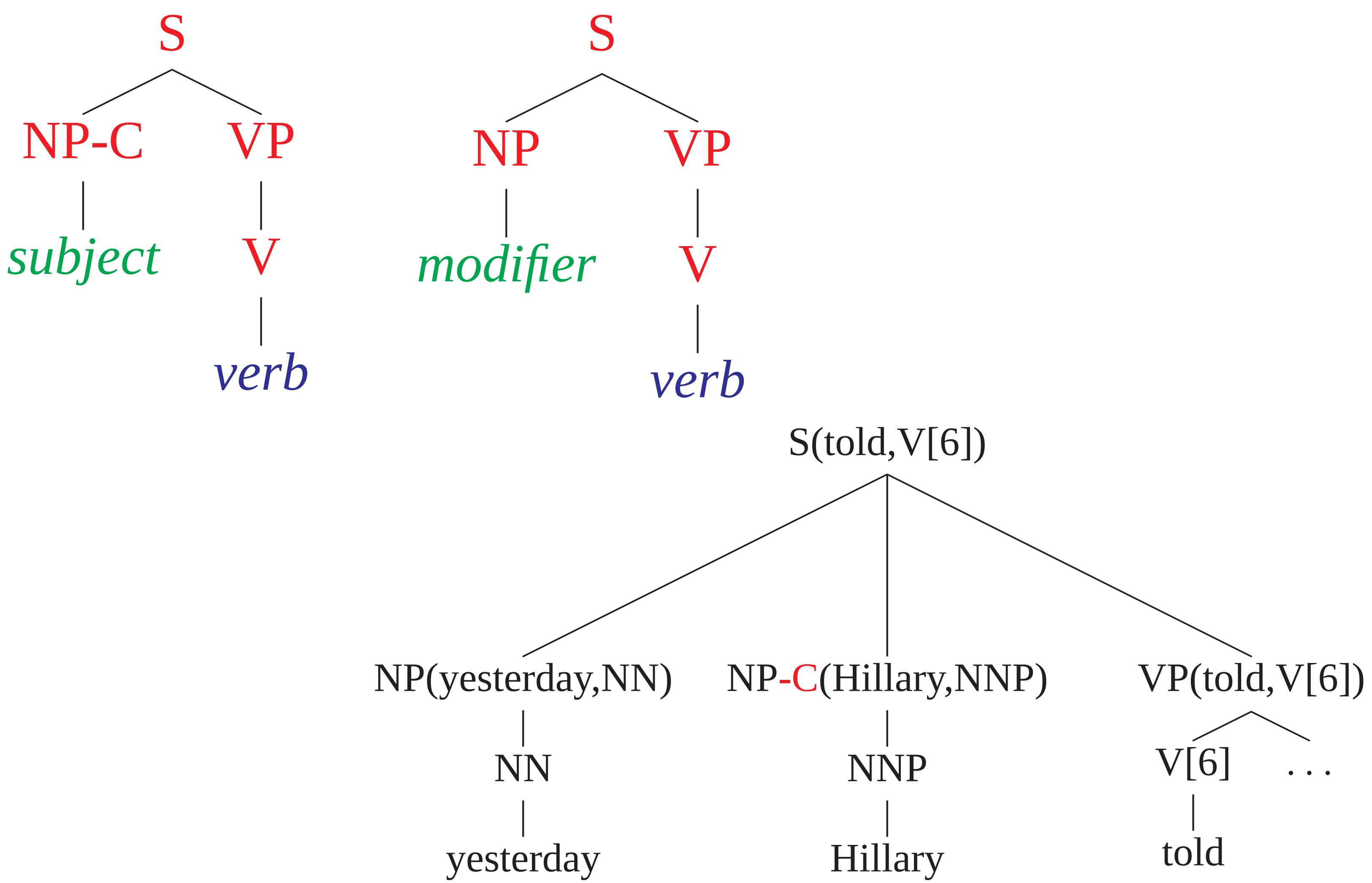
- *Bill* is the object
- *yesterday* is a temporal modifier
- **But nothing to distinguish them.**

Complements vs. Adjuncts

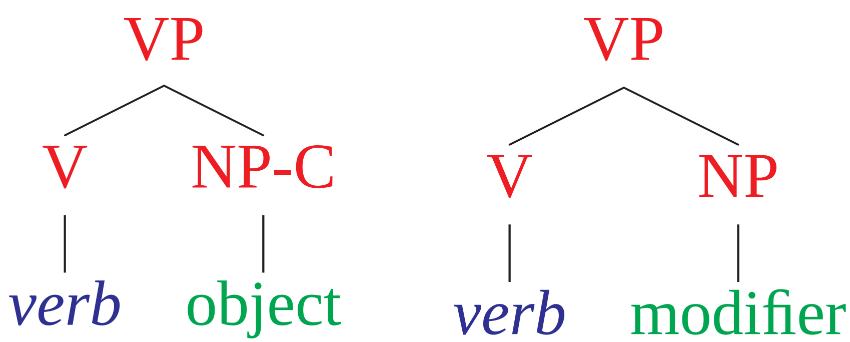
- Complements are closely related to the head they modify, adjuncts are more indirectly related
- Complements are usually arguments of the thing they modify
  yesterday Hillary told . . . $\Rightarrow$ *Hillary* is doing the *telling*
- Adjuncts add modifying information: time, place, manner etc.
  yesterday Hillary told . . . $\Rightarrow$ *yesterday* is a *temporal modifier*
- Complements are usually required, adjuncts are optional

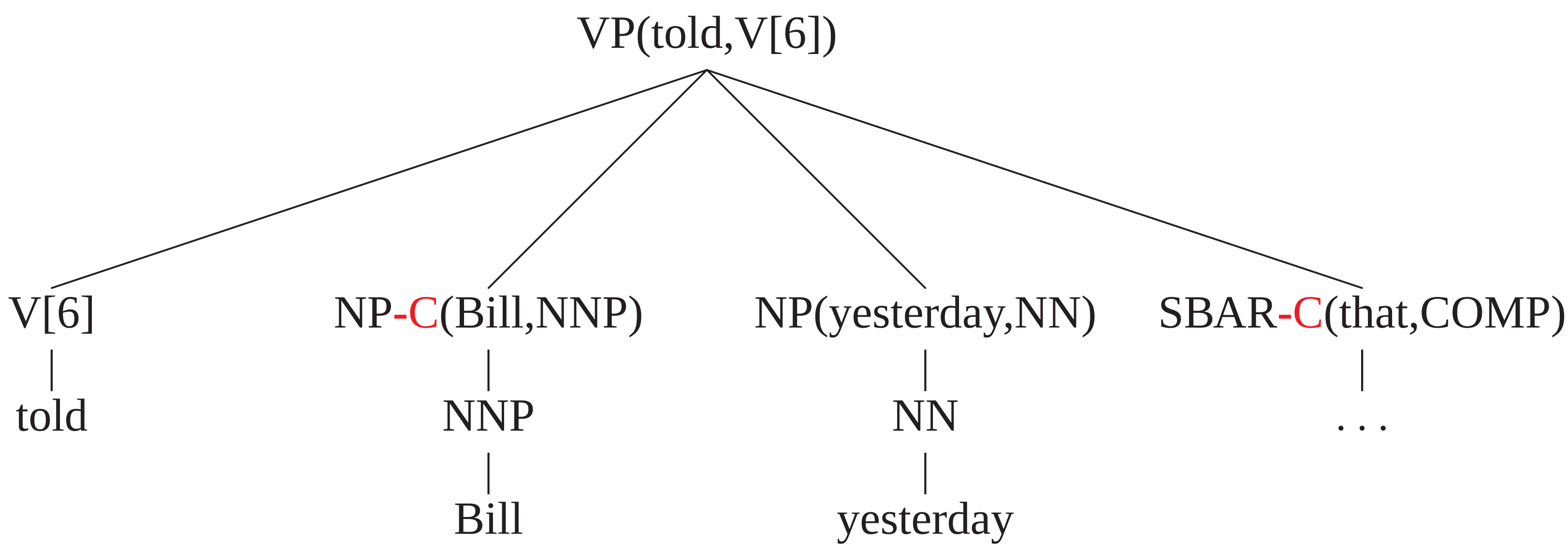
yesterday Hillary told . . . (grammatical)
vs. Hillary told . . . (grammatical)
vs. yesterday told . . . (ungrammatical)

Adding Tags Making the Complement/Adjunct Distinction



Adding Tags Making the Complement/Adjunct Distinction





Adding Subcategorization Probabilities

- Step 1: generate category of head child

$$\text{S(told,V[6])}$$

$$\Downarrow$$

$$\begin{array}{c}\text{S(told,V[6])}\\ | \\ \text{VP(told,V[6])}\end{array}$$

$$P_h(\text{VP} \mid \text{S, told, V[6]})$$

Adding Subcategorization Probabilities

- Step 2: choose left **subcategorization frame**

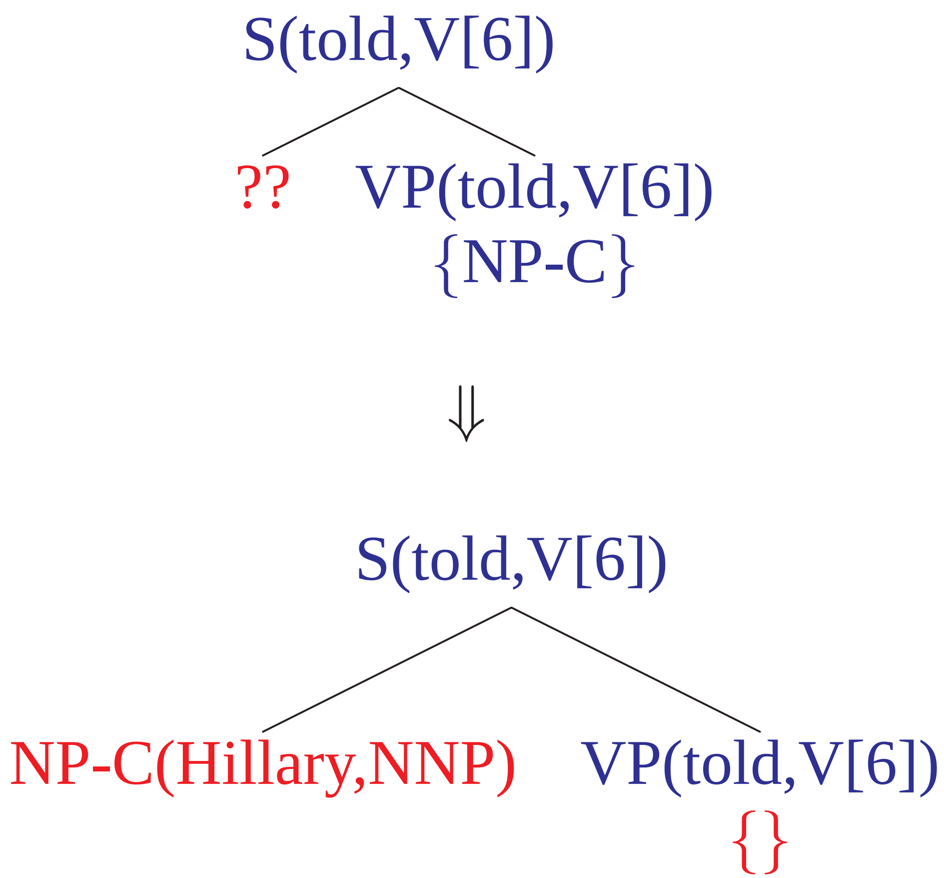
S(told,V[6])
|
VP(told,V[6])

$\Downarrow$

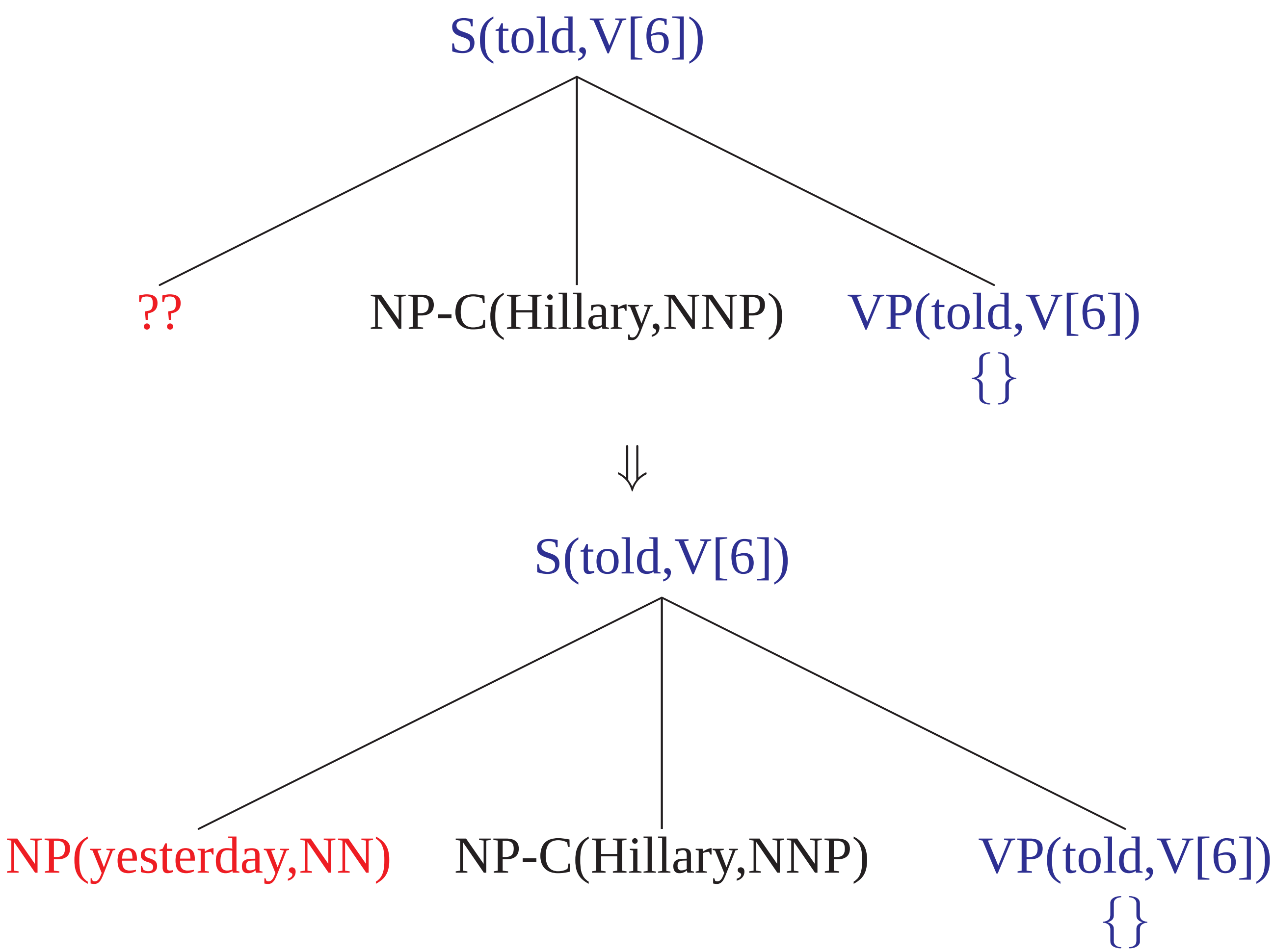
S(told,V[6])
|
VP(told,V[6])
{NP-C}

$$P_h(\text{VP} \mid \text{S, told, V[6]}) \times P_{lc}(\{\text{NP-C}\} \mid \text{S, VP, told, V[6]})$$

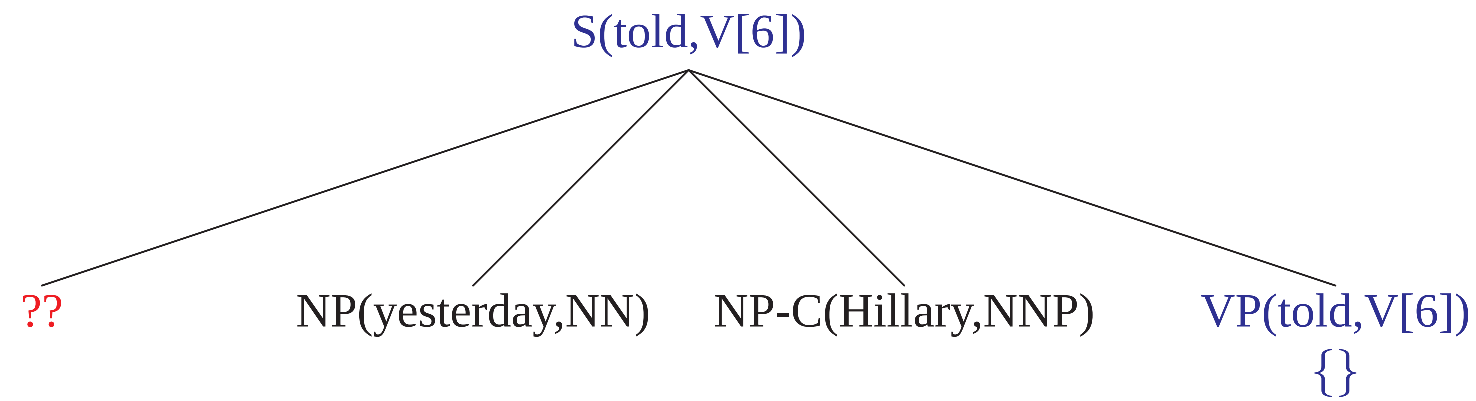
- Step 3: generate left modifiers in a Markov chain



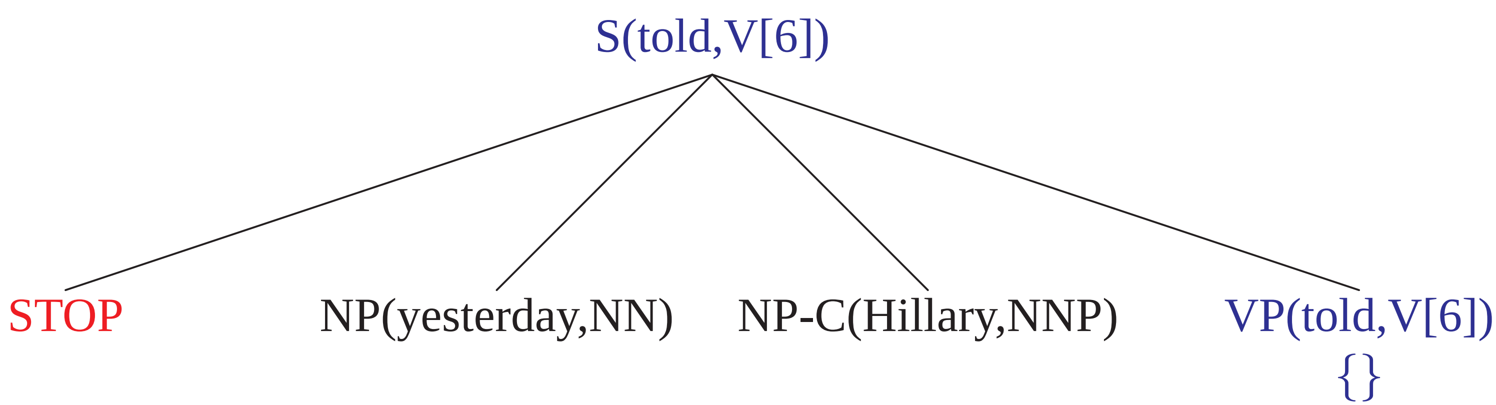
$P_h(\text{VP} \mid \text{S, told, V[6]}) \times P_{lc}(\{\text{NP-C}\} \mid \text{S, VP, told, V[6]}) \times$
$P_d(\text{NP-C(Hillary,NNP)} \mid \text{S,VP,told,V[6],LEFT,}\{\text{NP-C}\})$



$$P_h(\text{VP} \mid \text{S, told, V[6]}) \times P_{lc}(\{\text{NP-C}\} \mid \text{S, VP, told, V[6]})$$
$$P_d(\text{NP-C(Hillary,NNP)} \mid \text{S,VP,told,V[6],LEFT,}\{\text{NP-C}\}) \times$$
$$P_d(\text{NP(yesterday,NN)} \mid \text{S,VP,told,V[6],LEFT,}\{\})$$

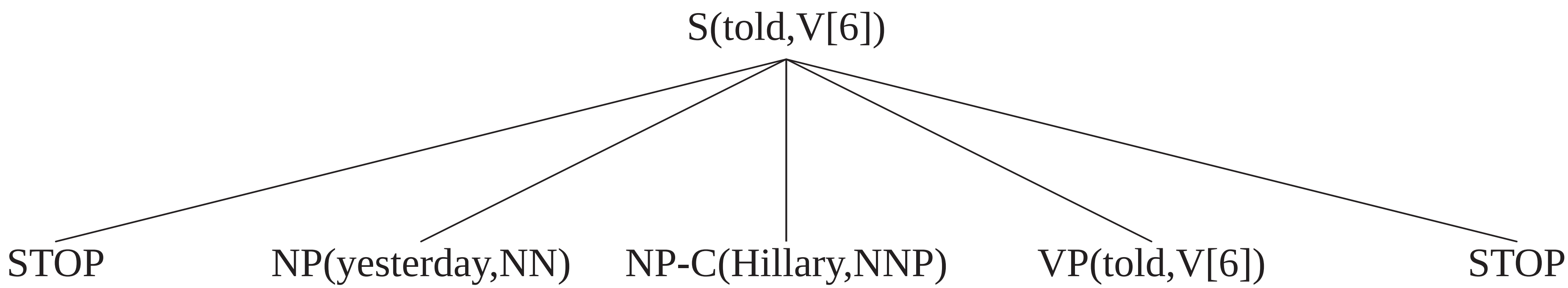


$$\Downarrow$$



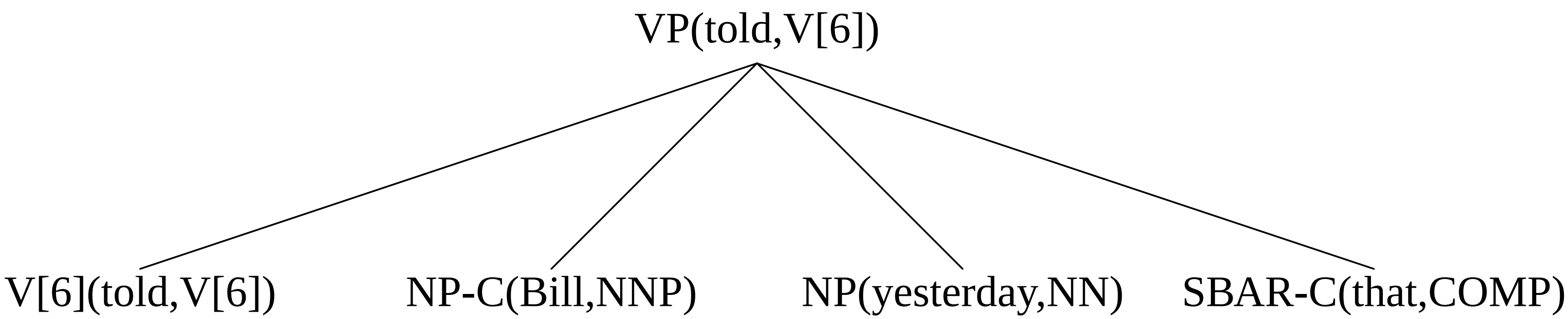
$$
\begin{aligned}
&P_h(\text{VP} \mid \text{S, told, V[6]}) \times P_{lc}(\{\text{NP-C}\} \mid \text{S, VP, told, V[6]})\\
&P_d(\text{NP-C(Hillary,NNP)} \mid \text{S,VP,told,V[6],LEFT,}\{\text{NP-C}\})\times\\
&P_d(\text{NP(yesterday,NN)} \mid \text{S,VP,told,V[6],LEFT,}\{\})\times\\
&P_d(\text{STOP} \mid \text{S,VP,told,V[6],LEFT,}\{\})
\end{aligned}
$$

The Final Probabilities



$$
\begin{aligned}
&P_h(\text{VP} \mid \text{S, told, V[6]}) \times \\
&P_{lc}(\{\text{NP-C}\} \mid \text{S, VP, told, V[6]}) \times \\
&P_d(\text{NP-C(Hillary,NNP)} \mid \text{S,VP,told,V[6],LEFT},\Delta = 1,\{\text{NP-C}\}) \times \\
&P_d(\text{NP(yesterday,NN)} \mid \text{S,VP,told,V[6],LEFT},\Delta = 0,\{\}) \times \\
&P_d(\text{STOP} \mid \text{S,VP,told,V[6],LEFT},\Delta = 0,\{\}) \times \\
&P_{rc}(\{\} \mid \text{S, VP, told, V[6]}) \times \\
&P_d(\text{STOP} \mid \text{S,VP,told,V[6],RIGHT},\Delta = 1,\{\})
\end{aligned}
$$

Another Example

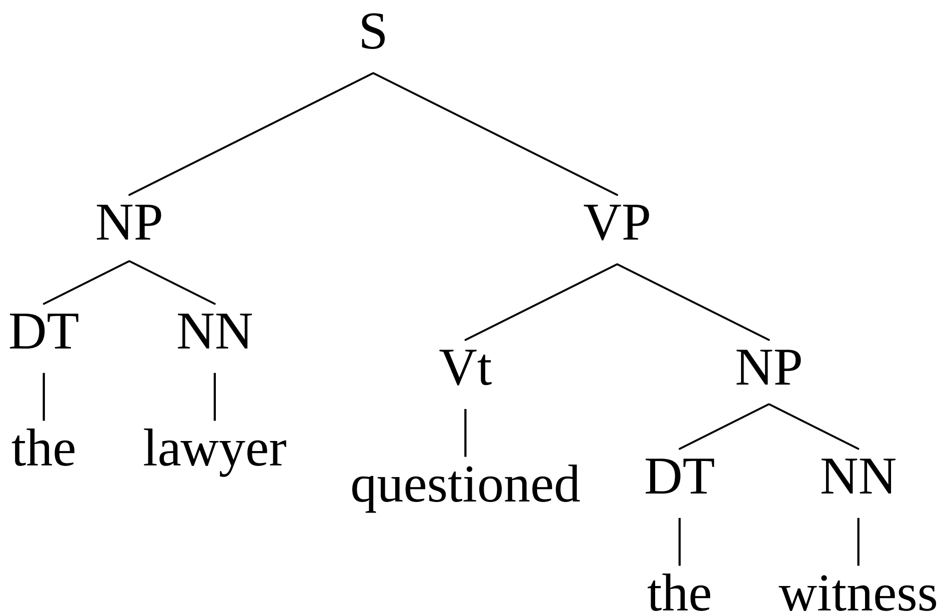


$$
\begin{aligned}
&P_h(\text{V[6]} \mid \text{VP, told, V[6]}) \times \\
&P_{lc}(\{\} \mid \text{VP, V[6], told, V[6]}) \times \\
&P_d(\text{STOP} \mid \text{VP,V[6],told,V[6],LEFT,}\Delta = 1, \{\}) \times \\
&P_{rc}(\{\text{NP-C, SBAR-C}\} \mid \text{VP, V[6], told, V[6]}) \times \\
&P_d(\text{NP-C(Bill,NNP)} \mid \text{VP,V[6],told,V[6],RIGHT,}\Delta = 1, \{\text{NP-C, SBAR-C}\}) \times \\
&P_d(\text{NP(yesterday,NN)} \mid \text{VP,V[6],told,V[6],RIGHT,}\Delta = 0, \{\text{SBAR-C}\}) \times \\
&P_d(\text{SBAR-C(that,COMP)} \mid \text{VP,V[6],told,V[6],RIGHT,}\Delta = 0, \{\text{SBAR-C}\}) \times \\
&P_d(\text{STOP} \mid \text{VP,V[6],told,V[6],RIGHT,}\Delta = 0, \{\})
\end{aligned}
$$

Summary

- Identify heads of rules $\Rightarrow$ dependency representations
- Presented two variants of PCFG methods applied to *lexicalized grammars*.
  - Break generation of rule down into small (markov process) steps
  - Build dependencies back up (distance, subcategorization)

Evaluation: Representing Trees as Constituents



$$\Downarrow$$

Label	Start Point	End Point
NP	1	2
NP	4	5
VP	3	5
S	1	5

Precision and Recall

Label	Start Point	End Point
NP	1	2
NP	4	5
NP	4	8
PP	6	8
NP	7	8
VP	3	8
S	1	8

Label	Start Point	End Point
NP	1	2
NP	4	5
PP	6	8
NP	7	8
VP	3	8
S	1	8

- G = number of constituents in gold standard = 7
- P = number in parse output = 6
- C = number correct = 6

$$\text{Recall} = 100\% \times \frac{C}{G} = 100\% \times \frac{6}{7} \qquad \text{Precision} = 100\% \times \frac{C}{P} = 100\% \times \frac{6}{6}$$

Results

Method	Recall	Precision
PCFGs (Charniak 97)	70.6%	74.8%
Conditional Models – Decision Trees (Magerman 95)	84.0%	84.3%
Lexical Dependencies (Collins 96)	85.3%	85.7%
Conditional Models – Logistic (Ratnaparkhi 97)	86.3%	87.5%
Generative Lexicalized Model (Charniak 97)	86.7%	86.6%
Model 1 (no subcategorization)	87.5%	87.7%
Model 2 (subcategorization)	88.1%	88.3%

Effect of the Different Features

MODEL	A	V	R	P
Model 1	NO	NO	75.0%	76.5%
Model 1	YES	NO	86.6%	86.7%
Model 1	YES	YES	87.8%	88.2%
Model 2	NO	NO	85.1%	86.8%
Model 2	YES	NO	87.7%	87.8%
Model 2	YES	YES	88.7%	89.0%

Results on Section 0 of the WSJ Treebank. Model 1 has no subcategorization, Model 2 has subcategorization. A = YES, V = YES mean that the adjacency/verb conditions respectively were used in the distance measure. **R/P** = recall/precision.

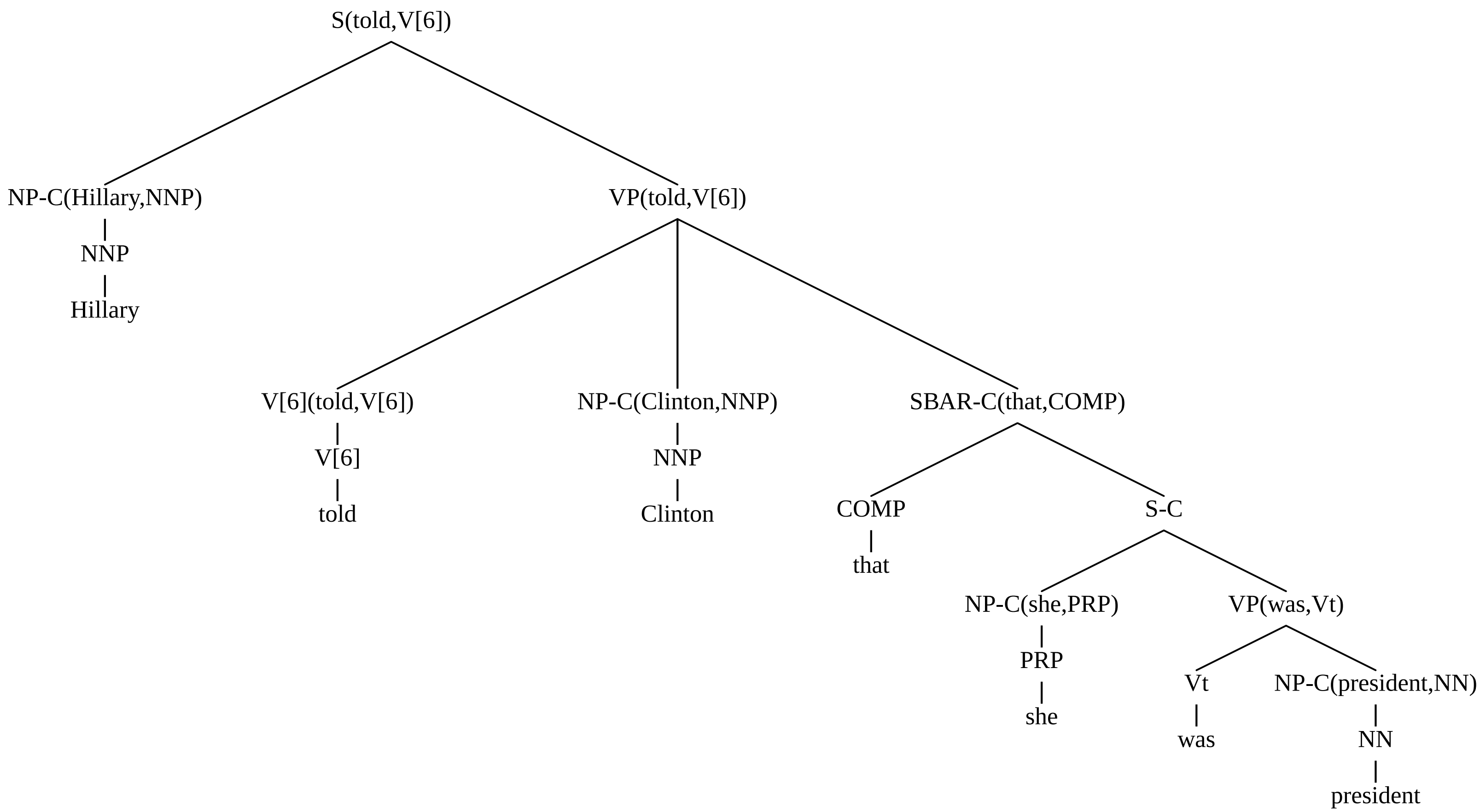
Weaknesses of Precision and Recall

Label	Start Point	End Point
NP	1	2
NP	4	5
NP	4	8
PP	6	8
NP	7	8
VP	3	8
S	1	8

Label	Start Point	End Point
NP	1	2
NP	4	5
PP	6	8
NP	7	8
VP	3	8
S	1	8

NP attachment:
(S (NP The men) (VP dumped (NP (NP sacks) (PP of (NP the substance)))))

VP attachment:
(S (NP The men) (VP dumped (NP sacks) (PP of (NP the substance))))



(__	__	told	V[6]	TOP	S	__	SPECIAL)
(told	V[6]	Hillary	NNP	S	VP	NP-C	LEFT)
(told	V[6]	Clinton	NNP	VP	V[6]	NP-C	RIGHT)
(told	V[6]	that	COMP	VP	V[6]	SBAR-C	RIGHT)
(that	COMP	was	Vt	SBAR-C	COMP	S-C	RIGHT)
(was	Vt	she	PRP	S-C	VP	NP-C	LEFT)
(was	Vt	president	NN	VP	Vt	NP-C	RIGHT)

Dependency Accuracies

- All parses for a sentence with n words have n dependencies
  Report a single figure, dependency accuracy

- Model 2 with all features scores 88.3% dependency accuracy
  (91% if you ignore non-terminal labels on dependencies)

- Can calculate precision/recall on particular dependency types
  e.g., look at all subject/verb dependencies ⇒
  all dependencies with label **(S,VP,NP-C,LEFT)**

$$\text{Recall} = \frac{\text{number of subject/verb dependencies correct}}{\text{number of subject/verb dependencies in gold standard}}$$

$$\text{Precision} = \frac{\text{number of subject/verb dependencies correct}}{\text{number of subject/verb dependencies in parser's output}}$$

R	CP	P	Count	Relation	Rec	Prec
1	29.65	29.65	11786	NPB TAG TAG L	94.60	93.46
2	40.55	10.90	4335	PP TAG NP-C R	94.72	94.04
3	48.72	8.17	3248	S VP NP-C L	95.75	95.11
4	54.03	5.31	2112	NP NPB PP R	84.99	84.35
5	59.30	5.27	2095	VP TAG NP-C R	92.41	92.15
6	64.18	4.88	1941	VP TAG VP-C R	97.42	97.98
7	68.71	4.53	1801	VP TAG PP R	83.62	81.14
8	73.13	4.42	1757	TOP TOP S R	96.36	96.85
9	74.53	1.40	558	VP TAG SBAR-C R	94.27	93.93
10	75.83	1.30	518	QP TAG TAG R	86.49	86.65
11	77.08	1.25	495	NP NPB NP R	74.34	75.72
12	78.28	1.20	477	SBAR TAG S-C R	94.55	92.04
13	79.48	1.20	476	NP NPB SBAR R	79.20	79.54
14	80.40	0.92	367	VP TAG ADVP R	74.93	78.57
15	81.30	0.90	358	NPB TAG NPB L	97.49	92.82
16	82.18	0.88	349	VP TAG TAG R	90.54	93.49
17	82.97	0.79	316	VP TAG SG-C R	92.41	88.22

Accuracy of the 17 most frequent dependency types in section 0 of the treebank, as recovered by model 2. R = rank; CP = cumulative percentage; P = percentage; Rec = Recall; Prec = precision.

Type	Sub-type	Description	Count	Recall	Precision
Complement to a verb	`S VP NP-C L`	Subject	3248	95.75	95.11
	`VP TAG NP-C R`	Object	2095	92.41	92.15
6495 = 16.3% of all cases	`VP TAG SBAR-C R`		558	94.27	93.93
	`VP TAG SG-C R`		316	92.41	88.22
	`VP TAG S-C R`		150	74.67	78.32
	`S VP S-C L`		104	93.27	78.86
	`S VP SG-C L`		14	78.57	68.75
	...				
	TOTAL		6495	93.76	92.96
Other complements	`PP TAG NP-C R`		4335	94.72	94.04
	`VP TAG VP-C R`		1941	97.42	97.98
7473 = 18.8% of all cases	`SBAR TAG S-C R`		477	94.55	92.04
	`SBAR WHNP SG-C R`		286	90.56	90.56
	`PP TAG SG-C R`		125	94.40	89.39
	`SBAR WHADVP S-C R`		83	97.59	98.78
	`PP TAG PP-C R`		51	84.31	70.49
	`SBAR WHNP S-C R`		42	66.67	84.85
	`SBAR TAG SG-C R`		23	69.57	69.57
	`PP TAG S-C R`		18	38.89	63.64
	`SBAR WHPP S-C R`		16	100.00	100.00
	`S ADJP NP-C L`		15	46.67	46.67
	`PP TAG SBAR-C R`		15	100.00	88.24
	...				
	TOTAL		7473	94.47	94.12

Type	Sub-type	Description	Count	Recall	Precision
PP modification	NP NPB PP R		2112	84.99	84.35
	VP TAG PP R		1801	83.62	81.14
4473 = 11.2% of all cases	S VP PP L		287	90.24	81.96
	ADJP TAG PP R		90	75.56	78.16
	ADVP TAG PP R		35	68.57	52.17
	NP NP PP R		23	0.00	0.00
	PP PP PP L		19	21.05	26.67
	NAC TAG PP R		12	50.00	100.00
	...				
	TOTAL		4473	82.29	81.51
Coordination	NP NP NP R		289	55.71	53.31
	VP VP VP R		174	74.14	72.47
763 = 1.9% of all cases	S S S R		129	72.09	69.92
	ADJP TAG TAG R		28	71.43	66.67
	VP TAG TAG R		25	60.00	71.43
	NX NX NX R		25	12.00	75.00
	SBAR SBAR SBAR R		19	78.95	83.33
	PP PP PP R		14	85.71	63.16
	...				
	TOTAL		763	61.47	62.20

Type	Sub-type	Description	Count	Recall	Precision
Mod'n within BaseNPs	NPB TAG TAG L		11786	94.60	93.46
	NPB TAG NPB L		358	97.49	92.82
12742 = 29.6% of all cases	NPB TAG TAG R		189	74.07	75.68
	NPB TAG ADJP L		167	65.27	71.24
	NPB TAG QP L		110	80.91	81.65
	NPB TAG NAC L		29	51.72	71.43
	NPB NX TAG L		27	14.81	66.67
	NPB QP TAG L		15	66.67	76.92
	...				
	TOTAL		12742	93.20	92.59
Mod'n to NPs	NP NPB NP R	Appositive	495	74.34	75.72
	NP NPB SBAR R	Relative clause	476	79.20	79.54
1418 = 3.6% of all cases	NP NPB VP R	Reduced relative	205	77.56	72.60
	NP NPB SG R		63	88.89	81.16
	NP NPB PRN R		53	45.28	60.00
	NP NPB ADVP R		48	35.42	54.84
	NP NPB ADJP R		48	62.50	69.77
	...				
	TOTAL		1418	73.20	75.49

Type	Sub-type	Description	Count	Recall	Precision
Sentential head	TOP TOP S R		1757	96.36	96.85
	TOP TOP SINV R		89	96.63	94.51
1917 = 4.8% of all cases	TOP TOP NP R		32	78.12	60.98
	TOP TOP SG R		15	40.00	33.33
	...				
	TOTAL		1917	94.99	94.99
Adjunct to a verb	VP TAG ADVP R		367	74.93	78.57
	VP TAG TAG R		349	90.54	93.49
2242 = 5.6% of all cases	VP TAG ADJP R		259	83.78	80.37
	S VP ADVP L		255	90.98	84.67
	VP TAG NP R		187	66.31	74.70
	VP TAG SBAR R		180	74.44	72.43
	VP TAG SG R		159	60.38	68.57
	S VP TAG L		115	86.96	90.91
	S VP SBAR L		81	88.89	85.71
	VP TAG ADVP L		79	51.90	49.40
	S VP PRN L		58	25.86	48.39
	S VP NP L		45	66.67	63.83
	S VP SG L		28	75.00	52.50
	VP TAG PRN R		27	3.70	12.50
	VP TAG S R		11	9.09	100.00
	...				
	TOTAL		2242	75.11	78.44

Some Conclusions about Errors in Parsing

- "Core" sentential structure (complements, NP chunks) recovered with over 90% accuracy.

- Attachment ambiguities involving adjuncts are resolved with much lower accuracy ($\approx 80\%$ for PP attachment, $\approx 50 - 60\%$ for coordination).