CSE 517
Natural Language Processing
Winter 2015

Parsing (Trees)

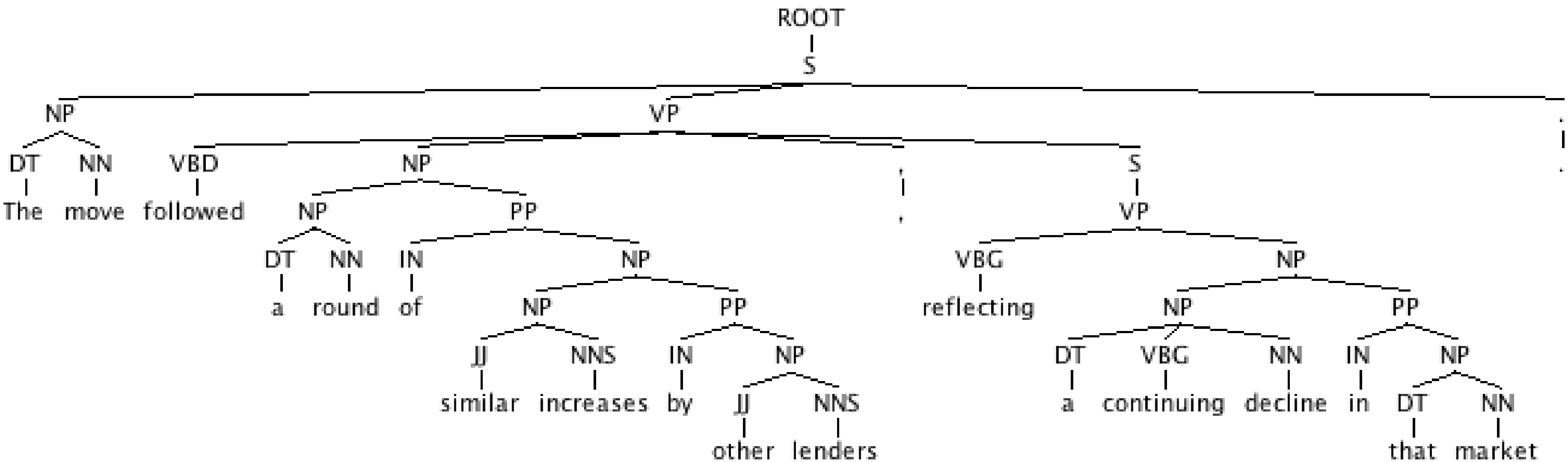
Yejin Choi - University of Washington

[Slides from Dan Klein, Michael Collins, Luke Zettlemoyer and Ray Mooney]

Topics

- Parse Trees
- (Probabilistic) Context Free Grammars
  - Supervised learning
  - Parsing: most likely tree, marginal distributions
- Treebank Parsing (English, edited text)

Parse Trees



The move followed a round of similar increases by other lenders, reflecting a continuing decline in that market

Penn Treebank Non-terminals

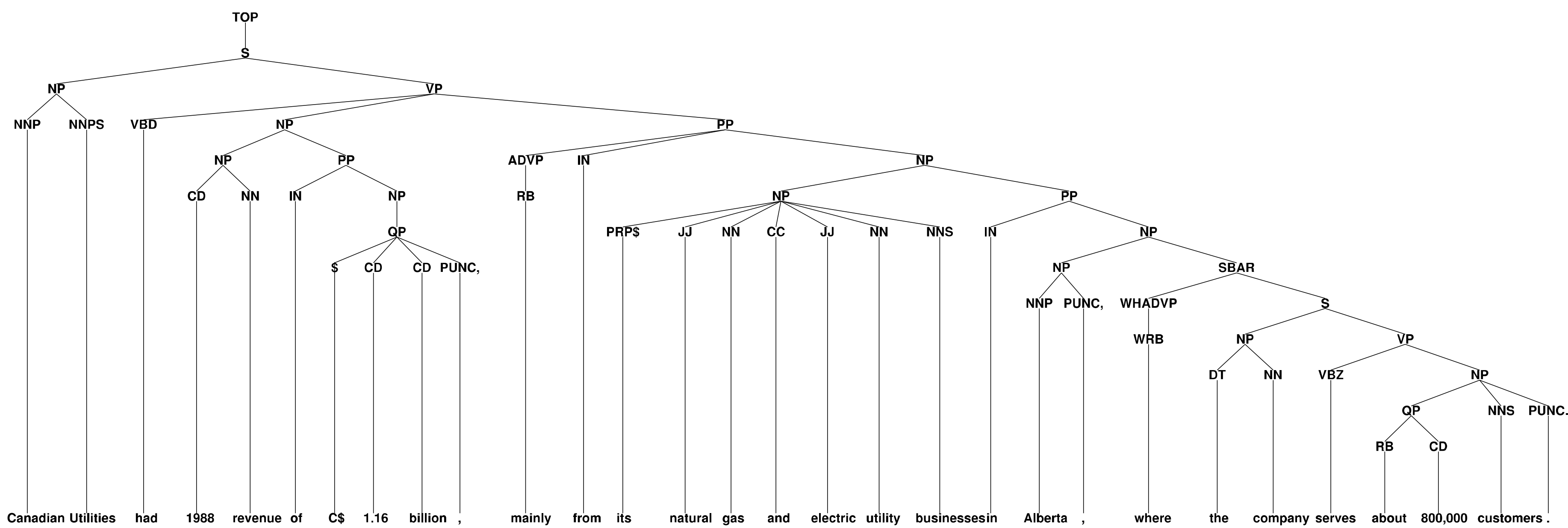
Table 1.2. The Penn Treebank syntactic tagset

ADJP	Adjective phrase
ADVP	Adverb phrase
NP	Noun phrase
PP	Prepositional phrase
S	Simple declarative clause
SBAR	Subordinate clause
SBARQ	Direct question introduced by *wh*-element
SINV	Declarative sentence with subject-aux inversion
SQ	Yes/no questions and subconstituent of SBARQ excluding *wh*-element
VP	Verb phrase
WHADVP	Wh-adverb phrase
WHNP	Wh-noun phrase
WHPP	Wh-prepositional phrase
X	Constituent of unknown or uncertain category
*	"Understood" subject of infinitive or imperative
0	Zero variant of *that* in subordinate clauses
T	Trace of wh-Constituent

The Penn Treebank: Size

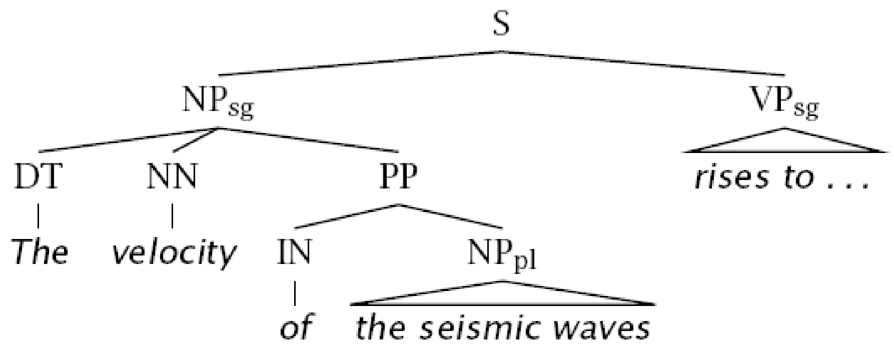
- Penn WSJ Treebank = 50,000 sentences with associated trees
- Usual set-up: 40,000 training sentences, 2400 test sentences

An example tree:



Phrase Structure Parsing

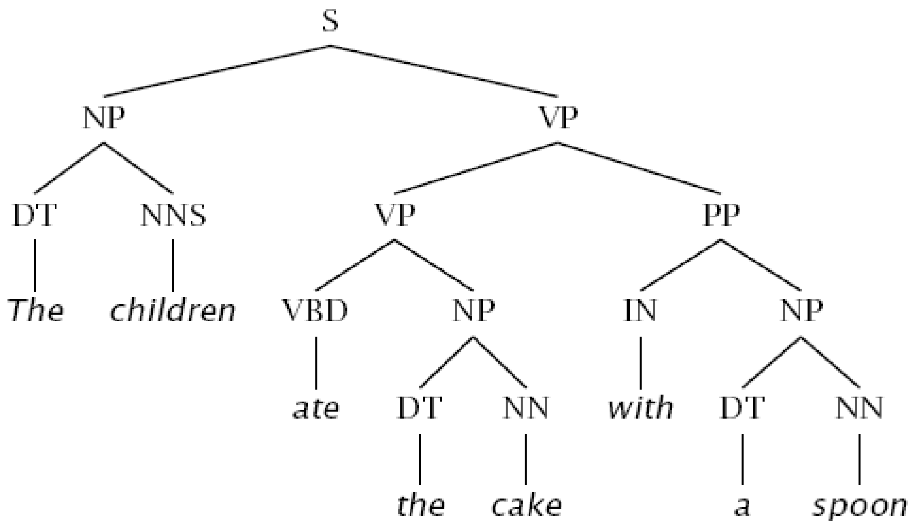
- Phrase structure parsing organizes syntax into *constituents* or *brackets*
- In general, this involves nested trees
- Linguists can, and do, argue about details
- Lots of ambiguity
- Not the only kind of syntax…



new art critics write reviews with computers

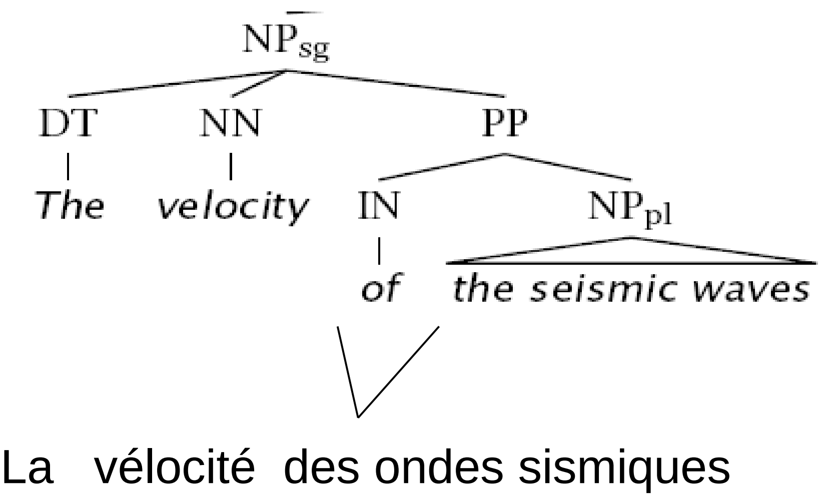
Constituency Tests

- How do we know what nodes go in the tree?
- Classic constituency tests:
  - Substitution by proform
    - he, she, it, they, ...
  - Question / answer
  - Deletion
  - Movement / dislocation
  - Conjunction / coordination
- Cross-linguistic arguments, too



Conflicting Tests

- Constituency isn't always clear
  - Units of transfer:
    - think about ~ penser à
    - talk about ~ hablar de
  - Phonological reduction:
    - I will go → I'll go
    - I want to go → I wanna go
    - a le centre → au centre

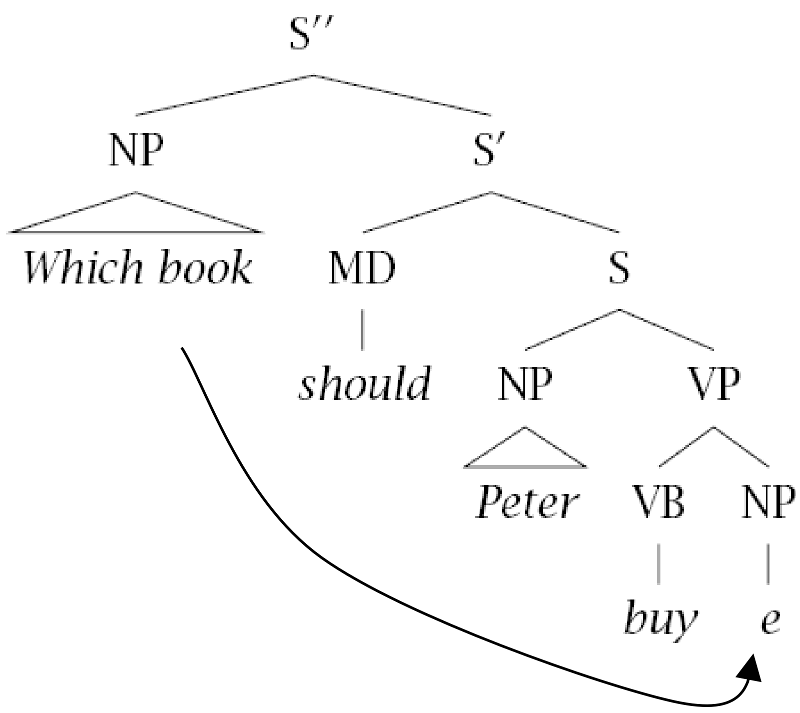


  - Coordination
    - He went to and came from the store.

Non-Local Phenomena

- Dislocation / gapping
  - Which book should Peter buy?
  - A debate arose which continued until the election.

- Binding
  - Reference
    - The IRS audits itself
  - Control
    - I want to go
    - I want you to go



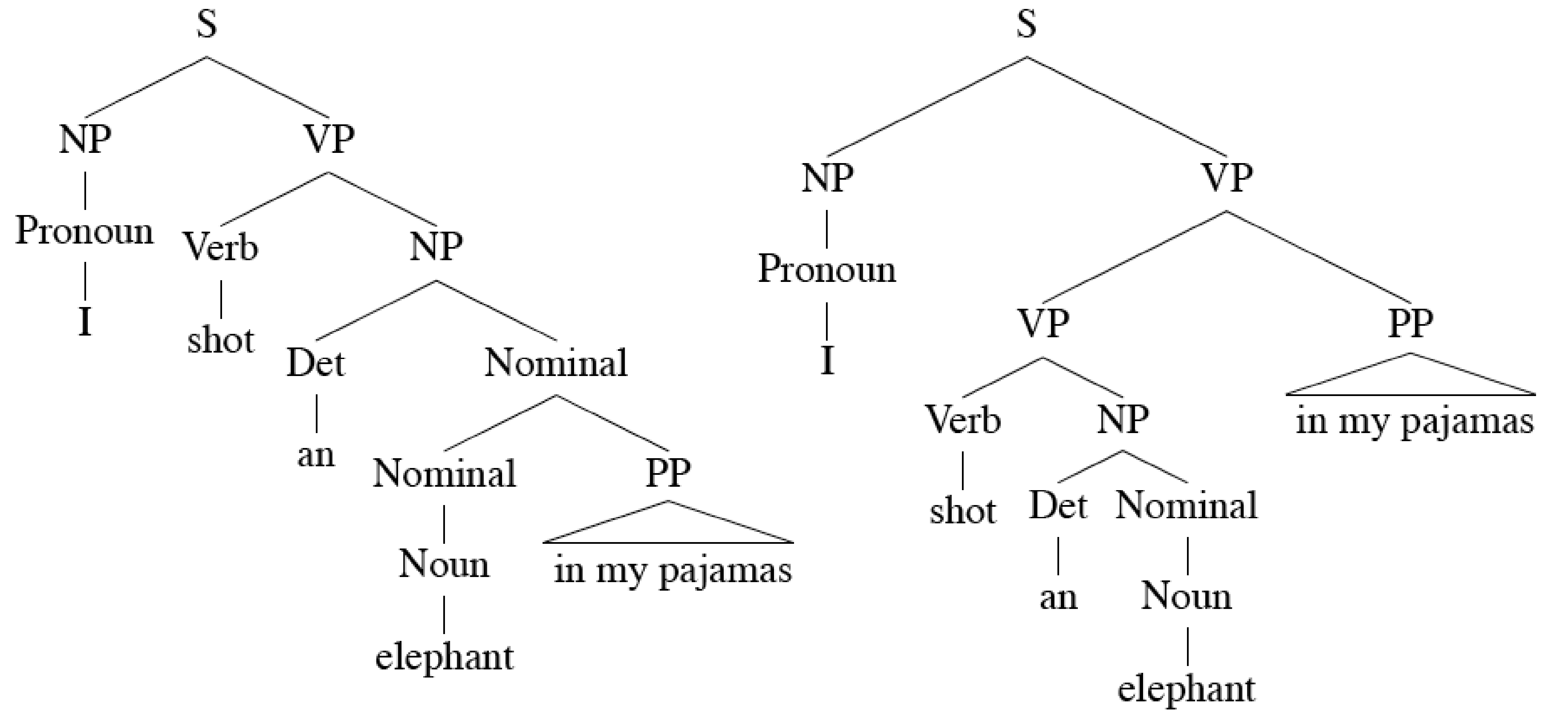
Classical NLP: Parsing

- Write symbolic or logical rules:

Grammar (CFG)		Lexicon
ROOT → S	NP → NP PP	NN → interest
S → NP VP	VP → VBP NP	NNS → raises
NP → DT NN	VP → VBP NP PP	VBP → interest
NP → NN NNS	PP → IN NP	VBZ → raises
		…

- Use deduction systems to prove parses from words
  - Minimal grammar on “Fed raises” sentence: 36 parses
  - Simple 10-rule grammar: 592 parses
  - Real-size grammar: many millions of parses
- This scaled very badly, didn’t yield broad-coverage tools

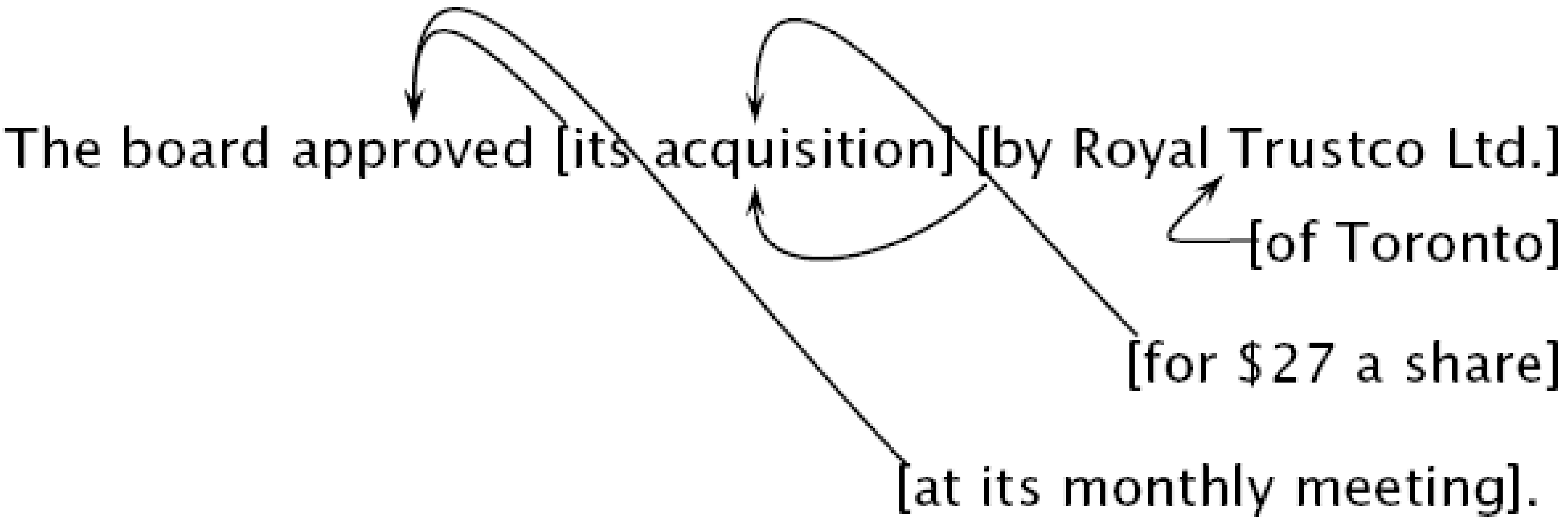
Ambiguity



Examples from J&M

Attachments

- I cleaned the dishes from dinner
- I cleaned the dishes with detergent
- I cleaned the dishes in my pajamas
- I cleaned the dishes in the sink



Syntactic Ambiguities I

- Prepositional phrases:
  They cooked the beans in the pot on the stove with handles.
- Particle vs. preposition:
  The puppy tore up the staircase.
- Complement structures
  The tourists objected to the guide that they couldn't hear.
  She knows you like the back of her hand.
- Gerund vs. participial adjective
  Visiting relatives can be boring.
  Changing schedules frequently confused passengers.

Syntactic Ambiguities II

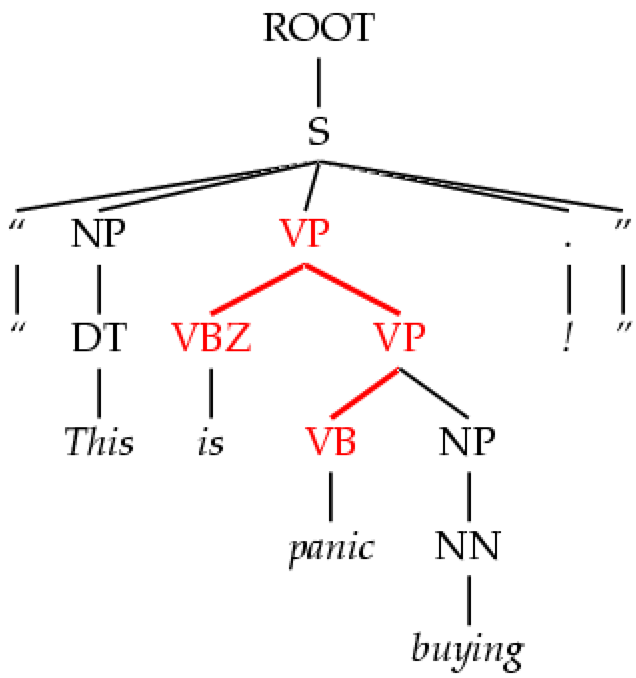
- Modifier scope within NPs
  impractical design requirements
  plastic cup holder
- Multiple gap constructions
  The chicken is ready to eat.
  The contractors are rich enough to sue.
- Coordination scope:
  Small rats and mice can squeeze into holes or cracks in the wall.

Dark Ambiguities

- *Dark ambiguities*: most analyses are shockingly bad (meaning, they don't have an interpretation you can get your mind around)

This analysis corresponds to the correct parse of

"This will panic buyers ! "



- Unknown words and new usages
- Solution: We need mechanisms to focus attention on the best ones, probabilistic techniques do this

Context-Free Grammars

- A context-free grammar is a tuple <*N*, Σ , *S*, *R*>
  - *N* : the set of non-terminals
    - Phrasal categories: S, NP, VP, ADJP, etc.
    - Parts-of-speech (pre-terminals): NN, JJ, DT, VB
  - Σ : the set of terminals (the words)
  - *S* : the start symbol
    - Often written as ROOT or TOP
    - *Not* usually the sentence non-terminal S
  - *R* : the set of rules
    - Of the form $X \rightarrow Y_1\ Y_2 \ldots Y_n$, with $X \in N$, $n \geq 0$, $Y_i \in (N \cup \Sigma)$
    - Examples: $S \rightarrow NP\ VP$, $VP \rightarrow VP\ CC\ VP$
    - Also called rewrites, productions, or local trees

Example Grammar

N = {S, NP, VP, PP, DT, Vi, Vt, NN, IN}
S = S
Σ = {sleeps, saw, man, woman, telescope, the, with, in}

R =

S	⇒	NP	VP
VP	⇒	Vi	
VP	⇒	Vt	NP
VP	⇒	VP	PP
NP	⇒	DT	NN
NP	⇒	NP	PP
PP	⇒	IN	NP

Vi	⇒	sleeps
Vt	⇒	saw
NN	⇒	man
NN	⇒	woman
NN	⇒	telescope
DT	⇒	the
IN	⇒	with
IN	⇒	in

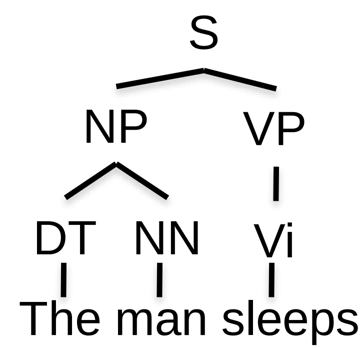
S=sentence, VP-verb phrase, NP=noun phrase, PP=prepositional phrase, DT=determiner, Vi=intransitive verb, Vt=transitive verb, NN=noun, IN=preposition

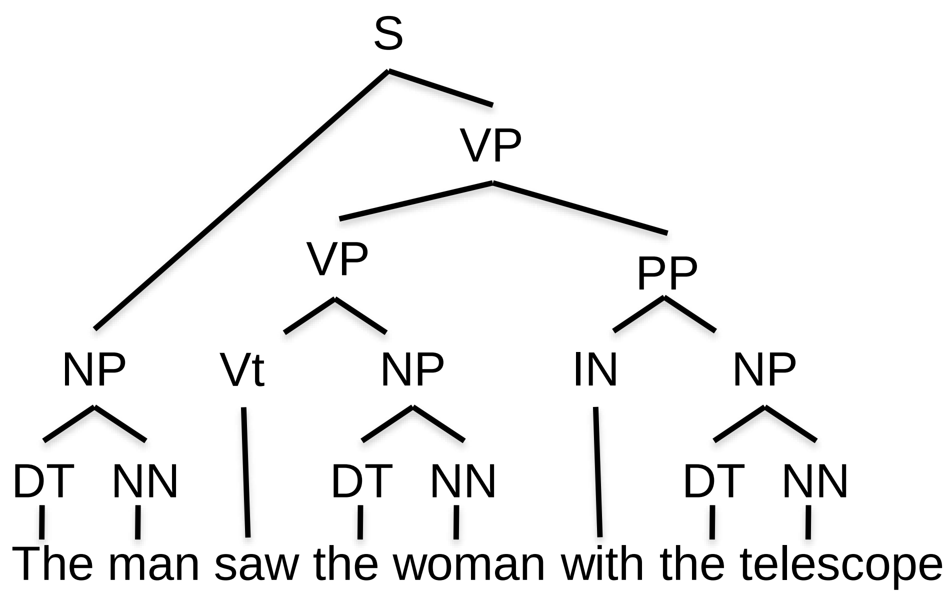
Example Parses

R =

S	⇒	NP	VP
VP	⇒	Vi	
VP	⇒	Vt	NP
VP	⇒	VP	PP
NP	⇒	DT	NN
NP	⇒	NP	PP
PP	⇒	IN	NP

Vi	⇒	sleeps
Vt	⇒	saw
NN	⇒	man
NN	⇒	woman
NN	⇒	telescope
DT	⇒	the
IN	⇒	with
IN	⇒	in





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

Probabilistic Context-Free Grammars

- A context-free grammar is a tuple <*N*, Σ ,*S*, *R*>
  - *N* : the set of non-terminals
    - Phrasal categories: S, NP, VP, ADJP, etc.
    - Parts-of-speech (pre-terminals): NN, JJ, DT, VB, etc.
  - Σ : the set of terminals (the words)
  - *S* : the start symbol
    - Often written as ROOT or TOP
    - *Not* usually the sentence non-terminal S
  - *R* : the set of rules
    - Of the form $X \rightarrow Y_1\ Y_2 \ldots Y_n$, with $X \in N$, $n \geq 0$, $Y_i \in (N \cup \Sigma)$
    - Examples: $S \rightarrow NP\ VP$, $VP \rightarrow VP\ CC\ VP$
- A PCFG adds a distribution q:
  - Probability q(r) for each $r \in R$, such that for all $X \in N$:

$$\sum_{\alpha \rightarrow \beta \in R : \alpha = X} q(\alpha \rightarrow \beta) = 1$$

PCFG Example

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	$\Rightarrow$	woman	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 t with rules

$$\alpha_1 \rightarrow \beta_1, \alpha_2 \rightarrow \beta_2, \ldots, \alpha_n \rightarrow \beta_n$$

is

$$p(t) = \prod_{i=1}^{n} q(\alpha_i \rightarrow \beta_i)$$

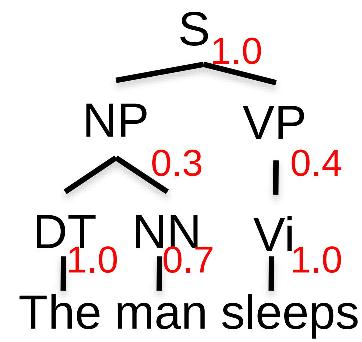
where $q(\alpha \rightarrow \beta)$ is the probability for rule $\alpha \rightarrow \beta$.

PCFG Example

S	⇒	NP	VP	1.0
VP	⇒	Vi		0.4
VP	⇒	Vt	NP	0.4
VP	⇒	VP	PP	0.2
NP	⇒	DT	NN	0.3
NP	⇒	NP	PP	0.7
PP	⇒	P	NP	1.0

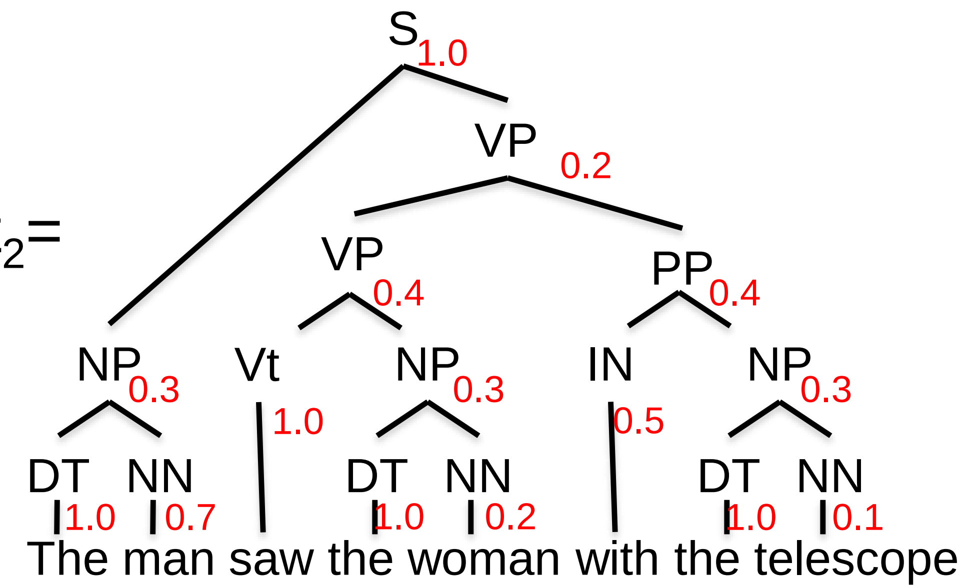
Vi	⇒	sleeps	1.0
Vt	⇒	saw	1.0
NN	⇒	man	0.7
NN	⇒	woman	0.2
NN	⇒	telescope	0.1
DT	⇒	the	1.0
IN	⇒	with	0.5
IN	⇒	in	0.5

t_1=



$p(t_1)=1.0*0.3*1.0*0.7*0.4*1.0$

t_2=



$p(t_s)=1.8*0.3*1.0*0.7*0.2*0.4*1.0*0.3*1.0*0.2*0.4*0.5*0.3*1.0*0.1$

PCFGs: Learning and Inference

- Model
  - The probability of a tree t with n rules $\alpha_i \rightarrow \beta_i$, i = 1..n

$$p(t) = \prod_{i=1}^{n} q(\alpha_i \rightarrow \beta_i)$$

- Learning
  - Read the rules off of labeled sentences, use ML estimates for probabilities

$$q_{ML}(\alpha \rightarrow \beta) = \frac{\text{Count}(\alpha \rightarrow \beta)}{\text{Count}(\alpha)}$$

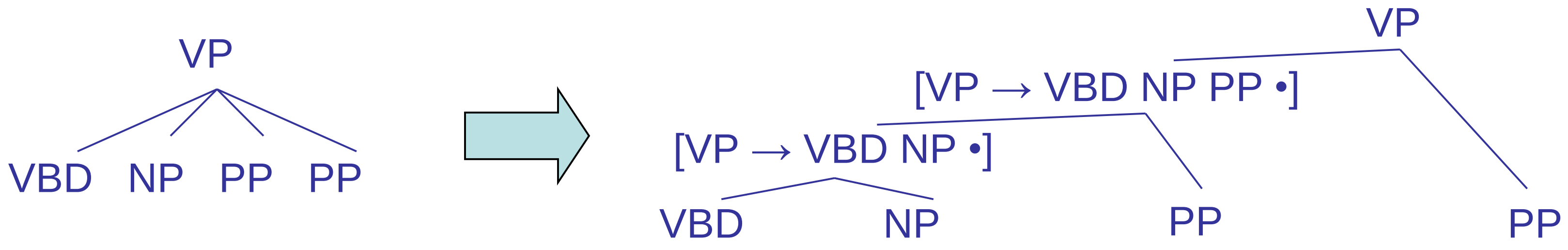
  - and use all of our standard smoothing tricks!

- Inference
  - For input sentence s, define T(s) to be the set of trees whole *yield* is s (whole leaves, read left to right, match the words in s)

$$t^*(s) = \arg \max_{t \in \mathcal{T}(s)} p(t)$$

Chomsky Normal Form

- Chomsky normal form:
  - All rules of the form X → Y Z or X → w
  - In principle, this is no limitation on the space of (P)CFGs
    - N-ary rules introduce new non-terminals



    - Unaries / empties are "promoted"
  - In practice it's kind of a pain:
    - Reconstructing n-aries is easy
    - Reconstructing unaries is trickier
    - The straightforward transformations don't preserve tree scores
  - Makes parsing algorithms simpler!

Original Grammar

CNF Conversion Example

S → NP VP	0.8
S → Aux NP VP	0.1
S → VP	**0.1**
NP → Pronoun	0.2
NP → Proper-Noun	0.2
NP → Det Nominal	0.6
Nominal → Noun	0.3
Nominal → Nominal Noun	0.2
Nominal → Nominal PP	0.5
VP → Verb	**0.2**
VP → Verb NP	0.5
VP → VP PP	0.3
PP → Prep NP	1.0

Lexicon:

Noun → book | flight | meal | money
0.1 0.5 0.2 0.2

Verb → book | include | prefer
0.5 0.2 0.3

Det → the | a | that | this
0.6 0.2 0.1 0.1

Pronoun → I | he | she | me
0.5 0.1 0.1 0.3

Proper-Noun → Houston | NWA
0.8 0.2

Aux → does
1.0

Prep → from | to | on | near | through
0.25 0.25 0.1 0.2 0.2

Original Grammar		Chomsky Normal Form	
S → NP VP	0.8	S → NP VP	0.8
S → Aux NP VP	0.1	S → X1 VP	0.1
		X1 → Aux NP	1.0
S → VP	0.1		
NP → Pronoun	0.2		
NP → Proper-Noun	0.2		
NP → Det Nominal	0.6		
Nominal → Noun	0.3		
Nominal → Nominal Noun	0.2		
Nominal → Nominal PP	0.5		
VP → Verb	0.2		
VP → Verb NP	0.5		
VP → VP PP	0.3		
PP → Prep NP	1.0		

Lexicon (See previous slide for full list) :

Noun → book | flight | meal | money
0.1 0.5 0.2 0.2

Verb → book | include | prefer
0.5 0.2 0.3

Original Grammar		Chomsky Normal Form	
S → NP VP	0.8	S → NP VP	0.8
S → Aux NP VP	0.1	S → X1 VP	0.1
		X1 → Aux NP	1.0
S → VP	**0.1**	S → book | include | prefer	
		S → Verb NP	
		S → VP PP	
NP → Pronoun	0.2		
NP → Proper-Noun	0.2		
NP → Det Nominal	0.6		
Nominal → Noun	0.3		
Nominal → Nominal Noun	0.2		
Nominal → Nominal PP	0.5		
VP → Verb	**0.2**		
VP → Verb NP	**0.5**		
VP → VP PP	**0.3**		
PP → Prep NP	1.0		

Lexicon (See previous slide for full list) :

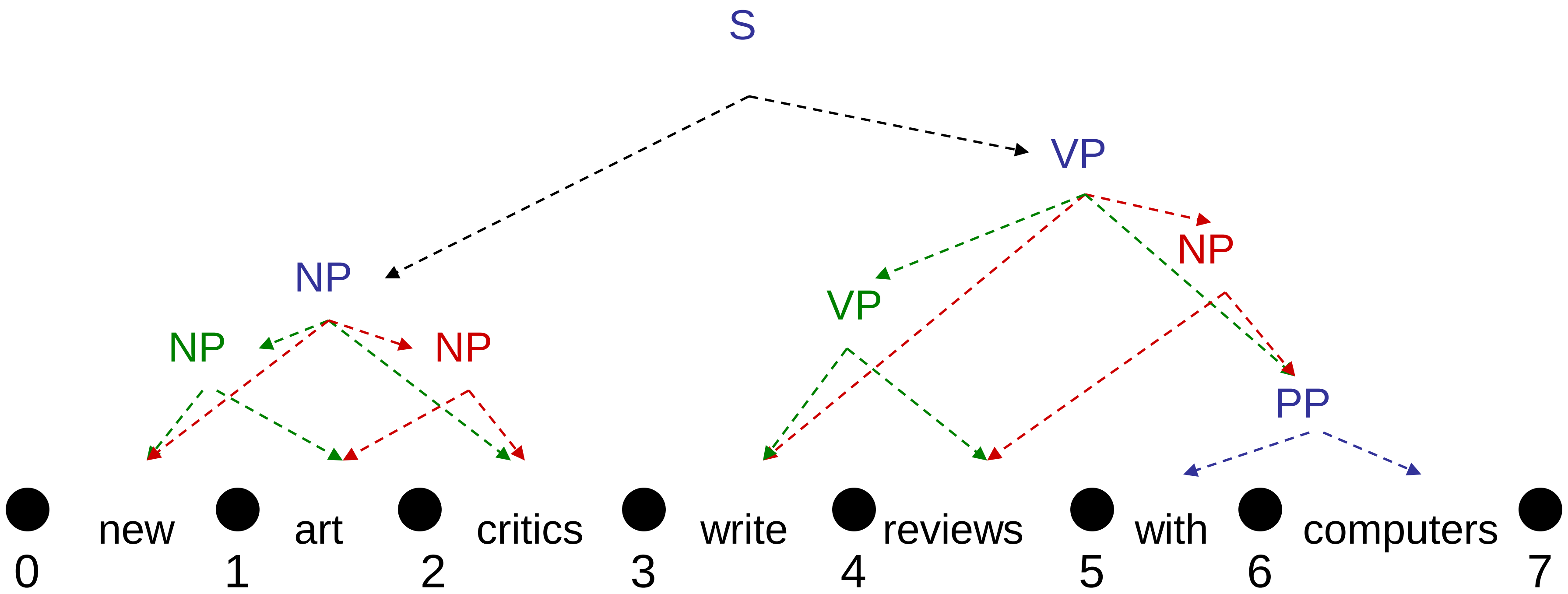
Noun → book | flight | meal | money
0.1 0.5 0.2 0.2

Verb → book | include | prefer
0.5 0.2 0.3

Original Grammar		Chomsky Normal Form	
S → NP VP	0.8	S → NP VP	0.8
S → Aux NP VP	0.1	S → X1 VP	0.1
		X1 → Aux NP	1.0
S → VP	**0.1**	S → book | include | prefer 0.01 0.004 0.006	
		S → Verb NP	0.05
		S → VP PP	0.03
NP → Pronoun	0.2	NP → I | he | she | me 0.1 0.02 0.02 0.06	
NP → Proper-Noun	0.2	NP → Houston | NWA 0.16 .04	
NP → Det Nominal	0.6	NP → Det Nominal	0.6
Nominal → Noun	0.3	Nominal → book | flight | meal | money 0.03 0.15 0.06 0.06	
Nominal → Nominal Noun	0.2	Nominal → Nominal Noun	0.2
Nominal → Nominal PP	0.5	Nominal → Nominal PP	0.5
VP → Verb	**0.2**	VP → book | include | prefer 0.1 0.04 0.06	
VP → Verb NP	**0.5**	VP → Verb NP	0.5
VP → VP PP	**0.3**	VP → VP PP	0.3
PP → Prep NP	1.0	PP → Prep NP	1.0

Lexicon (See previous slide for full list) :
Noun → book | flight | meal | money
0.1 0.5 0.2 0.2
Verb → book | include | prefer
0.5 0.2 0.3

The Parsing Problem



A Recursive Parser

```
bestScore(X,i,j,s)
  if (j == i)
      return q(X->s[i])
  else
      return max q(X->YZ) *
       k,X->YZ
               bestScore(Y,i,k,s) *
               bestScore(Z,k+1,j,s)
```

- Will this parser work?
- Why or why not?
- Memory/time requirements?
- Q: Remind you of anything? Can we adapt this to other models / inference tasks?

Dynamic Programming

- We will store: score of the max parse of x_i to x_j with root non-terminal X

$$\pi(i, j, X)$$

- So we can compute the most likely parse:

$$\pi(1, n, S) = \arg \max_{t \in \mathcal{T}_G(s)}$$

- Via the recursion:

$$\pi(i, j, X) = \max_{\substack{X \rightarrow Y Z \in R, \\ s \in \{i \ldots (j-1)\}}} (q(X \rightarrow Y Z) \times \pi(i, s, Y) \times \pi(s+1, j, Z))$$

- With base case:

$$\pi(i, i, X) = \begin{cases} q(X \rightarrow x_i) & \text{if } X \rightarrow x_i \in R \\ 0 & \text{otherwise} \end{cases}$$

The CKY Algorithm

- Input: a sentence s = x_1 .. x_n and a PCFG = <N, Σ ,S, R, q>
- Initialization: For i = 1 … n and all X in N

$$\pi(i,i,X) \;=\; \begin{cases} q(X \rightarrow x_i) & \text{if } X \rightarrow x_i \in R \\ 0 & \text{otherwise} \end{cases}$$

- For l = 1 … (n-1) [iterate all phrase lengths]
  - For i = 1 … (n-l) and j = i+l [iterate all phrases of length l]
    - For all X in N [iterate all non-terminals]

$$\pi(i,j,X) = \max_{\substack{X \rightarrow Y Z \in R, \\ s \in \{i \ldots (j-1)\}}} \left(q(X \rightarrow Y Z) \times \pi(i,s,Y) \times \pi(s+1,j,Z) \right)$$

    - also, store back pointers

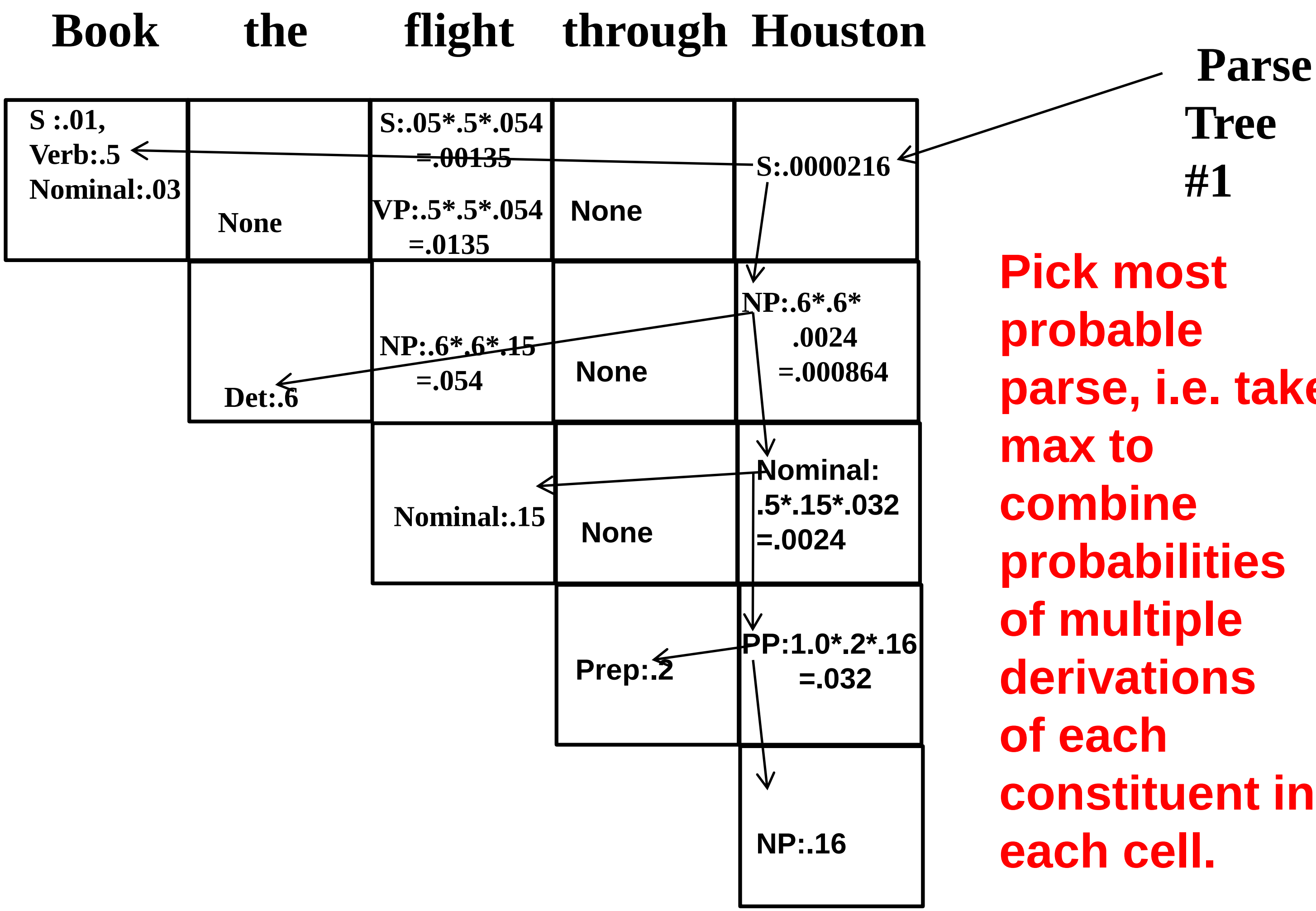
$$bp(i,j,X) = \arg \max_{\substack{X \rightarrow Y Z \in R, \\ s \in \{i \ldots (j-1)\}}} \left(q(X \rightarrow Y Z) \times \pi(i,s,Y) \times \pi(s+1,j,Z) \right)$$

Probabilistic CKY Parser

S → NP VP	0.8
S → X1 VP	0.1
X1 → Aux NP	1.0
S → book | include | prefer 0.01 0.004 0.006	
S → Verb NP	0.05
S → VP PP	0.03
NP → I | he | she | me 0.1 0.02 0.02 0.06	
NP → Houston | NWA 0.16 .04	
Det→ the | a | an 0.6 0.1 0.05	
NP → Det Nominal	0.6
Nominal → book | flight | meal | money 0.03 0.15 0.06 0.06	
Nominal → Nominal Nominal	0.2
Nominal → Nominal PP	0.5
Verb→ book | include | prefer 0.5 0.04 0.06	
VP → Verb NP	0.5
VP → VP PP	0.3
Prep → through | to | from 0.2 0.3 0.3	
PP → Prep NP	1.0

Book	the	flight	through	Houston
S :.01, Verb:.5 Nominal:.03	None	S:.05*.5*.054 =.00135 VP:.5*.5*.054 =.0135	None	S:.03*.0135*.032 =.00001296 S:.05*.5* .000864 =.0000216
	Det:.6	NP:.6*.6*.15 =.054	None	NP:.6*.6* .0024 =.000864
		Nominal:.15	None	Nominal: .5*.15*.032 =.0024
			Prep:.2	PP:1.0*.2*.16 =.032
				NP:.16

Probabilistic CKY Parser



Pick most probable parse, i.e. take max to combine probabilities of multiple derivations of each constituent in each cell.

Probabilistic CKY Parser

Book	the	flight	through	Houston
S :.01, Verb:.5 Nominal:.03	None	S:.05*.5*.054 =.00135 VP:.5*.5*.054 =.0135	None	S:.00001296 S:.0000216
	Det:.6	NP:.6*.6*.15 =.054	None	NP:.6*.6* .0024 =.000864
		Nominal:.15	None	Nominal: .5*.15*.032 =.0024
			Prep:.2	PP:1.0*.2*.16 =.032
				NP:.16

Parse Tree #2

Pick most probable parse, i.e. take max to combine probabilities of multiple derivations of each constituent in each cell.

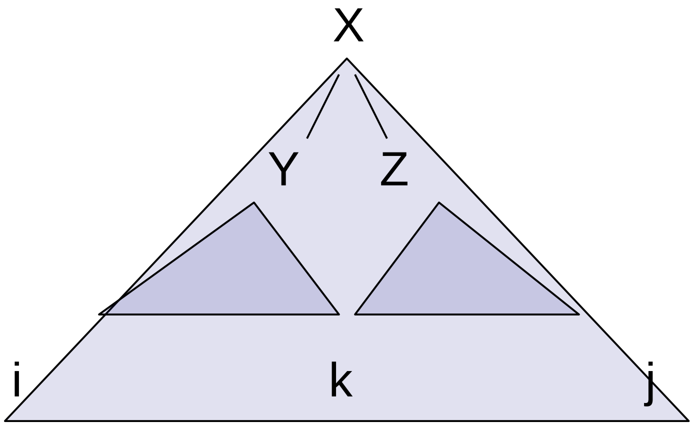
Memory

- How much memory does this require?
  - Have to store the score cache
  - Cache size: |symbols|*n^2 doubles
  - For the plain treebank grammar:
    - X ~ 20K, n = 40, double ~ 8 bytes = ~ 256MB
    - Big, but workable.
- Pruning: Beams
  - score[X][i][j] can get too large (when?)
  - Can keep beams (truncated maps score[i][j]) which only store the best few scores for the span [i,j]
- Pruning: Coarse-to-Fine
  - Use a smaller grammar to rule out most X[i,j]
  - Much more on this later…

Time: Theory

- How much time will it take to parse?

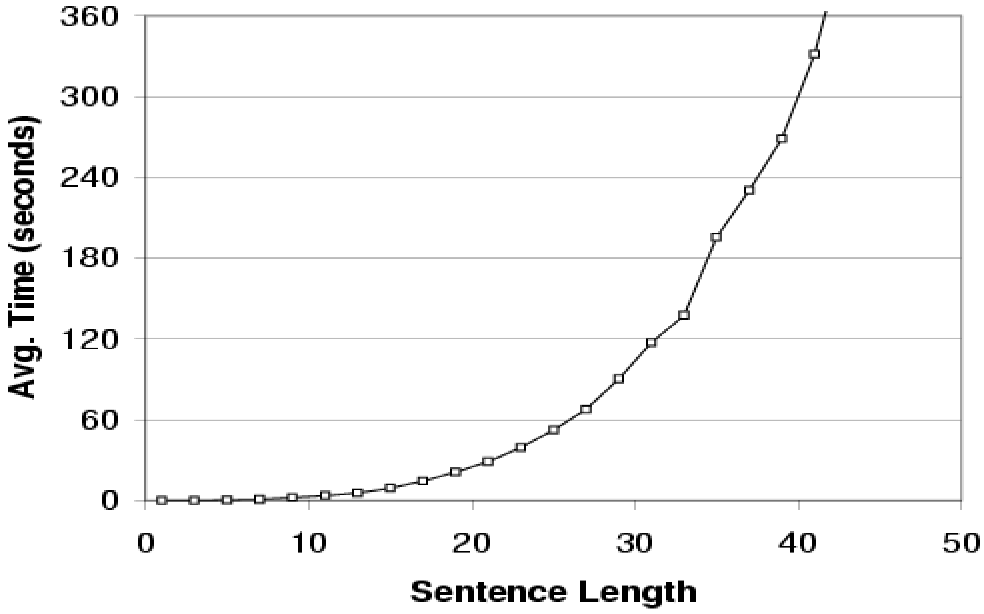
  - For each diff (<= n)
    - For each i (<= n)
      - For each rule X → Y Z
        - For each split point k
          Do constant work



  - Total time: |rules|*n^3
  - Something like 5 sec for an unoptimized parse of a 20-word sentences

Time: Practice

- Parsing with the vanilla treebank grammar:



~ 20K Rules

(not an optimized parser!)

Observed exponent:

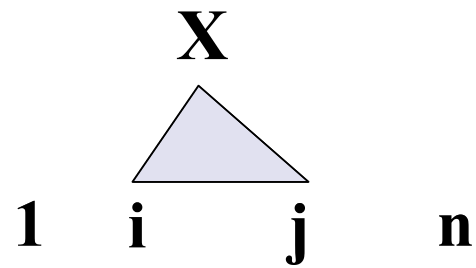
3.6

- Why's it worse in practice?
  - Longer sentences "unlock" more of the grammar
  - All kinds of systems issues don't scale

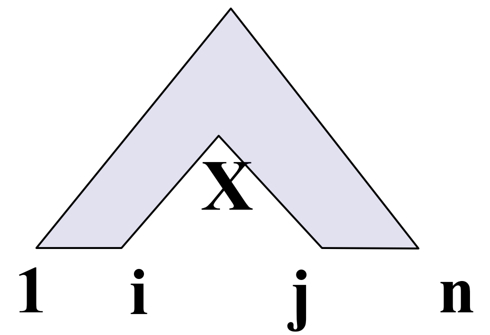
Other Dynamic Programs

Can also compute other quantities:

- *Best Inside:* score of the max parse of w_i to w_j with root non-terminal X

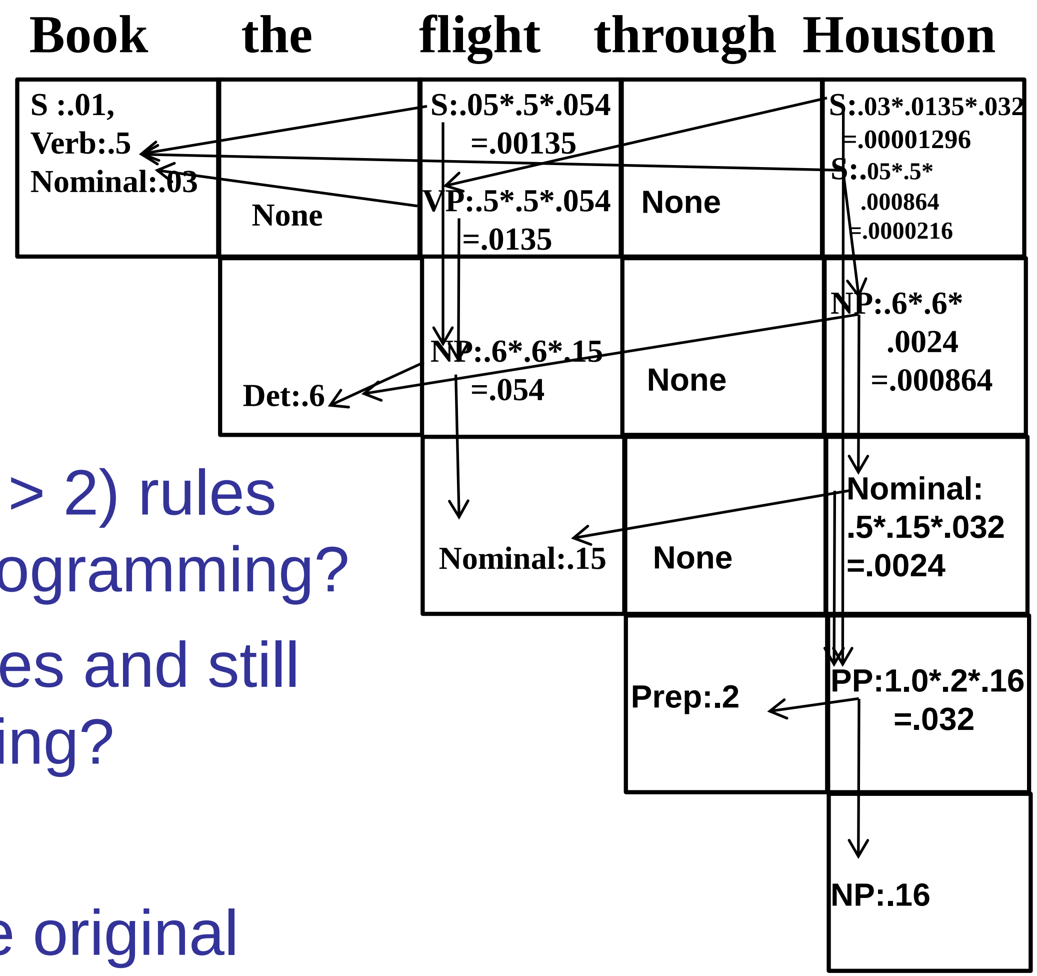


- *Best Outside:* score of the max parse of w_0 to w_n with a gap from w_i to w_j rooted with non-terminal X
  - see notes for derivation, it is a bit more complicated



- Sum Inside/Outside: Do sums instead of maxes

Why Chomsky Normal Form?



Inference:

- Can we keep N-ary (N > 2) rules and still do dynamic programming?
- Can we keep unary rules and still do dynamic programming?

Learning:

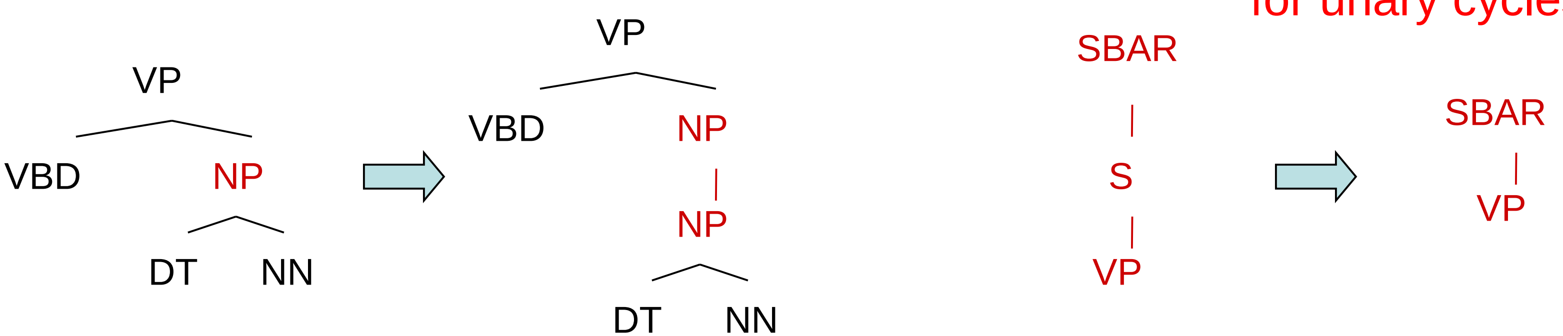
- Can we reconstruct the original trees?

CNF + Unary Closure

We need unaries to be non-cyclic

- Calculate *closure Close(R) for unary rules in R*
  - Add $X \rightarrow Y$ if there exists a rule chain $X \rightarrow Z_1$, $Z_1 \rightarrow Z_2, \ldots, Z_k \rightarrow Y$ with $q(X \rightarrow Y) = q(X \rightarrow Z_1) * q(Z_1 \rightarrow Z_2) * \ldots * q(Z_k \rightarrow Y)$
  - Add $X \rightarrow X$ with $q(X \rightarrow X)=1$ for all X in N

WARNING: Watch out for unary cycles!



- Rather than zero or more unaries, always exactly one
- Alternate unary and binary layers
- What about $X \rightarrow Y$ with different unary paths (and scores)?

The CKY Algorithm

- Input: a sentence s = x_1 .. x_n and a PCFG = <N, Σ ,S, R, q>
- Initialization: For i = 1 … n and all X in N

$$\pi(i,i,X) = \begin{cases} q(X \rightarrow x_i) & \text{if } X \rightarrow x_i \in R \\ 0 & \text{otherwise} \end{cases}$$

- For l = 1 … (n-1) [iterate all phrase lengths]
  - For i = 1 … (n-l) and j = i+l [iterate all phrases of length l]
    - For all X in N [iterate all non-terminals]

$$\pi(i,j,X) = \max_{\substack{X \rightarrow Y Z \in R, \\ s \in \{i \dots (j-1)\}}} \left(q(X \rightarrow Y Z) \times \pi(i,s,Y) \times \pi(s+1,j,Z) \right)$$

    - also, store back pointers

$$bp(i,j,X) = \arg \max_{\substack{X \rightarrow Y Z \in R, \\ s \in \{i \dots (j-1)\}}} \left(q(X \rightarrow Y Z) \times \pi(i,s,Y) \times \pi(s+1,j,Z) \right)$$

CKY with Unary Closure

- Input: a sentence s = x_1 .. x_n and a PCFG = <*N*, Σ ,*S*, *R*, *q*>
- Initialization: For i = 1 … n:
  - Step 1: for all X in N:

$$\pi(i,i,X) = \begin{cases} q(X \to x_i) & \text{if } X \to x_i \in R \\ 0 & \text{otherwise} \end{cases}$$

  - Step 2: for all X in N:

$$\pi_U(i,i,X) = \max_{X \to Y \in Close(R)} (q(X \to Y) \times \pi(i,i,Y))$$

- For l = 1 … (n-1) [iterate all phrase lengths]
  - For i = 1 … (n-l) and j = i+l [iterate all phrases of length l]
    - Step 1: (Binary)
      - For all X in N [iterate all non-terminals]

$$\pi_B(i,j,X) = \max_{X \to YZ \in R, s \in \{i \ldots (j-1)\}} (q(X \to YZ) \times \pi_U(i,s,Y) \times \pi_U(s+1,j,Z))$$

    - Step 2: (Unary)
      - For all X in N [iterate all non-terminals]

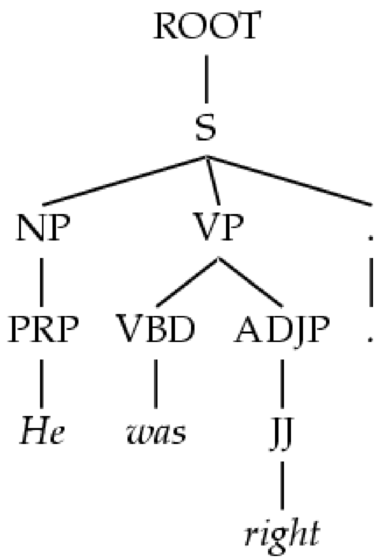
$$\pi_U(i,j,X) = \max_{X \to Y \in Close(R)} (q(X \to Y) \times \pi_B(i,j,Y))$$

Treebank Sentences

```
( (S (NP-SBJ The move)
     (VP followed
         (NP (NP a round)
             (PP of
                 (NP (NP similar increases)
                     (PP by
                         (NP other lenders))
                     (PP against
                         (NP Arizona real estate loans)))))
         ,
         (S-ADV (NP-SBJ *)
                (VP reflecting
                    (NP (NP a continuing decline)
                        (PP-LOC in
                                (NP that market))))))
     .))
```

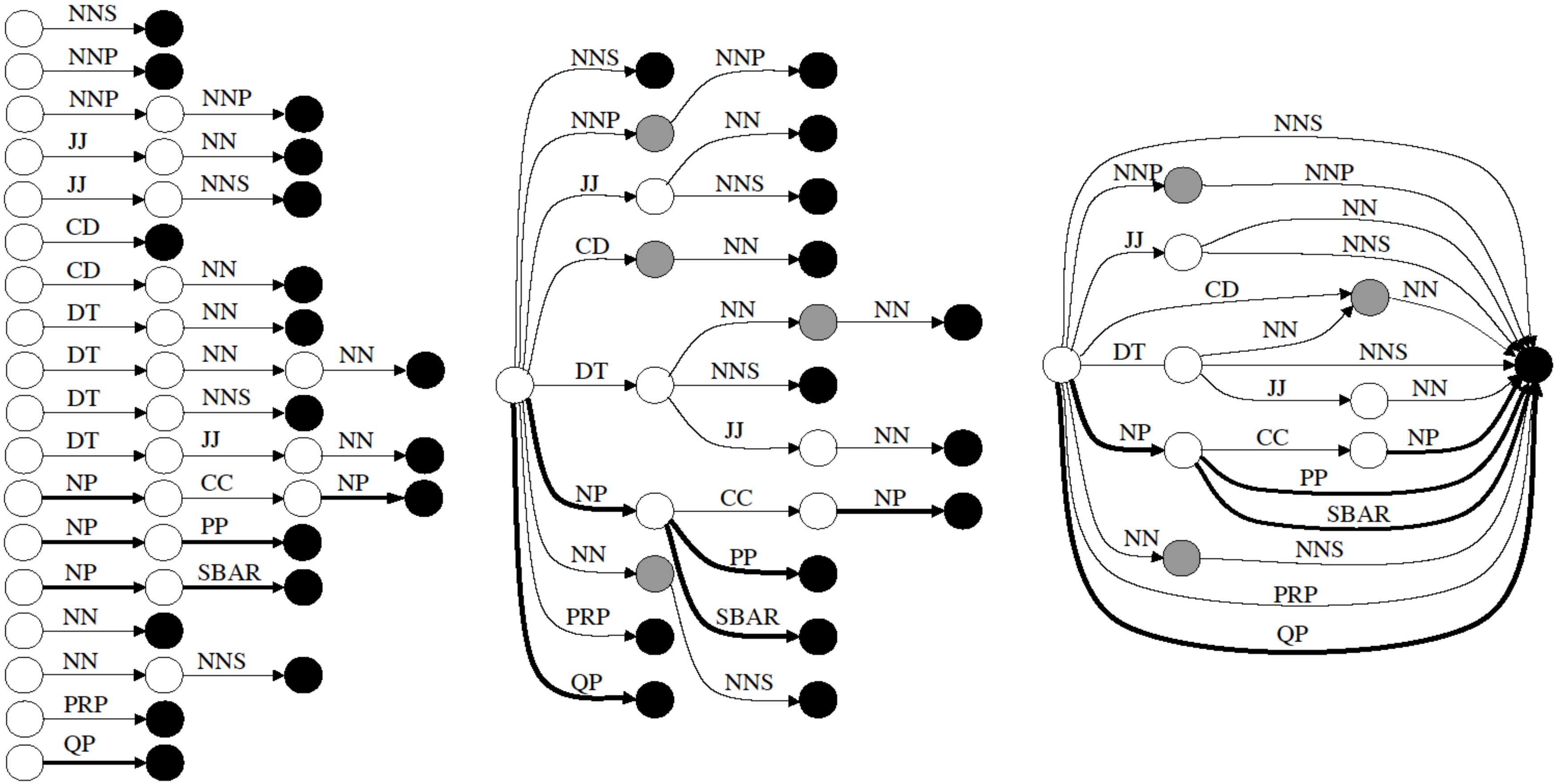
Treebank Grammars

- Need a PCFG for broad coverage parsing.
- Can take a grammar right off the trees (doesn't work well):



ROOT → S	1
S → NP VP .	1
NP → PRP	1
VP → VBD ADJP	1
…..	

- Better results by enriching the grammar (e.g., lexicalization).
- Can also get reasonable parsers without lexicalization.



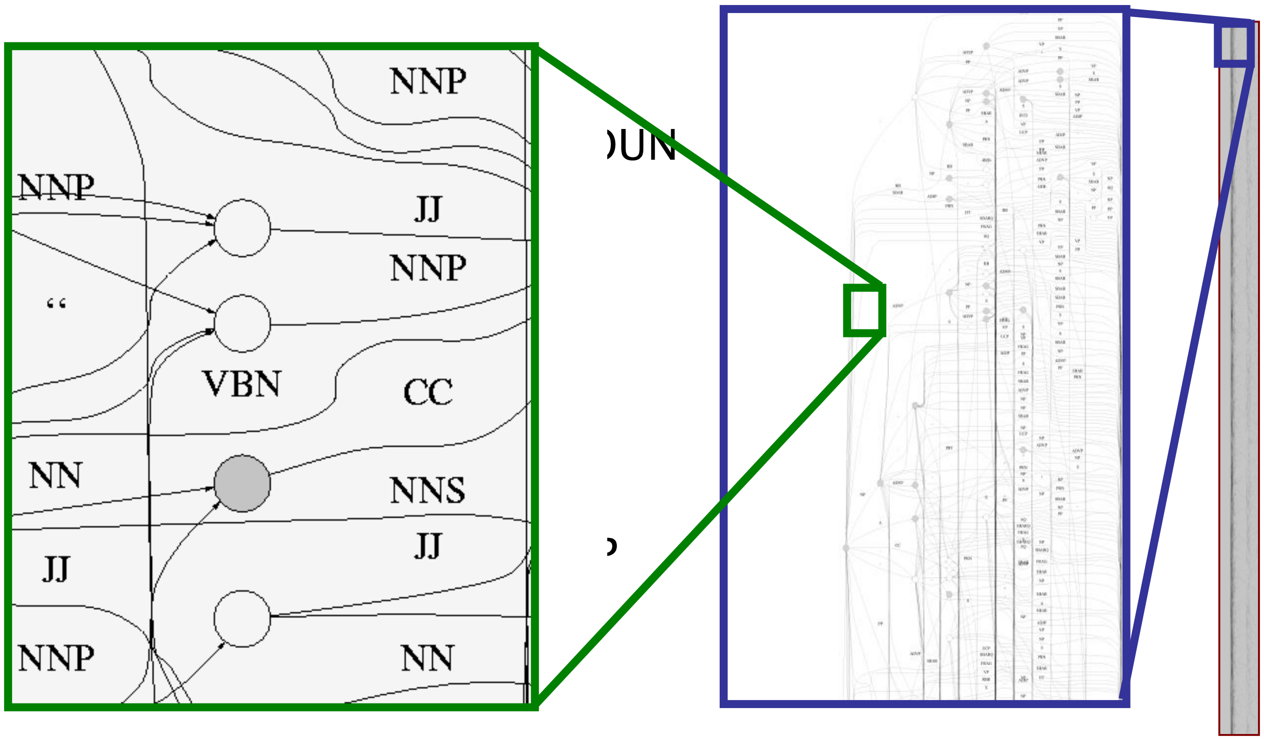
LIST TRIE Min FSA

Grammar encodings: Non-black states are active, non-white states are accepting, and bold transitions are phrasal. FSAs for a subset of the rules for the category NP.

Treebank Grammar Scale

- Treebank grammars can be enormous
  - As FSAs, the raw grammar has ~10K states, excluding the lexicon
  - Better parsers usually make the grammars larger, not smaller

NP:



Typical Experimental Setup

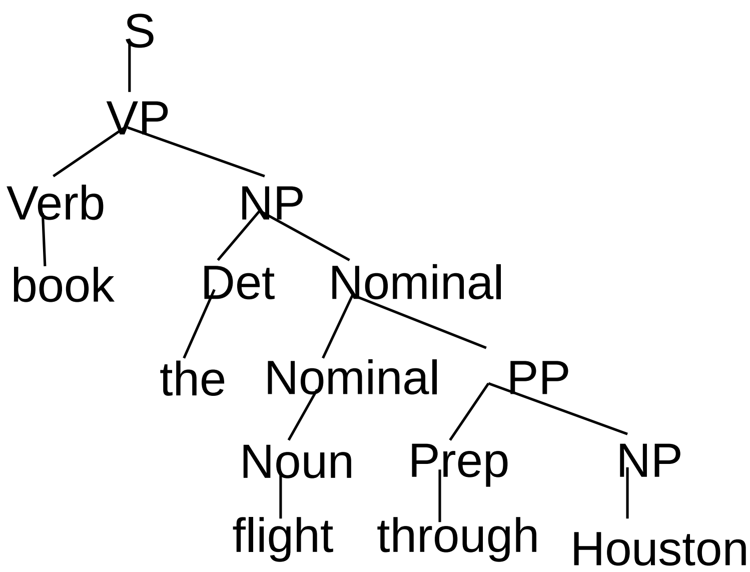
- Corpus: Penn Treebank, WSJ

Training:	sections	02-21
Development:	section	22 (here, first 20 files)
Test:	section	23

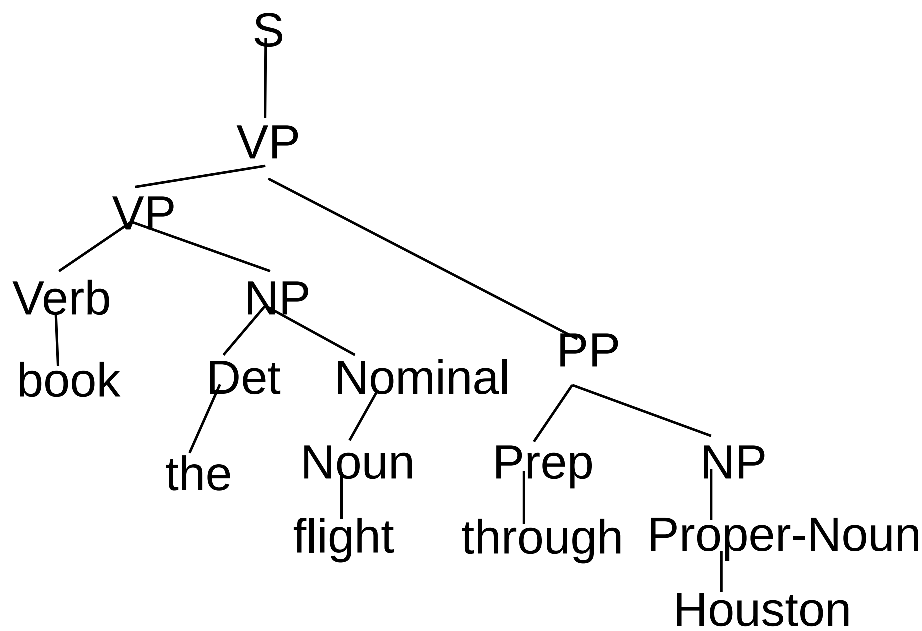
- Accuracy – F1: harmonic mean of per-node labeled precision and recall.
- Here: also size – number of symbols in grammar.
  - Passive / complete symbols: NP, NP^S
  - Active / incomplete symbols: NP $\rightarrow$ NP CC •

How to Evaluate?

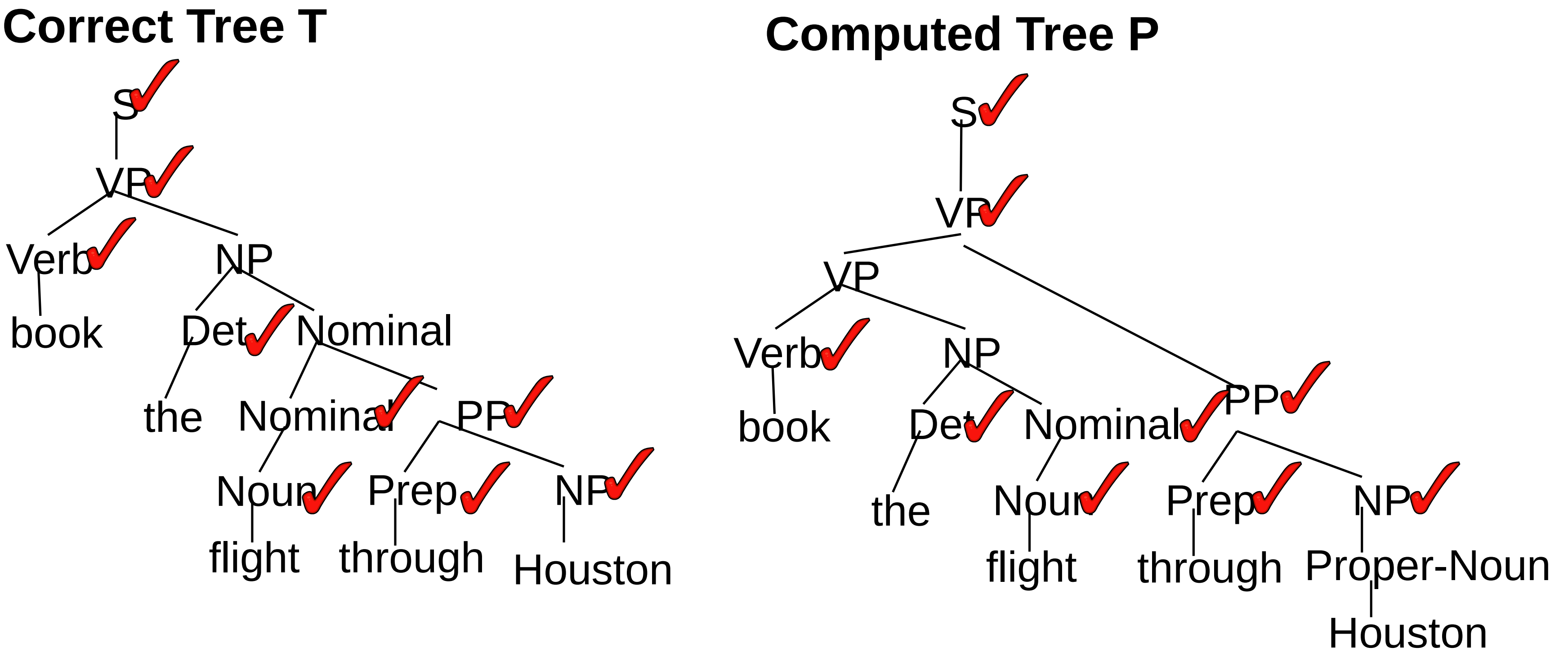
Correct Tree T



Computed Tree P



PARSEVAL Example



Constituents: 11

Constituents: 12

Correct Constituents: 10

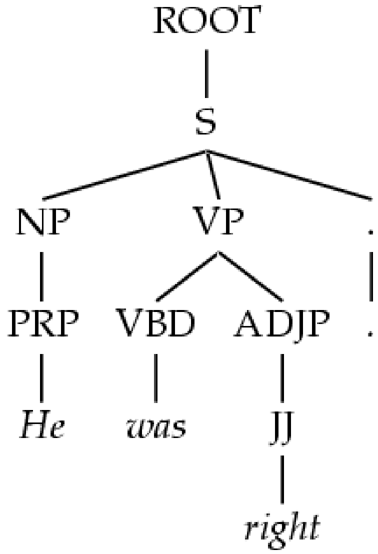
Recall = 10/11= 90.9%   Precision = 10/12=83.3% F_1 = 87.4%

Evaluation Metric

- PARSEVAL metrics measure the fraction of the constituents that match between the computed and human parse trees. If P is the system's parse tree and T is the human parse tree (the "gold standard"):
  - Recall = (# correct constituents in P) / (# constituents in T)
  - Precision = (# correct constituents in P) / (# constituents in P)
- Labeled Precision and labeled recall require getting the non-terminal label on the constituent node correct to count as correct.
- F1 is the harmonic mean of precision and recall.
  - F1= (2 * Precision * Recall) / (Precision + Recall)

Treebank PCFGs [Charniak 96]

- Use PCFGs for broad coverage parsing
- Can take a grammar right off the trees (doesn't work well):



ROOT → S 1

S → NP VP . 1

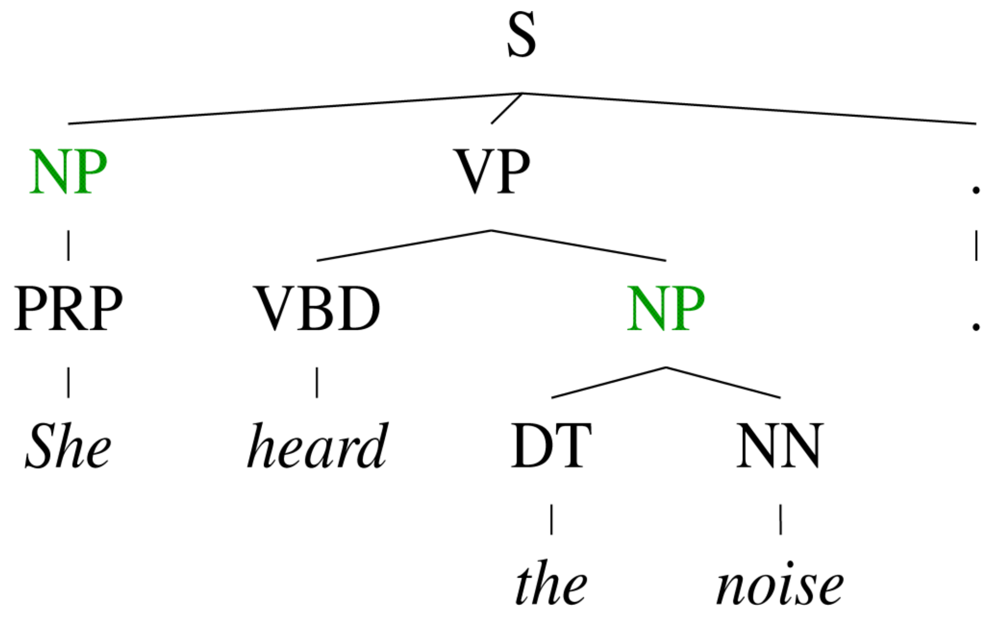
NP → PRP 1

VP → VBD ADJP 1

.....

Model	*F1*
Baseline	72.0

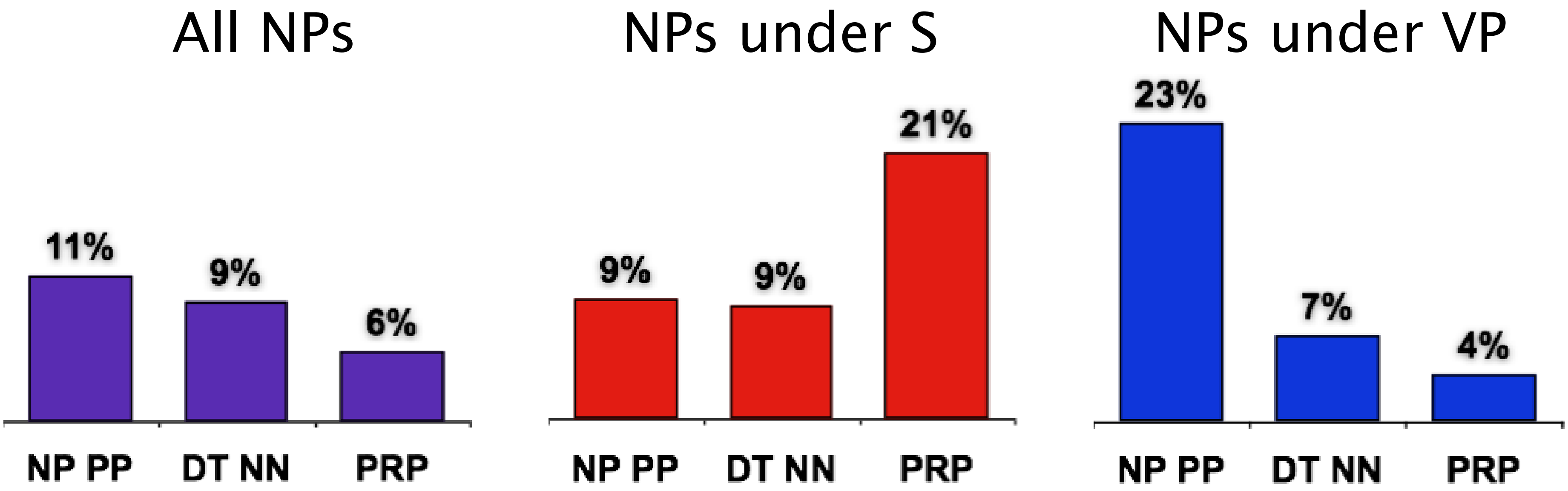
Conditional Independence?



- Not every NP expansion can fill every NP slot
  - A grammar with symbols like “NP” won’t be context-free
  - Statistically, conditional independence too strong

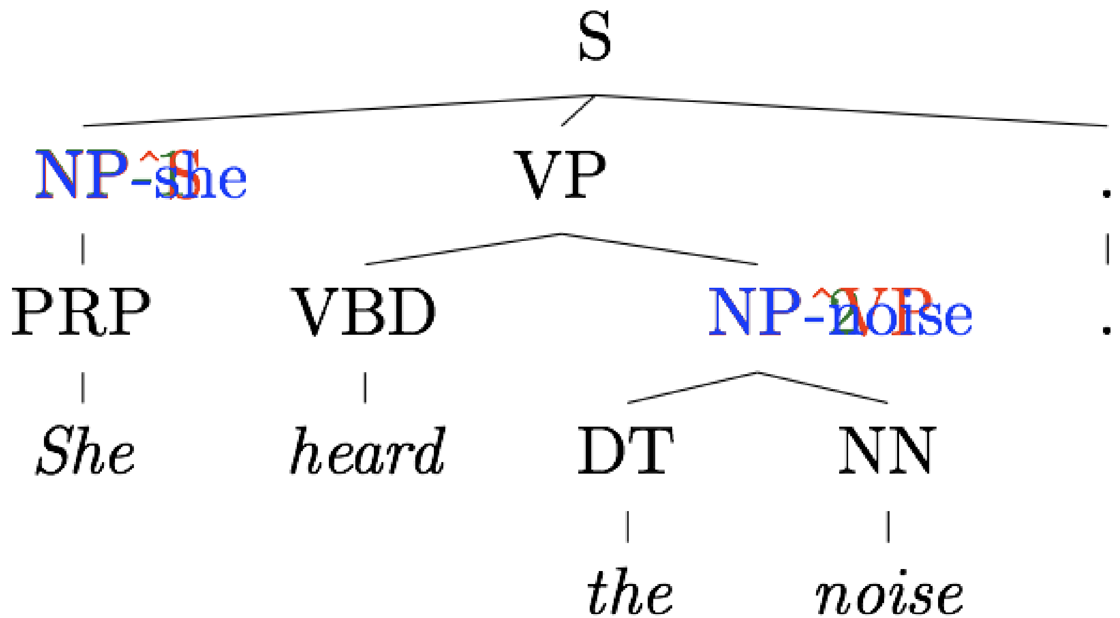
Non-Independence

- Independence assumptions are often too strong.



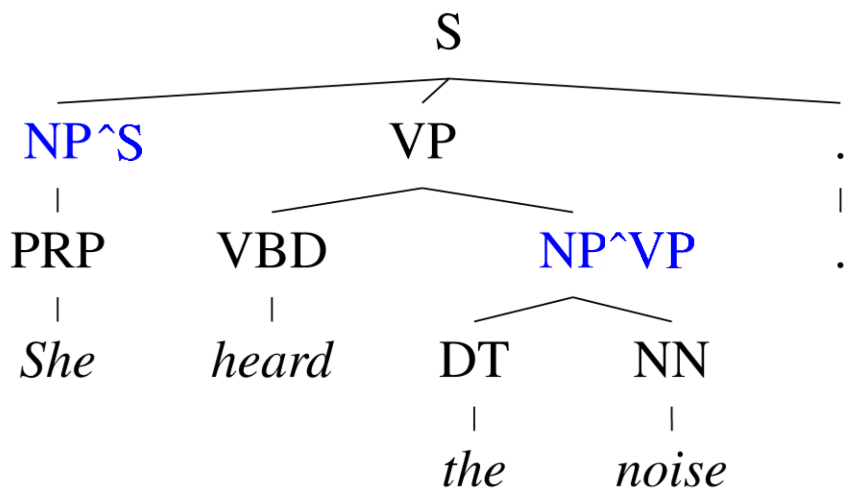
- Example: the expansion of an NP is highly dependent on the parent of the NP (i.e., subjects vs. objects).
- Also: the subject and object expansions are correlated!

Grammar Refinement



- Structure Annotation [Johnson ’98, Klein&Manning ’03]
- Lexicalization [Collins ’99, Charniak ’00]
- Latent Variables [Matsuzaki et al. 05, Petrov et al. ’06]

The Game of Designing a Grammar



- Annotation refines base treebank symbols to improve statistical fit of the grammar
- Structural annotation

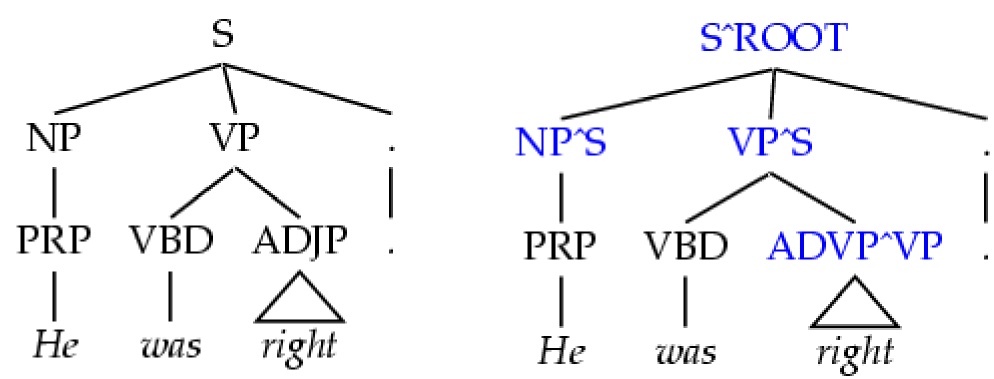
Vertical Markovization

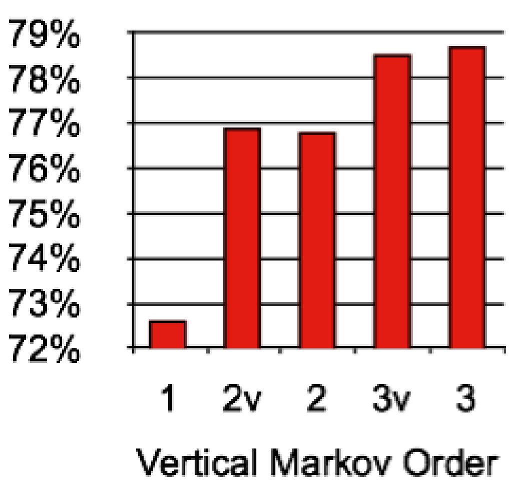
- Vertical Markov order: rewrites depend on past k ancestor nodes.

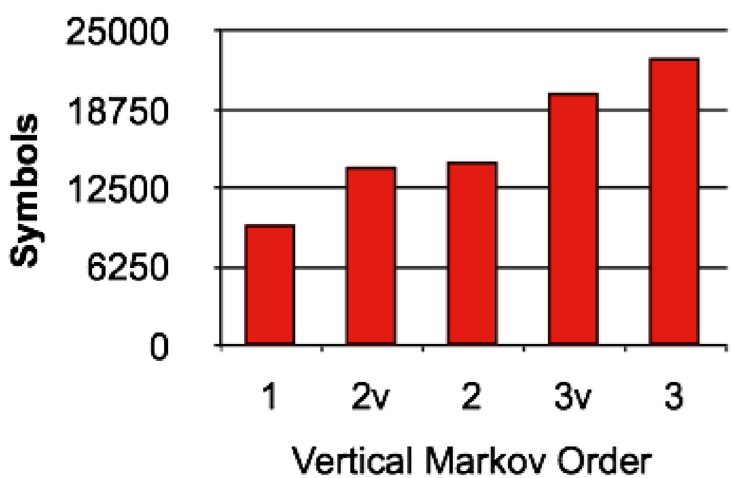
  (cf. parent annotation)

Order 1

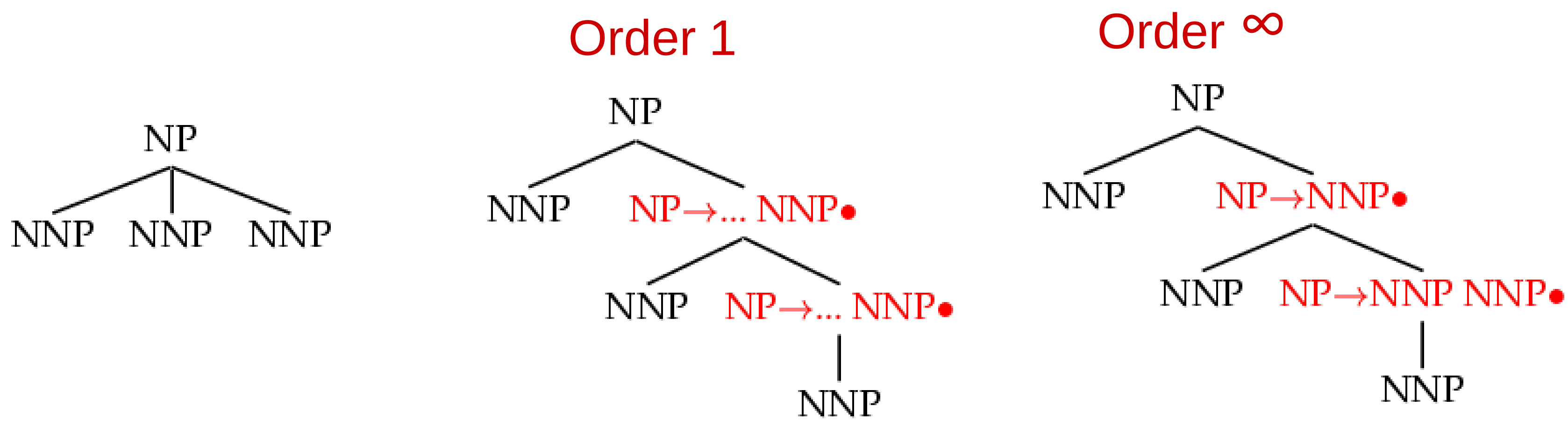
Order 2

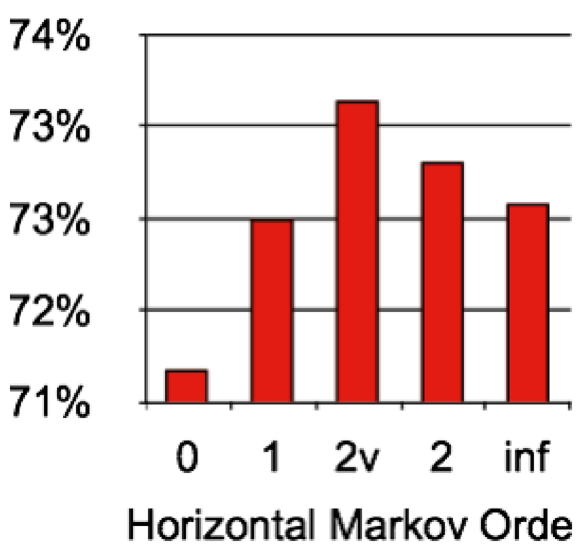


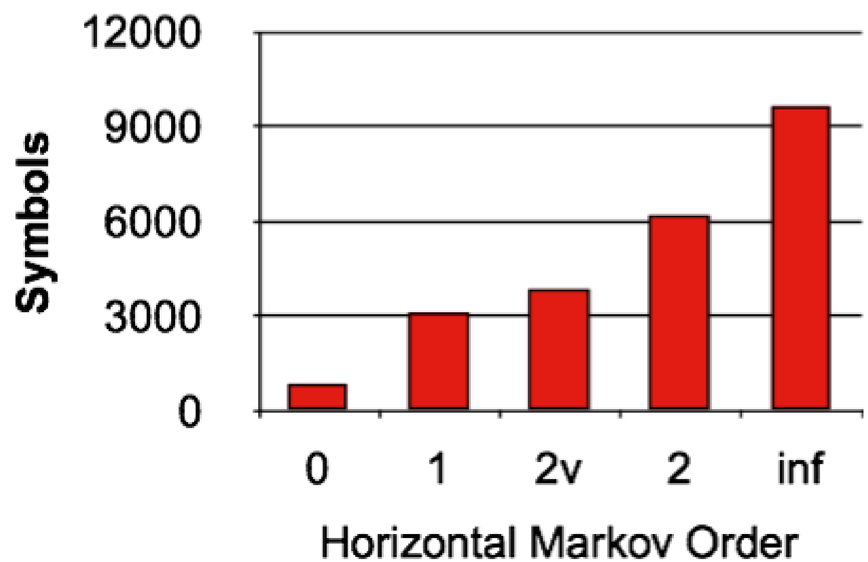




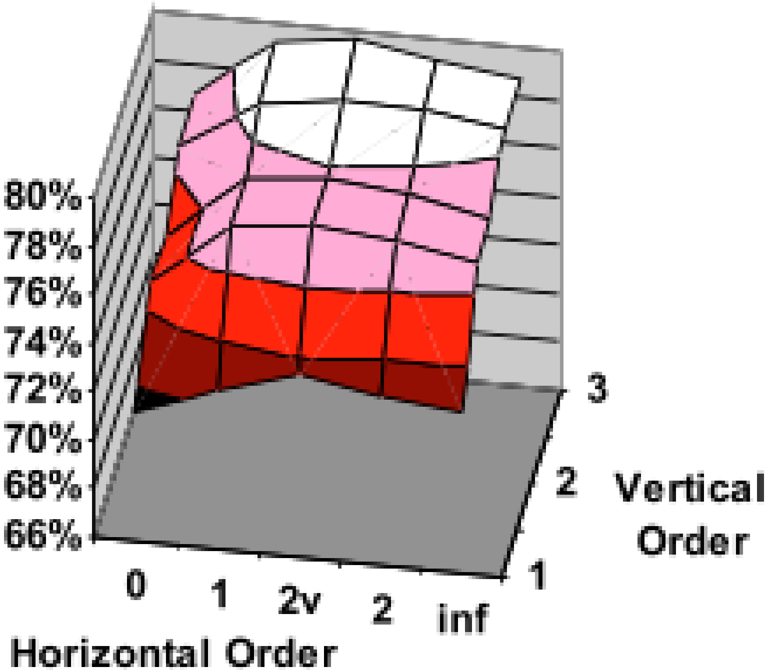
Horizontal Markovization

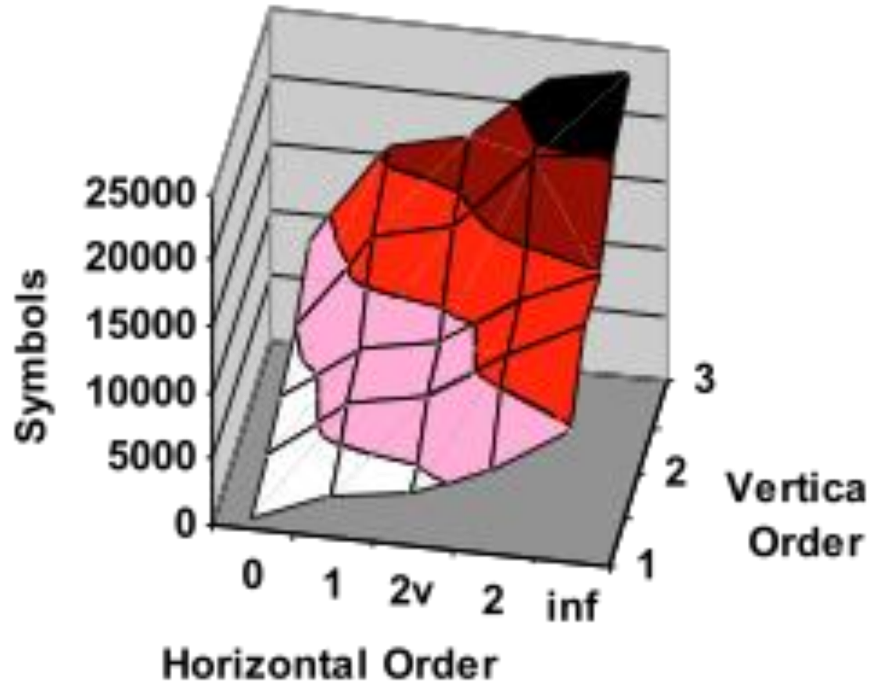






Vertical and Horizontal





- Raw treebank: v=1, h=∞
- Johnson 98: v=2, h=∞
- Collins 99: v=2, h=2
- Best F1: v=3, h=2v

Model	F1	Size
Base: v=h=2v	77.8	7.5K

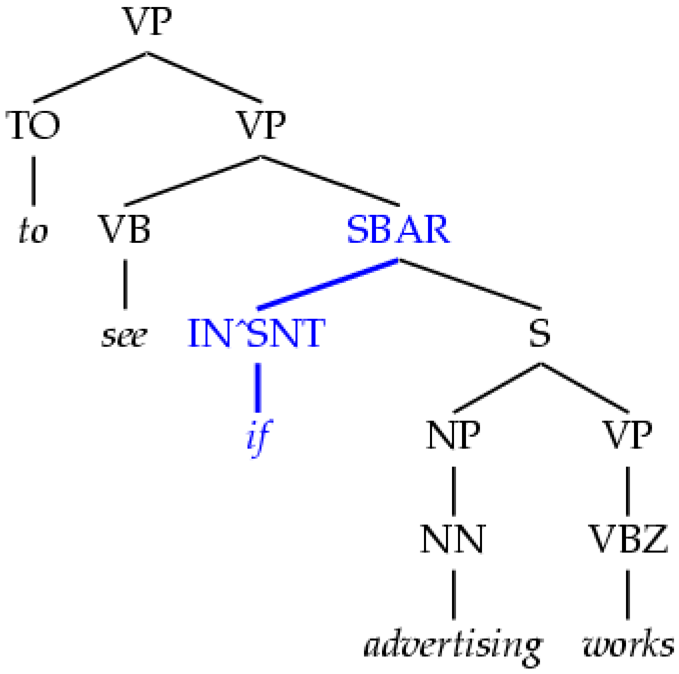
Unlexicalized PCFG Grammar Size

Vertical Order		Horizontal Markov Order $h = 0$	$h = 1$	$h \leq 2$	$h = 2$	$h = \infty$
$v = 1$	No annotation	71.27 (854)	72.5 (3119)	73.46 (3863)	72.96 (6207)	72.62 (9657)
$v \leq 2$	Sel. Parents	74.75 (2285)	77.42 (6564)	77.77 (7619)	77.50 (11398)	76.91 (14247)
$v = 2$	All Parents	74.68 (2984)	77.42 (7312)	77.81 (8367)	77.50 (12132)	76.81 (14666)
$v \leq 3$	Sel. GParents	76.50 (4943)	78.59 (12374)	79.07 (13627)	78.97 (19545)	78.54 (20123)
$v = 3$	All GParents	76.74 (7797)	79.18 (15740)	79.74 (16994)	79.07 (22886)	78.72 (22002)

Figure 2: Markovizations: F_1 and grammar size.

Tag Splits

- Problem: Treebank tags are too coarse.
- Example: Sentential, PP, and other prepositions are all marked IN.
- Partial Solution:
  - Subdivide the IN tag.



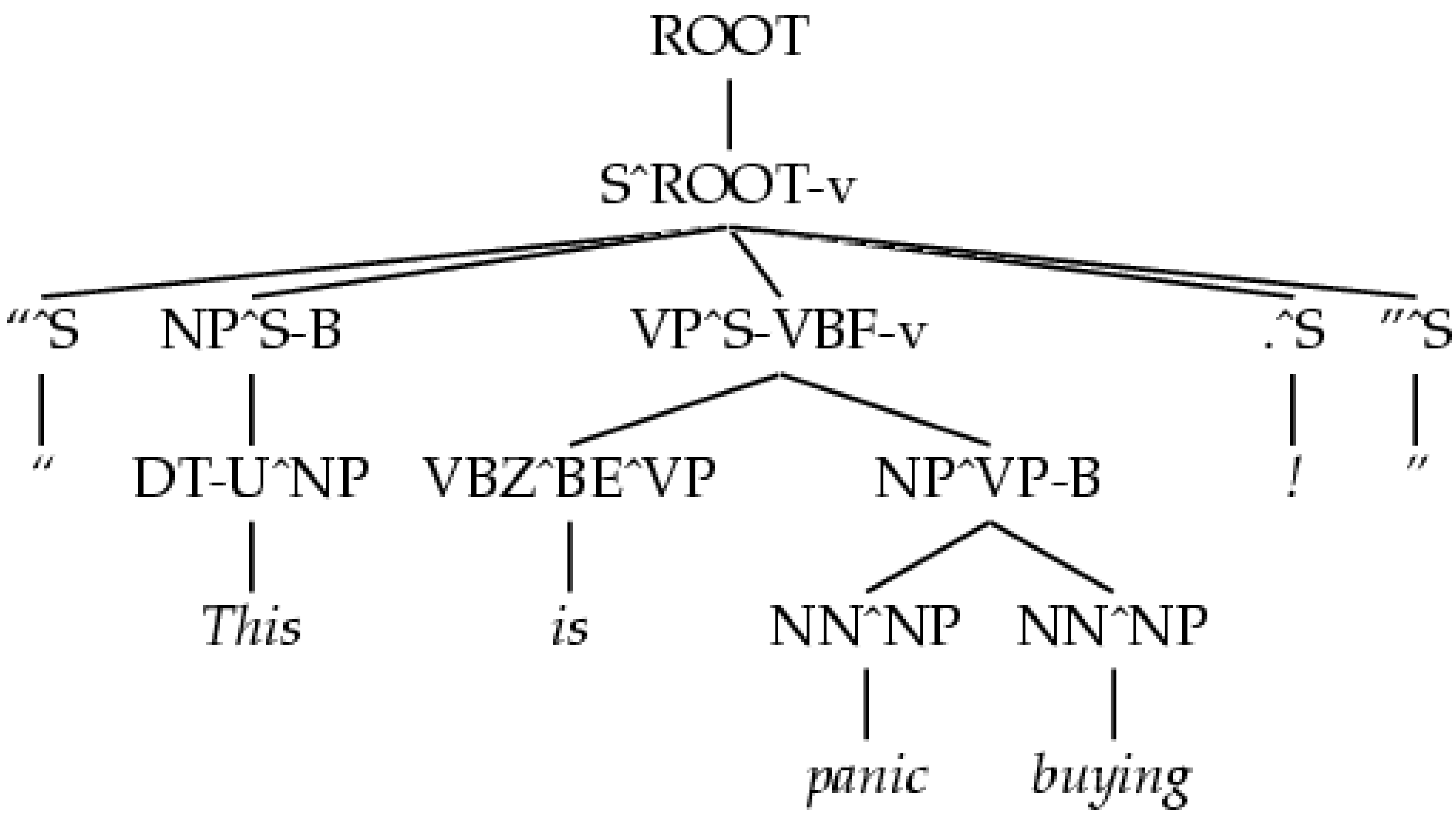
Annotation	F1	Size
Previous	78.3	8.0K
SPLIT-IN	80.3	8.1K

Other Tag Splits

- UNARY-DT: mark demonstratives as DT^U ("the X" vs. "those")
- UNARY-RB: mark phrasal adverbs as RB^U ("quickly" vs. "very")
- TAG-PA: mark tags with non-canonical parents ("not" is an RB^VP)
- SPLIT-AUX: mark auxiliary verbs with –AUX [cf. Charniak 97]
- SPLIT-CC: separate "but" and "&" from other conjunctions
- SPLIT-%: "%" gets its own tag.

F1	Size
80.4	8.1K
80.5	8.1K
81.2	8.5K
81.6	9.0K
81.7	9.1K
81.8	9.3K

A Fully Annotated (Unlex) Tree

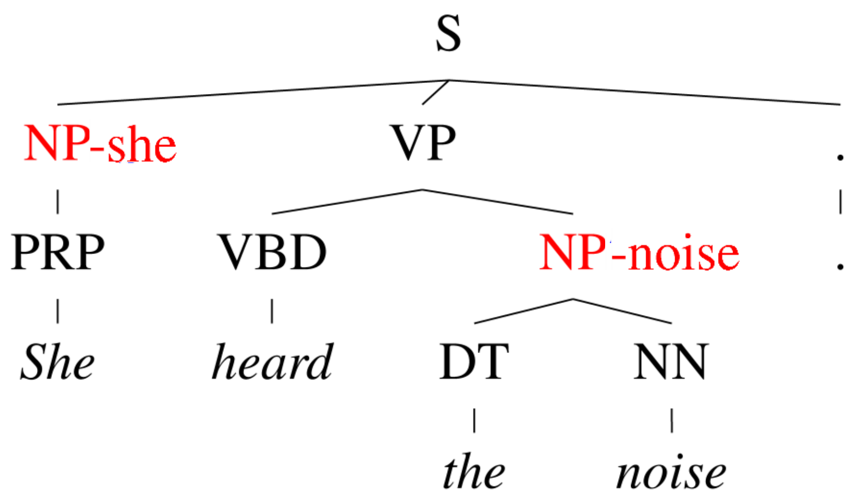


Some Test Set Results

Parser	LP	LR	**F1**
Magerman 95	84.9	84.6	**84.7**
Collins 96	86.3	85.8	**86.0**
Unlexicalized	86.9	85.7	**86.3**
Charniak 97	87.4	87.5	**87.4**
Collins 99	88.7	88.6	**88.6**

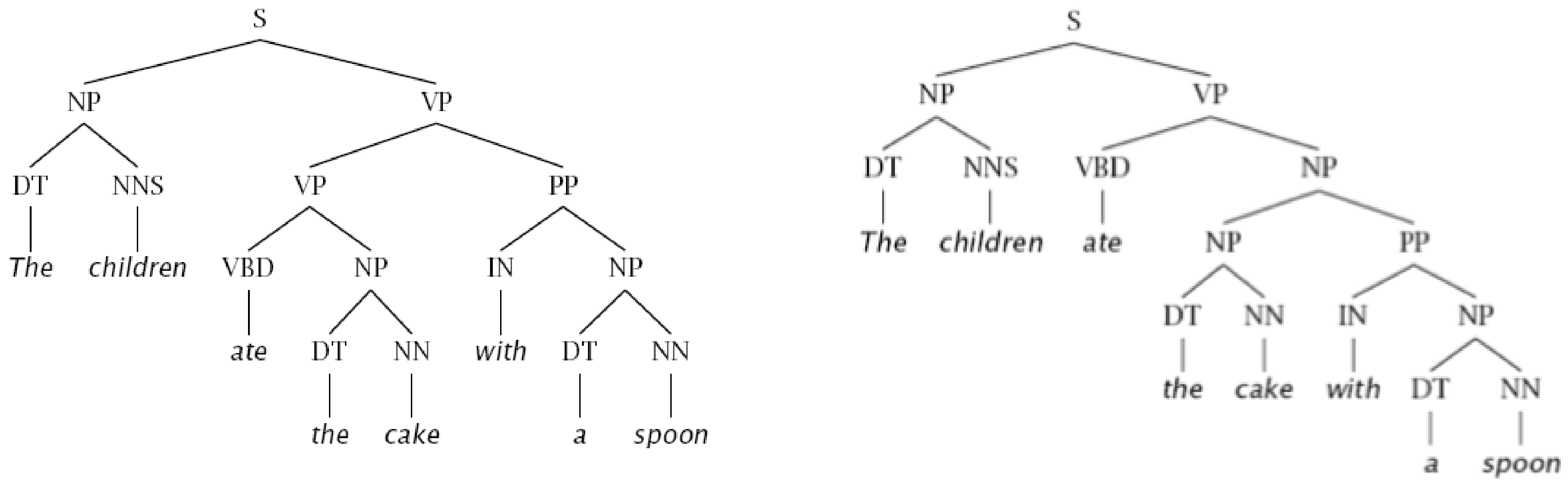
- Beats “first generation” lexicalized parsers.
- Lots of room to improve – more complex models next.

The Game of Designing a Grammar



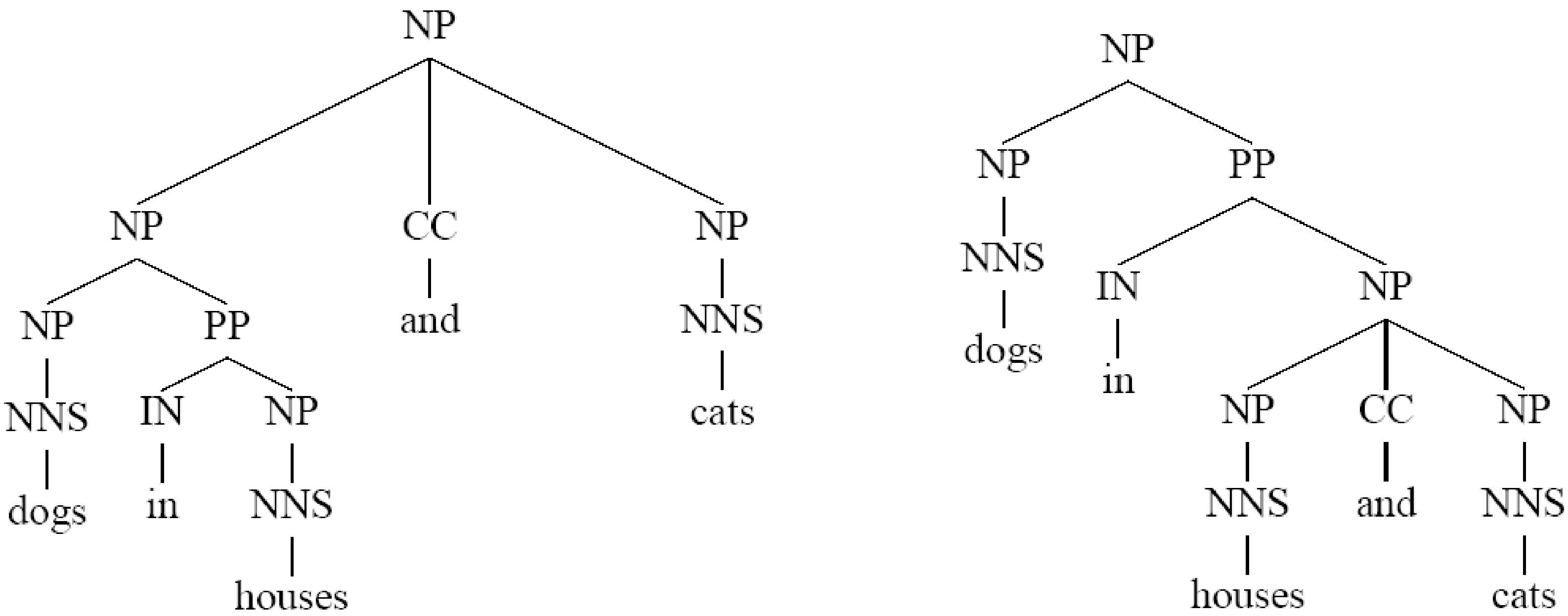
- Annotation refines base treebank symbols to improve statistical fit of the grammar
- Structural annotation [Johnson '98, Klein and Manning 03]
- Head lexicalization [Collins '99, Charniak '00]

Problems with PCFGs



- If we do no annotation, these trees differ only in one rule:
  - VP → VP PP
  - NP → NP PP
- Parse will go one way or the other, regardless of words
- We addressed this in one way with unlexicalized grammars (how?)
- Lexicalization allows us to be sensitive to specific words

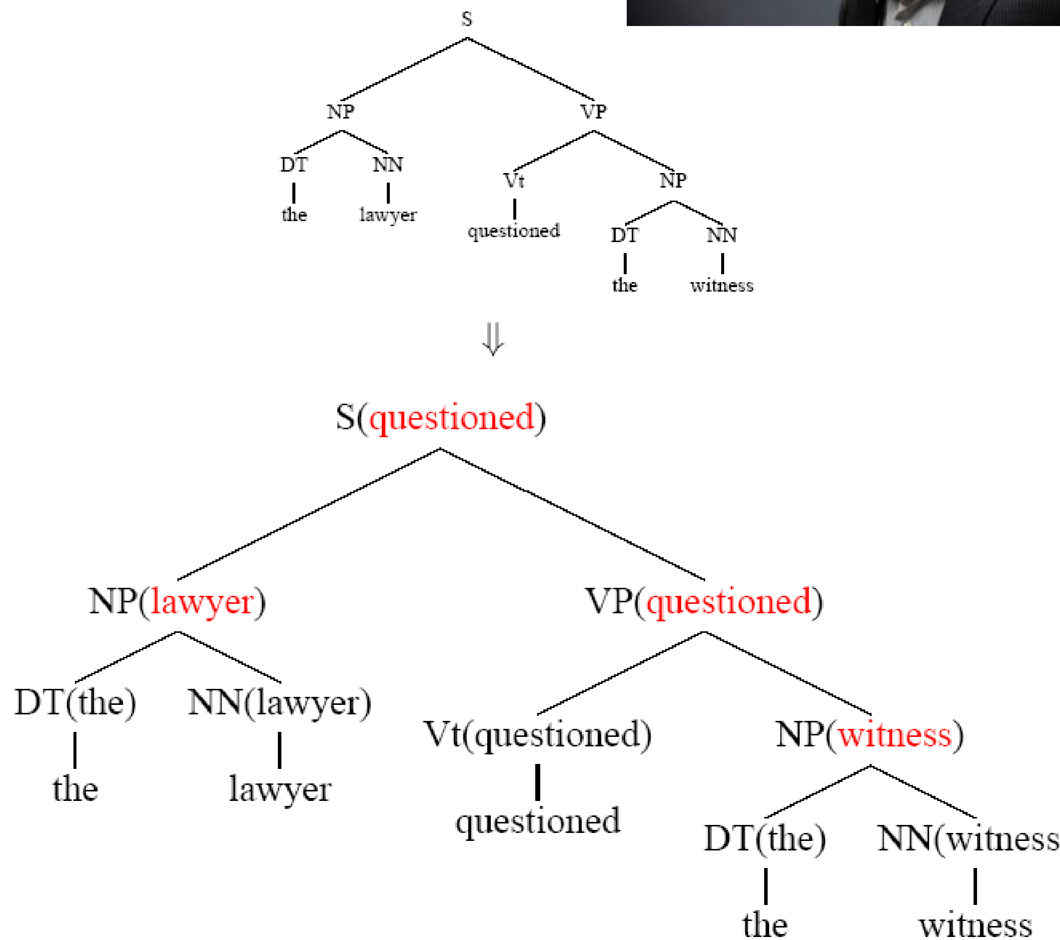
Problems with PCFGs



- What's different between basic PCFG scores here?
- What (lexical) correlations need to be scored?

Lexicalize Trees!

- Add “headwords” to each phrasal node
  - Headship not in (most) treebanks
  - Usually *use head rules*, e.g.:
    - NP:
      - Take leftmost NP
      - Take rightmost N*
      - Take rightmost JJ
      - Take right child
    - VP:
      - Take leftmost VB*
      - Take leftmost VP
      - Take left child



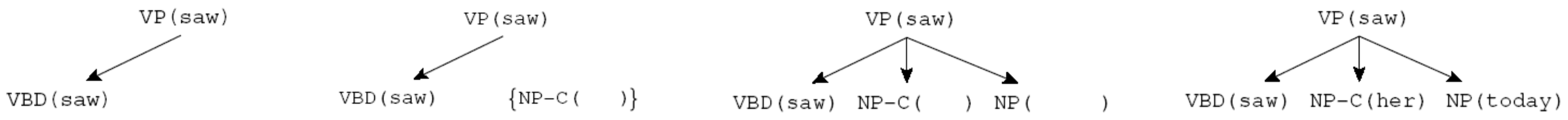
Lexicalized PCFGs?

- Problem: we now have to estimate probabilities like

```
VP(saw) -> VBD(saw) NP-C(her) NP(today)
```

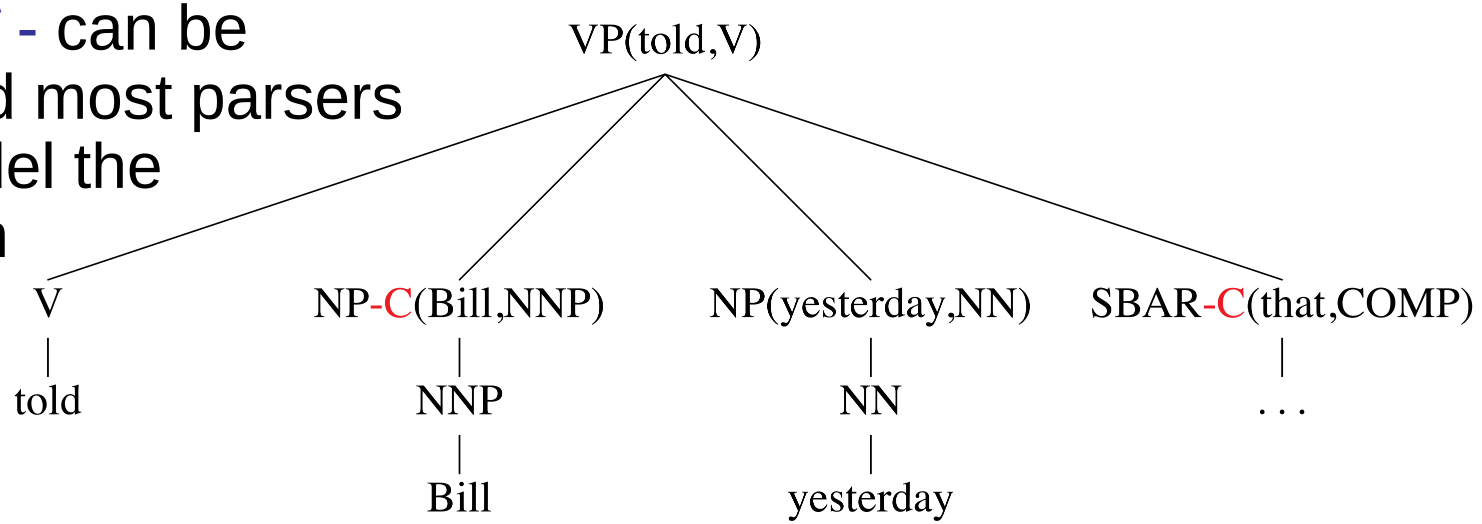
- Never going to get these atomically off of a treebank

- Solution: break up derivation into smaller steps



Complement / Adjunct Distinction

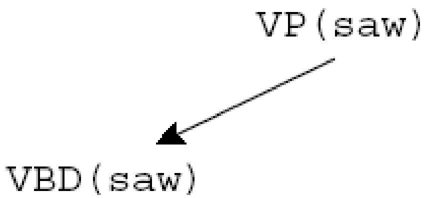
- **warning** - can be tricky, and most parsers don't model the distinction



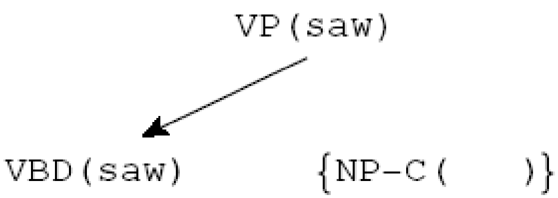
- Complement: defines a property/argument (often obligatory), ex: [capitol [of Rome]]
- Adjunct: modifies / describes something (always optional), ex: [quickly ran]
- A Test for Adjuncts: [X Y] --> can claim X and Y
  - [they ran and it happened quickly] vs. [capitol and it was of Rome]

Lexical Derivation Steps

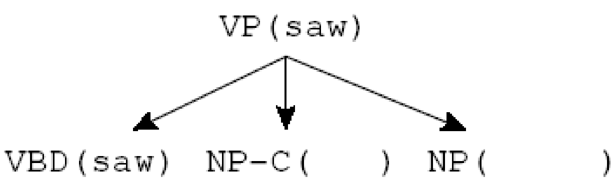
- Main idea: define a linguistically-motivated Markov process for generating children given the parent



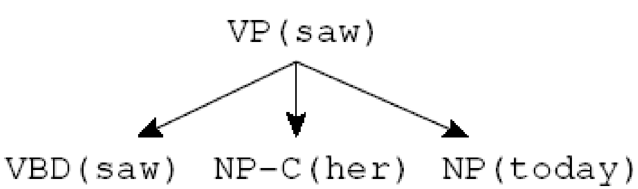
Step 1: Choose a head tag and word



Step 2: Choose a complement bag



Step 3: Generate children (incl. adjuncts)

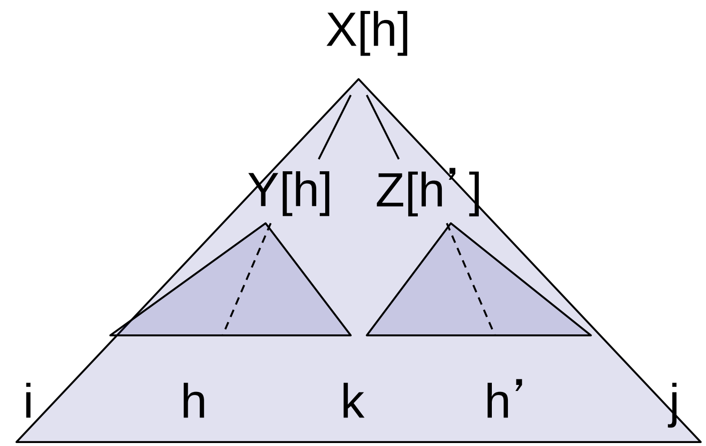


Step 4: Recursively derive children

Lexicalized CKY

(VP->VBD...NP •)[saw]

(VP->VBD •)[saw]　　NP[her]

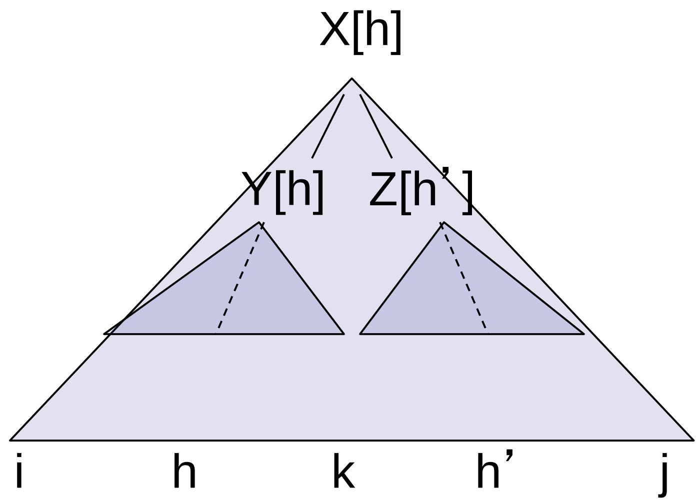



```
bestScore(X,i,j,h)
  if (j = i+1)
    return tagScore(X,s[i])
  else
    return
      max    max    score(X[h]->Y[h] Z[h']) *
      k,h',         bestScore(Y,i,k,h) *
      X->YZ         bestScore(Z,k,j,h')
             max    score(X[h]->Y[h'] Z[h]) *
      k,h',         bestScore(Y,i,k,h') *
      X->YZ         bestScore(Z,k,j,h)
```

still cubic time?

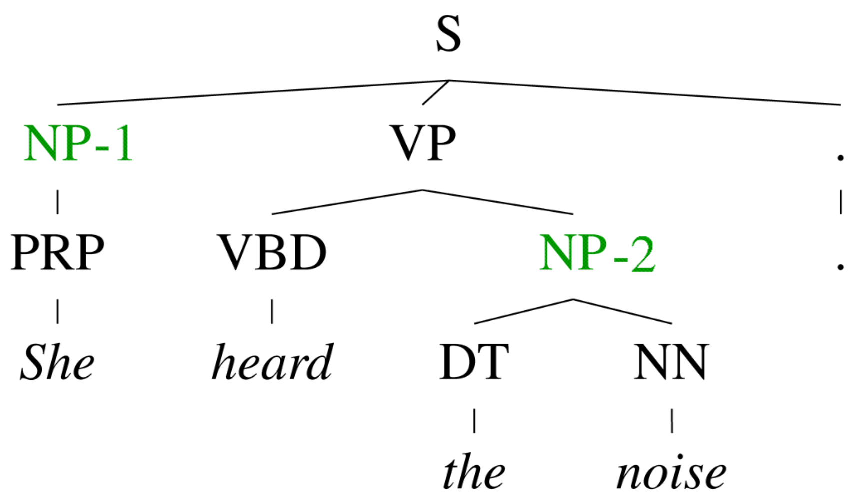
Pruning with Beams

- The Collins parser prunes with per-cell beams [Collins 99]
  - Essentially, run the **$O(n^5)$** CKY
  - Remember only a few hypotheses for each span <i,j>.
  - If we keep K hypotheses at each span, then we do at most $O(nK^2)$ work per span (why?)
  - Keeps things more or less cubic
- Also: certain spans are forbidden entirely on the basis of punctuation (crucial for speed)



Model	*F1*
Naïve Treebank Grammar	72.6
Klein & Manning '03	86.3
Collins 99	88.6

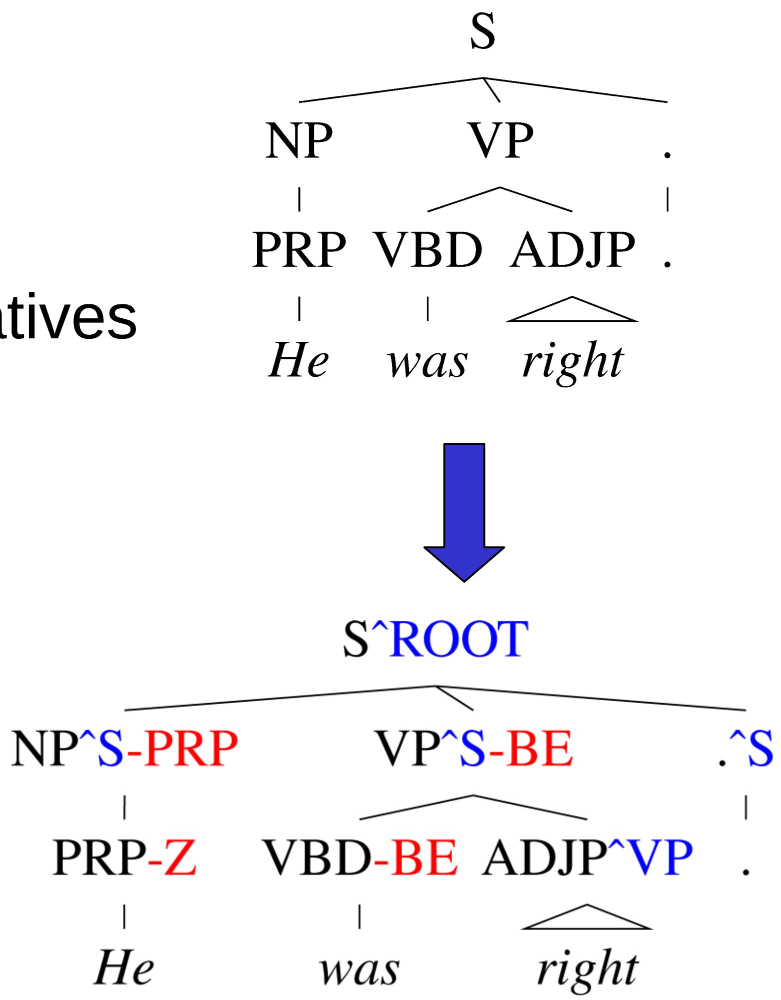
The Game of Designing a Grammar



- Annotation refines base treebank symbols to improve statistical fit of the grammar
- Parent annotation [Johnson ’98]
- Head lexicalization [Collins ’99, Charniak ’00]
- Automatic clustering?

Manual Annotation

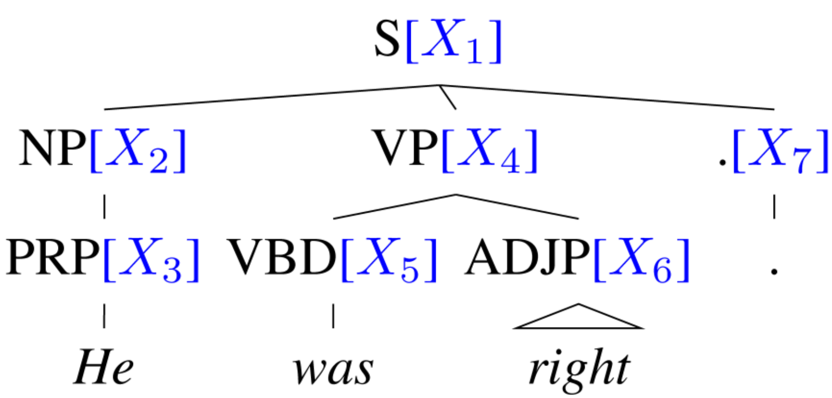
- Manually split categories
  - NP: subject vs object
  - DT: determiners vs demonstratives
  - IN: sentential vs prepositional
- Advantages:
  - Fairly compact grammar
  - Linguistic motivations
- Disadvantages:
  - Performance leveled out
  - Manually annotated

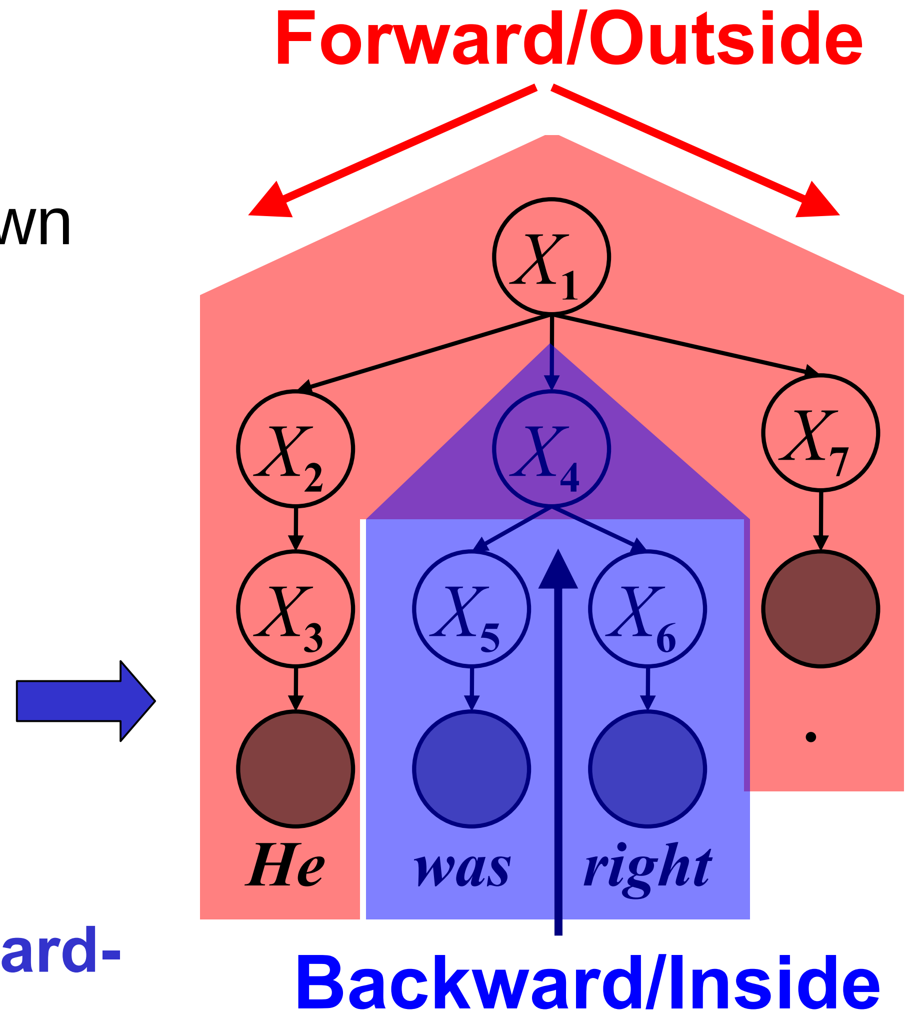


Learning Latent Annotations

Latent Annotations:

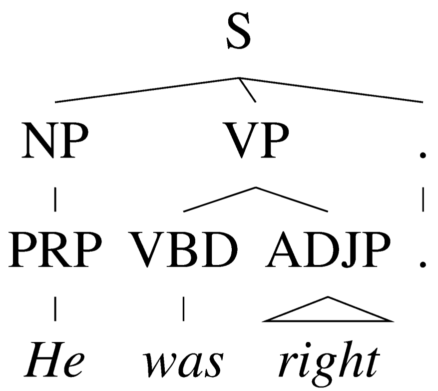
- Brackets are known
- Base categories are known
- Hidden variables for subcategories





Can learn with EM: like Forward-Backward for HMMs.

Automatic Annotation Induction



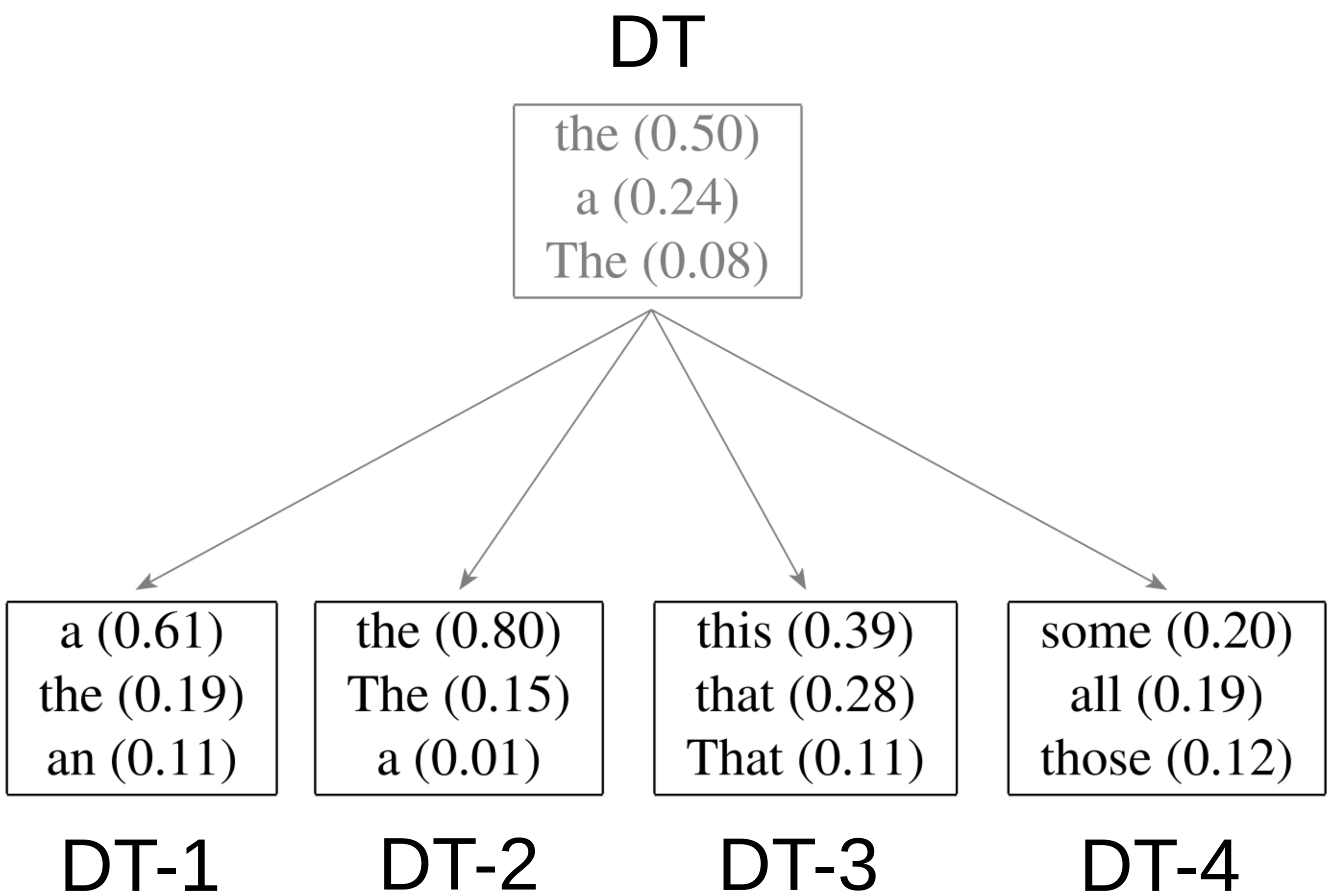
- Advantages:
  - Automatically learned:

    Label *all* nodes with latent variables.

    Same number k of subcategories for all categories.
- Disadvantages:
  - Grammar gets too large
  - Most categories are oversplit while others are undersplit.

Model	*F1*
Klein & Manning ’03	86.3
Matsuzaki et al. ’05	86.7

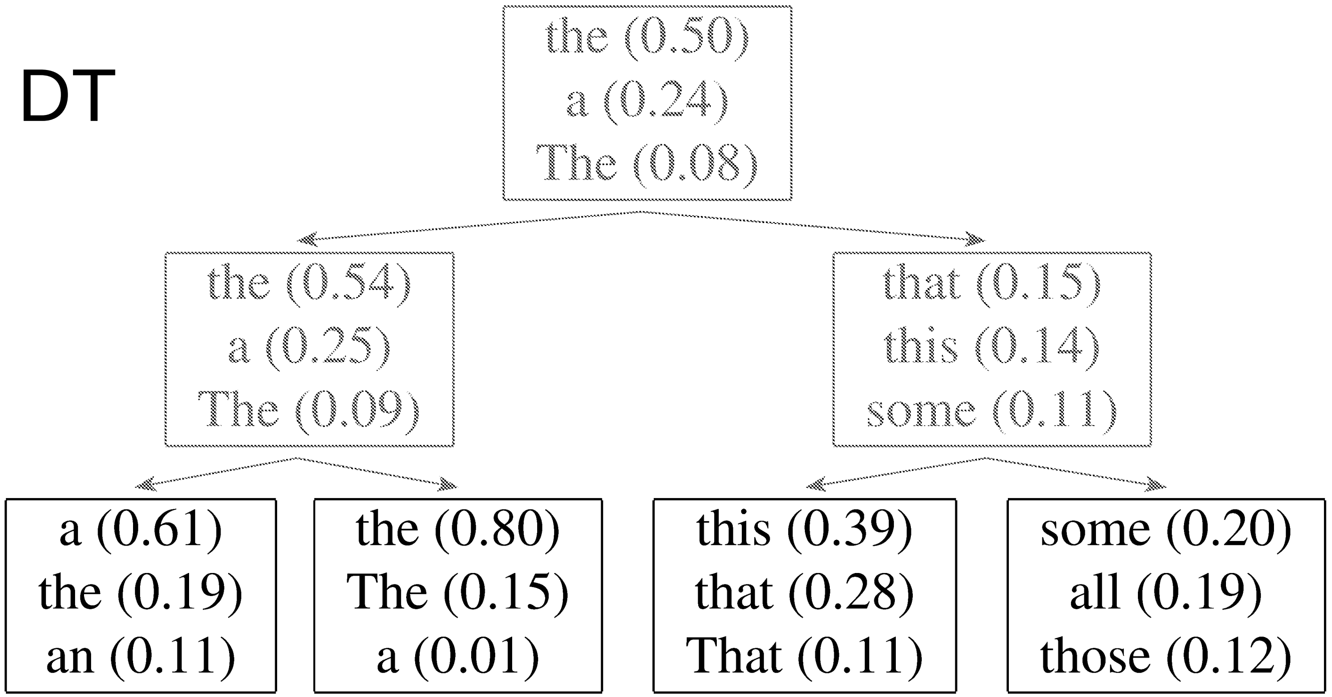
Refinement of the DT tag



Hierarchical refinement

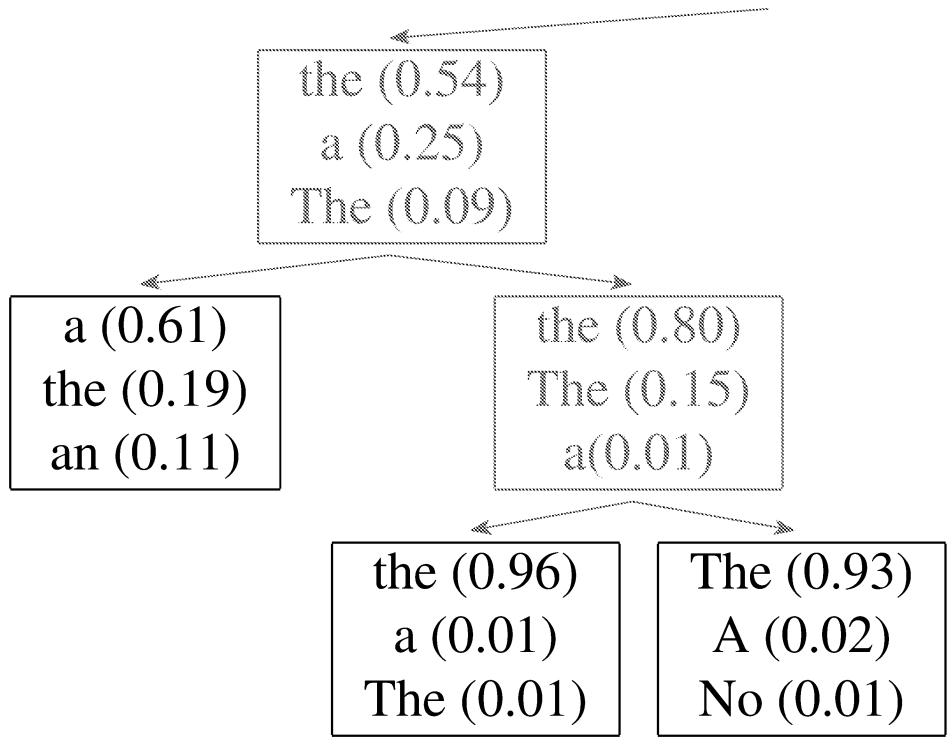
- Repeatedly learn more fine-grained subcategories
- start with two (per non-terminal), then keep splitting
- initialize each EM run with the output of the last

DT



Adaptive Splitting

- Want to split complex categories more
- Idea: split everything, roll back splits which were least useful

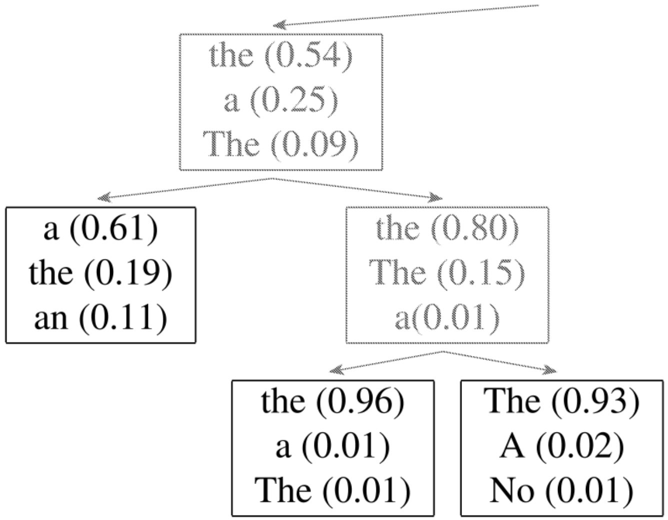


Adaptive Splitting

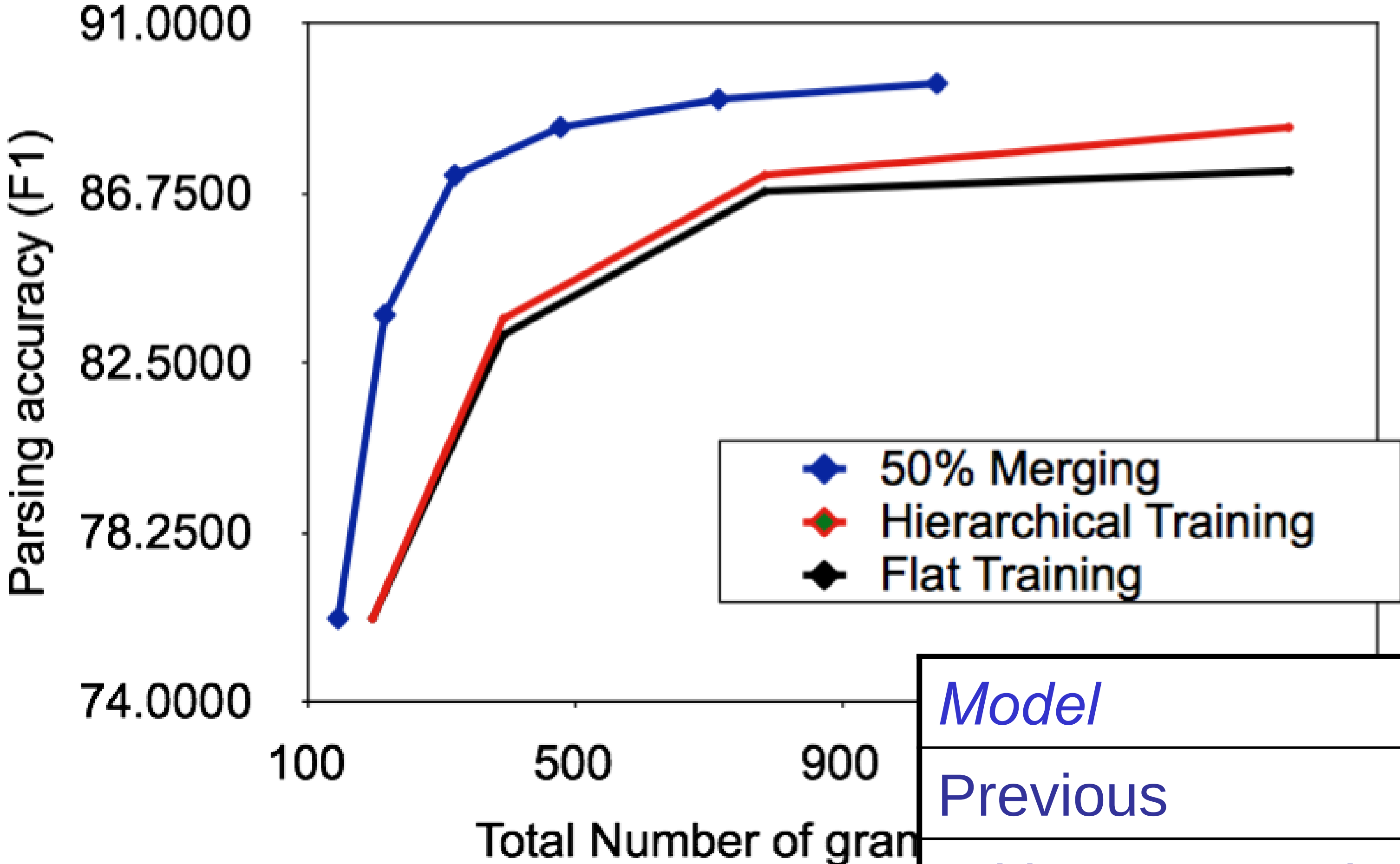
- Evaluate loss in likelihood from removing each split =

$$\frac{\text{Data likelihood with split reversed}}{\text{Data likelihood with split}}$$

- No loss in accuracy when 50% of the splits are reversed.

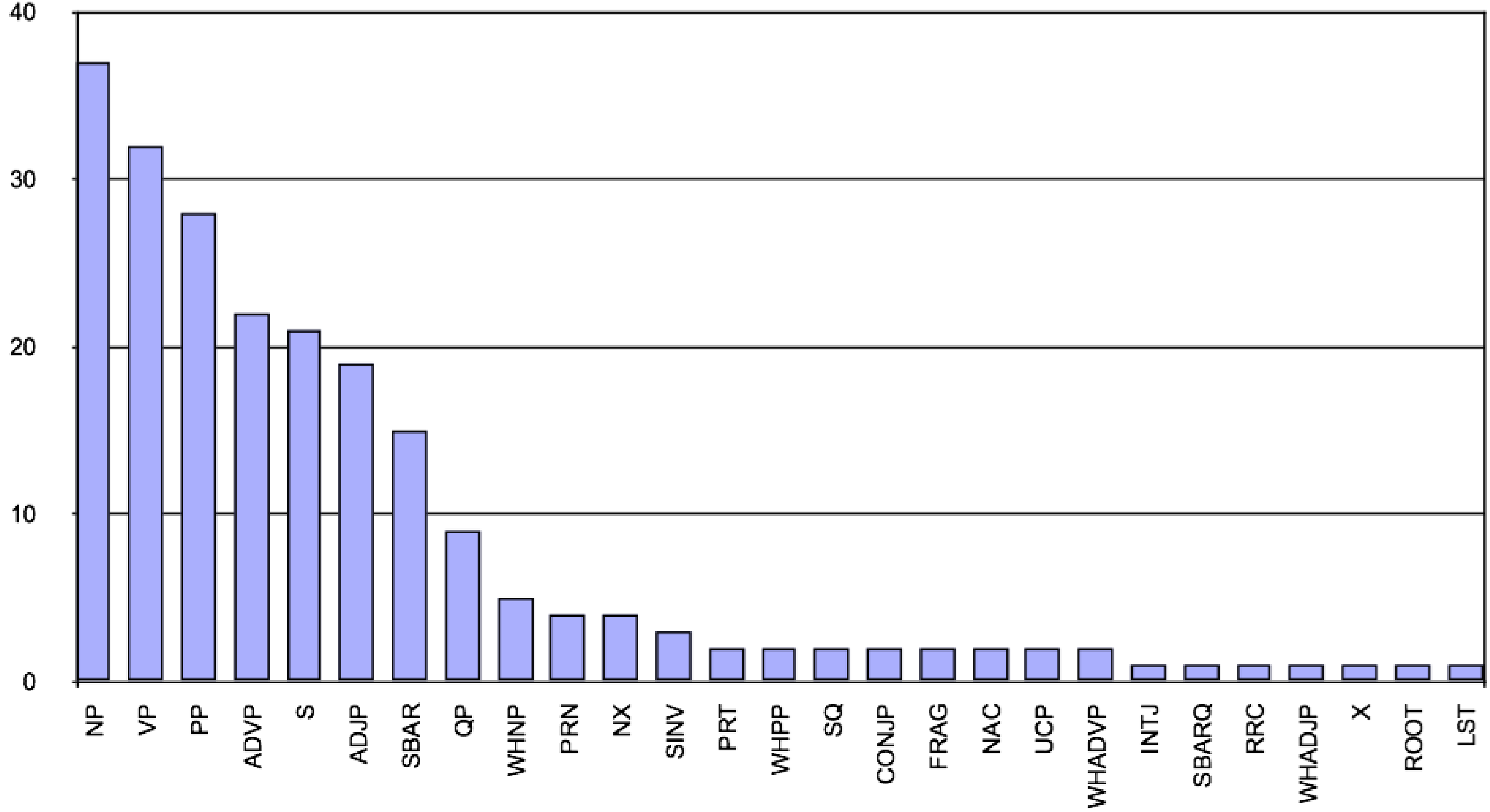


Adaptive Splitting Results



Model	*F1*
Previous	88.4
With 50% Merging	89.5

Number of Phrasal Subcategories



Final Results

Parser	*F1* *≤ 40 words*	*F1* *all words*
Klein & Manning ’03	86.3	85.7
Matsuzaki et al. ’05	86.7	86.1
Collins ’99	88.6	88.2
Charniak & Johnson ’05	90.1	89.6
Petrov et. al. 06	**90.2**	**89.7**

Learned Splits

- Proper Nouns (NNP):

NNP-14	Oct.	Nov.	Sept.
NNP-12	John	Robert	James
NNP-2	J.	E.	L.
NNP-1	Bush	Noriega	Peters
NNP-15	New	San	Wall
NNP-3	York	Francisco	Street

- Personal pronouns (PRP):

PRP-0	It	He	I
PRP-1	it	he	they
PRP-2	it	them	him

Learned Splits

- Relative adverbs (RBR):

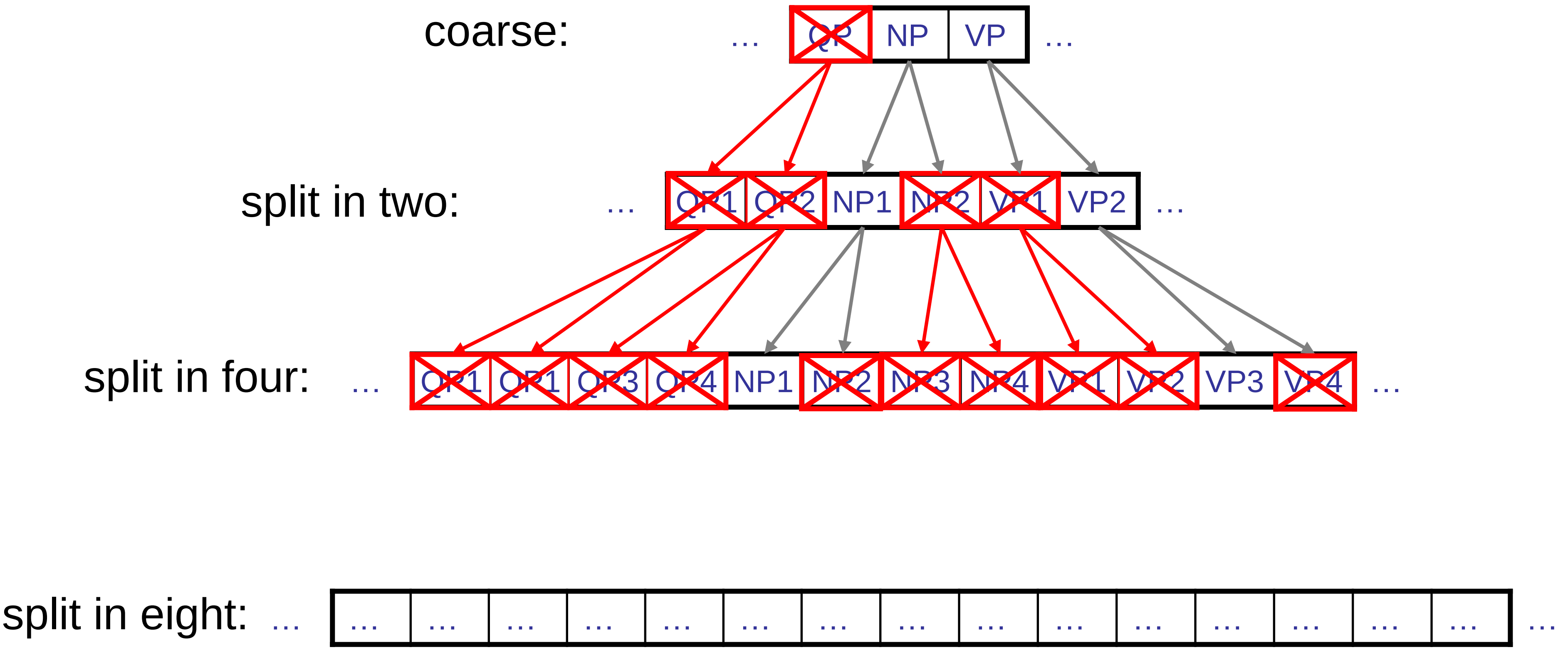
RBR-0	further	lower	higher
RBR-1	more	less	More
RBR-2	earlier	Earlier	later

- Cardinal Numbers (CD):

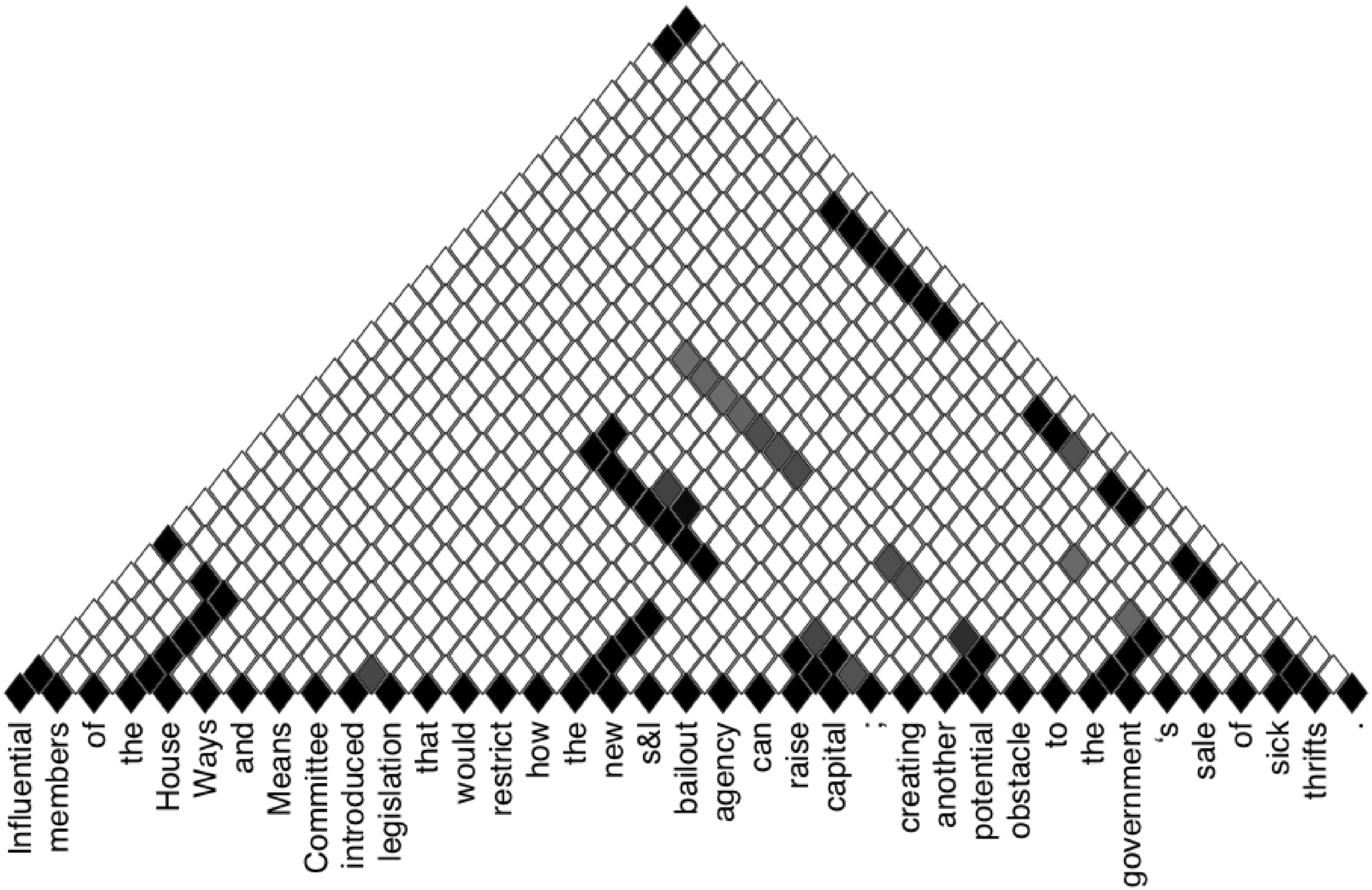
CD-7	one	two	Three
CD-4	1989	1990	1988
CD-11	million	billion	trillion
CD-0	1	50	100
CD-3	1	30	31
CD-9	78	58	34

Hierarchical Pruning

Parse multiple times, with grammars at different levels of granularity!



Bracket Posteriors



1621 min

111 min

35 min

15 min

(no search error)

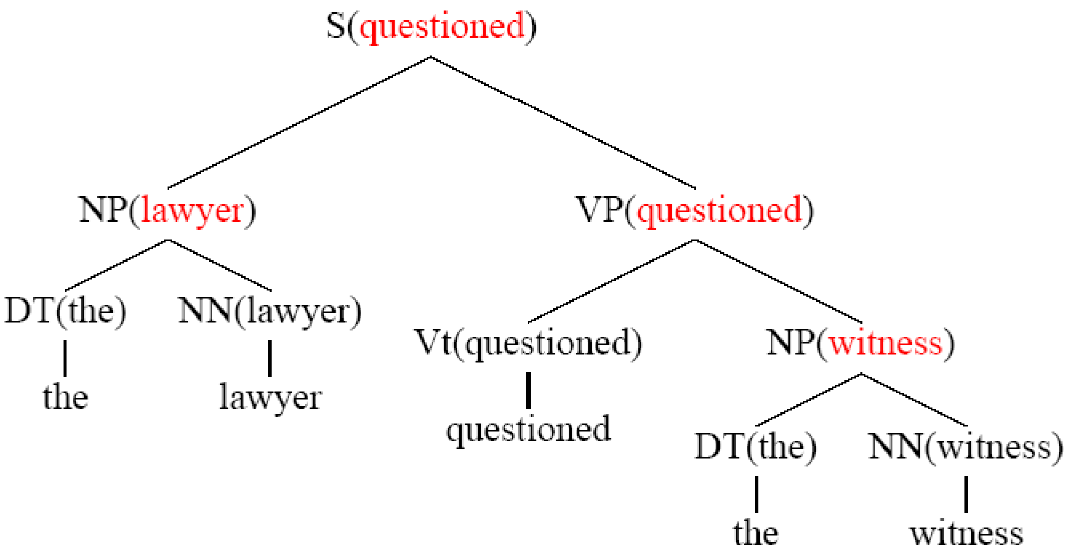
Final Results (Accuracy)

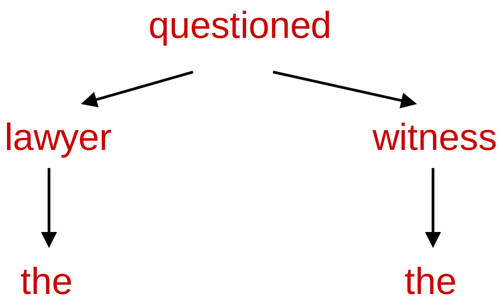
		≤ 40 words F1	all F1
ENG	Charniak&Johnson '05 (generative)	90.1	89.6
	Split / Merge	**90.6**	**90.1**
GER	Dubey '05	76.3	-
	Split / Merge	**80.8**	**80.1**
CHN	Chiang et al. '02	80.0	76.6
	Split / Merge	**86.3**	**83.4**

Still higher numbers from reranking / self-training methods

Dependency Parsing*

- Lexicalized parsers can be seen as producing *dependency trees*

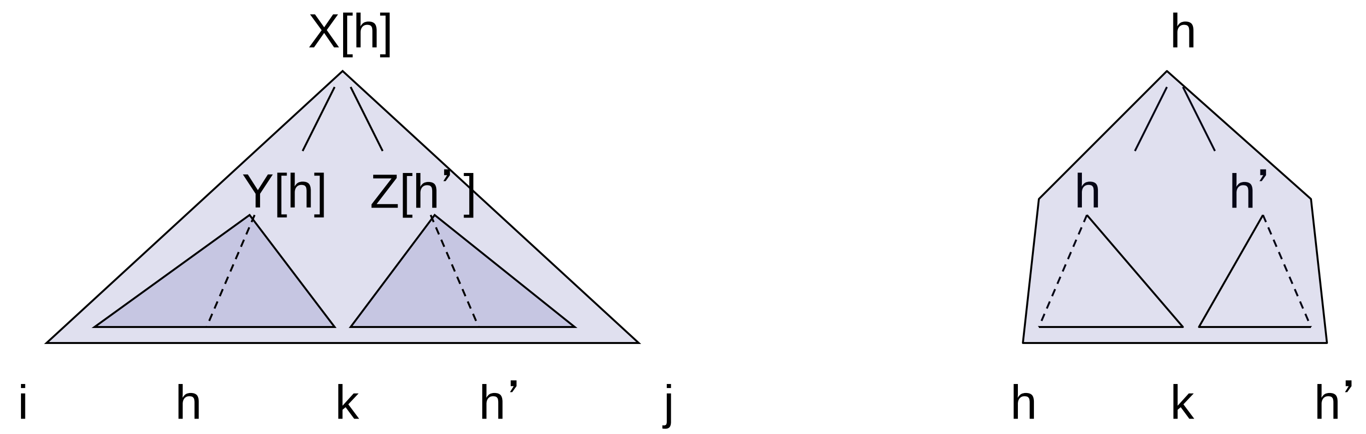




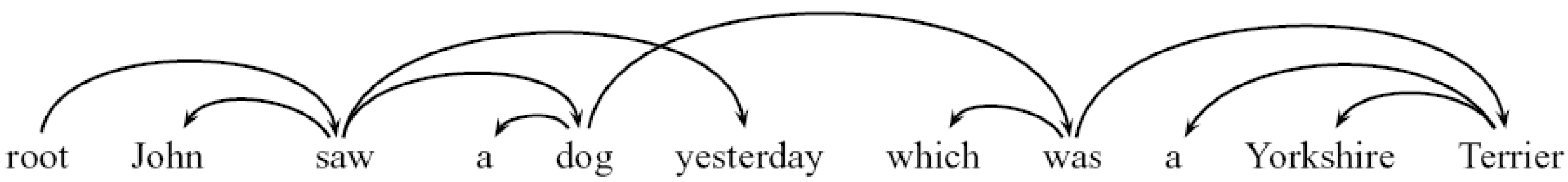
- Each local binary tree corresponds to an attachment in the dependency graph

Dependency Parsing*

- Pure dependency parsing is only cubic [Eisner 99]

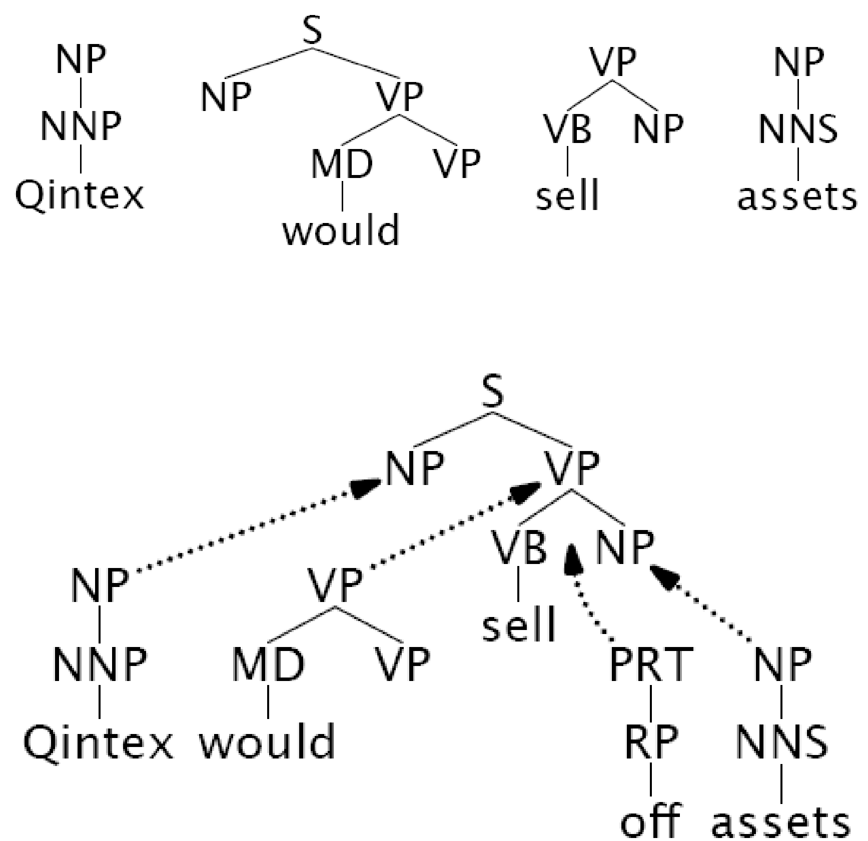


- Some work on *non-projective* dependencies
  - Common in, e.g. Czech parsing
  - Can do with MST algorithms [McDonald and Pereira 05]

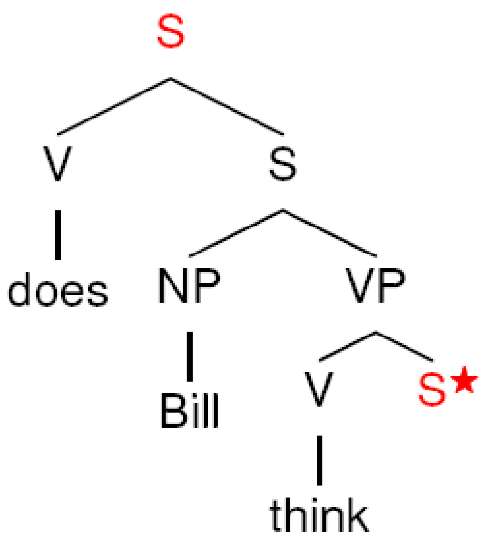


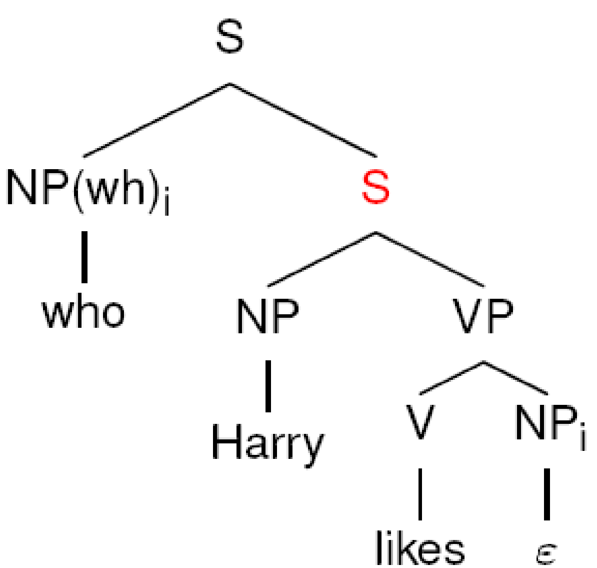
Tree-adjoining grammars*

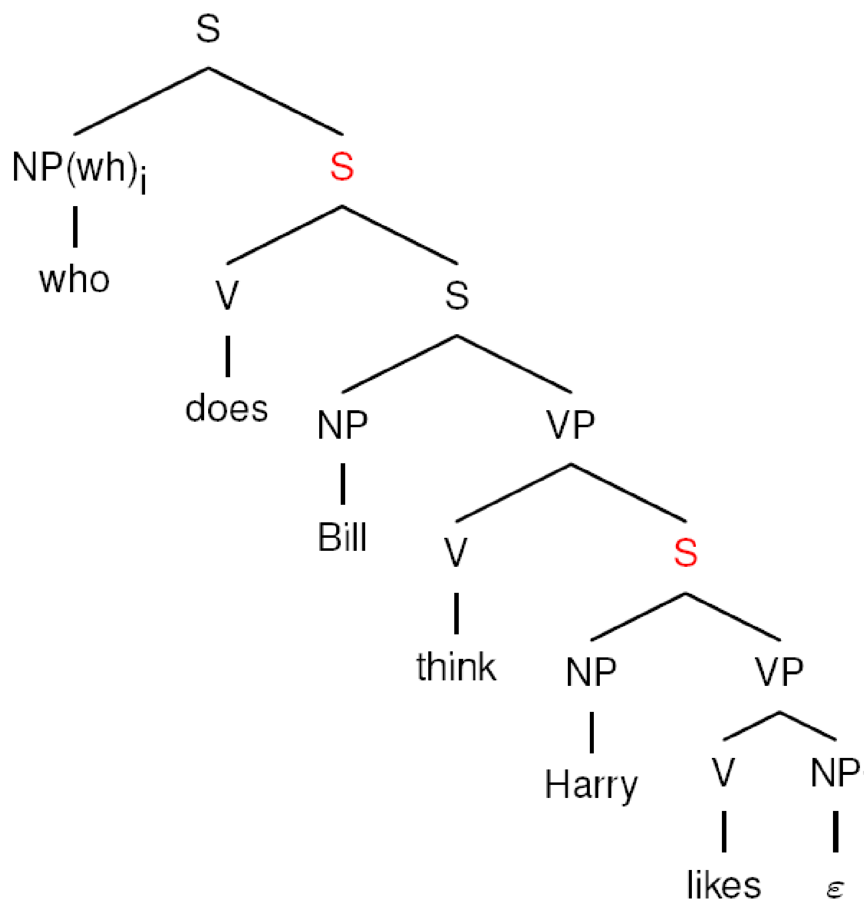
- Start with *local trees*
- Can insert structure with *adjunction* operators
- Mildly context-sensitive
- Models long-distance dependencies naturally
- … as well as other weird stuff that CFGs don’t capture well (e.g. cross-serial dependencies)



TAG: Long Distance*







CCG Parsing*

- Combinatory Categorial Grammar
  - Fully (mono-) lexicalized grammar
  - Categories encode argument sequences
  - Very closely related to the lambda calculus (more later)
  - Can have spurious ambiguities (why?)

$$\mathit{John} \vdash \mathsf{NP}$$

$$\mathit{shares} \vdash \mathsf{NP}$$

$$\mathit{buys} \vdash (\mathsf{S}\backslash\mathsf{NP})/\mathsf{NP}$$

$$\mathit{sleeps} \vdash \mathsf{S}\backslash\mathsf{NP}$$

$$\mathit{well} \vdash (\mathsf{S}\backslash\mathsf{NP})\backslash(\mathsf{S}\backslash\mathsf{NP})$$