

Natural Language Processing



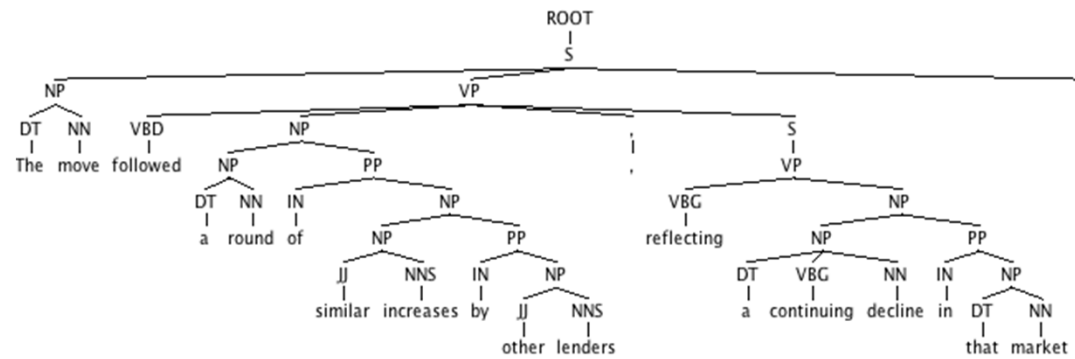
Syntax and Parsing

Dan Klein – UC Berkeley

Syntax



Parse Trees

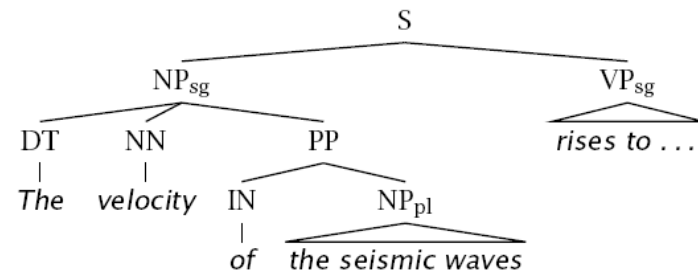


*The move followed a round of similar increases by other lenders,
reflecting a continuing decline in that market*



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
- Not the only kind of syntax...



new art critics write reviews with computers

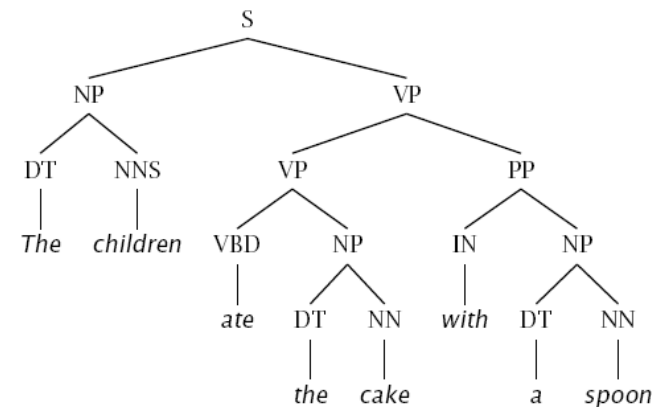


Constituency Tests

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

- Classic constituency tests:

- Substitution by *proform*
- Question answers
- Semantic grounds
 - Coherence
 - Reference
 - Idioms
- Dislocation
- Conjunction



- Cross-linguistic arguments, too



Conflicting Tests

- Constituency isn't always clear

- Units of transfer:

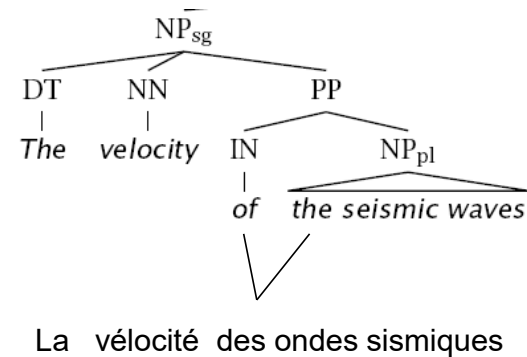
- think about ~ penser à
 - talk about ~ hablar de

- Phonological reduction:

- I will go → I'll go
 - I want to go → I wanna go
 - a le centre → au centre

- Coordination

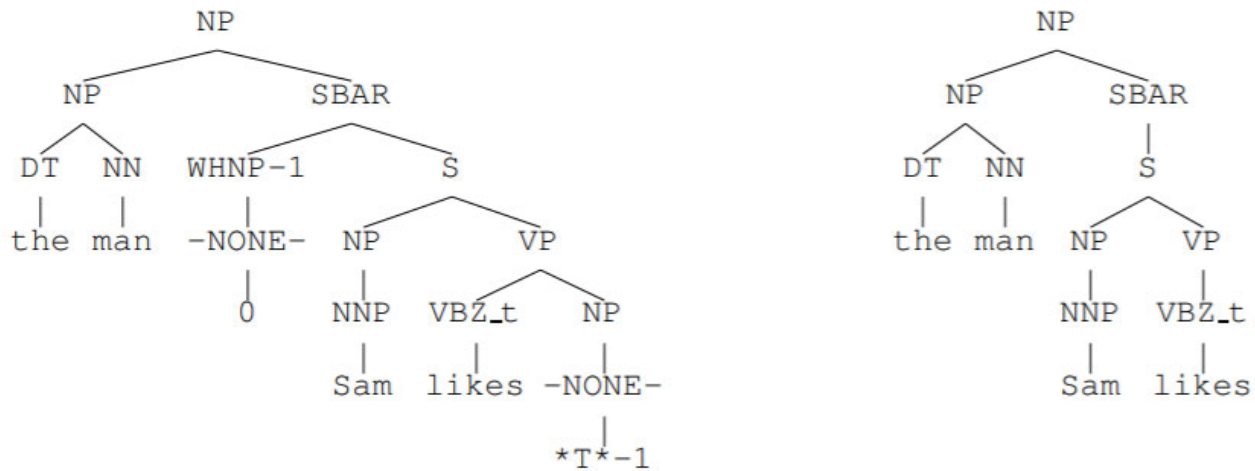
- He went to and came from the store.



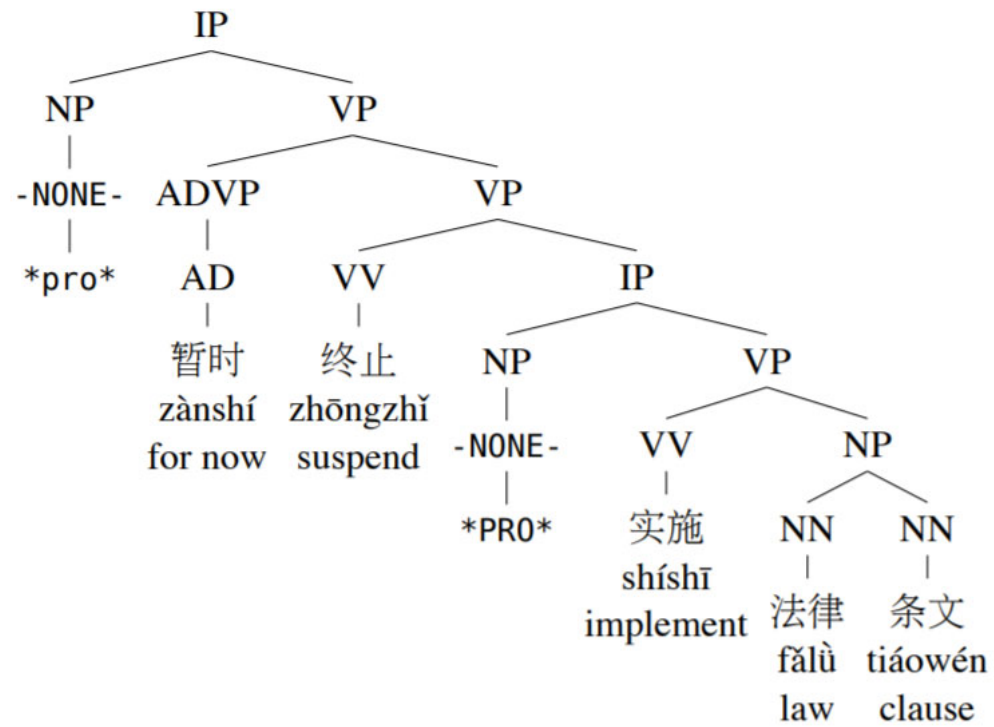


Structure Depth

- Q: Do we model deep vs surface structure?



[Example: Johnson 02]



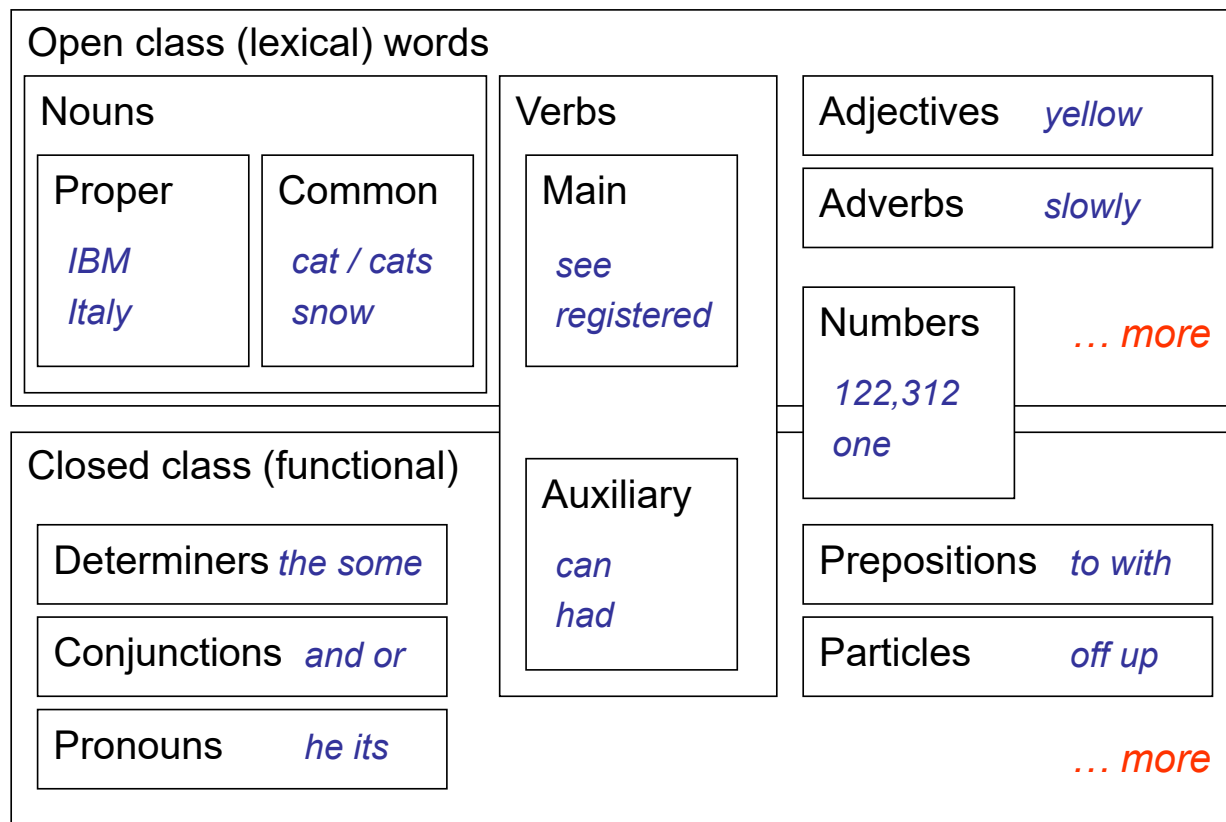
[Example: Cai et al 11]

Ambiguities



Parts-of-Speech (English)

- One basic kind of linguistic structure: syntactic word classes





Part-of-Speech Ambiguity

- Words can have multiple parts of speech

VBD VB
VBN VBZ VBP VBZ
NNP NNS NN NNS CD NN

Fed raises interest rates 0.5 percent

Mrs./NNP Shaefer/NNP never/RB got/VBD **around/RP** to/TO joining/VBG

All/DT we/PRP gotta/VBN do/VB is/VBZ go/VB **around/IN** the/DT corner/NN

Chateau/NNP Petrus/NNP costs/VBZ **around/RB** 250/CD

- Two basic sources of constraint:
 - Grammatical environment
 - Identity of the current word
- Many more possible features:
 - Suffixes, capitalization, name databases (gazetteers), etc...



Why POS Tagging?

- Historically useful in and of itself (more than you'd think)
 - Text-to-speech: record, lead
 - Lemmatization: saw[v] → see, saw[n] → saw
 - Quick-and-dirty NP-chunk detection: `grep {JJ | NN}* {NN | NNS}`
- Useful as a pre-processing step for parsing
 - Less tag ambiguity means fewer parses
 - However, some tag choices are better decided by parsers

DT NNP NN VBD VBN **IN** RP NN NNS
The Georgia branch had taken **on** loan commitments ...

DT NN IN NN **VDN** VBD NNS VBD
The average of interbank **offered** rates plummeted ...



Classical NLP: Parsing

- Write symbolic or logical rules:

Grammar (CFG)

ROOT → S	NP → NP PP
S → NP VP	VP → VBP NP
NP → DT NN	VP → VBP NP PP
NP → NN NNS	PP → IN NP

Lexicon

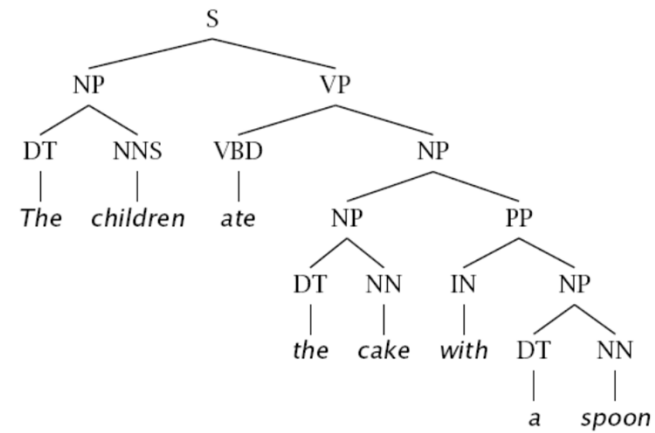
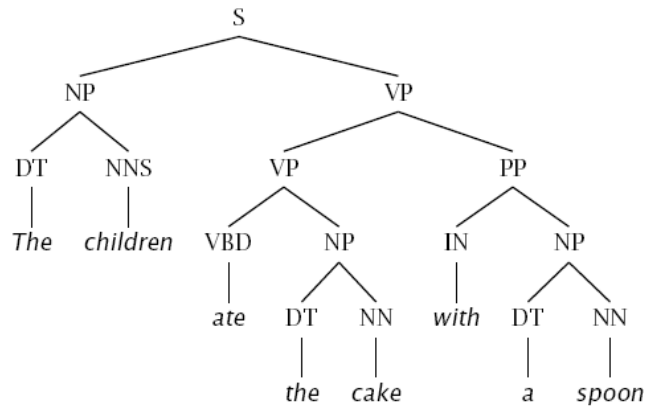
NN → interest
NNS → raises
VBP → interest
VBZ → raises

...

- 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: PP Attachment



The board approved [its acquisition] [by Royal Trustco Ltd.]
[of Toronto]
[for \$27 a share]
[at its monthly meeting].



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

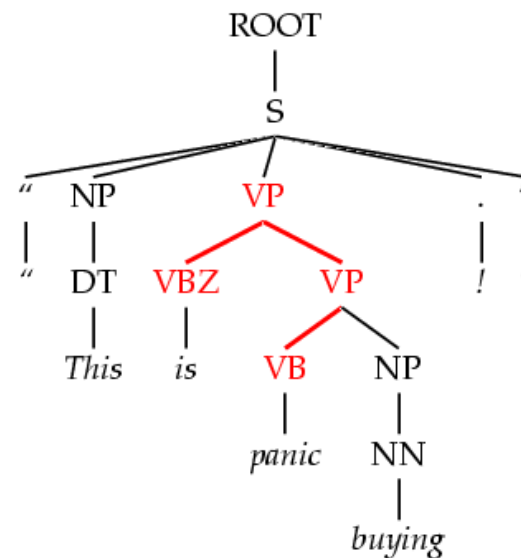
- **Modifier scope within NPs**
impractical design requirements
plastic cup holder
- **Multiple gap constructions**
The chicken is ready to eat.
The contractors are rich enough to sue.
- **Coordination scope:**
Small rats and mice can squeeze into holes or cracks in the wall.



Inaccessible Ambiguities

- *Inaccessible ambiguities*: most analyses are beyond implausible (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 so

PCFGs



Probabilistic Context-Free Grammars

- A context-free grammar is a tuple $\langle N, T, S, R \rangle$
 - 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* usually the sentence non-terminal S
 - R : the set of rules
 - Of the form $X \rightarrow Y_1 Y_2 \dots Y_k$, with $X, Y_i \in N$
 - Examples: $S \rightarrow NP VP$, $VP \rightarrow VP CC VP$
 - Also called rewrites, productions, or local trees
- A PCFG adds:
 - A top-down production probability per rule $P(Y_1 Y_2 \dots Y_k \mid X)$



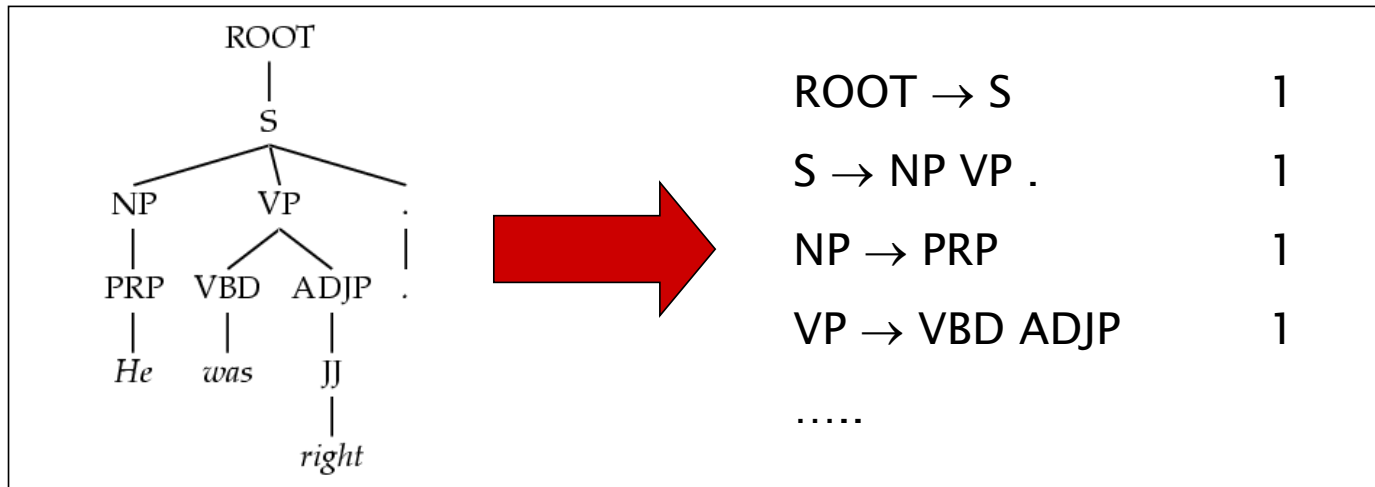
Treebank Sentences

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



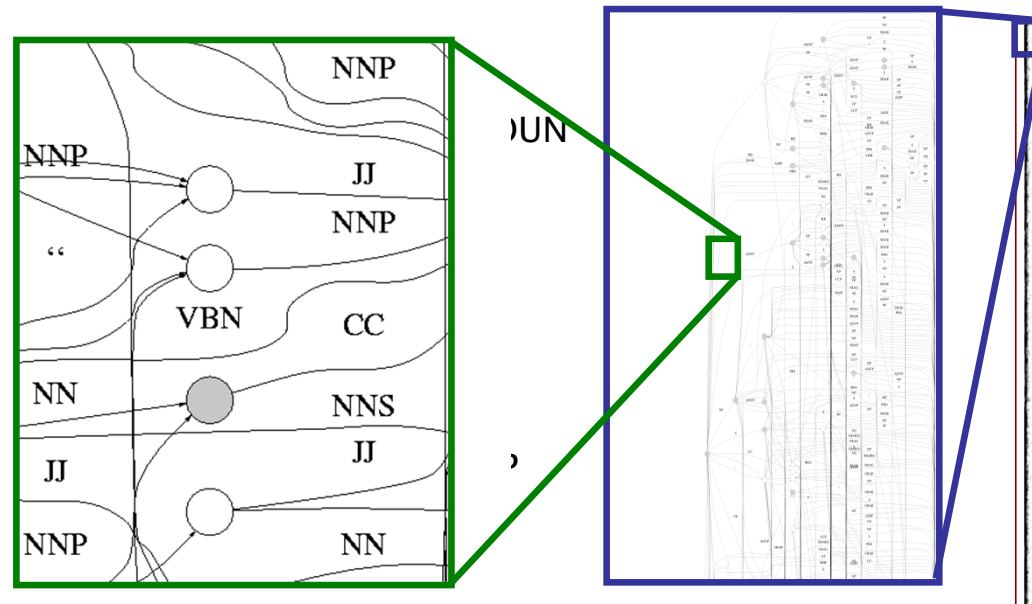
- Better results by enriching the grammar (e.g., lexicalization).
- Can also get state-of-the-art parsers without lexicalization.



Trebank 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

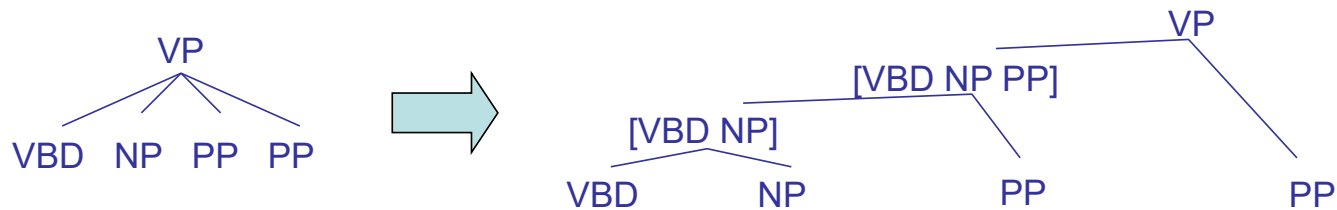
NP





Chomsky Normal Form

- **Chomsky normal form:**
 - All rules of the form $X \rightarrow YZ$ or $X \rightarrow w$
 - 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
 - Transformations don’t always preserve tree scores
- Makes parsing algorithms simpler!

CKY Parsing



A Recursive Parser

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

- Will this parser work?
- Why or why not?
- Memory requirements?
- Time requirements?



A Memoized Parser

- One small change:

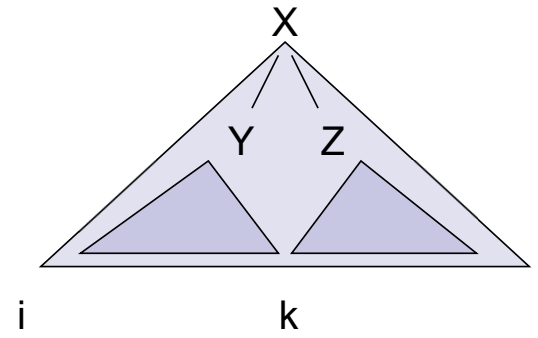
```
bestScore(X,i,j)
  if (scores[X][i][j] == null)
    if (j = i+1)
      score = tagScore(X,s[i])
    else
      score = max score(X->YZ) *
                bestScore(Y,i,k) *
                bestScore(Z,k,j)
    scores[X][i][j] = score
  return scores[X][i][j]
```



A Bottom-Up Parser (CKY)

- Can also organize things bottom-up

```
bestScores()  
  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

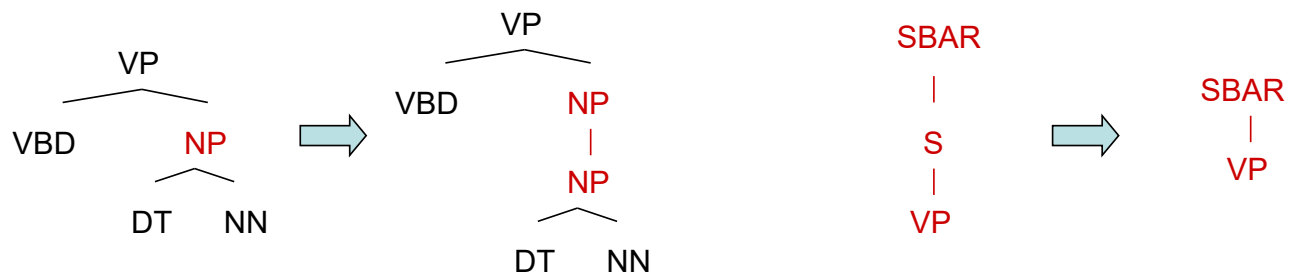
- Maybe we can handle unary rules?

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



CNF + 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)
```

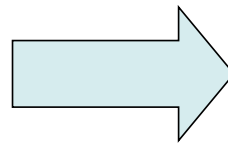
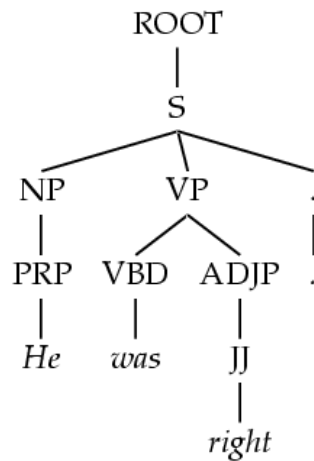
```
bestScoreU(X,i,j,s)
    if (j = i+1)
        return tagScore(X,s[i])
    else
        return max max score(X->Y) *
                    bestScoreB(Y,i,j)
```

Learning PCFGs



Treebank PCFGs [Charniak 96]

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

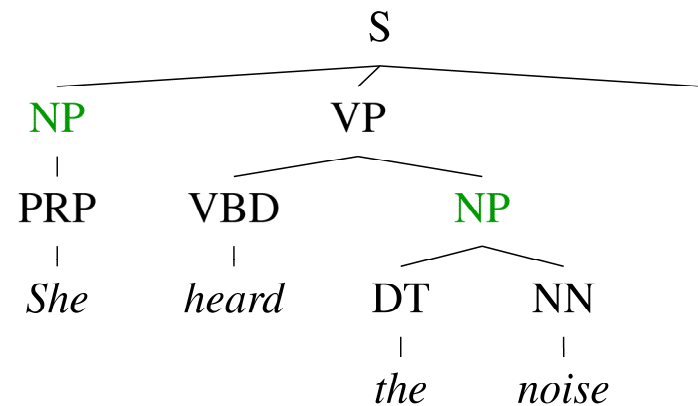


ROOT → S 1
S → NP VP . 1
NP → PRP 1
VP → VBD ADJP 1
.....

<i>Model</i>	<i>F1</i>
Baseline	72.0



Conditional Independence?

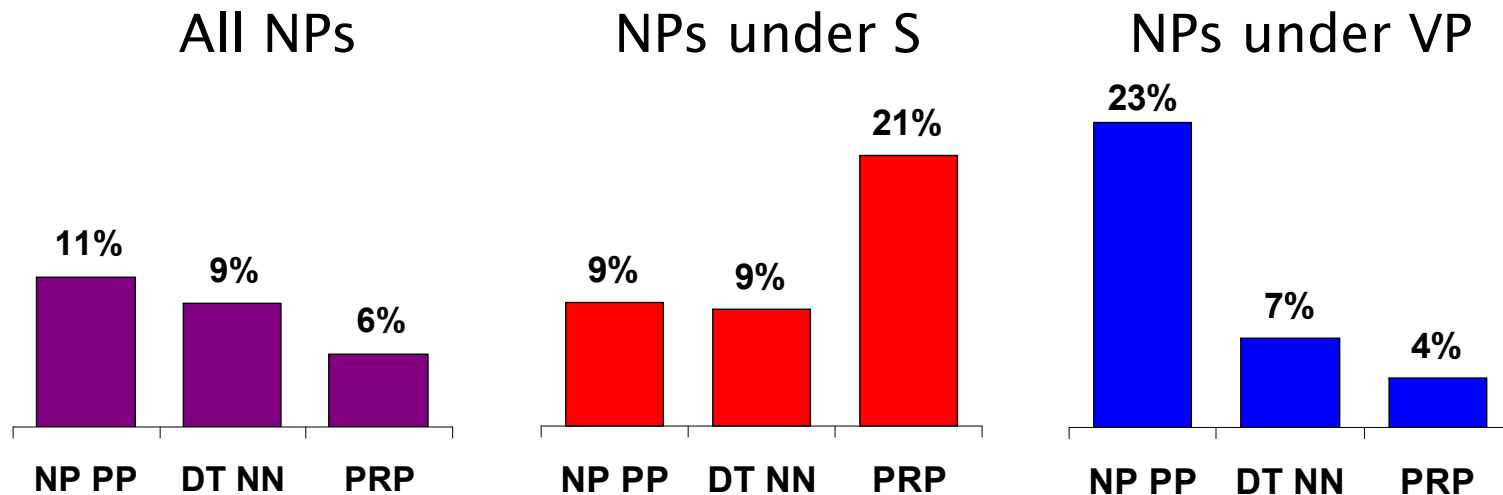


- Not every NP expansion can fill every NP slot
 - A grammar with symbols like “NP” won’t be context-free
 - Statistically, conditional independence too strong



Non-Independence

- Independence assumptions are often too strong.

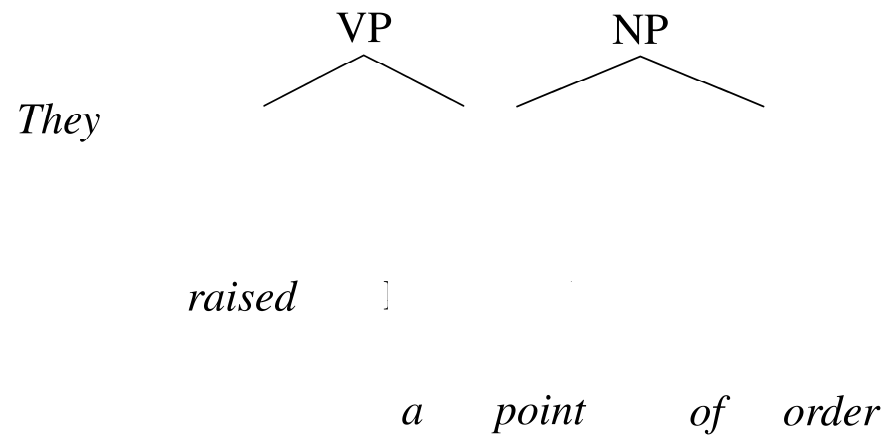


- Example: the expansion of an NP is highly dependent on the parent of the NP (i.e., subjects vs. objects).
- Also: the subject and object expansions are correlated!



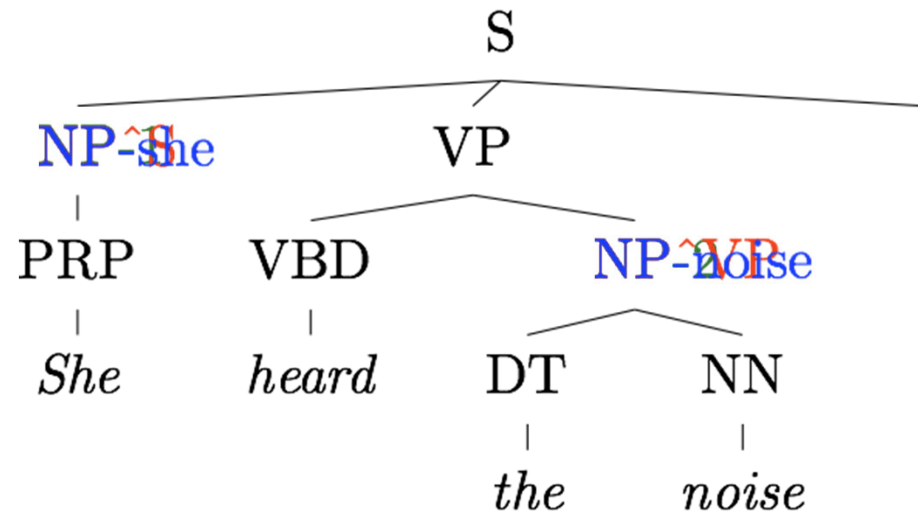
Grammar Refinement

- Example: PP attachment





Grammar Refinement

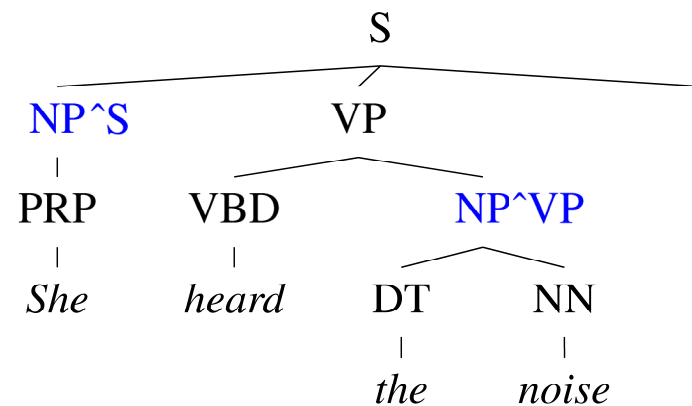


- Structure Annotation [Johnson '98, Klein&Manning '03]
- Lexicalization [Collins '99, Charniak '00]
- Latent Variables [Matsuzaki et al. 05, Petrov et al. '06]

Structural Annotation



The Game of Designing a Grammar

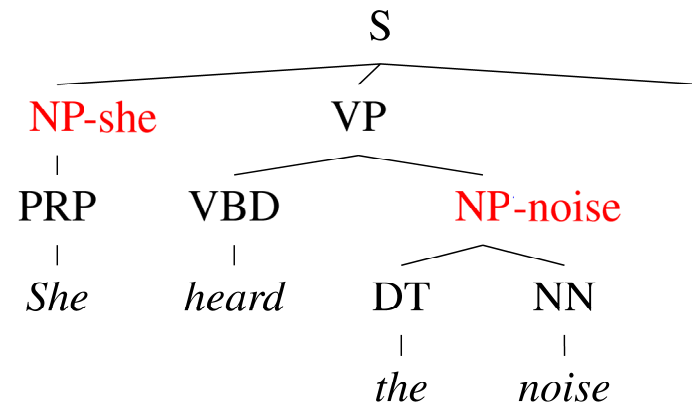


- Annotation refines base treebank symbols to improve statistical fit of the grammar
 - Structural annotation

Lexicalization



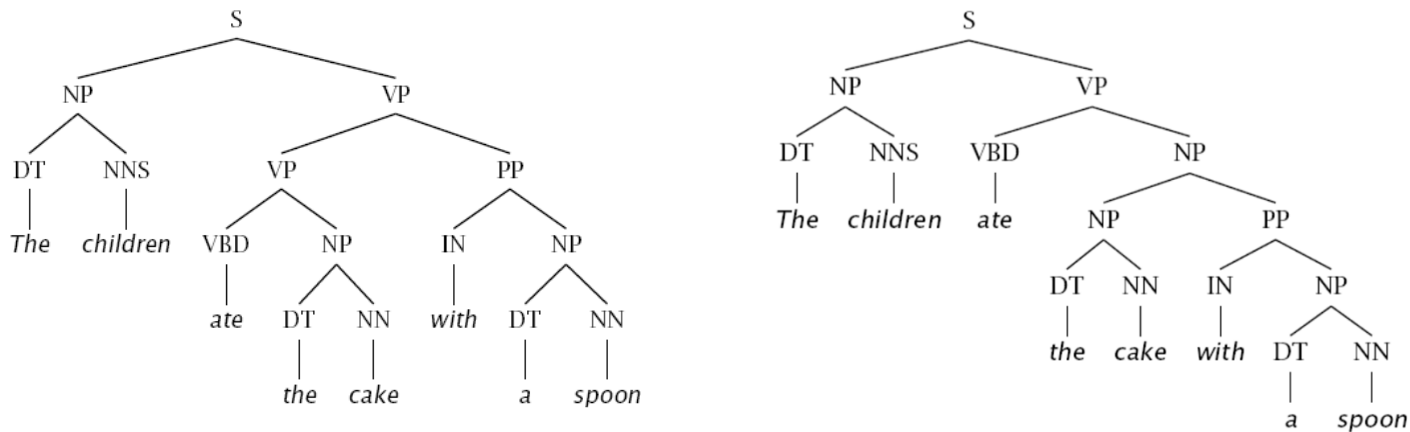
The Game of Designing a Grammar



- 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]



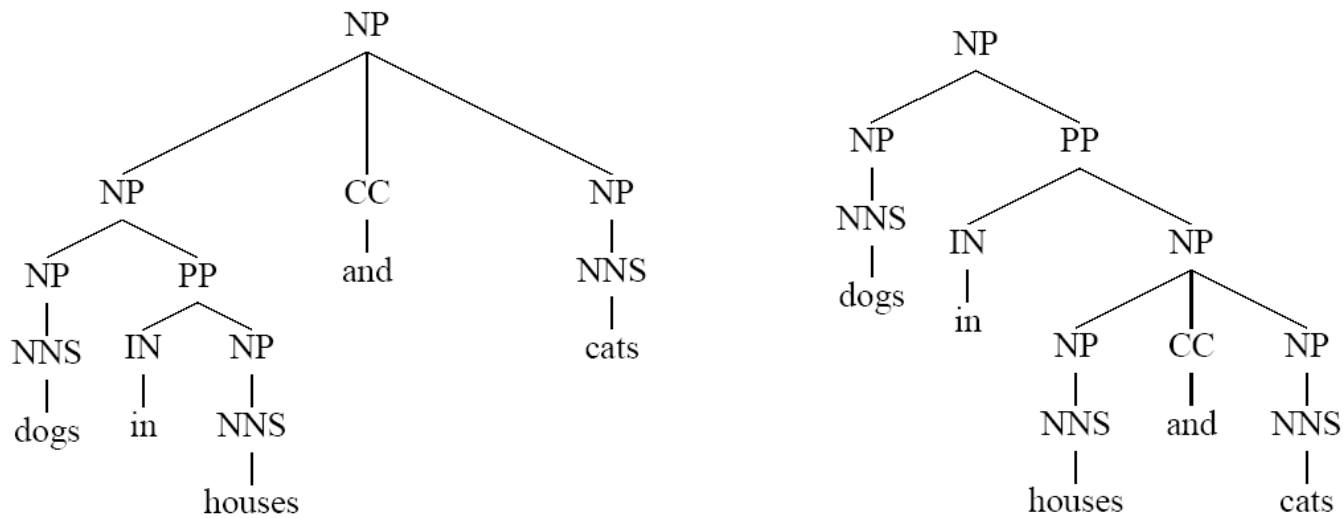
Problems with PCFGs



- If we do no annotation, these trees differ only in one rule:
 - $VP \rightarrow VP PP$
 - $NP \rightarrow NP PP$
- 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

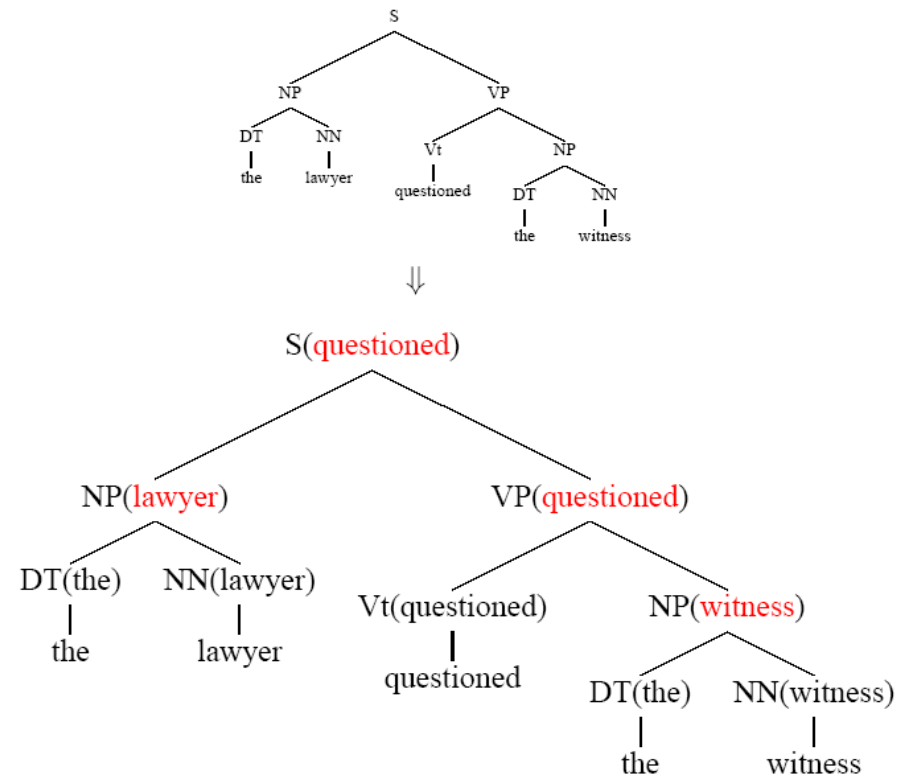


- What's different between basic PCFG scores here?
- What (lexical) correlations need to be scored?



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

$VP(\text{saw}) \rightarrow VBD(\text{saw}) NP-C(\text{her}) NP(\text{today})$

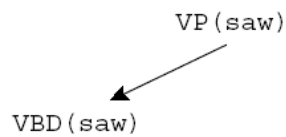
- Never going to get these atomically off of a treebank
- Solution: break up derivation into smaller steps



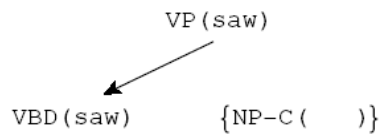


Lexical Derivation Steps

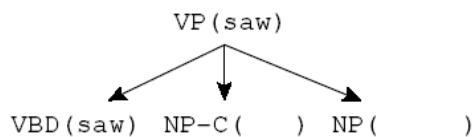
- A derivation of a local tree [Collins 99]



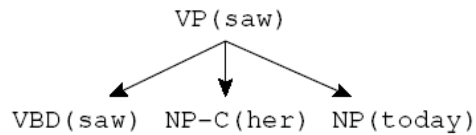
Choose a head tag and word



Choose a complement bag



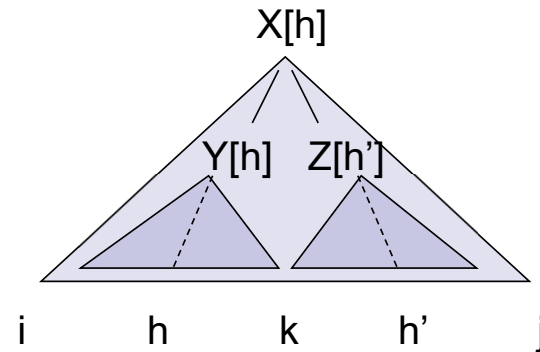
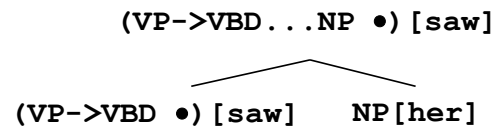
Generate children (incl. adjuncts)



Recursively derive children



Lexicalized CKY



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',X->YZ}

 bestScore(Y,i,k,h) *

 bestScore(Z,k,j,h')

max score(X[h]->Y[h'] Z[h]) *
 _{k,h',X->YZ}

 bestScore(Y,i,k,h') *

 bestScore(Z,k,j,h)



Results

- Some results
 - Collins 99 – 88.6 F1 (generative lexical)
 - Charniak and Johnson 05 – 89.7 / 91.3 F1 (generative lexical / reranked)
 - Petrov et al 06 – 90.7 F1 (generative unlexical)
 - McClosky et al 06 – 92.1 F1 (gen + rerank + self-train)
- However
 - Bilexical counts rarely make a difference (why?)
 - Gildea 01 – Removing bilexical counts costs < 0.5 F1

Natural Language Processing

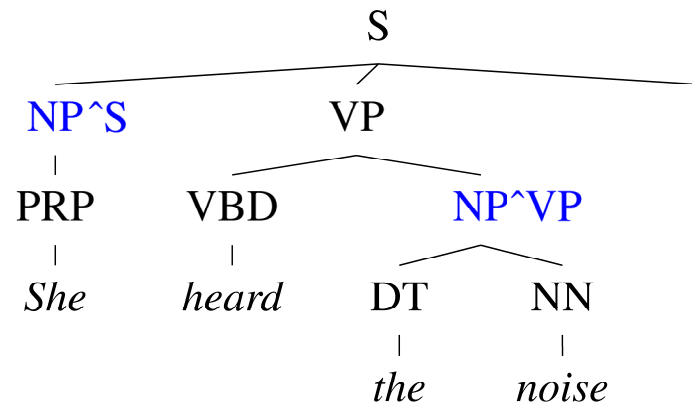


Syntax and Parsing

Dan Klein – UC Berkeley



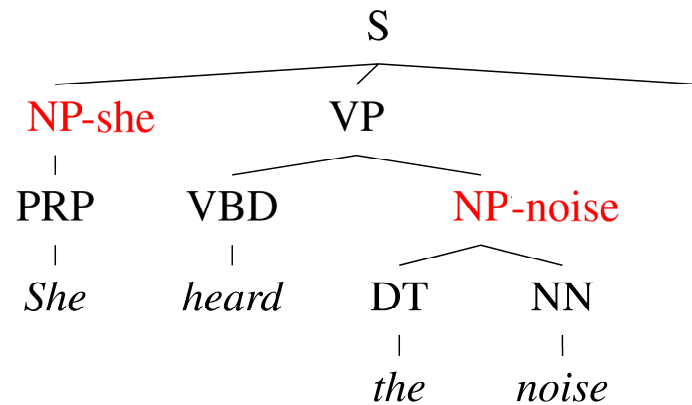
The Game of Designing a Grammar



- Annotation refines base treebank symbols to improve statistical fit of the grammar
 - Parent annotation [Johnson '98]



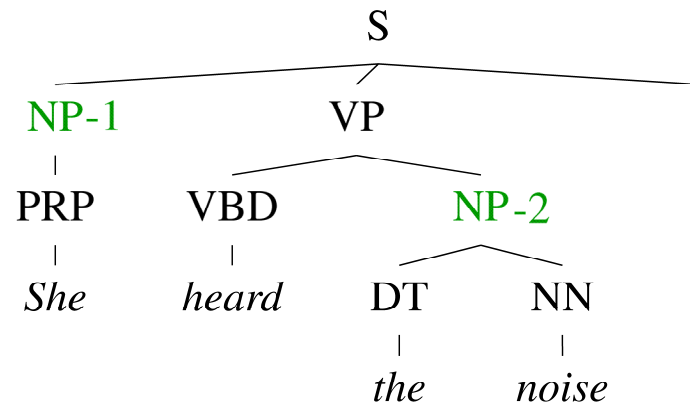
The Game of Designing a Grammar



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



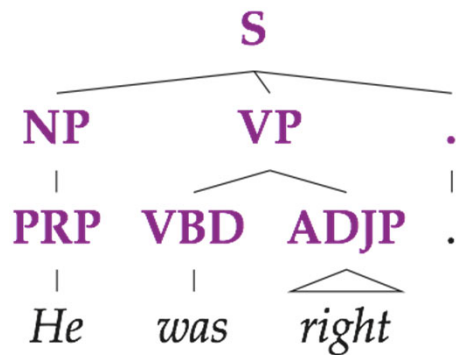
The Game of Designing a Grammar



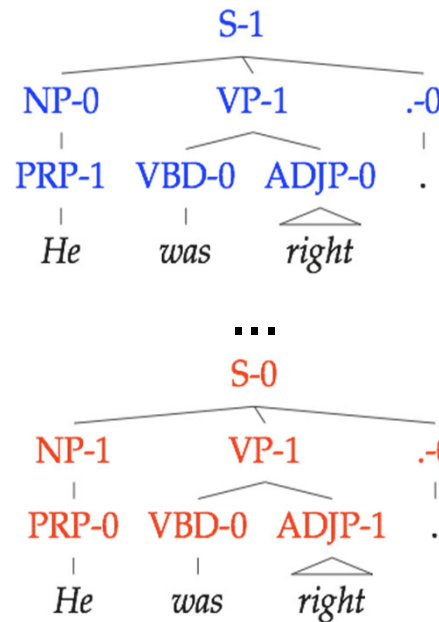
- Annotation refines base treebank symbols to improve statistical fit of the grammar
 - Parent annotation [Johnson '98]
 - Head lexicalization [Collins '99, Charniak '00]
 - Automatic clustering?



Latent Variable Grammars



Parse Tree T
Sentence w



Derivations $t : T$



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

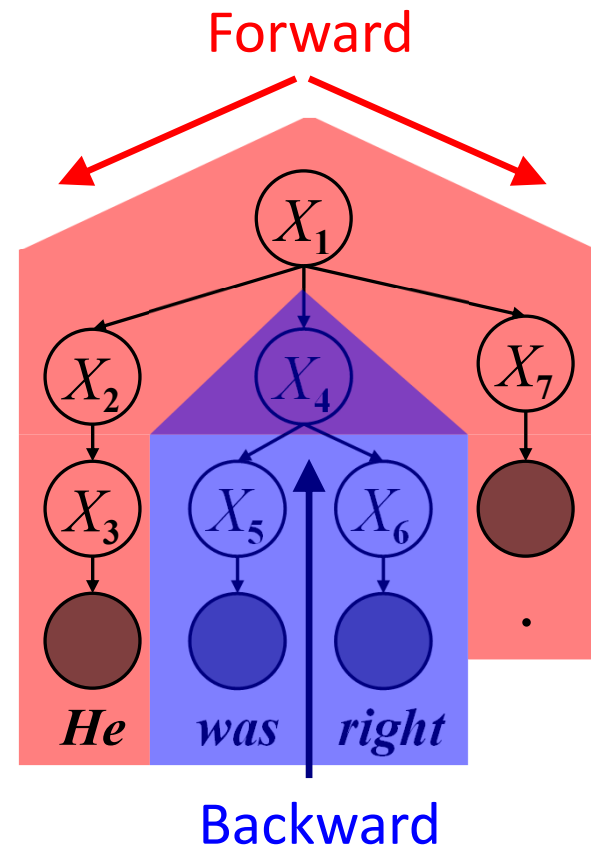
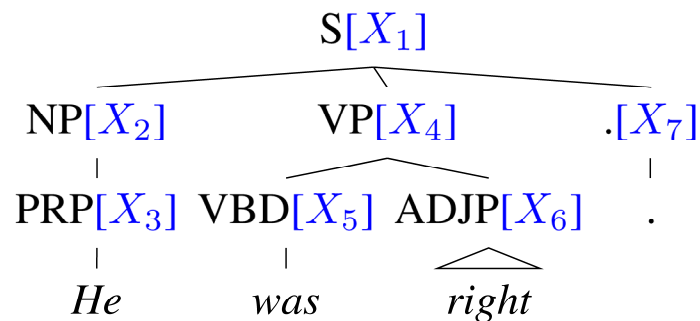
Parameters θ



Learning Latent Annotations

EM algorithm:

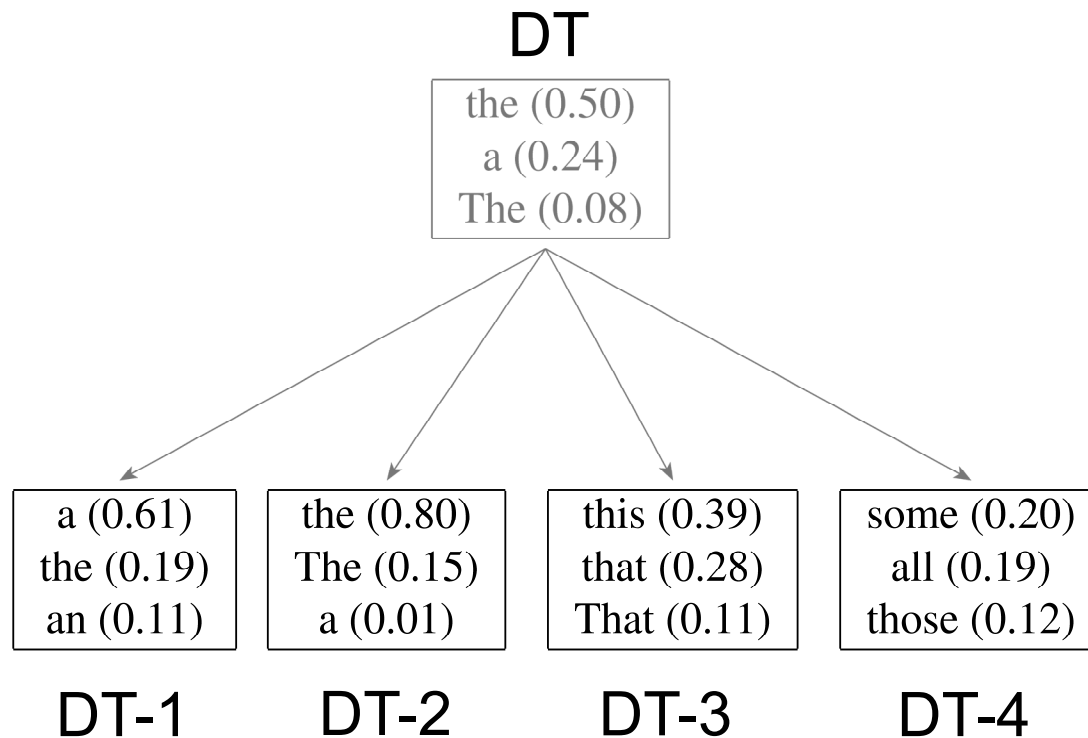
- Brackets are known
- Base categories are known
- Only induce subcategories



Just like Forward-Backward for HMMs.

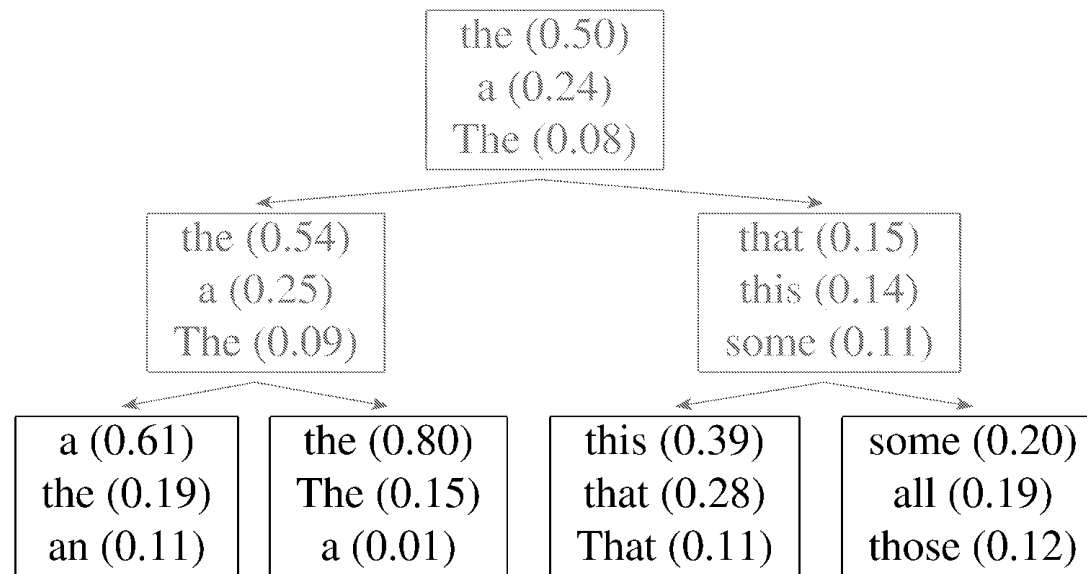


Refinement of the DT tag



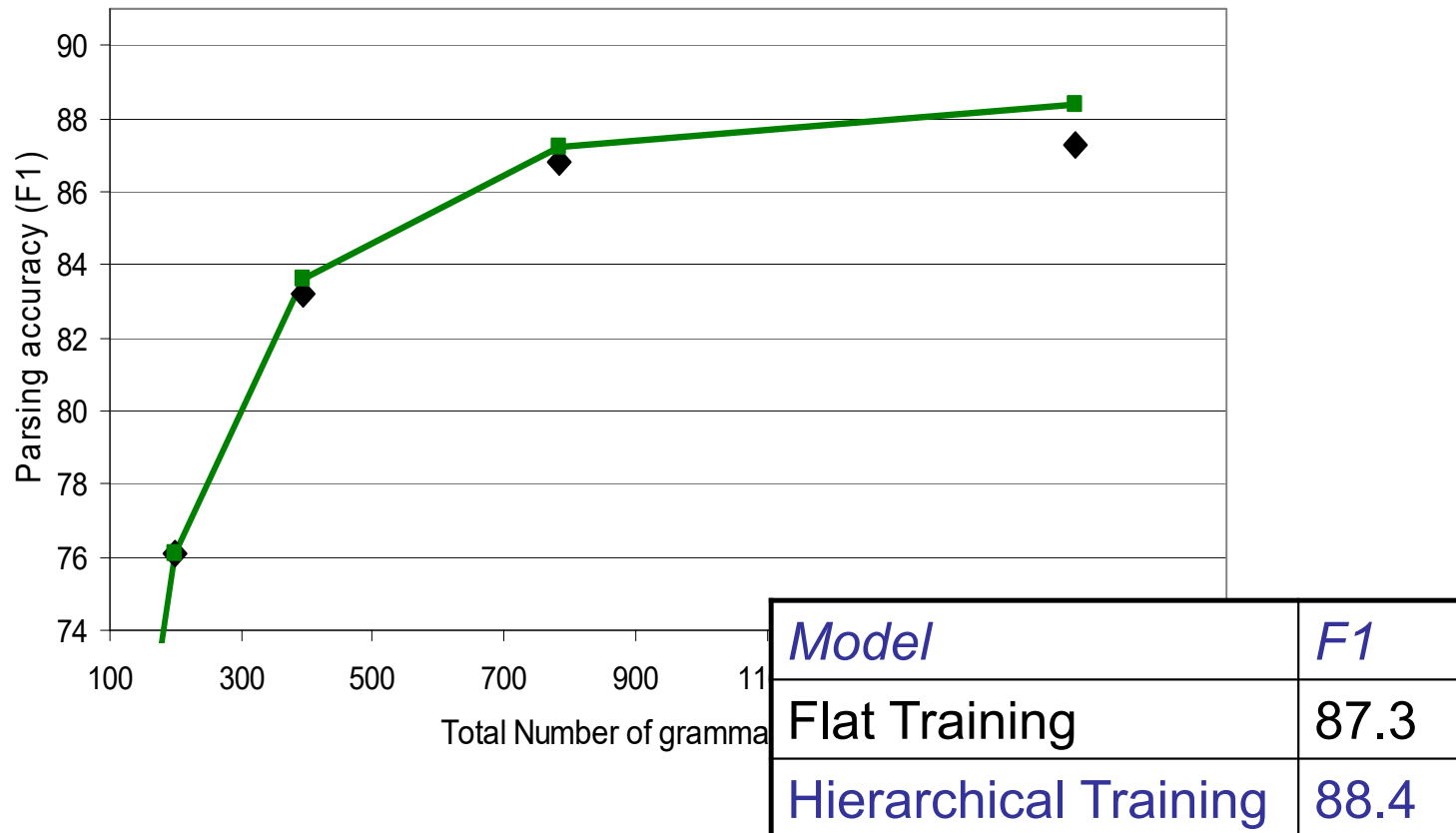


Hierarchical refinement





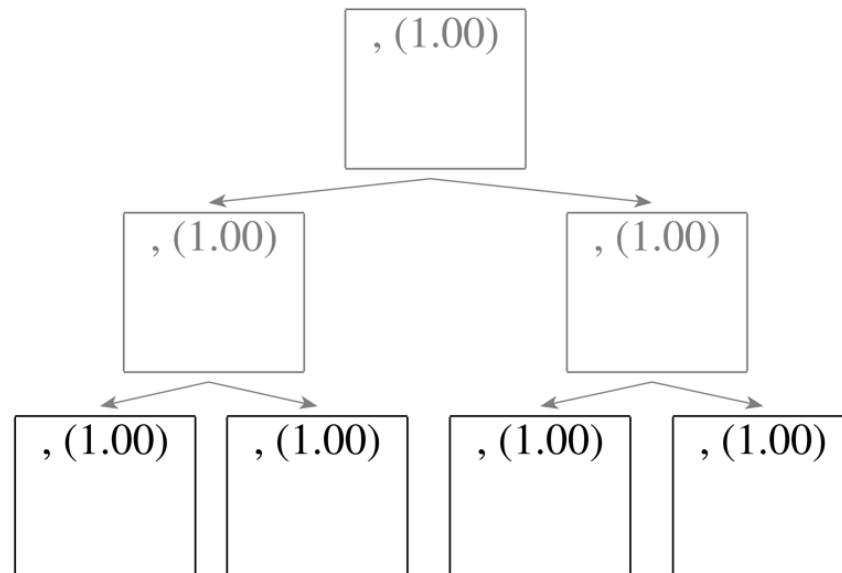
Hierarchical Estimation Results





Refinement of the , tag

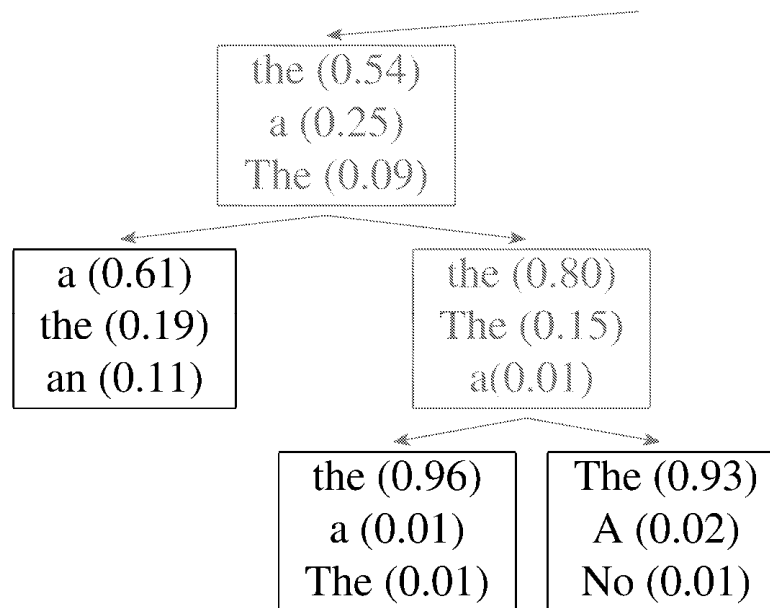
- Splitting all categories equally is wasteful:





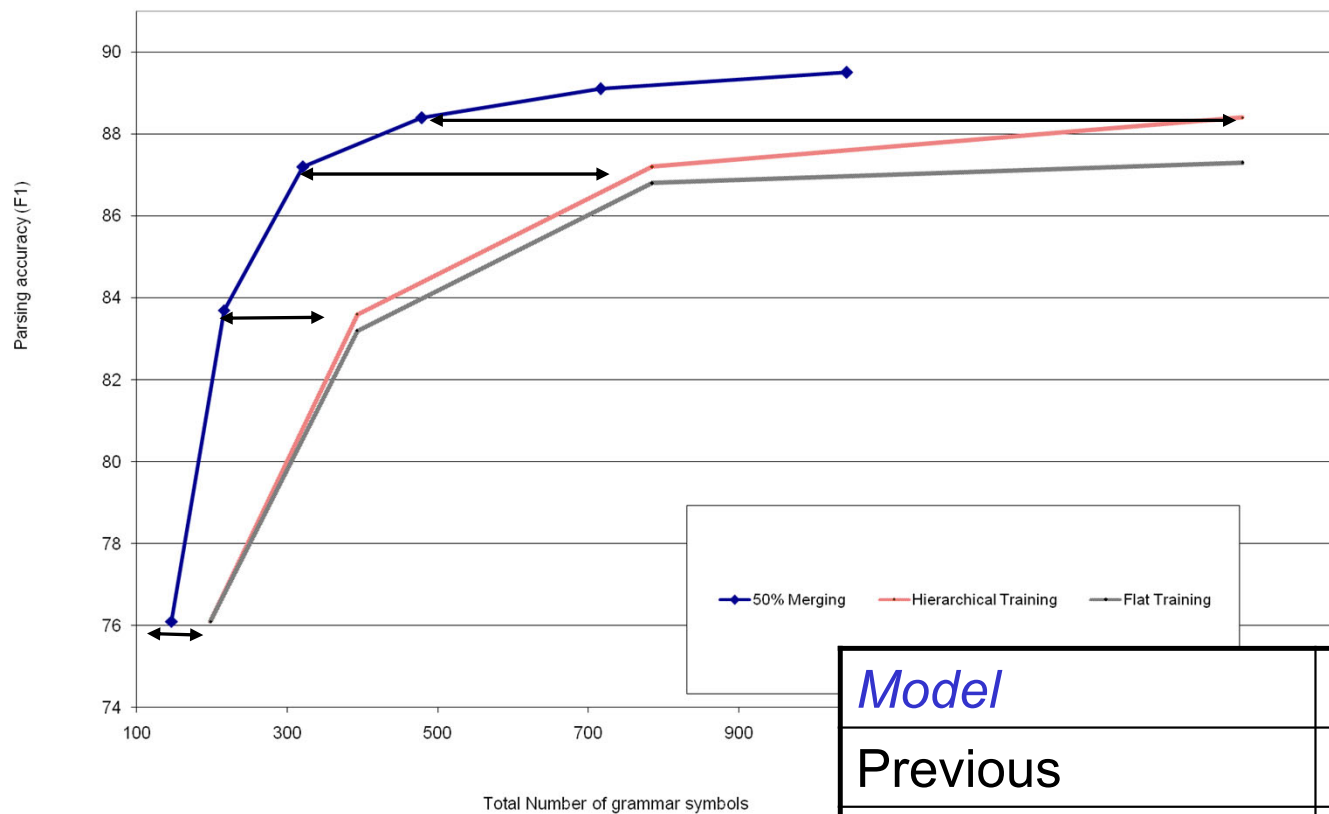
Adaptive Splitting

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





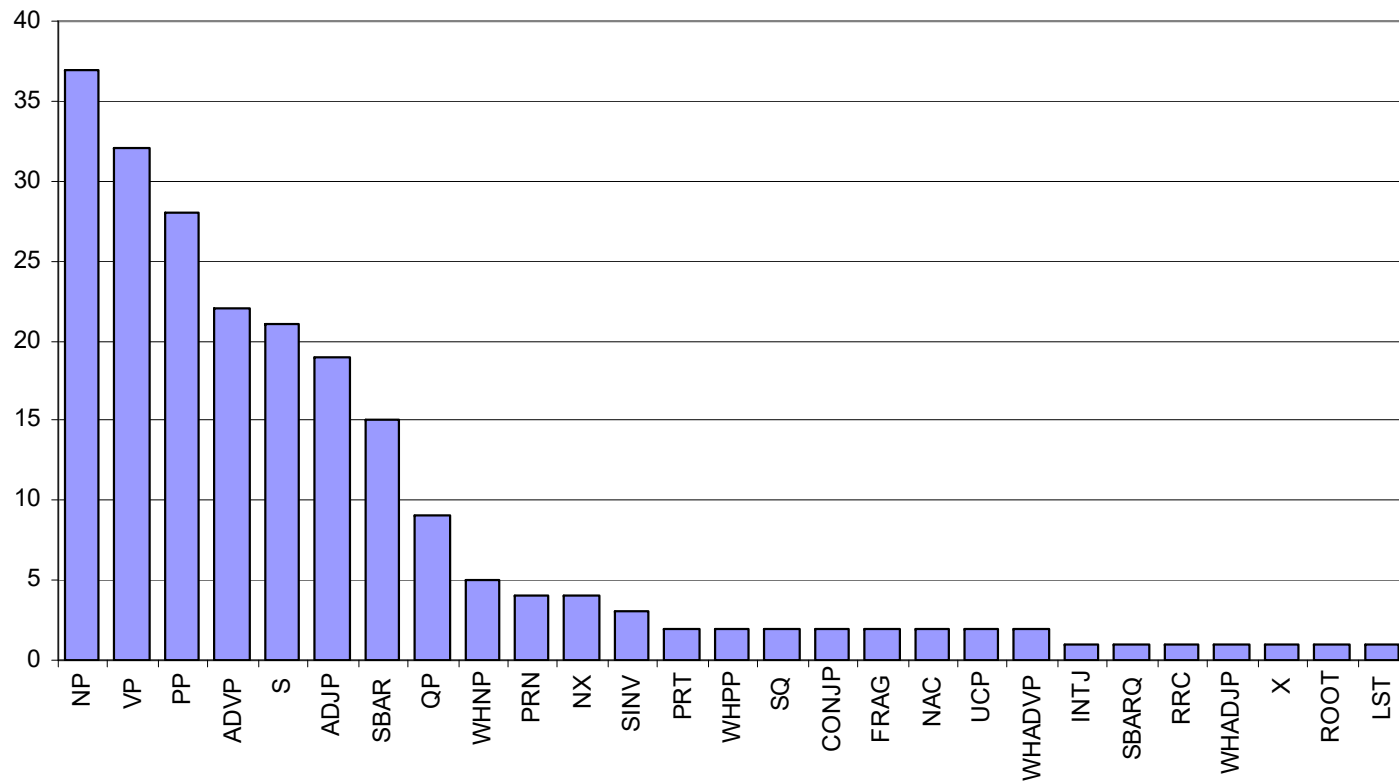
Adaptive Splitting Results



<i>Model</i>	<i>F1</i>
Previous	88.4
With 50% Merging	89.5

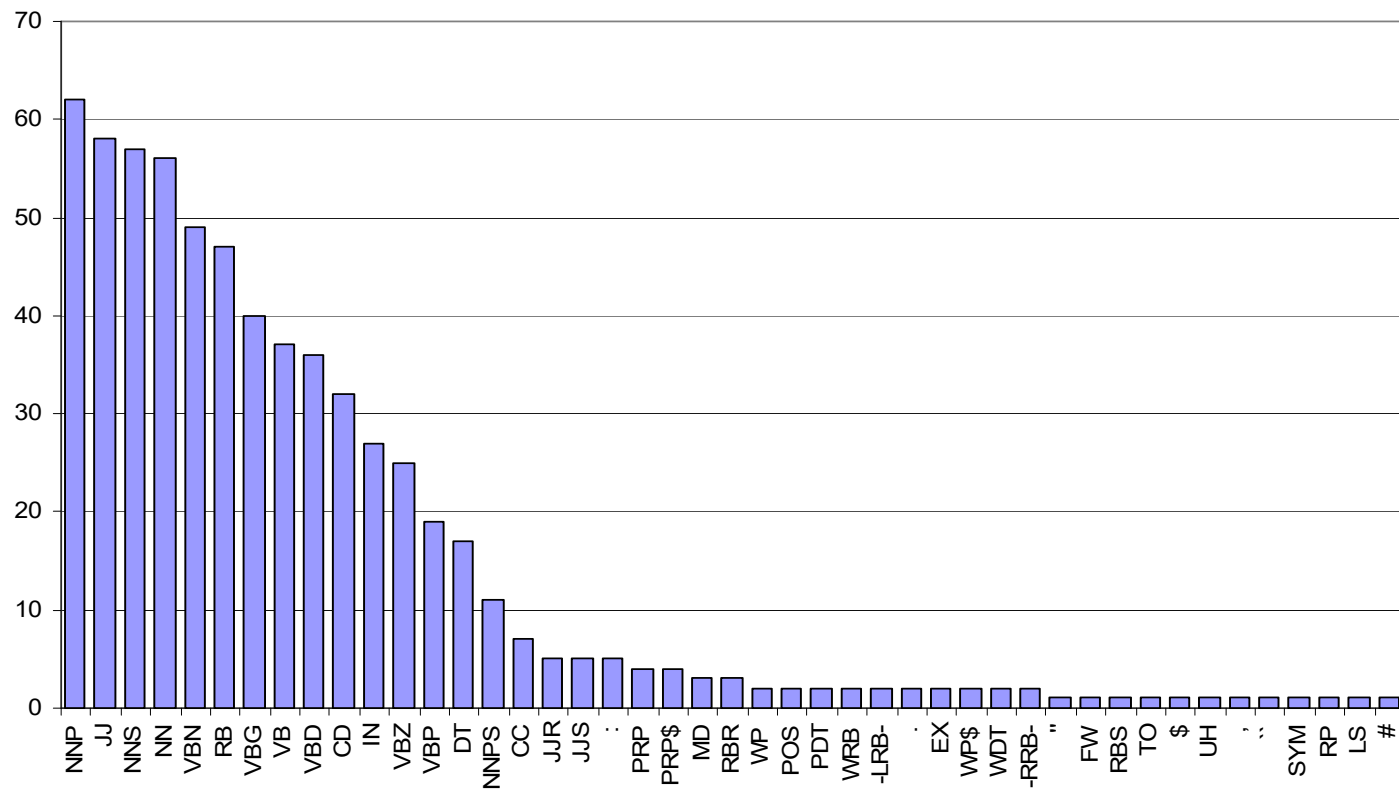


Number of Phrasal Subcategories





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

- Personal pronouns (PRP):

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



Learned Splits

- Relative adverbs (RBR):

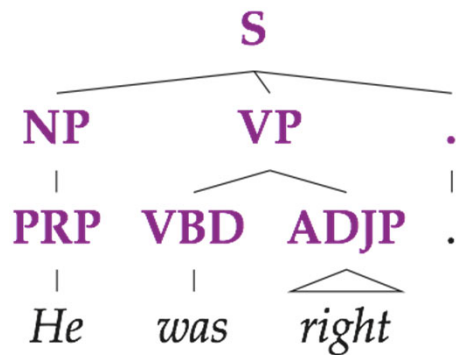
RBR-0	further	lower	higher
RBR-1	more	less	More
RBR-2	earlier	Earlier	later

- Cardinal Numbers (CD):

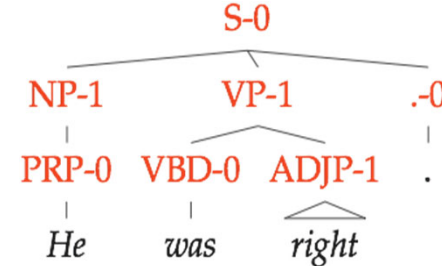
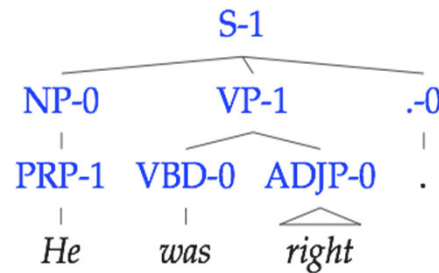
CD-7	one	two	Three
CD-4	1989	1990	1988
CD-11	million	billion	trillion
CD-0	1	50	100
CD-3	1	30	31
CD-9	78	58	34



Latent Variable Grammars



Parse Tree T
Sentence w



Derivations $t : T$



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

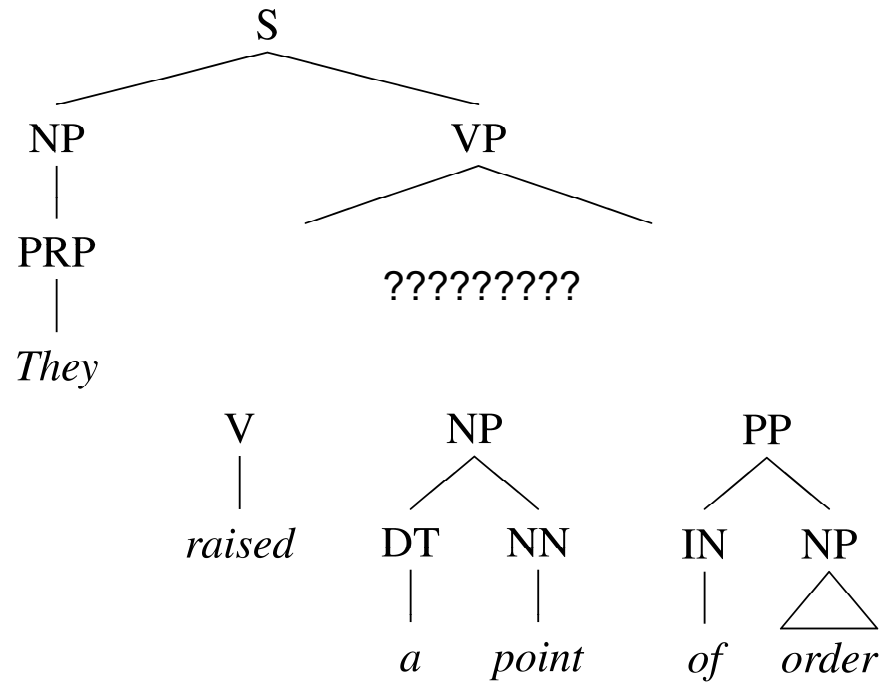
Lexicon		
$PRP_0 \rightarrow She$?	
$PRP_1 \rightarrow She$?	
...		
$VBD_0 \rightarrow was$?	
$VBD_1 \rightarrow was$?	
$VBD_2 \rightarrow was$?	
...		

Parameters θ



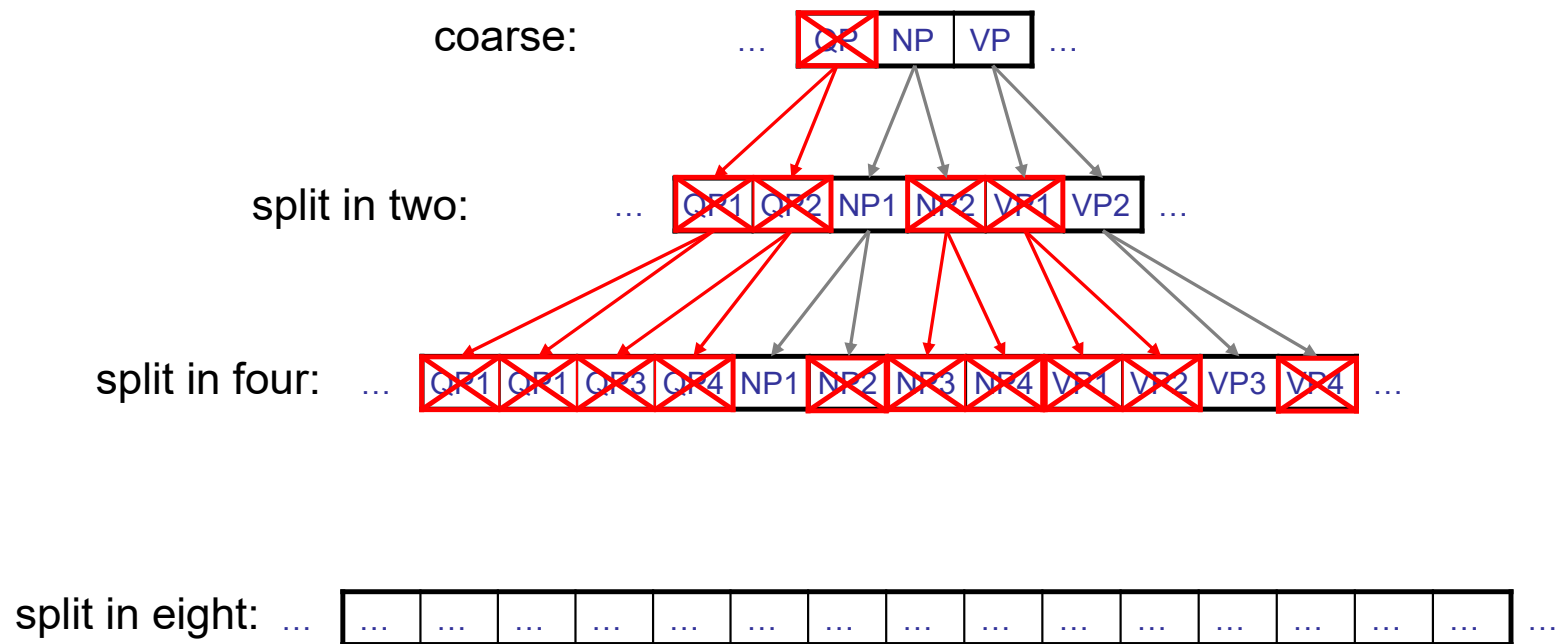
Coarse-to-Fine Inference

- Example: PP attachment



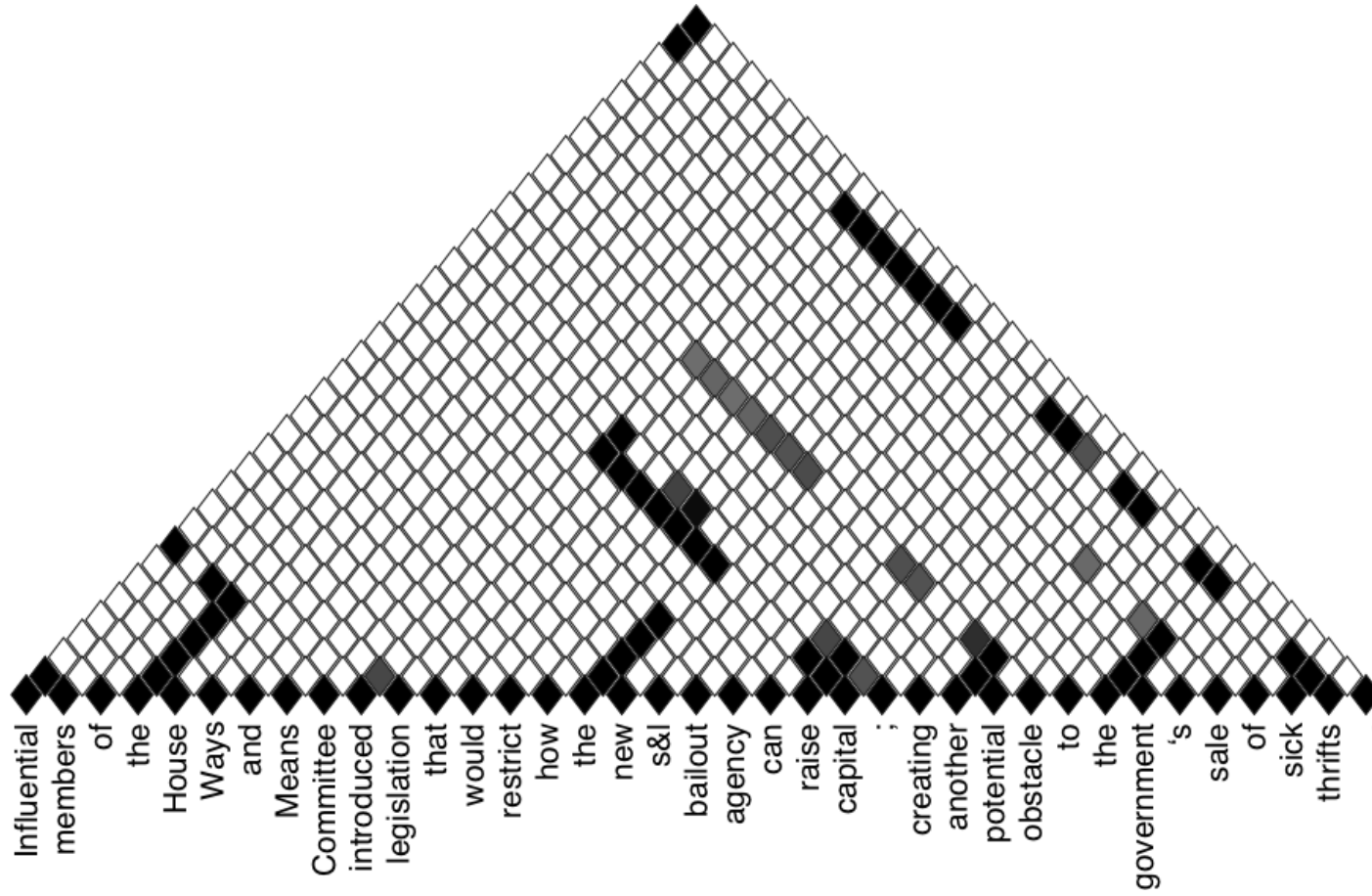


Hierarchical Pruning





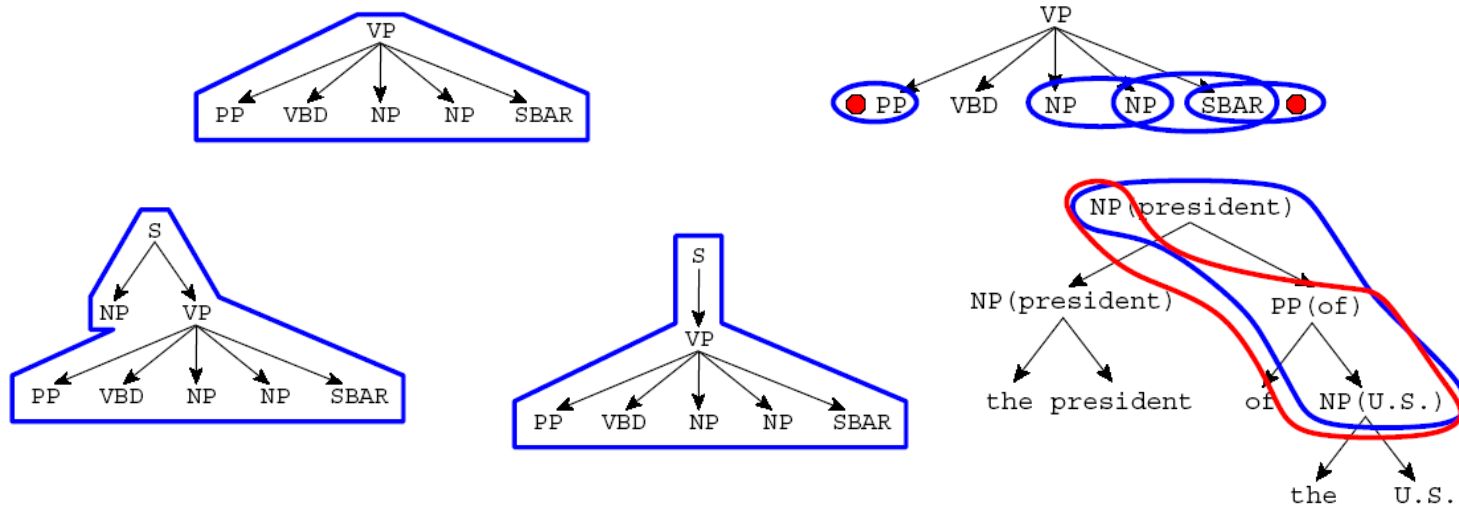
Bracket Posteriors





Parse Reranking

- Assume the number of parses is very small
- We can represent each parse T as a feature vector $\phi(T)$
 - Typically, all local rules are features
 - Also non-local features, like how right-branching the overall tree is
 - [Charniak and Johnson 05] gives a rich set of features



Natural Language Processing



Syntax and Parsing

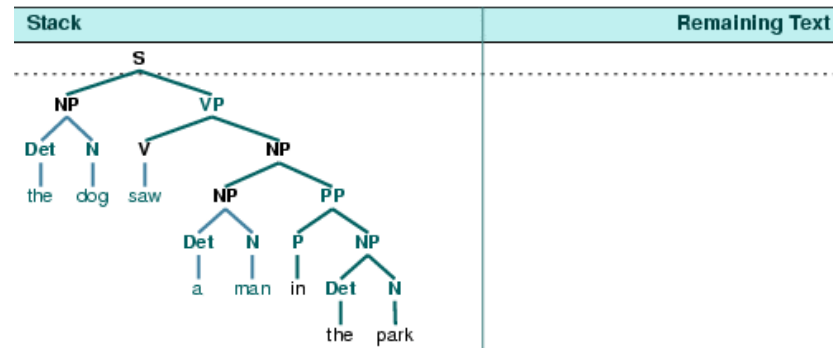
Dan Klein – UC Berkeley

Other Syntactic Models



Shift-Reduce Parsers

- Another way to derive a tree:

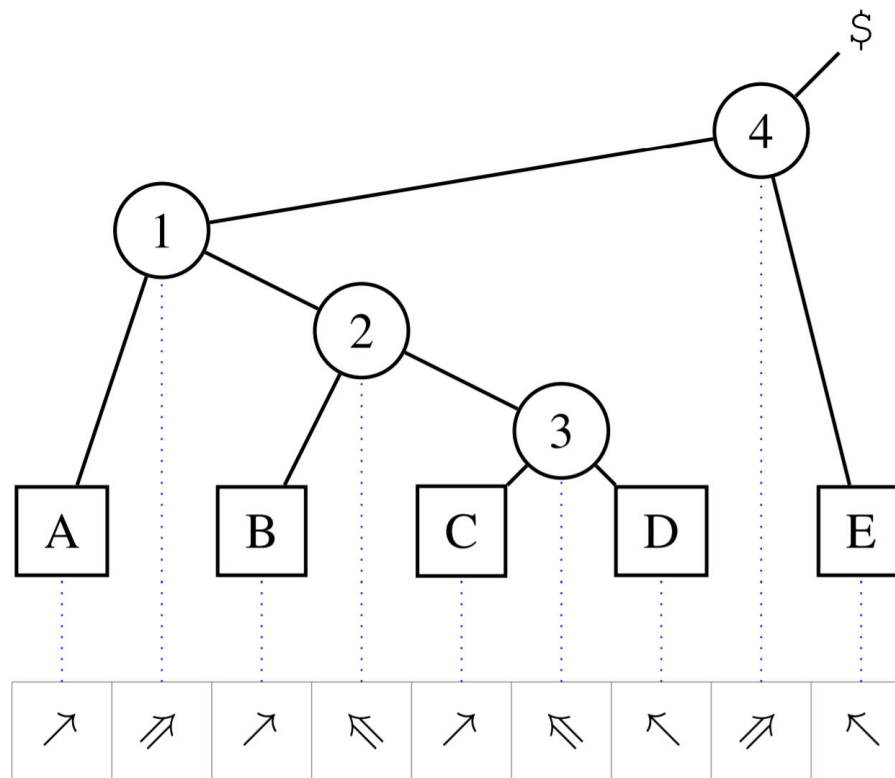


- Parsing
 - No useful dynamic programming search
 - Can still use beam search [Ratnaparkhi 97]



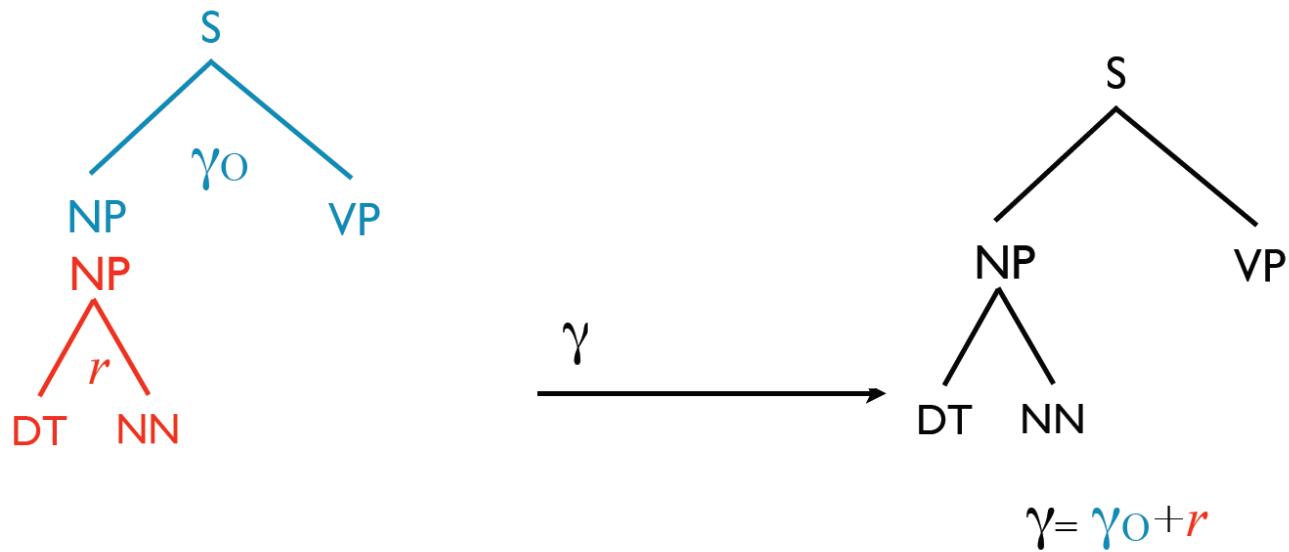
Other Transformations

- Example: Left-Corner Transforms, Tetra-Tags





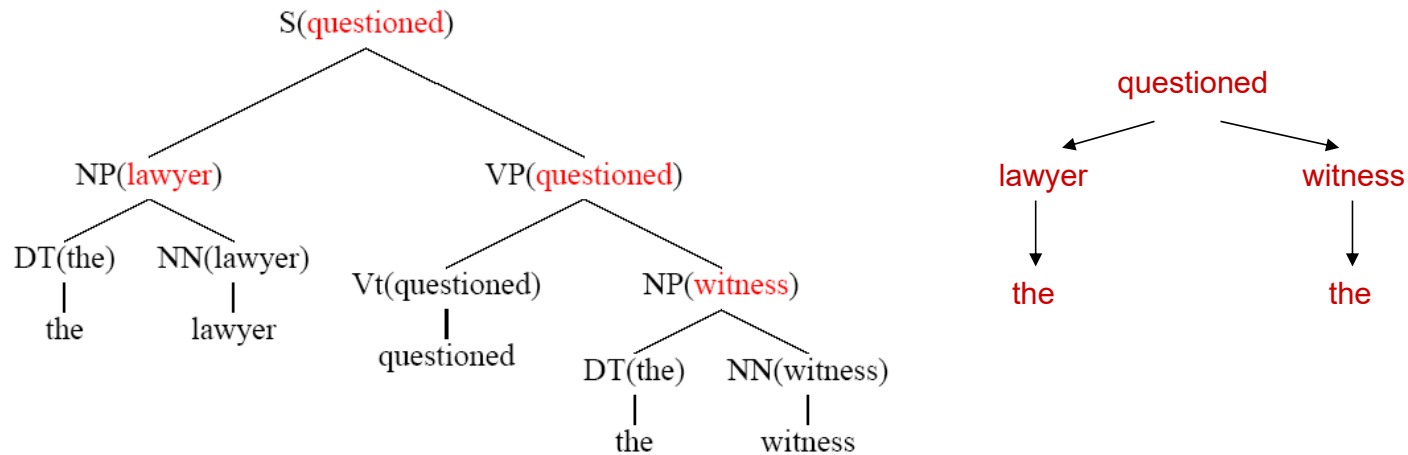
K-Best Parsing





Dependency Parsing

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

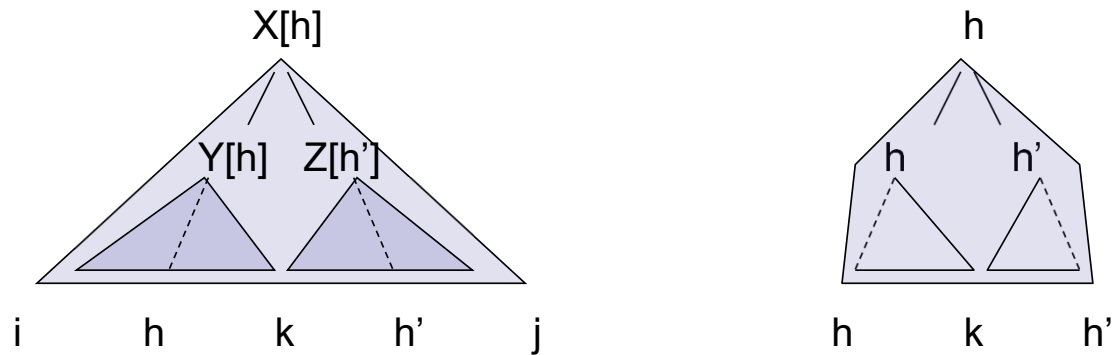


- Each local binary tree corresponds to an attachment in the dependency graph



Dependency Parsing

- Pure dependency parsing is only cubic [Eisner 99]



- Some work on *non-projective* dependencies

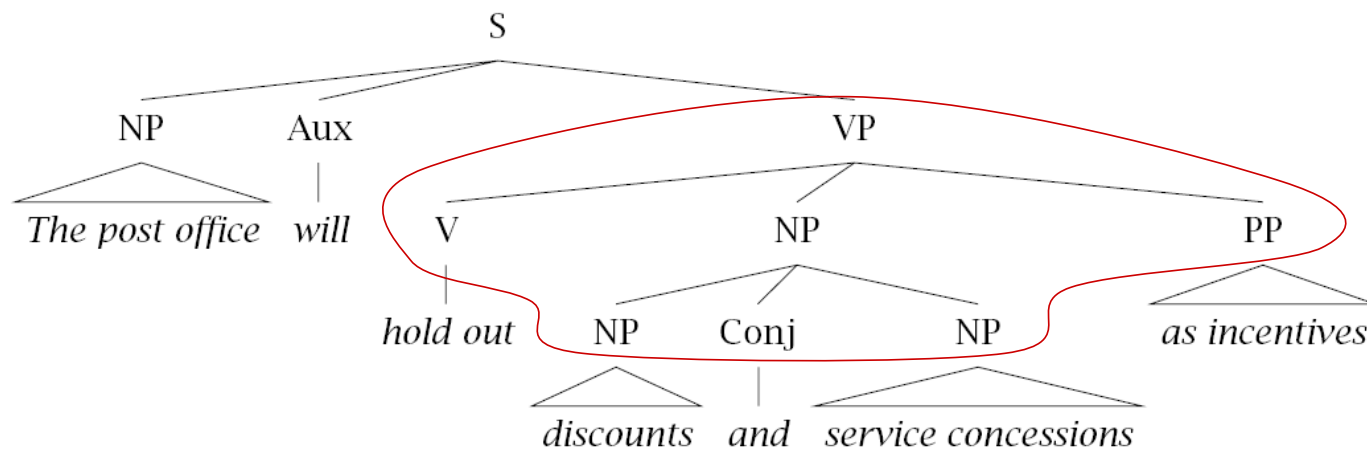
- Common in, e.g. Czech parsing
- Can do with MST algorithms [McDonald and Pereira 05]





Data-oriented parsing:

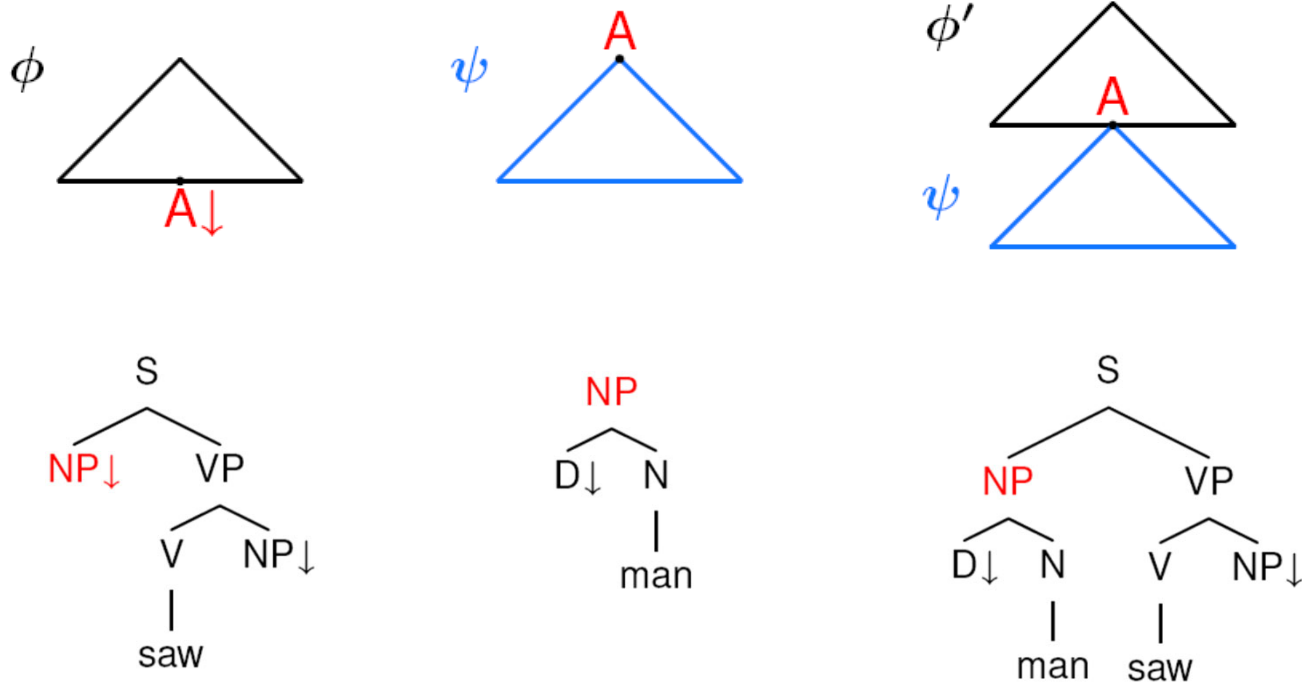
- Rewrite large (possibly lexicalized) subtrees in a single step



- Formally, a *tree-insertion grammar*
- Derivational ambiguity whether subtrees were generated atomically or compositionally
- Most probable *parse* is NP-complete



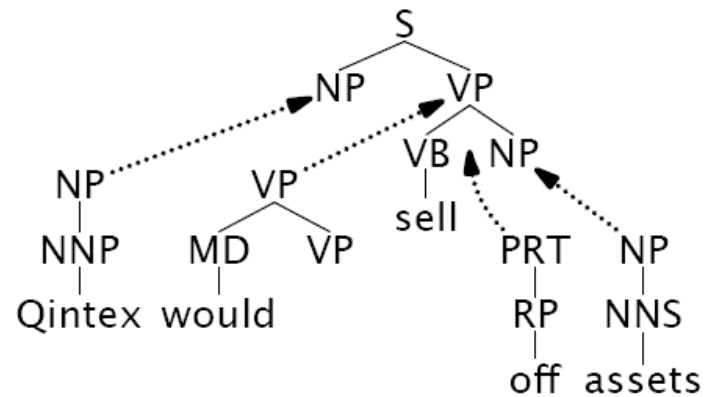
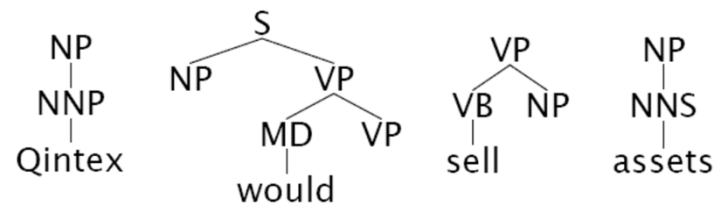
TIG: Insertion





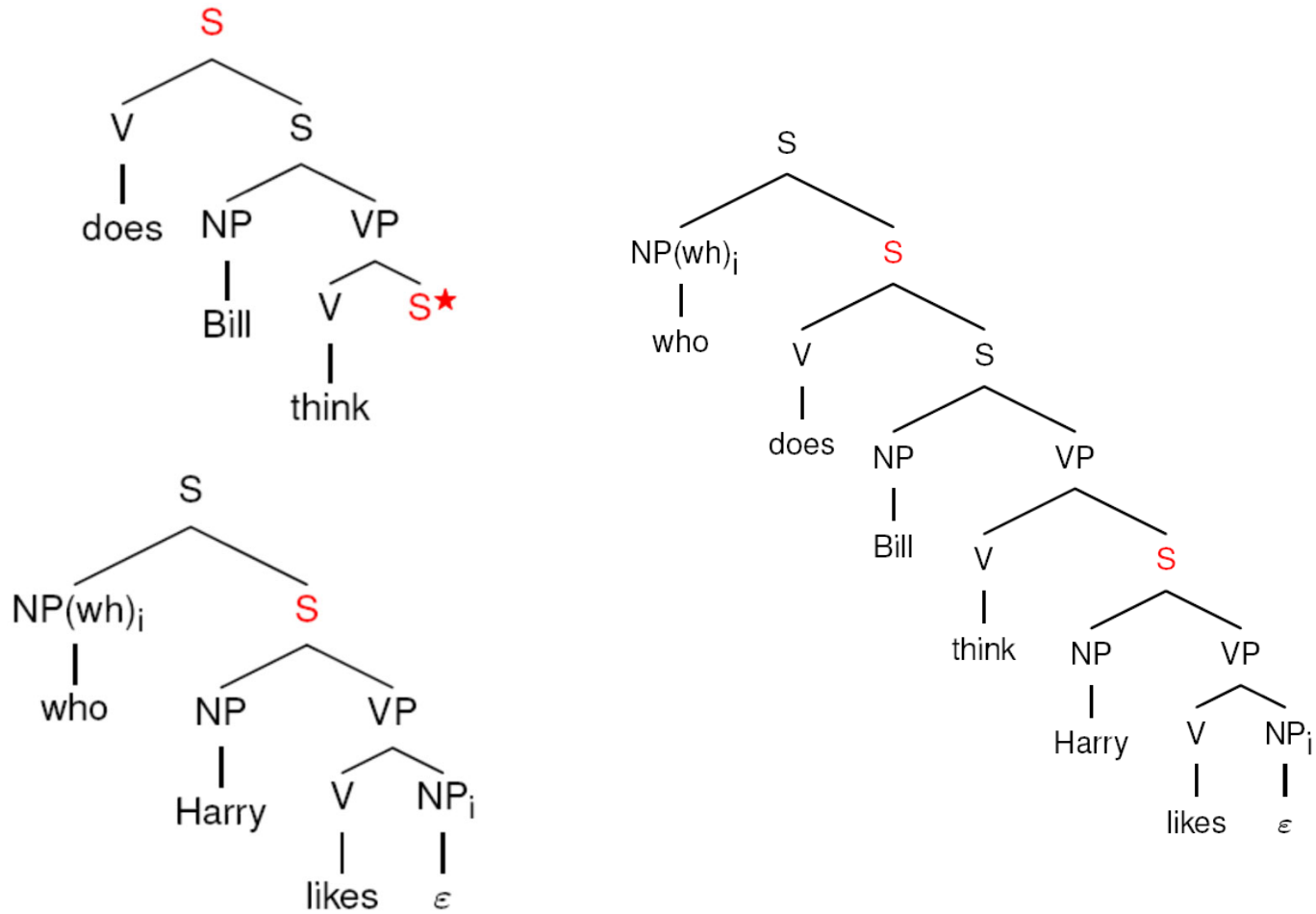
Tree-adjoining grammars

- 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?)

John \vdash NP

shares \vdash NP

buys \vdash (S\NP)/NP

sleeps \vdash S\NP

well \vdash (S\NP)\(S\NP)

