CS395T: Structured Models for NLP
Lecture 7: Parsing I

Greg Durrett

Adapted from Dan Klein – UC Berkeley

Administrivia

Project 1 due one week from today!

Recall: EM for HMMs

Maximize lower bound of log marginal likelihood:

$$\log \sum_{\mathbf{y}} P(\mathbf{x}, \mathbf{y}|\theta) \geq \mathbb{E}_{q(\mathbf{y})} \log P(\mathbf{x}, \mathbf{y}|\theta) + \text{Entropy}[q(\mathbf{y})]$$

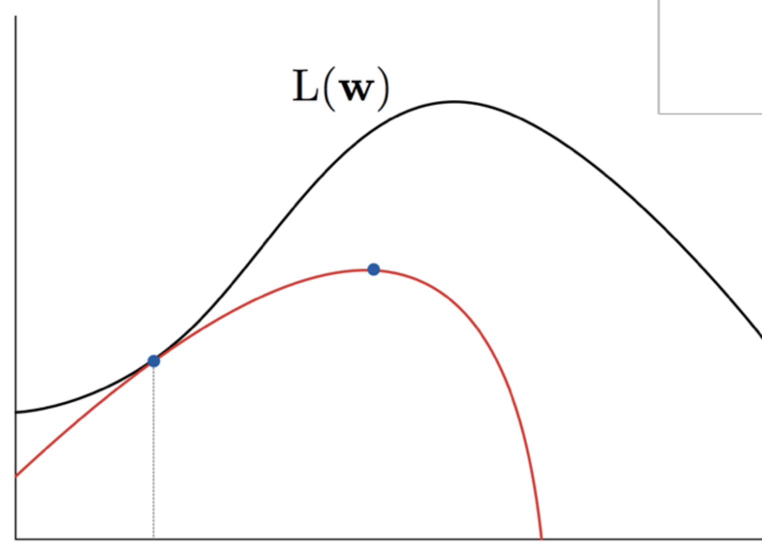
EM: alternating maximization

E-step: maximize w.r.t. q

$$q^t = P(\mathbf{y}|\mathbf{x}, \theta^{t-1})$$

M-step: maximize w.r.t. theta

$$\theta^t = \text{argmax}_\theta \mathbb{E}_{q^t} \log P(\mathbf{x}, \mathbf{y}|\theta)$$



Supervised learning from fractional annotation

Road Map

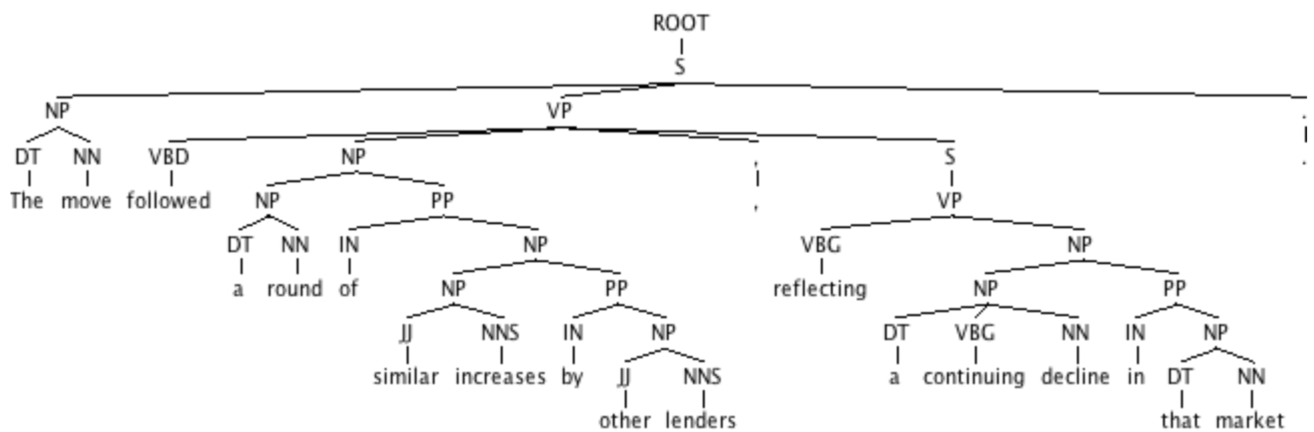
- Done: Sequences: generative, discriminative, supervised, unsupervised
- Now: trees (parsing) – a little more linguistics...
- This week: constituency – lots of generative models
- Next week: dependency (Project 2) – more discriminative models

This Lecture

- Constituency formalism
- (Probabilistic) Context-free Grammars
- CKY
- Refining grammars
- Next time: finish constituency + writing tips

Syntax

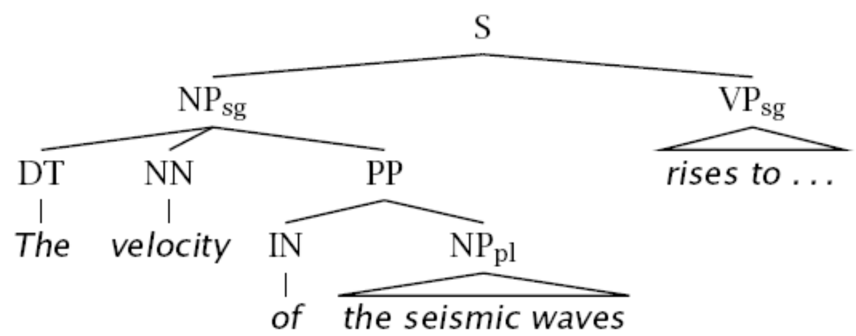
Parse Trees



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

Phrase Structure Parsing

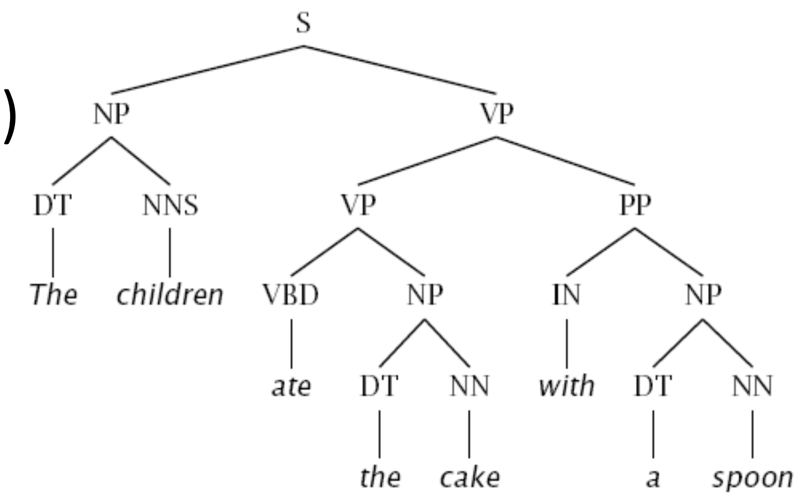
- Phrase structure parsing organizes syntax into *constituents* or *brackets*
- In general, this involves nested trees
- Linguists can, and do, argue about details
- Lots of ambiguity
- Dependency (next week) makes more sense for some languages



Constituency Tests

- How do we know what nodes go in the tree?

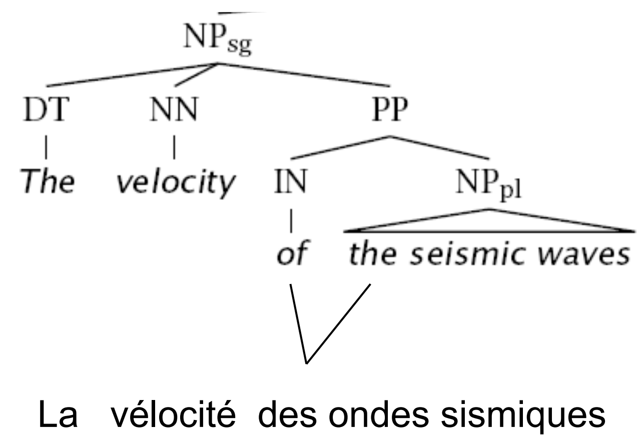
- Classic constituency tests:
  - Substitution by *proform*
  - Clefting (*It was with a spoon*...)
  - Answer ellipsis (What did you eat?)
  - Coordination



- Cross-linguistic arguments, too

Conflicting Tests

- Constituency isn't always clear
  - 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.



Classical NLP: Parsing

- Write symbolic or logical rules:

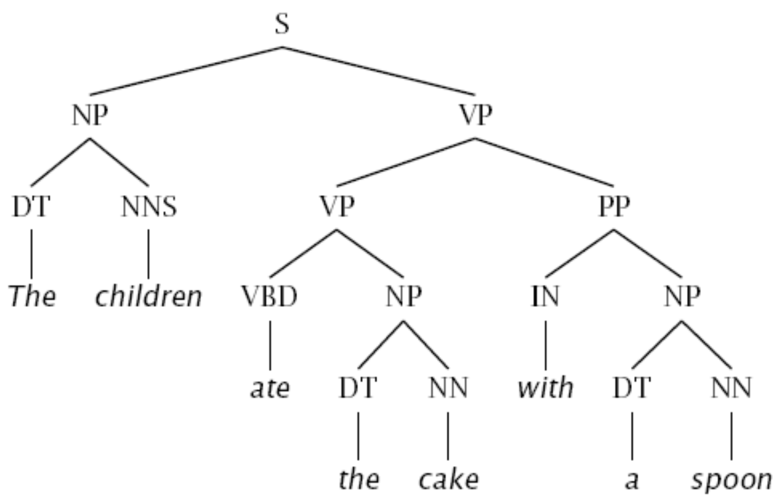
Grammar (CFG)		Lexicon
ROOT → S	NP → NP PP	NN → interest
S → NP VP	VP → VBP NP	NNS → raises
NP → DT NN	VP → VBP NP PP	VBP → interest
NP → NN NNS	PP → IN NP	VBZ → raises
		…

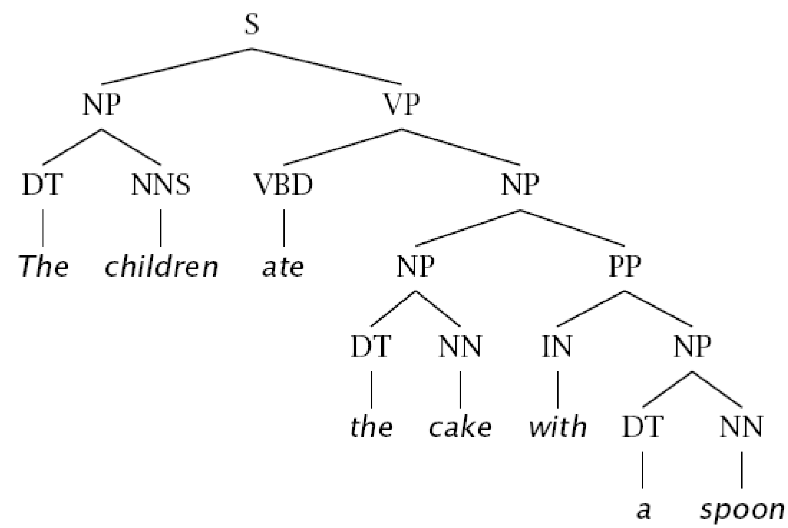
- Use deduction systems to prove parses from words
  - Minimal grammar on "Fed raises" sentence: 36 parses
  - Simple 10-rule grammar: 592 parses
  - Real-size grammar: many millions of parses

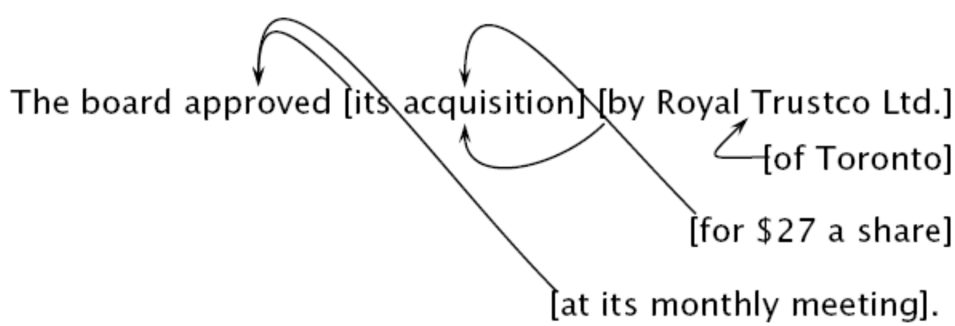
- This scaled very badly, didn't yield broad-coverage tools

Ambiguities

Ambiguities: PP Attachment







Attachments

- 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

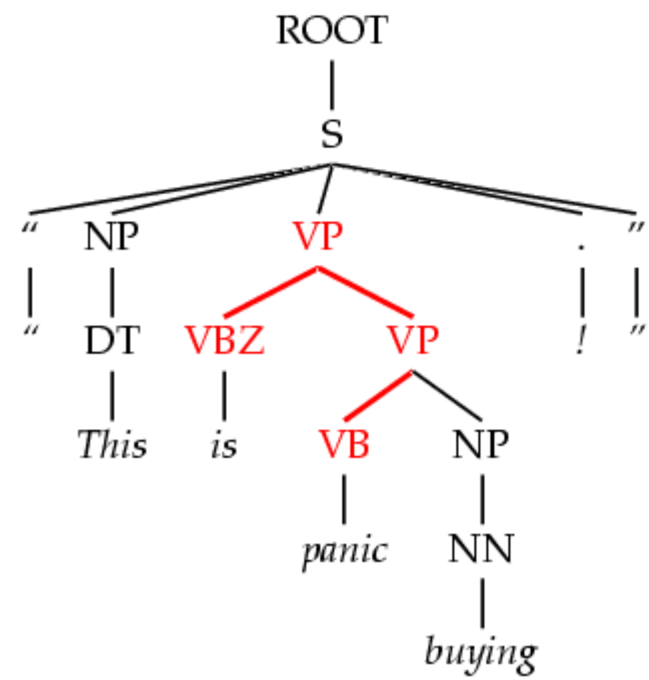
- Coordination scope:
  Small rats and mice can squeeze into holes or cracks in the wall.

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 probabilistic techniques handle this uncertainty

PCFGs

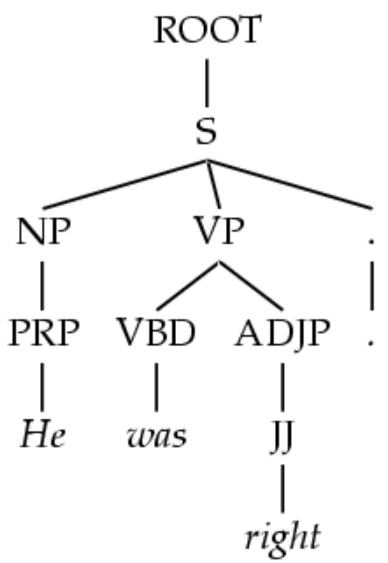
Probabilistic Context-Free Grammars

- A context-free grammar is a tuple <*N, T, S, R*>
  - *N* : the set of non-terminals
    - Phrasal categories: S, NP, VP, ADJP, etc.
    - Parts-of-speech (pre-terminals): NN, JJ, DT, VB
  - *T* : the set of terminals (the words)
  - *S* : the start symbol
    - Often written as ROOT or TOP (not S – not all “sentences” are sentences)
  - *R* : the set of rules
    - Of the form $X \rightarrow Y_1\ Y_2 \ldots Y_k$, with $X, Y_i \in N$
    - Examples: $S \rightarrow NP\ VP$, $VP \rightarrow VP\ CC\ VP$
    - Also called rewrites or productions

- A PCFG adds:
  - A top-down production probability per rule $P(Y_1\ Y_2 \ldots Y_k \mid X)$

Treebank Grammars

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

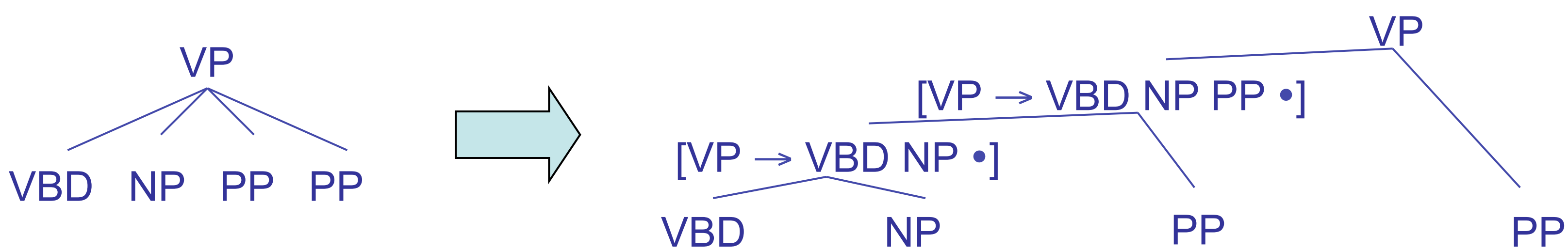


ROOT → S	1
S → NP VP .	1
NP → PRP	1
VP → VBD ADJP	1
…..	

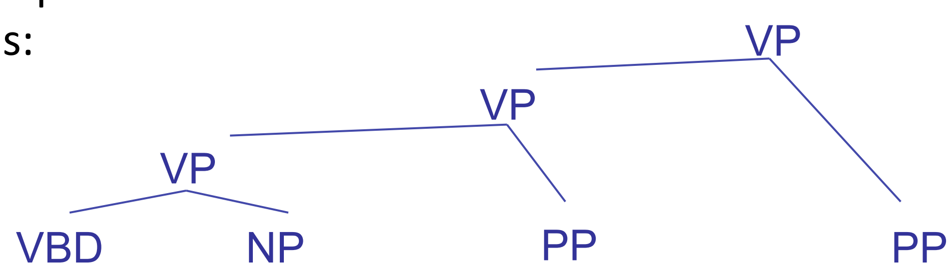
- Maximum-likelihood estimate: get P(NP -> PRP | NP) by counting + normalizing
- Better results by enriching the grammar (lexicalization, other techniques)

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"
    - NOT equivalent to this:



- In practice: *binarize* the grammar, keep unaries

CKY Parsing

A Recursive Parser

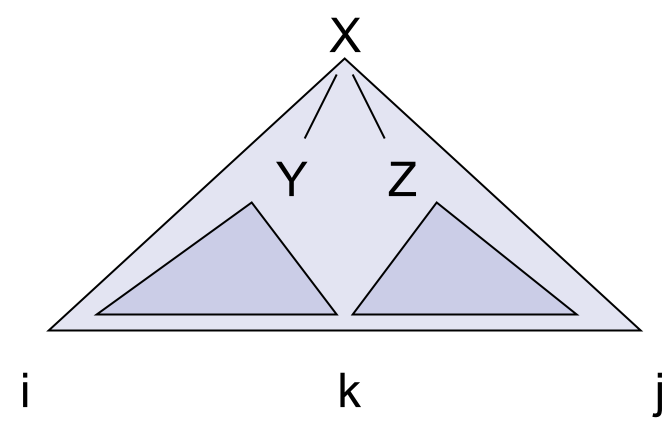
```
bestScore(X,i,j,s)
   if (j = i+1)
       return tagScore(X,s[i])
   else
       return max score(X->YZ) *
                   bestScore(Y,i,k,s) *
                   bestScore(Z,k,j,s)
```

- max over k and rule being applied
- Will this parser work?

A Bottom-Up Parser (CKY)

- Can also organize things bottom-up
- Not every tag/nonterminal can be built over every span!

```
bestScore(s)
  for (i : [0,n-1])
    for (X : tags[s[i]])
      score[X][i][i+1] =
          tagScore(X,s[i])
  for (diff : [2,n])
    for (i : [0,n-diff])
      j = i + diff
      for (X->YZ : rule)
        for (k : [i+1, j-1])
          score[X][i][j] = max score[X][i][j],
                               score(X->YZ) *
                               score[Y][i][k] *
                               score[Z][k][j]
```



Unary Rules

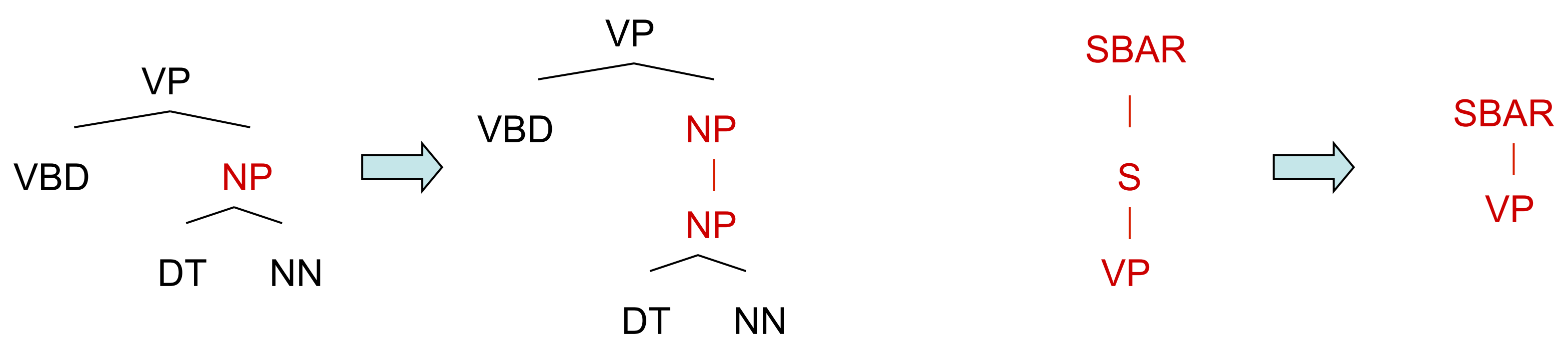
- Unary rules?

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

- Problem: dynamic program is self-referential!

Unary Closure

- We need unaries to be non-cyclic
  - Can address by pre-calculating the *unary closure*
  - Rather than having zero or more unaries, always have exactly one



  - Alternate unary and binary layers
  - Reconstruct unary chains afterwards

Alternating Layers

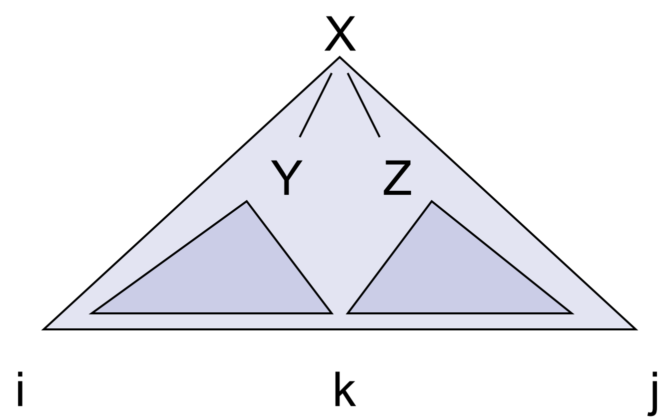
```
bestScoreB(X,i,j,s)
      return max max score(X->YZ) *
                      bestScoreU(Y,i,k) *
                      bestScoreU(Z,k,j)


bestScoreU(X,i,j,s)
   if (j = i+1)
       return tagScore(X,s[i])
   else
       return max max score(X->Y) *
                      bestScoreB(Y,i,j)
```

Analysis

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

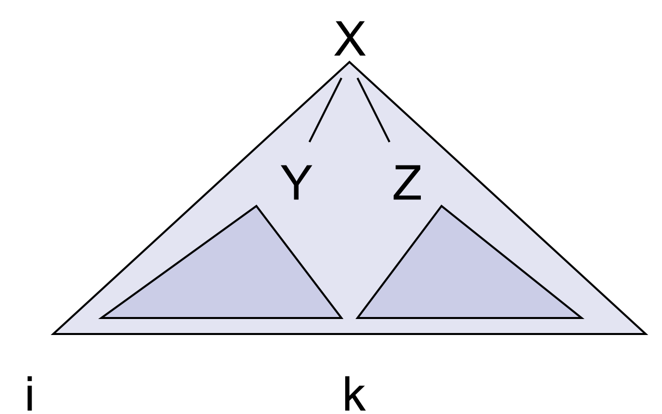


Time: Theory

- 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

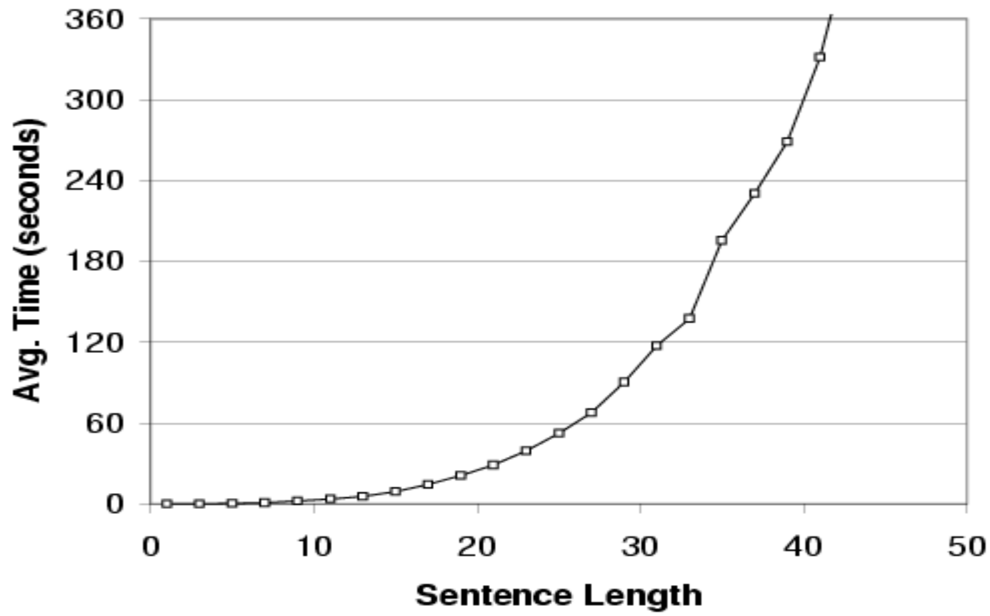
          Do constant work



  - Total time: |rules|*n^3
  - Simple grammar takes 0.1 sec to parse a 20-word sentence, bigger grammars can take 10+ seconds unoptimized

Time: Practice

- Parsing with the vanilla treebank grammar:



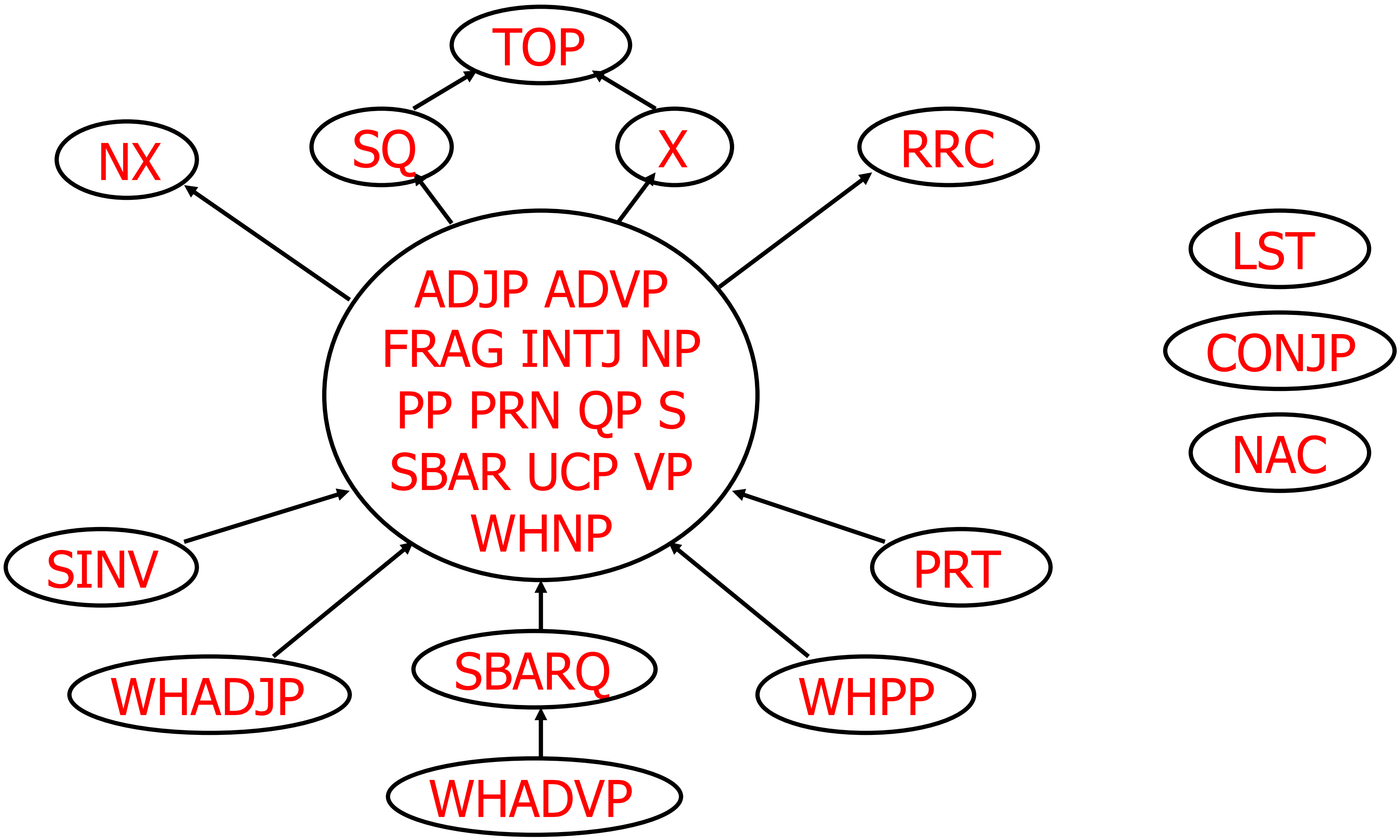
~ 20K Rules

(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

Same-Span Reachability



Efficient CKY

- Lots of tricks to make CKY efficient
  - Some of them are little engineering details:
    - E.g., first choose k, then enumerate through the Y:[i,k] which are non-zero, then loop through rules by left child.
    - Optimal layout of the dynamic program depends on grammar
  - Some are algorithmic improvements:
    - Pruning: rule out chunks of the chart based on a simpler model

Learning PCFGs

Typical Experimental Setup

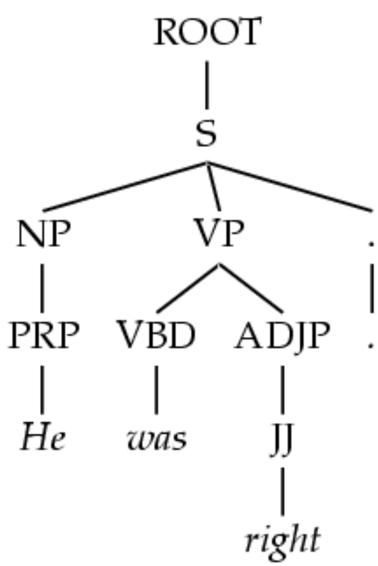
- Corpus: Penn Treebank, WSJ

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.

Treebank PCFGs [Charniak 96]

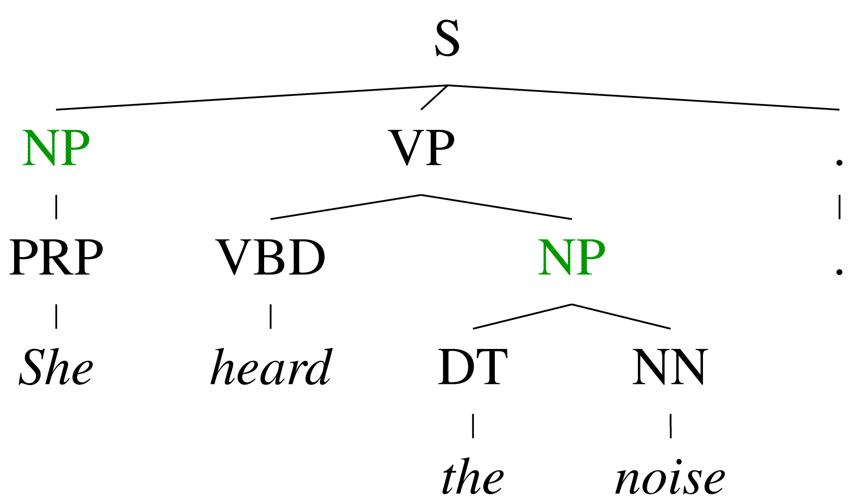
- Use PCFGs for broad coverage parsing
- Can take a grammar right off the trees (doesn't work well):



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

Model	*F1*
Baseline	72.0

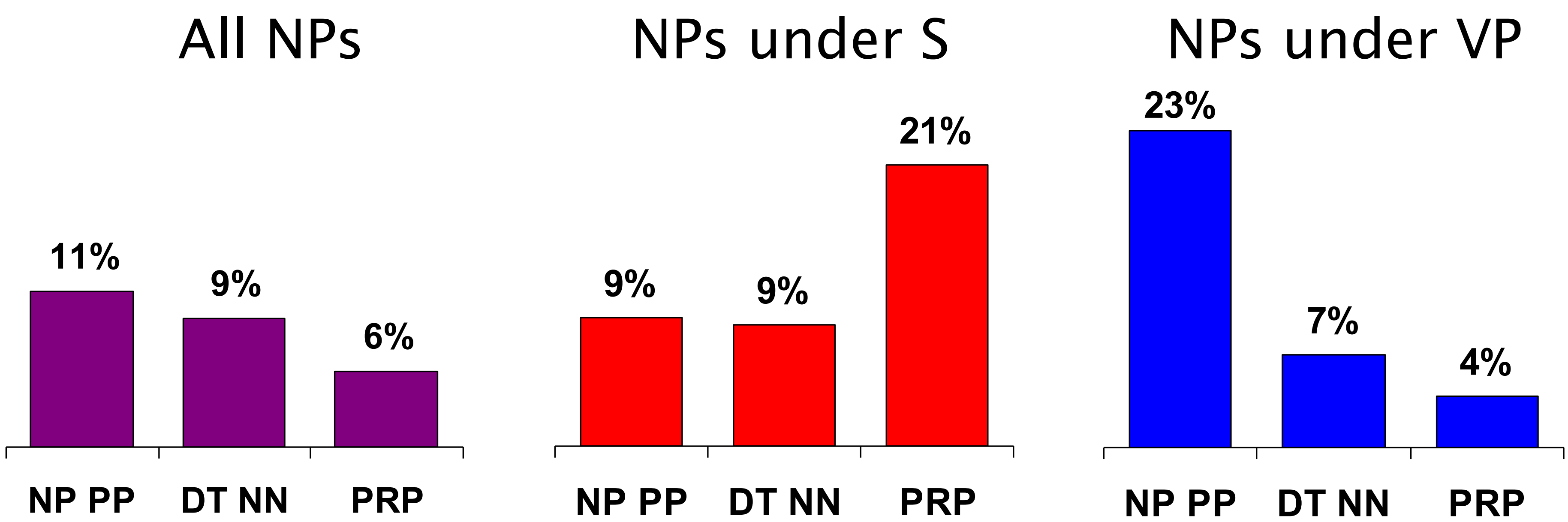
Conditional Independence?



- 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

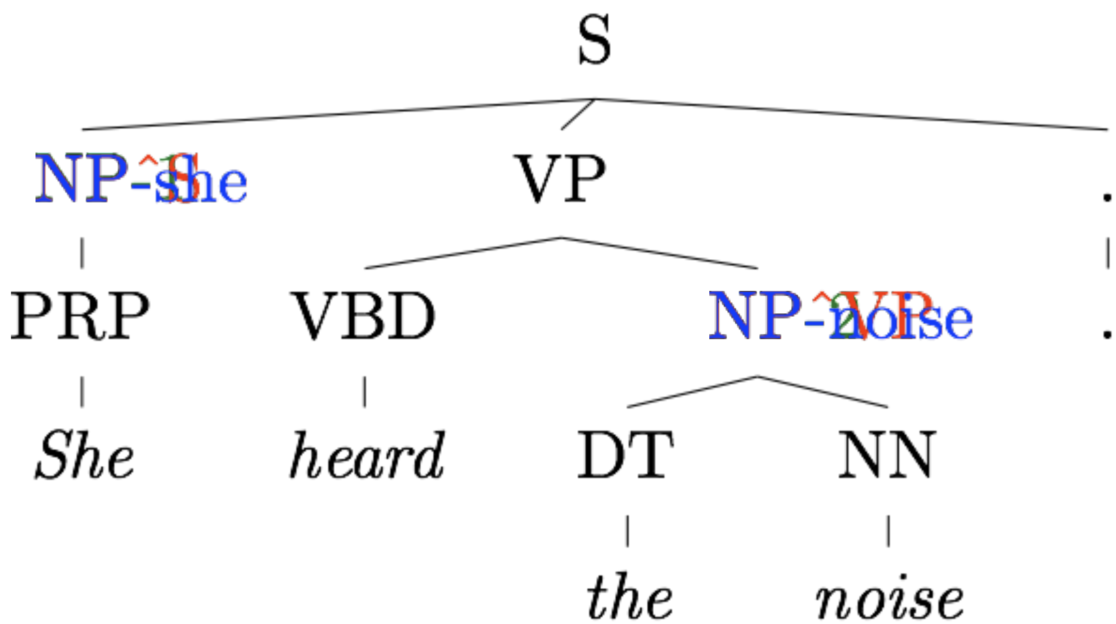
Non-Independence

- 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!

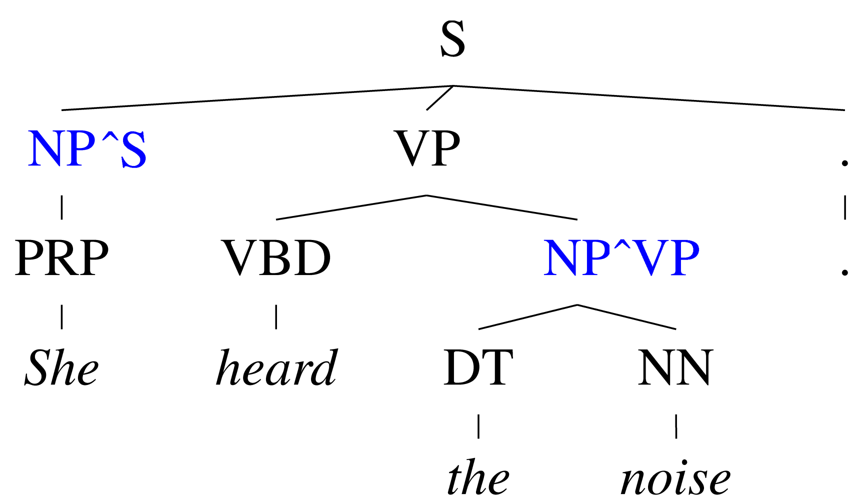
Grammar Refinement



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

Structural Annotation

The Game of Designing a Grammar

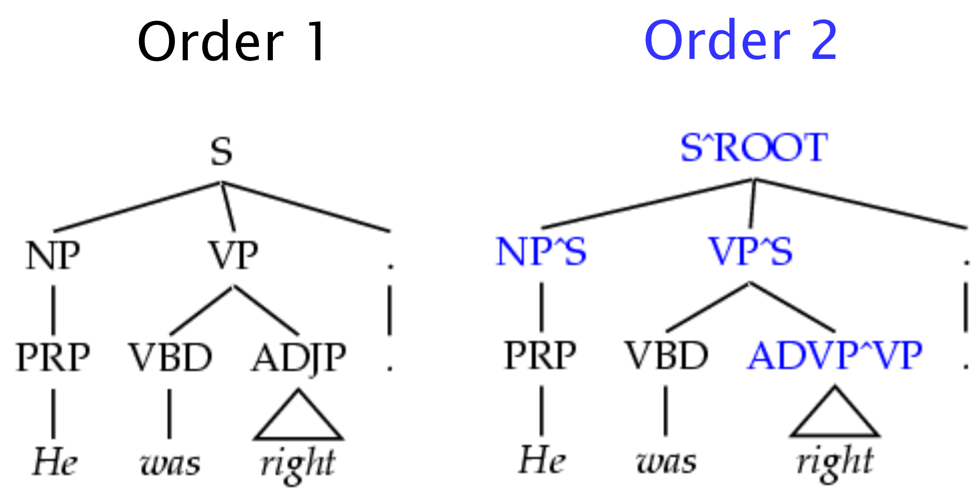


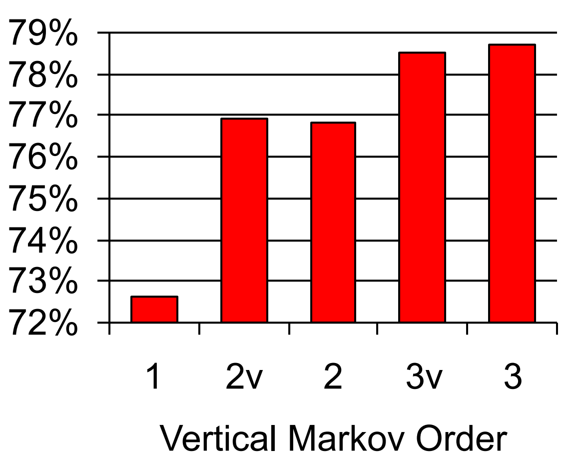
- Annotation refines base treebank symbols to improve statistical fit of the grammar
  - Structural annotation

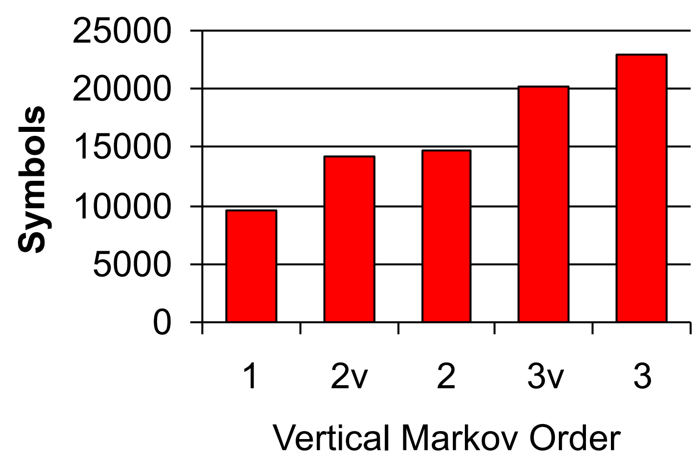
Vertical Markovization

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

  (cf. parent annotation)

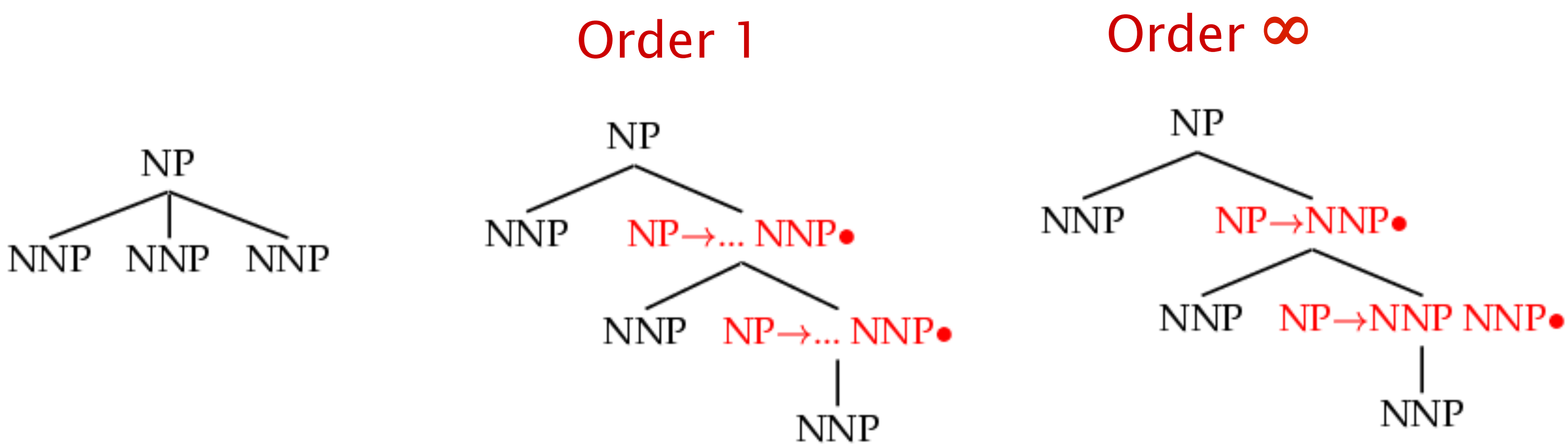


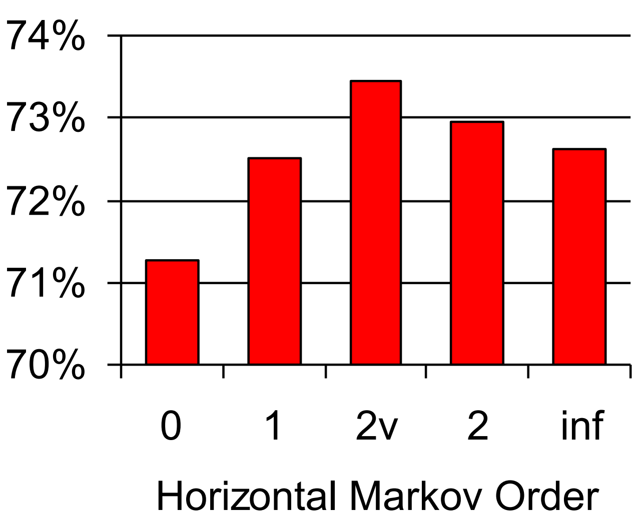


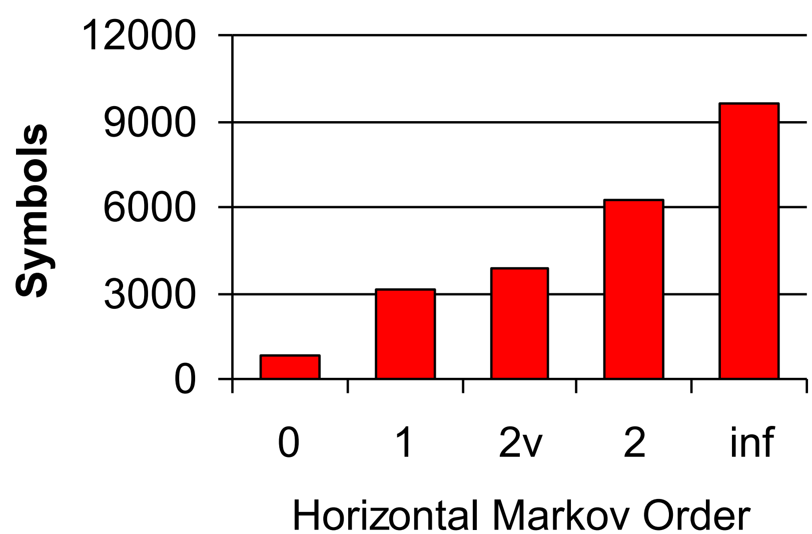


Klein and Manning (2003)

Horizontal Markovization





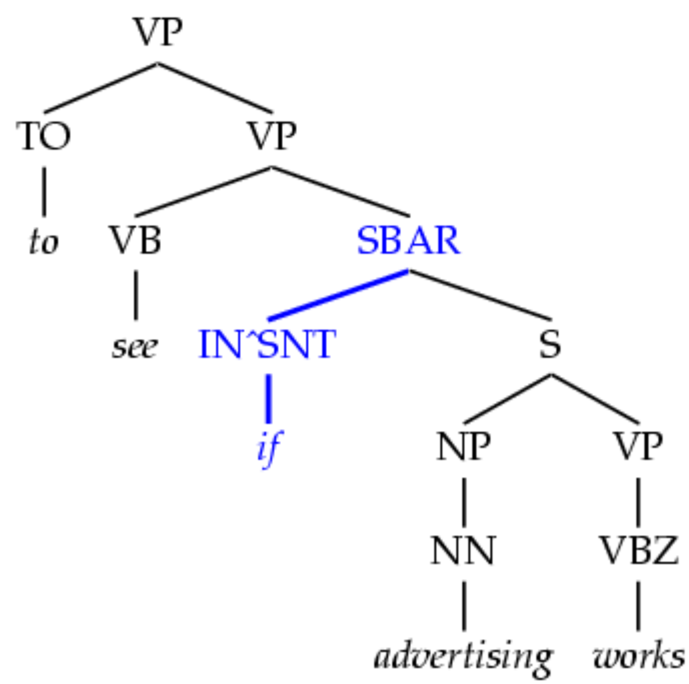


Klein and Manning (2003)

Tag Splits

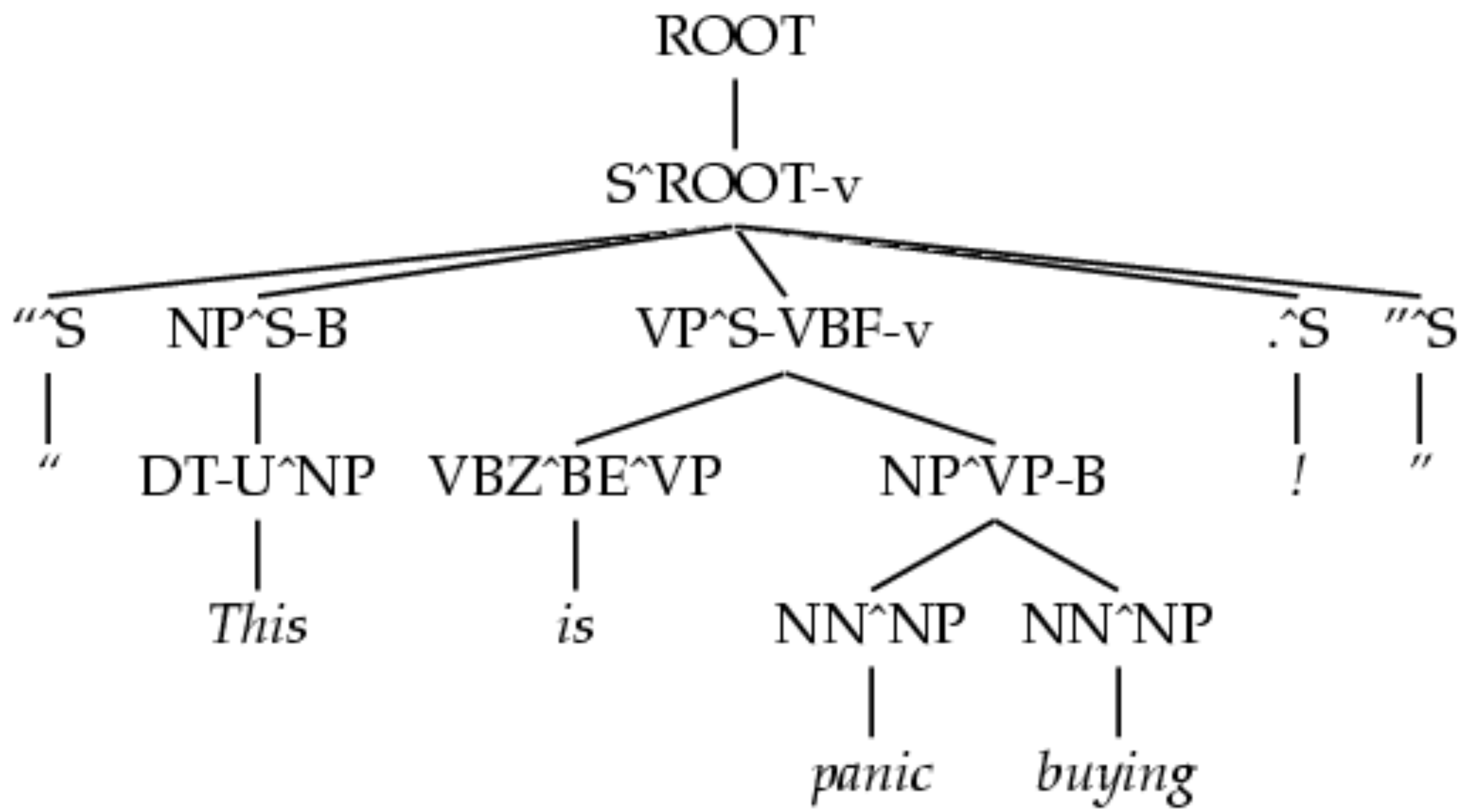
Klein and Manning (2003)

- 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

A Fully Annotated (Unlex) Tree



Some Test Set Results

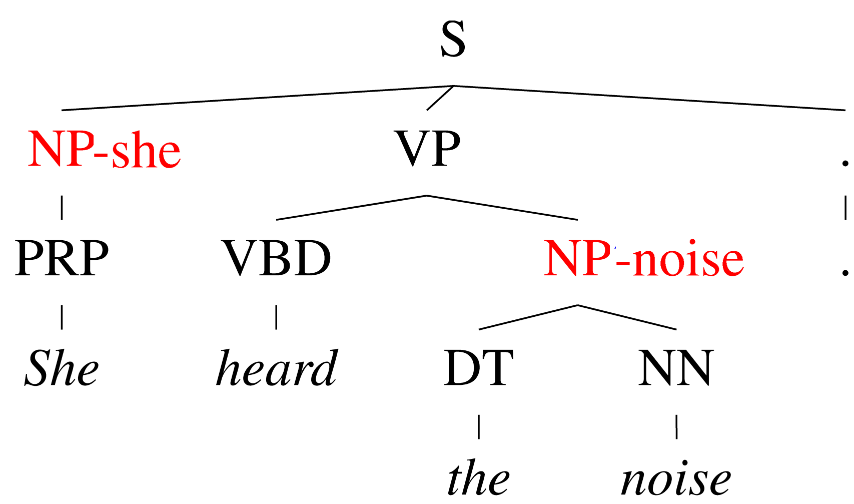
Parser	LP	LR	**F1**	CB	0 CB
Magerman 95	84.9	84.6	**84.7**	1.26	56.6
Collins 96	86.3	85.8	**86.0**	1.14	59.9
K+M 2003	86.9	85.7	**86.3**	1.10	60.3
Charniak 97	87.4	87.5	**87.4**	1.00	62.1
Collins 99	88.7	88.6	**88.6**	0.90	67.1

- Beats "first generation" lexicalized parsers.
- Baseline: ~72

Klein and Manning (2003)

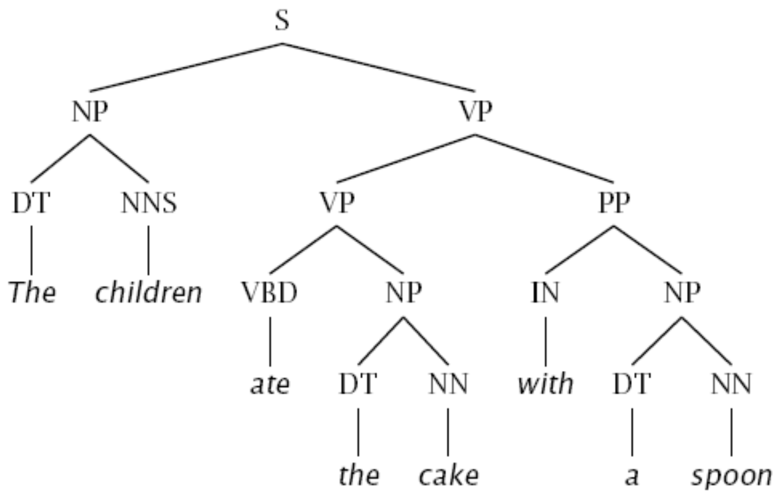
Lexicalization

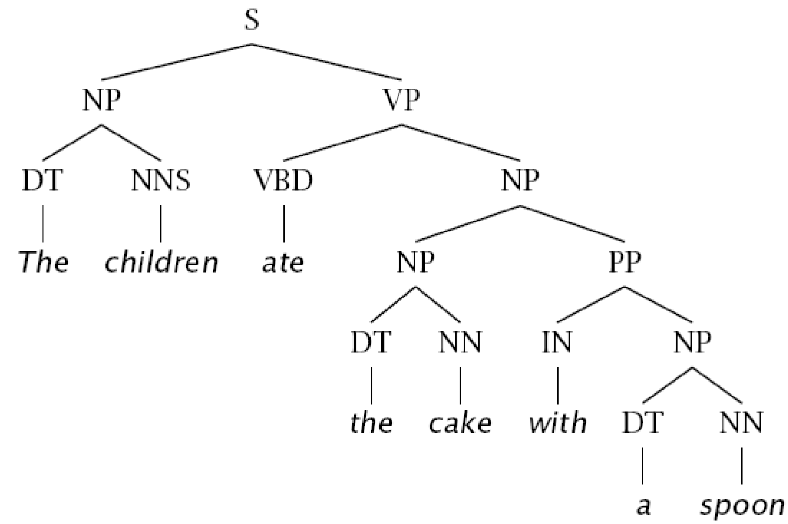
The Game of Designing a Grammar



- 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]

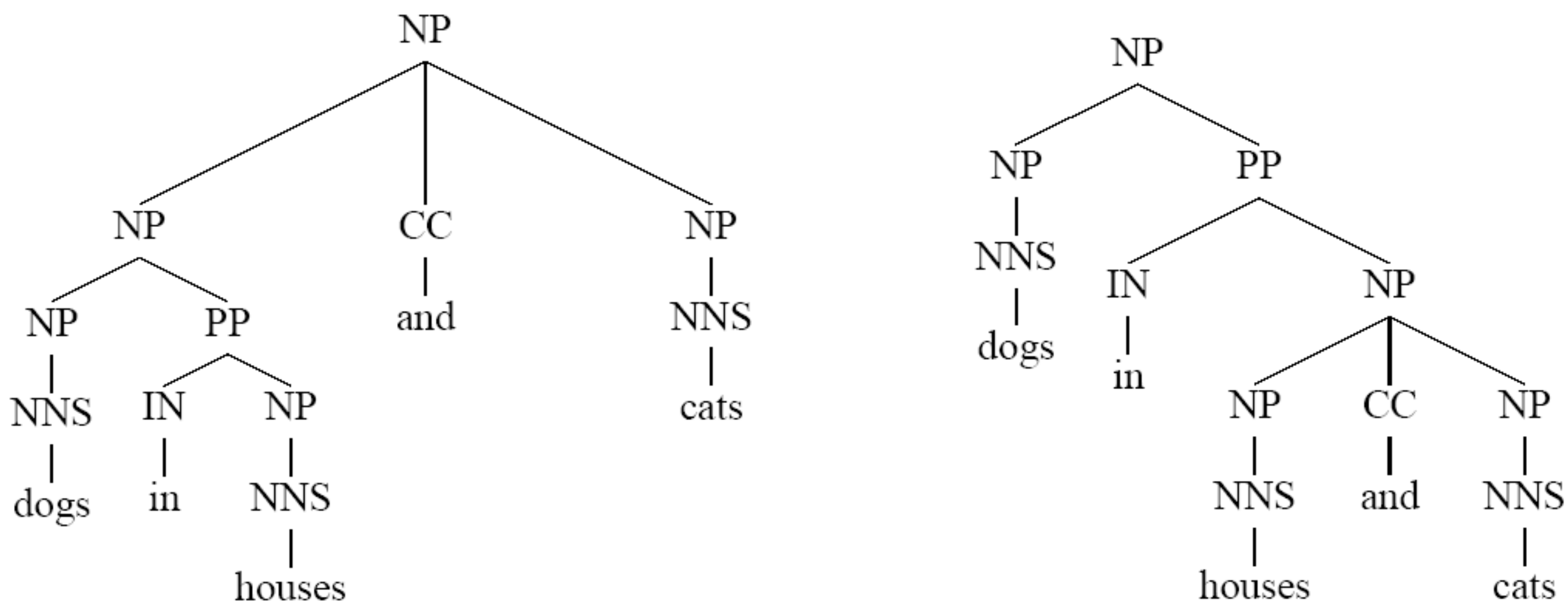
Problems with PCFGs





- 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
- Lexicalization allows us to be sensitive to specific words

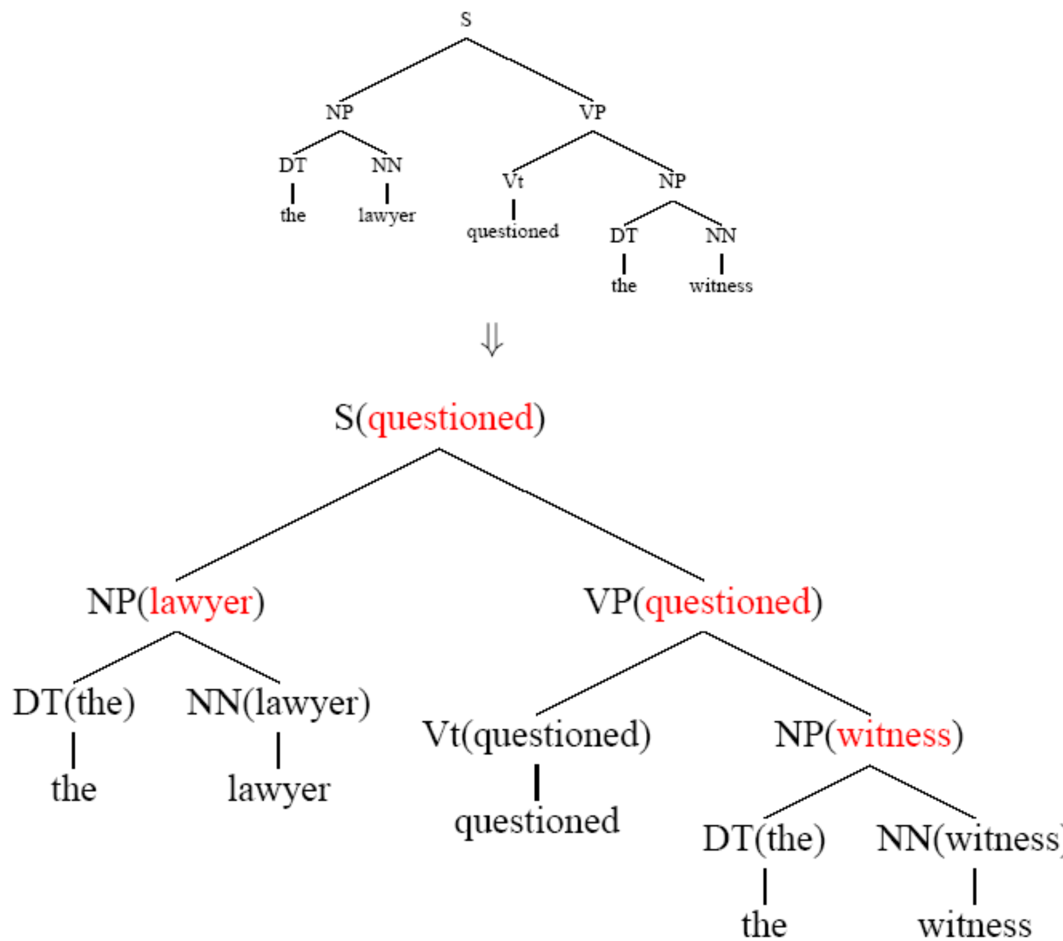
Problems with PCFGs



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

Lexicalized Trees

- Add “head words” to each phrasal node
  - Syntactic vs. semantic heads
  - 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



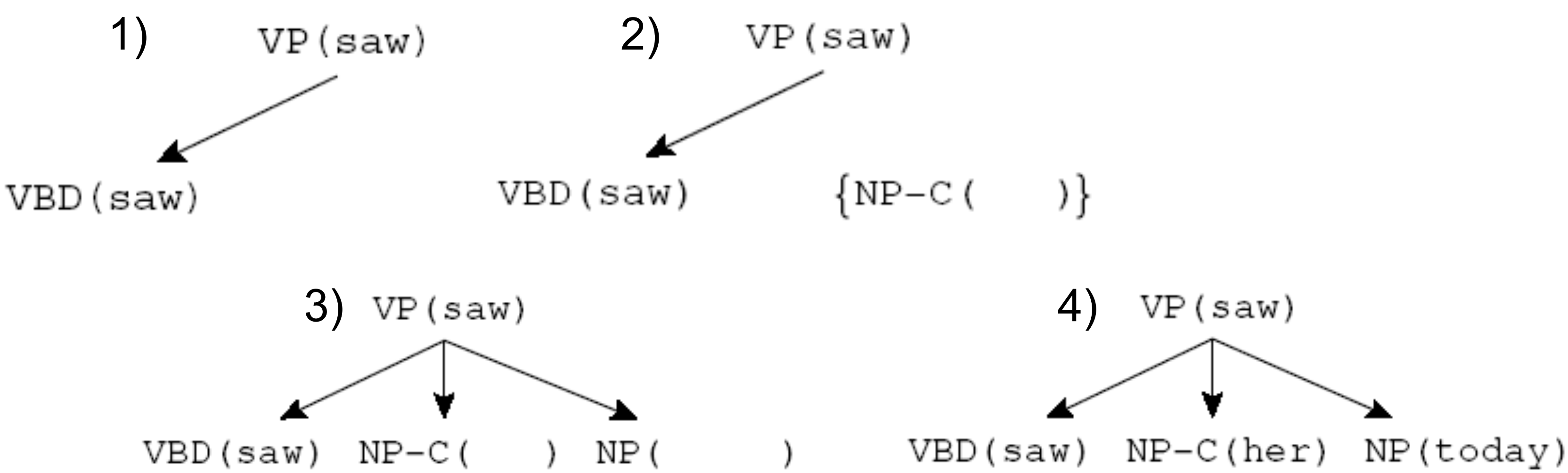
Lexicalized PCFGs?

- Problem: we now have to estimate probabilities like

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

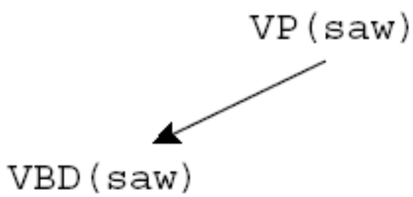
- Never going to get these atomically off of a treebank

- Solution: break up derivation into smaller steps

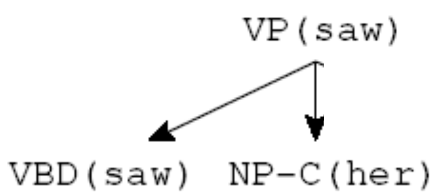


Lexical Derivation Steps

- A derivation of a local tree [Collins 99]

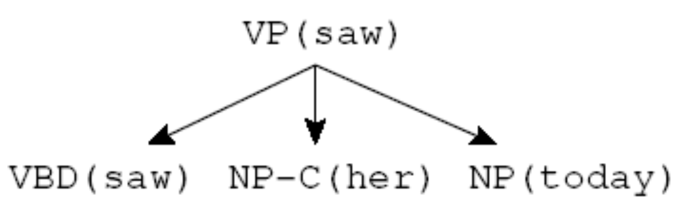


Choose a head tag and word
P(child symbol | parent, head word)



Generate children from head sequentially

P(child tag, child head | parent, head symbol, head word)



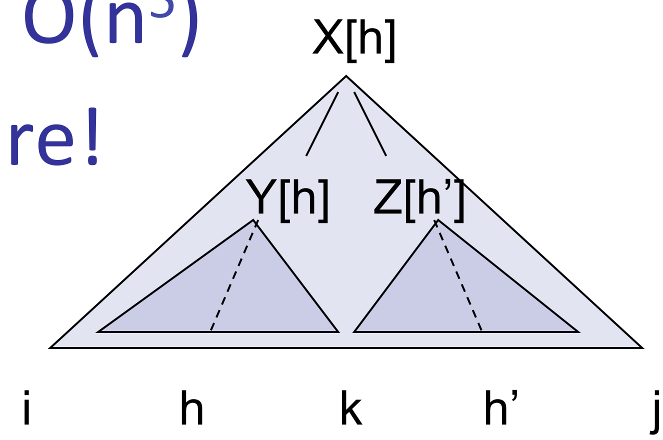
Finish generating the children; each new one conditions on the previous ones

Lexicalized CKY

- How big is the state space?
- Nonterminals = Num symbols x vocab size – way too large!
- Can’t use standard CKY

Lexicalized CKY

- Track index h of head in DP: $O(n^5)$
- Better algorithms next lecture!



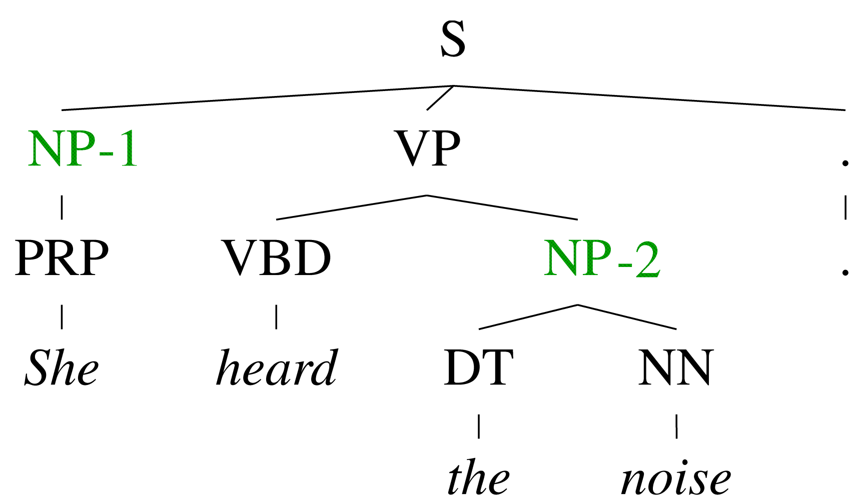

```
bestScore(X,i,j,h,s)
  if (j = i+1)
    return tagScore(X,s[i])
  else
    return
      max  max         score(X[h]->Y[h] Z[h']) *
          k,h',X->YZ
                       bestScore(Y,i,k,h,s) *
                       bestScore(Z,k,j,h',s)
           max         score(X[h]->Y[h'] Z[h]) *
    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

Results

- Some results
  - Collins 99 – 88.6 F1 (generative lexical)
  - Charniak and Johnson 05 – 89.7 / 91.3 F1 (generative lexical / reranked)
  - McClosky et al 06 – 92.1 F1 (gen + rerank + self-train)
  - 92.1 was SOTA for around 8 years!

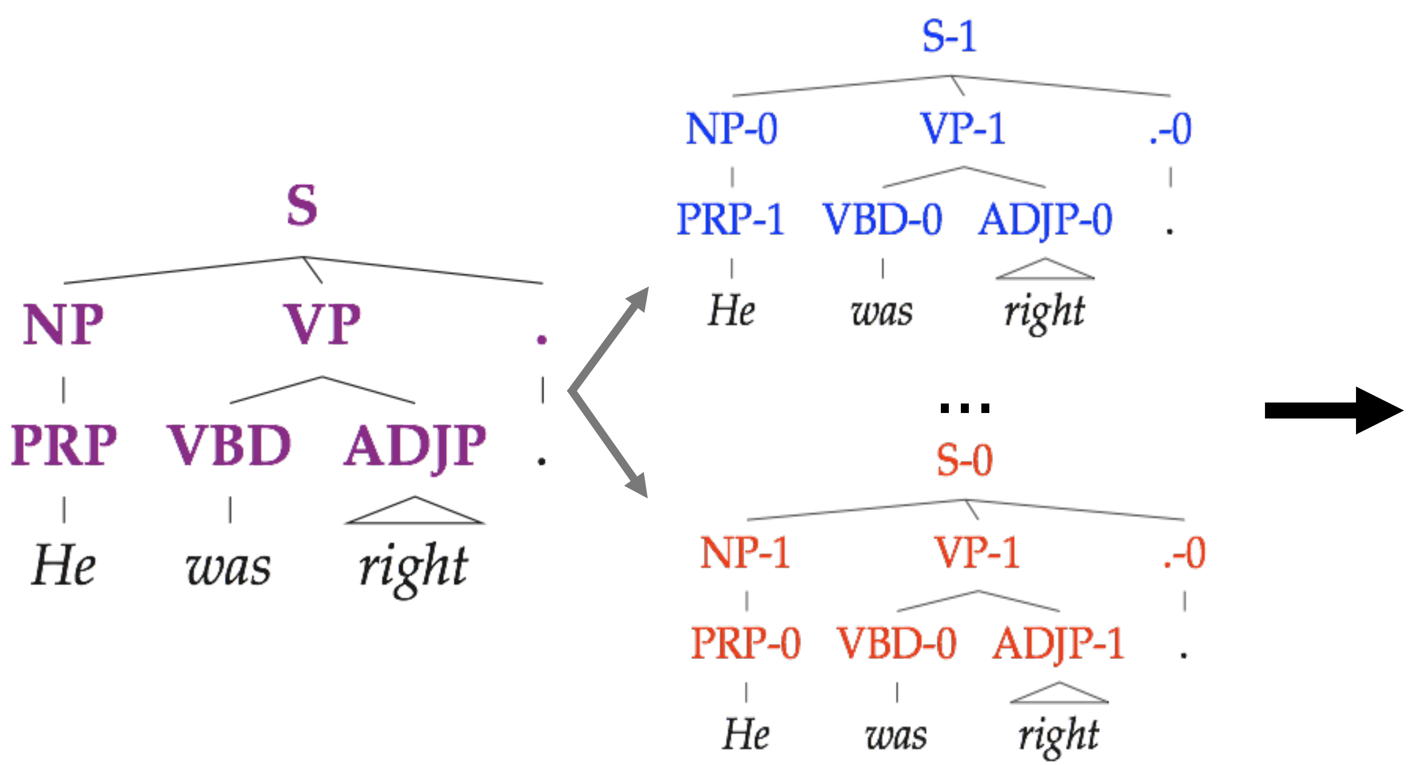
Latent Variable PCFGs

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?

Latent Variable Grammars



Grammar G	
$S_0 \rightarrow NP_0\ VP_0$	?
$S_0 \rightarrow NP_1\ VP_0$	?
$S_0 \rightarrow NP_0\ VP_1$	?
$S_0 \rightarrow NP_1\ VP_1$	?
$S_1 \rightarrow NP_0\ VP_0$	?
...	
$S_1 \rightarrow NP_1\ VP_1$	?
...	
$NP_0 \rightarrow PRP_0$	?
$NP_0 \rightarrow PRP_1$	?
...	

Lexicon	
$PRP_0 \rightarrow$ She	?
$PRP_1 \rightarrow$ She	?
...	
$VBD_0 \rightarrow$ was	?
$VBD_1 \rightarrow$ was	?
$VBD_2 \rightarrow$ was	?
...	

Parse Tree T
Sentence w

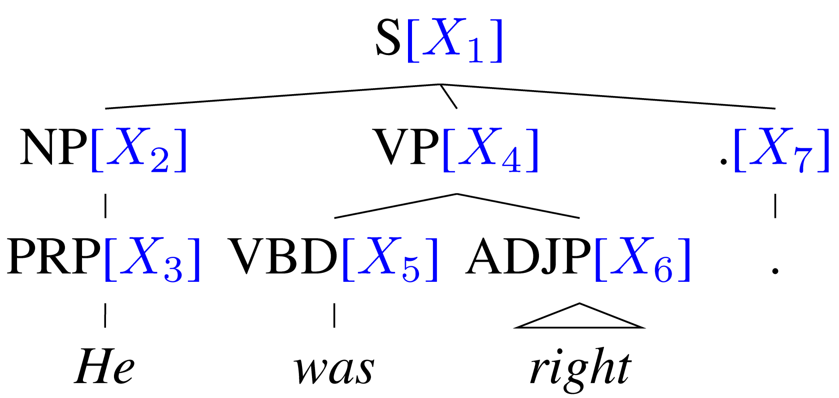
Derivations $t : T$

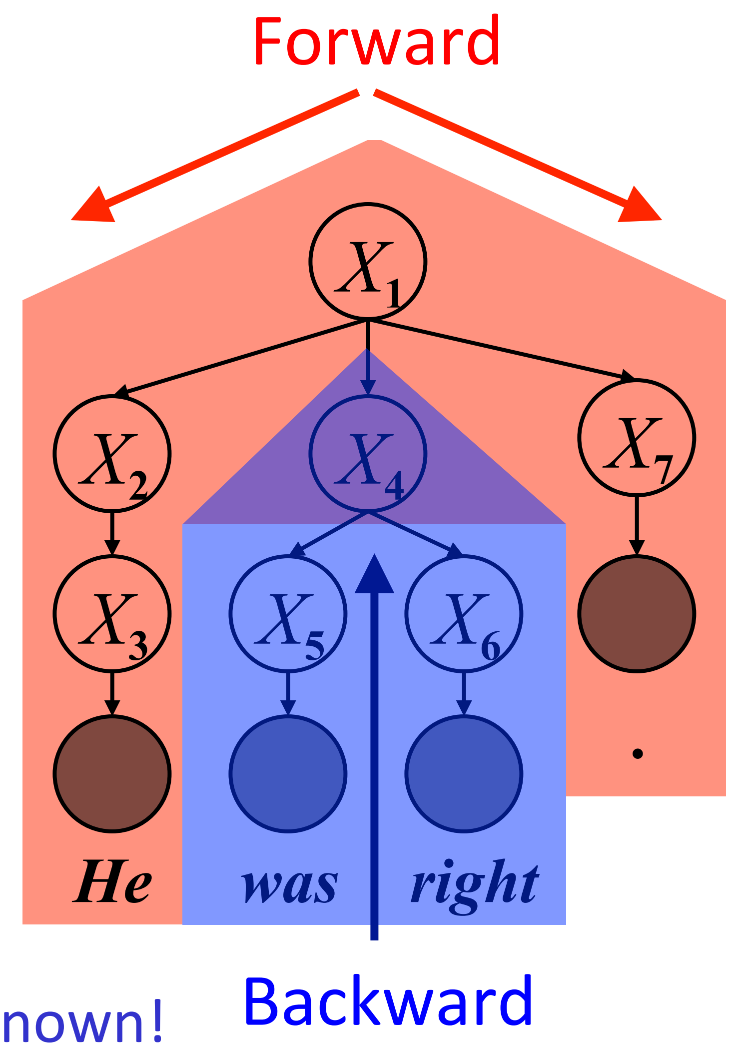
Parameters θ

Learning Latent Annotations

EM algorithm:

- Brackets are known
- Base categories are known
- Only induce subcategories

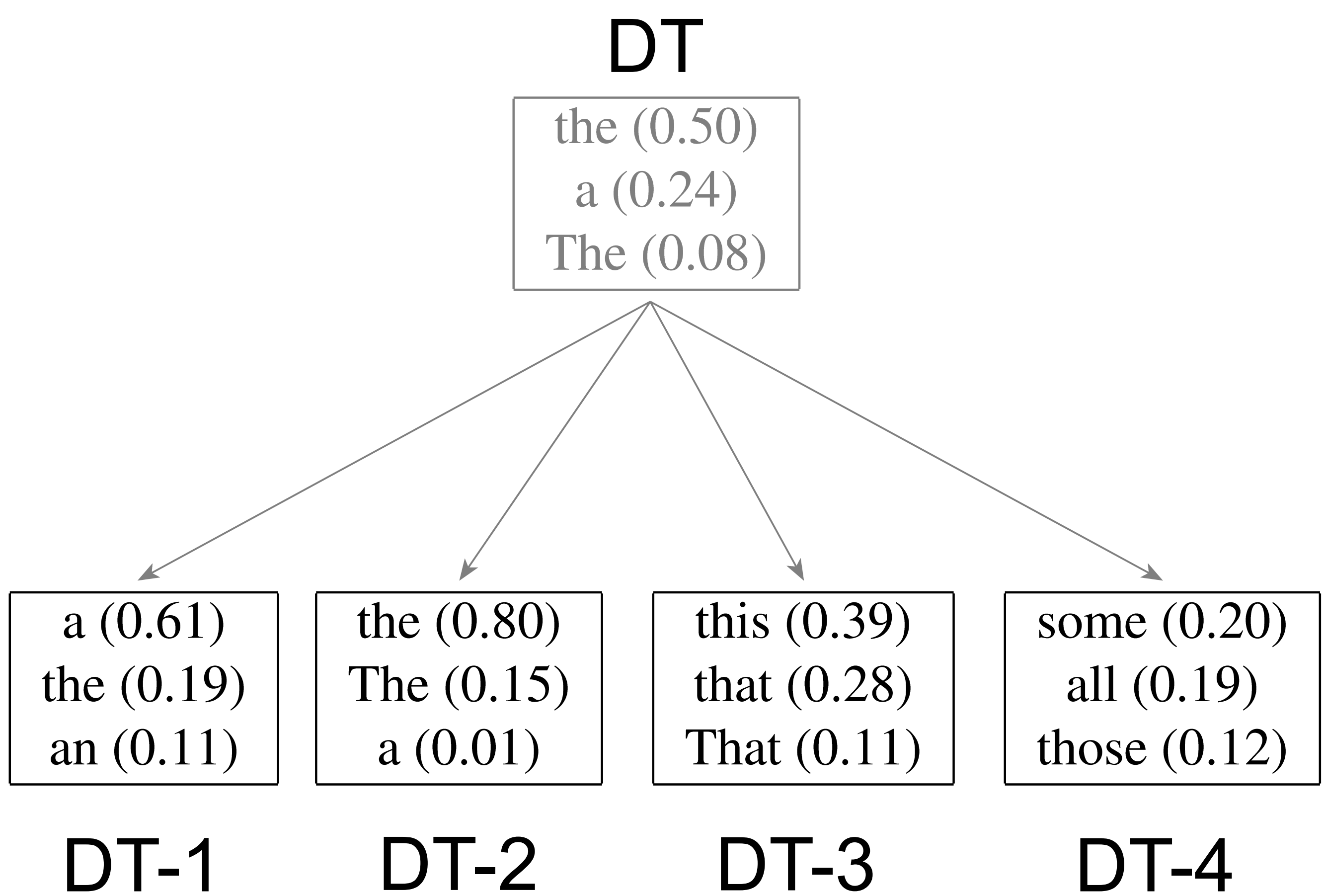




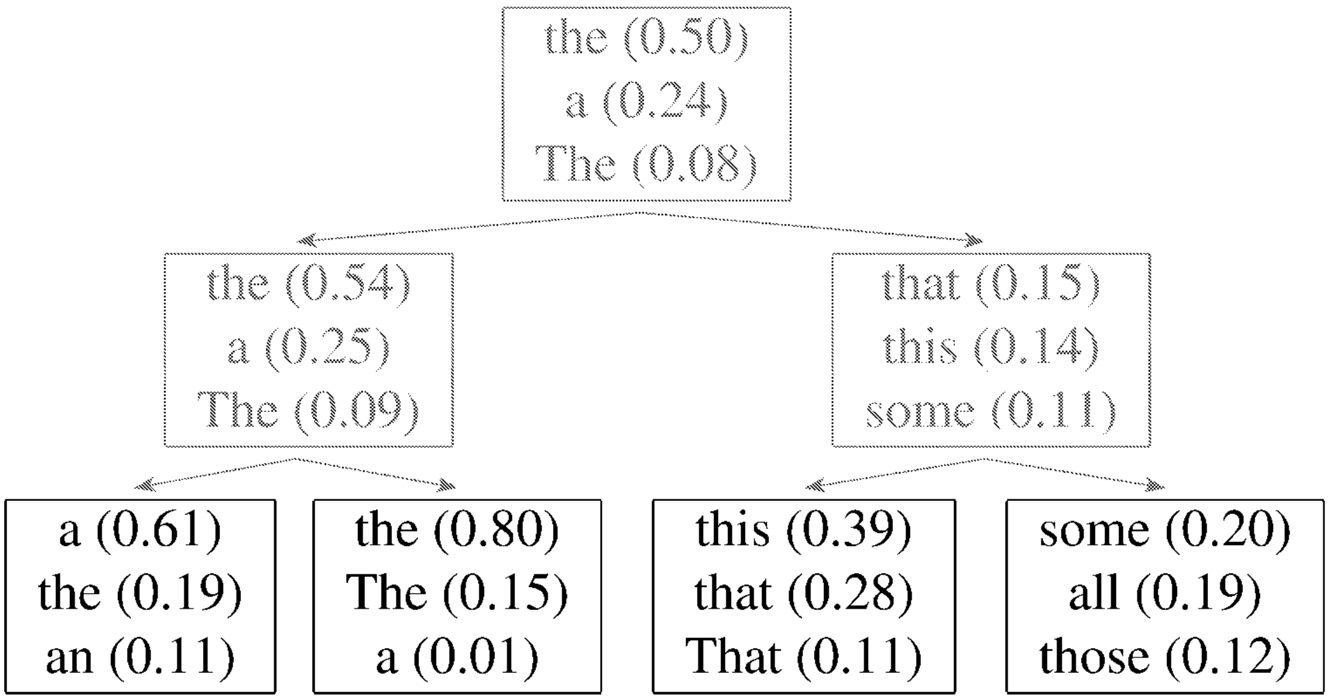
Just like Forward-Backward for HMMs.

Learn label *refinements*, base labels are known!

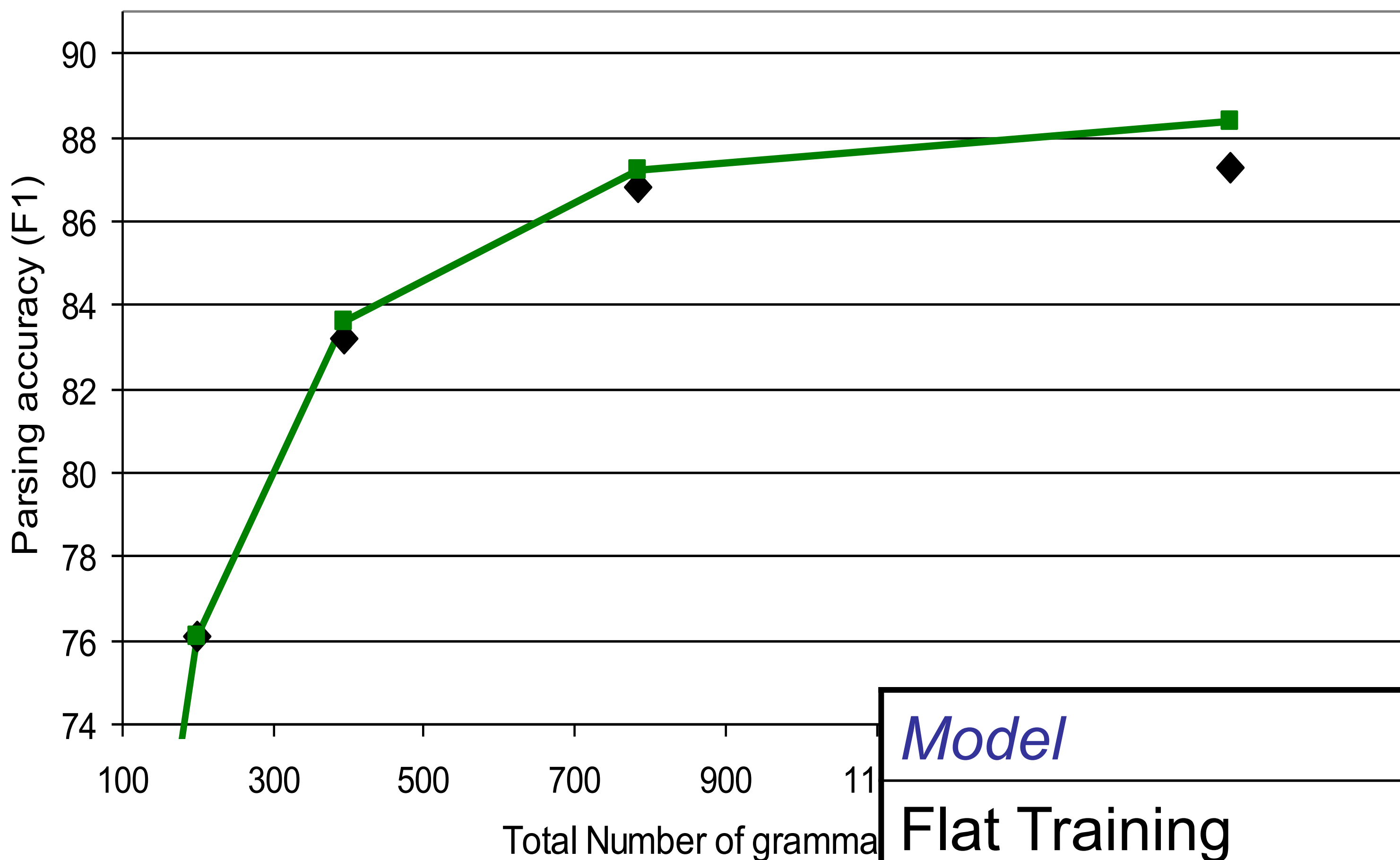
Refinement of the DT tag



Hierarchical refinement



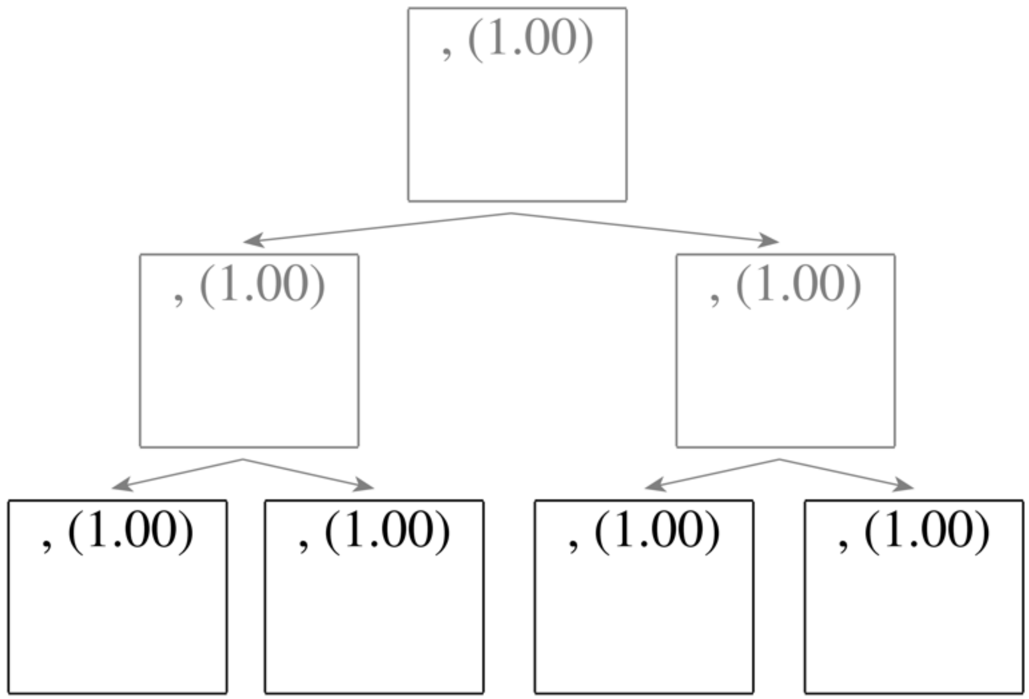
Hierarchical Estimation Results



Model	F1
Flat Training	87.3
Hierarchical Training	88.4

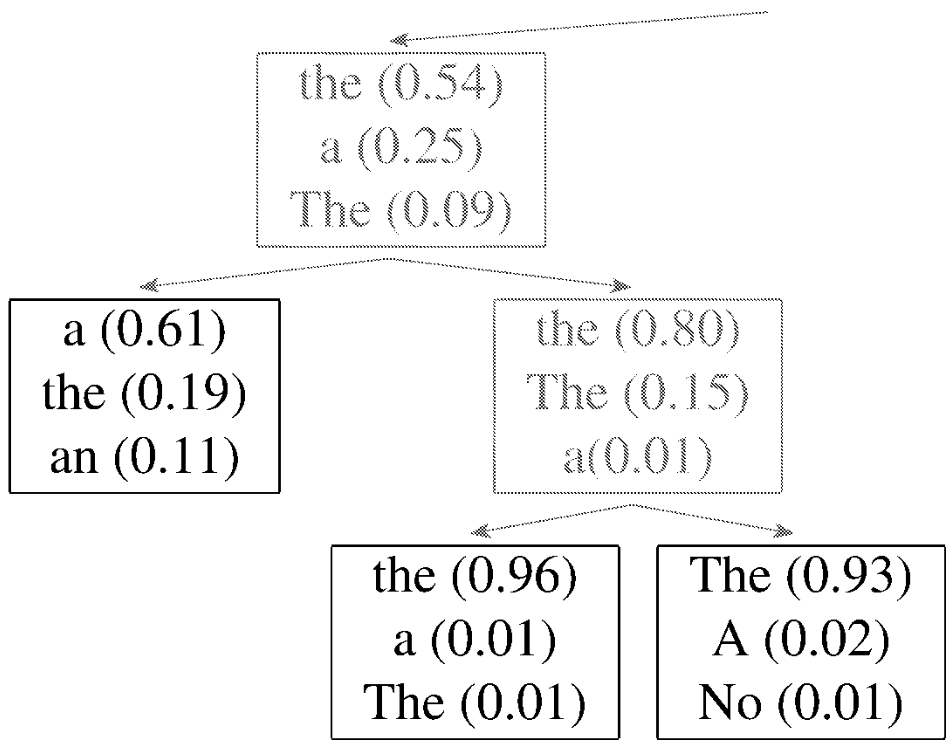
Refinement of the , tag

- Splitting all categories equally is wasteful:

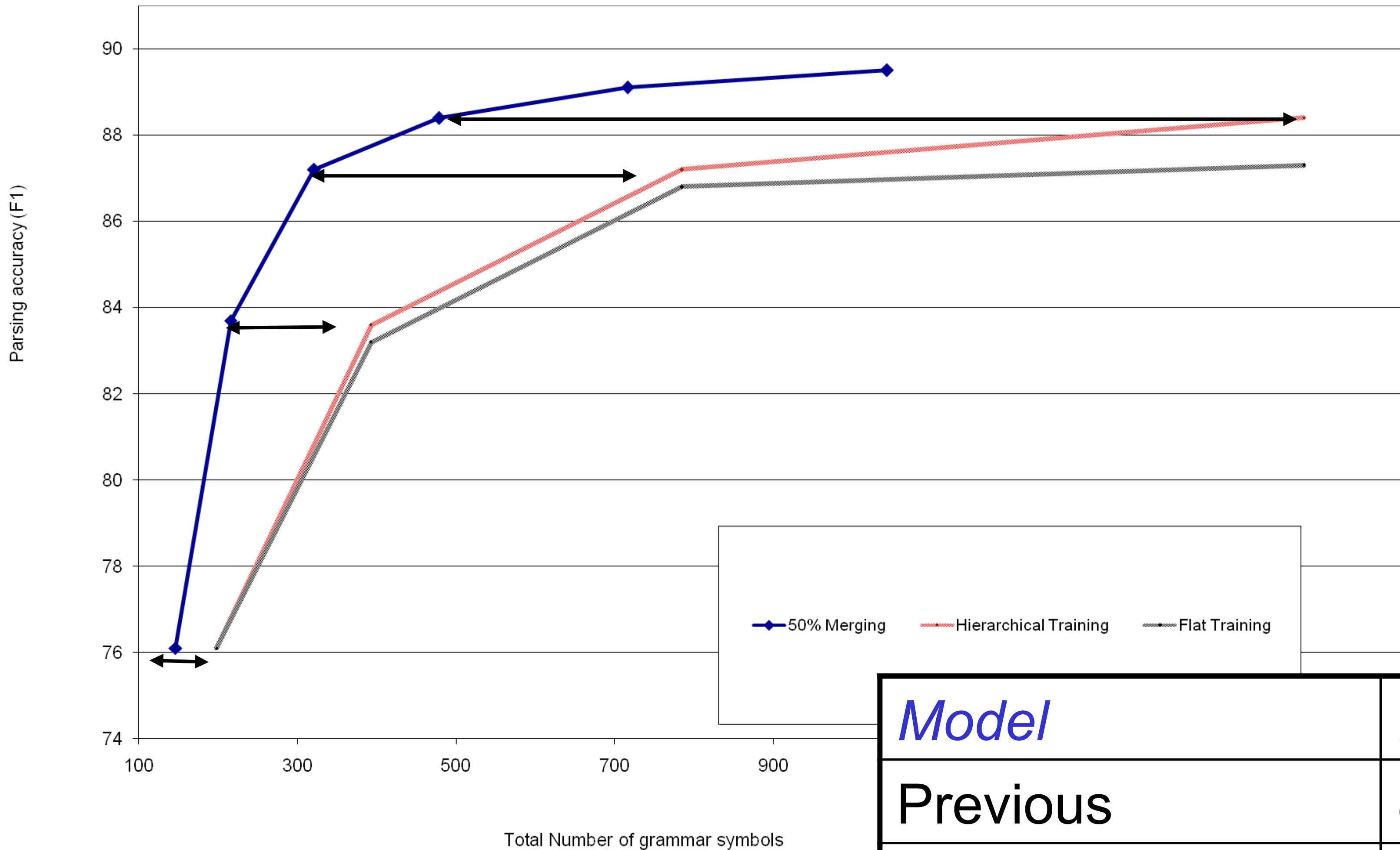


Adaptive Splitting

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

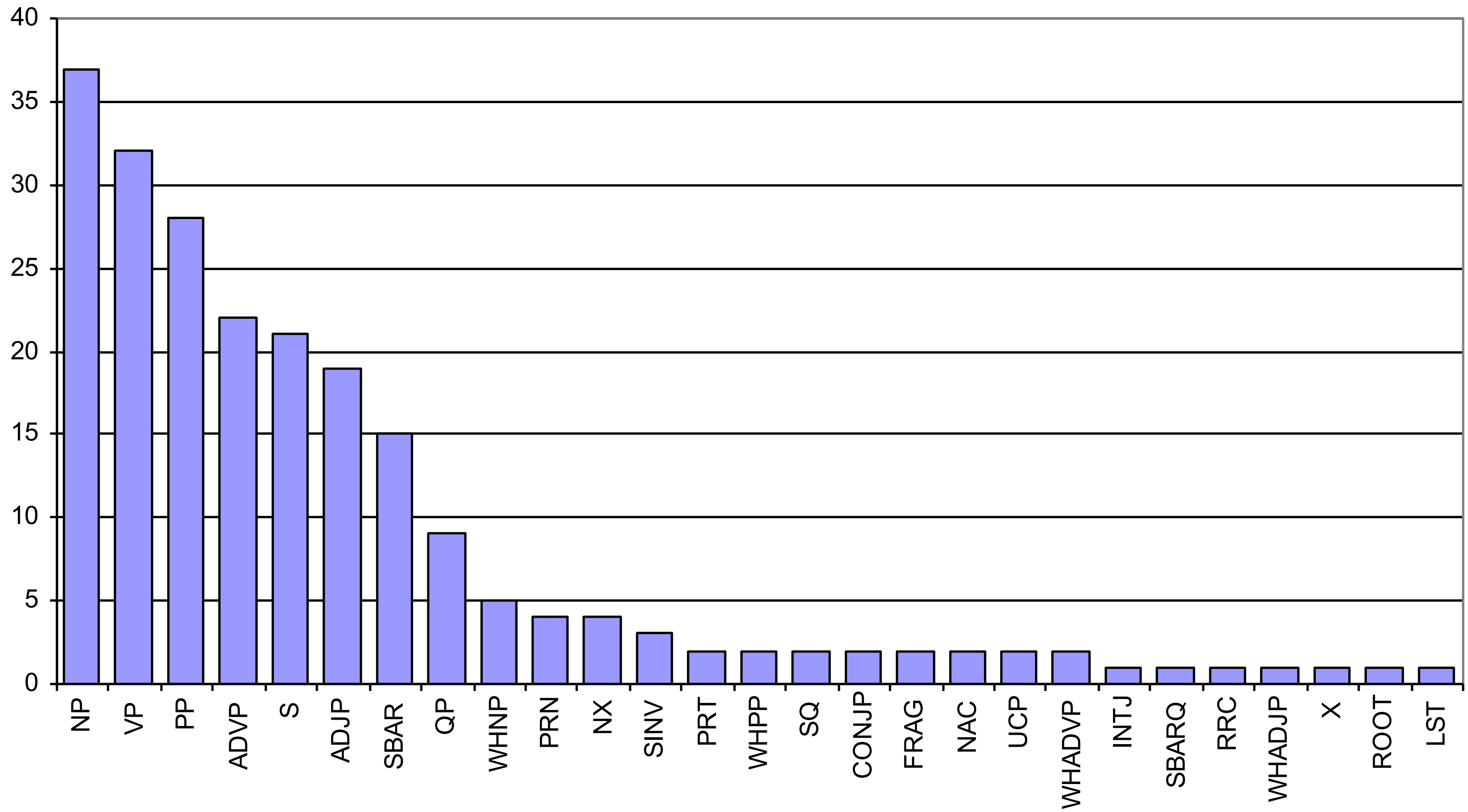


Adaptive Splitting Results

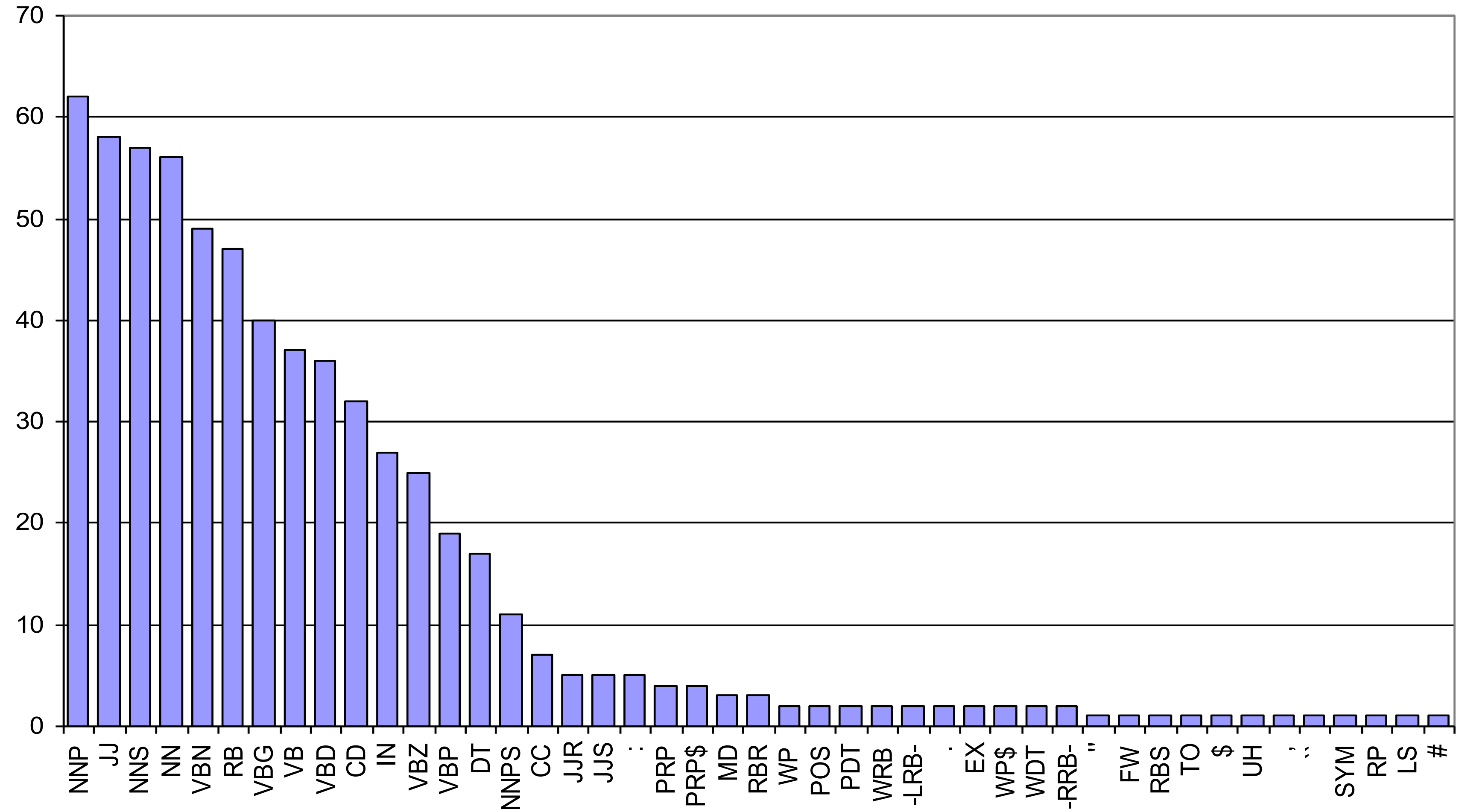


Model	*F1*
Previous	88.4
With 50% Merging	89.5

Number of Phrasal Subcategories



Number of Lexical Subcategories



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

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):

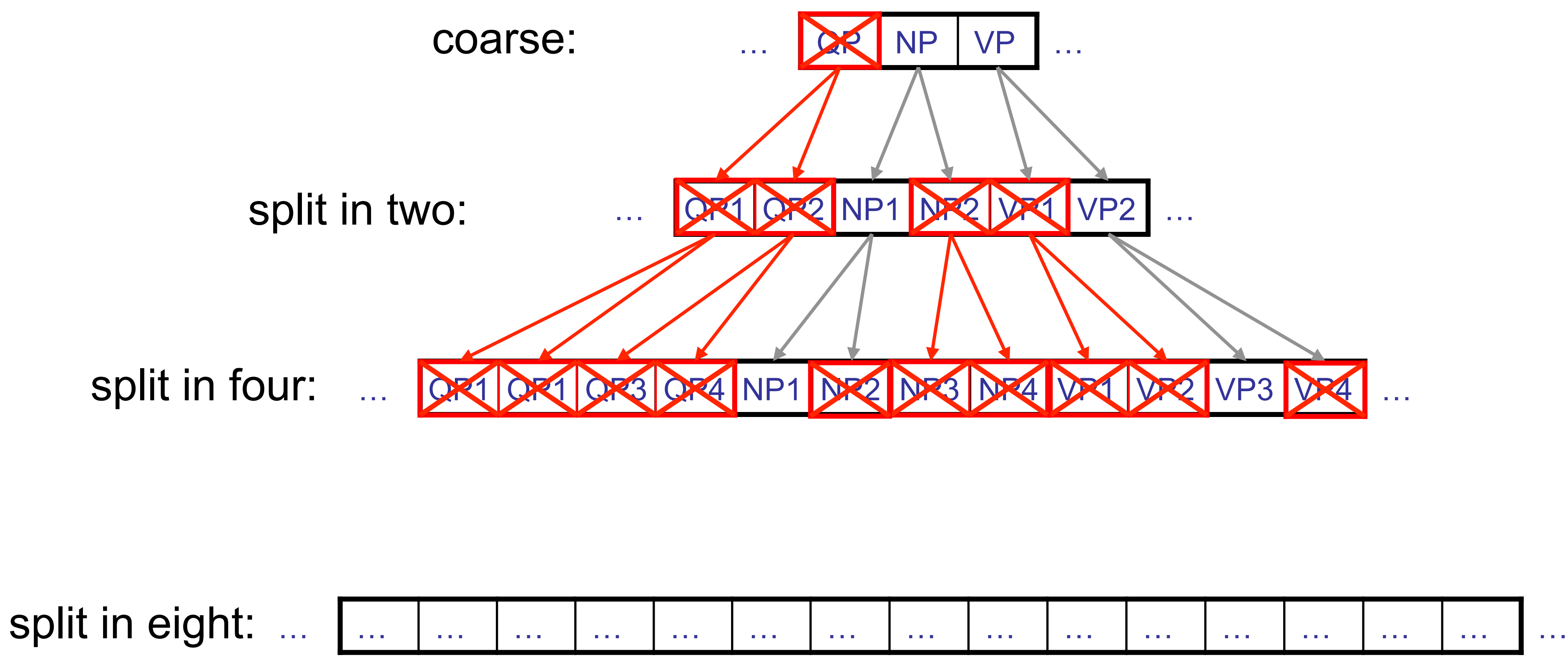
PRP-0	It	He	I
PRP-1	it	he	they
PRP-2	it	them	him

Learned Splits

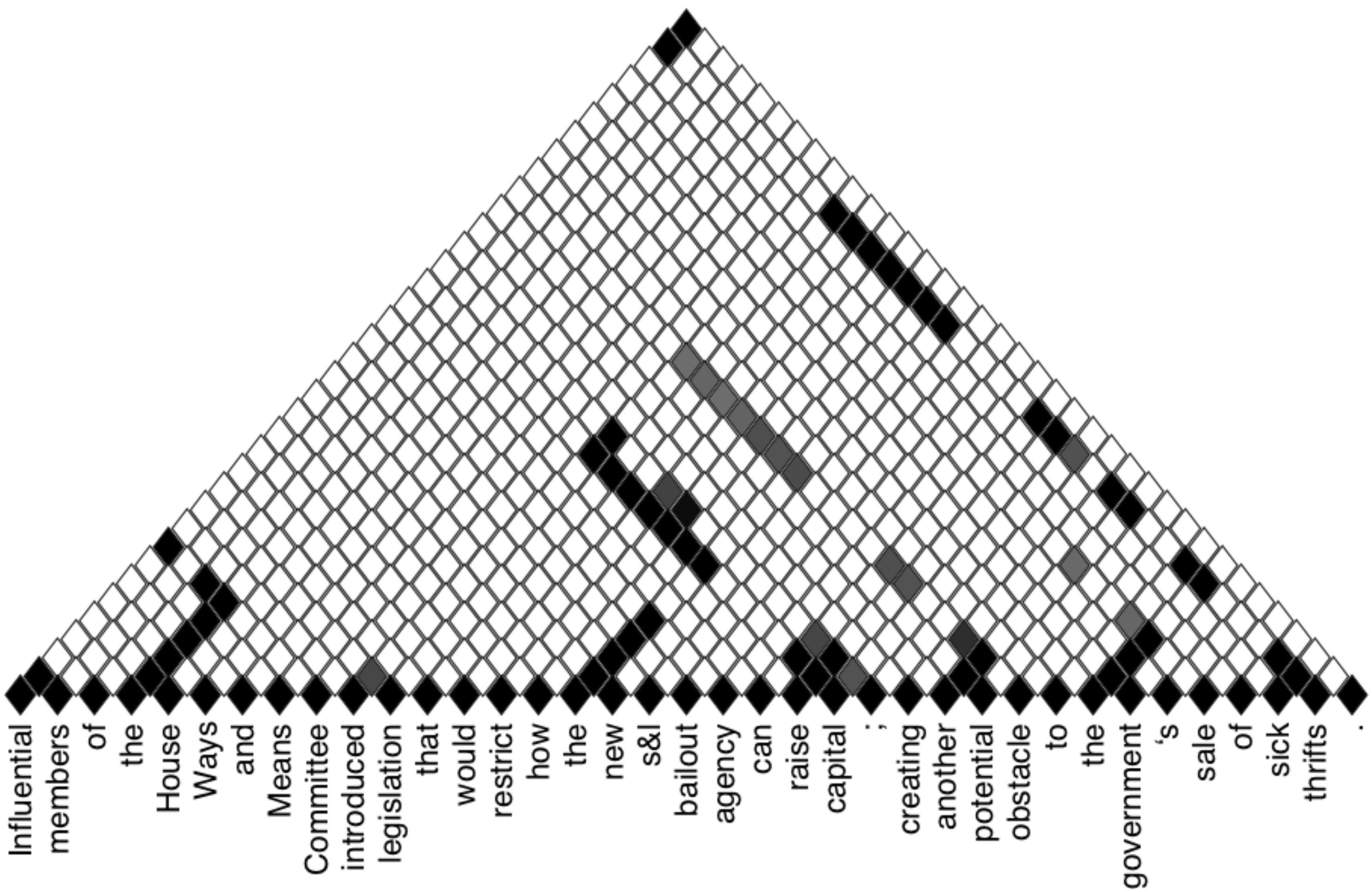
- Cardinal Numbers (CD):

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



Bracket Posteriors



Speedup

- **100x** speedup if you use the full coarse-to-fine hierarchy vs. just parsing with the finest grammar
- Parse the test set in 15 minutes – 2 sentences/second

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**

Ensemble of split-merge parsers = best cross-lingual parser for ~7 years!