

Statistical NLP

Winter 2011

Lectures 10-11: Parsing

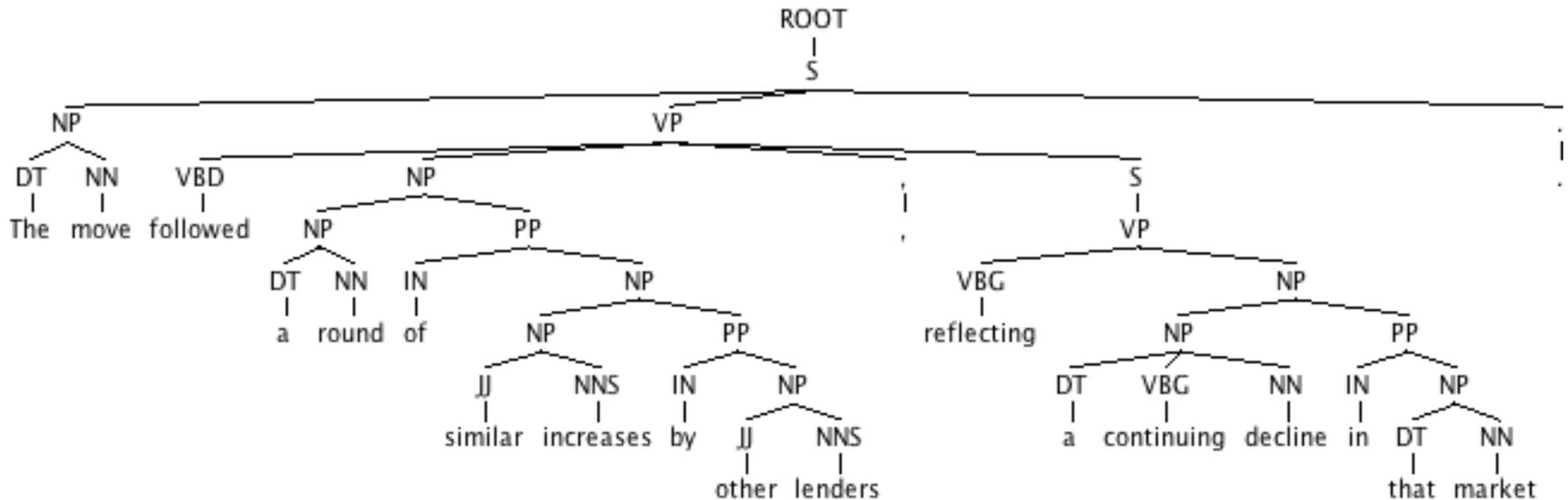
Luke Zettlemoyer - University of Washington

[Most slides from Dan Klein]

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

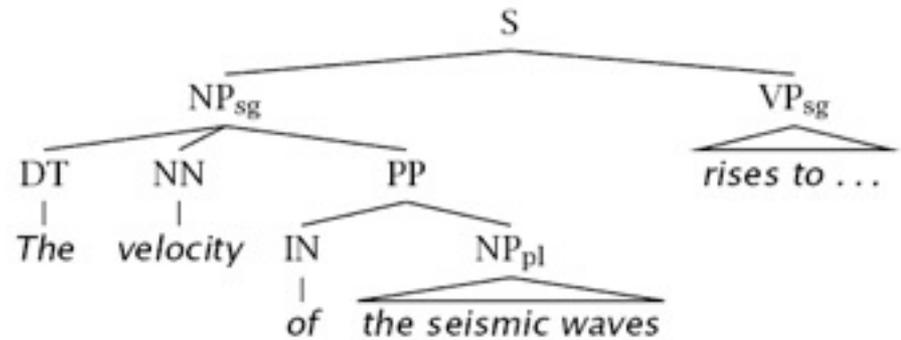
Table 3

The Penn Treebank syntactic tagset.

1.	ADJP	Adjective phrase
2.	ADVP	Adverb phrase
3.	NP	Noun phrase
4.	PP	Prepositional phrase
5.	S	Simple declarative clause
6.	SBAR	Clause introduced by subordinating conjunction or <i>0</i> (see below)
7.	SBARQ	Direct question introduced by <i>wh</i> -word or <i>wh</i> -phrase
8.	SINV	Declarative sentence with subject-aux inversion
9.	SQ	Subconstituent of SBARQ excluding <i>wh</i> -word or <i>wh</i> -phrase
10.	VP	Verb phrase
11.	WHADVP	<i>wh</i> -adverb phrase
12.	WHNP	<i>wh</i> -noun phrase
13.	WHPP	<i>wh</i> -prepositional phrase
14.	X	Constituent of unknown or uncertain category
Null elements		
1.	*	“Understood” subject of infinitive or imperative
2.	0	Zero variant of <i>that</i> in subordinate clauses
3.	T	Trace—marks position where moved <i>wh</i> -constituent is interpreted
4.	NIL	Marks position where preposition is interpreted in pied-piping contexts

Phrase Structure Parsing

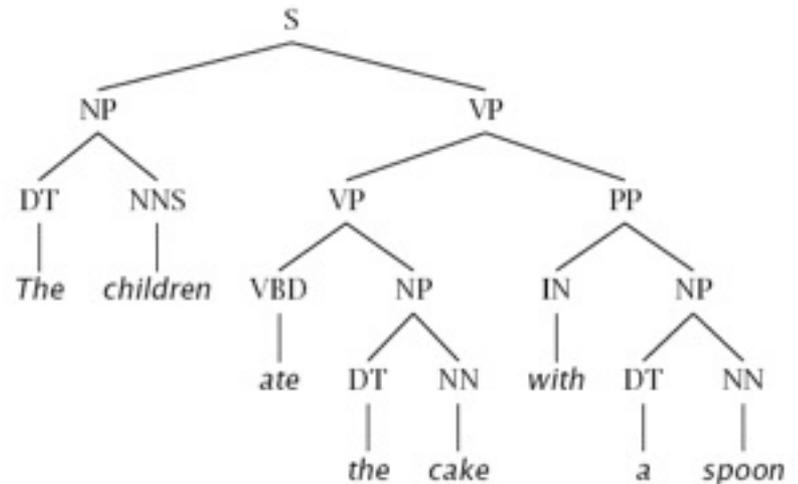
- Phrase structure parsing organizes syntax into *constituents* or *brackets*
- In general, this involves nested trees
- Linguists can, and do, argue about details
- Lots of ambiguity
- Not the only kind of syntax...



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

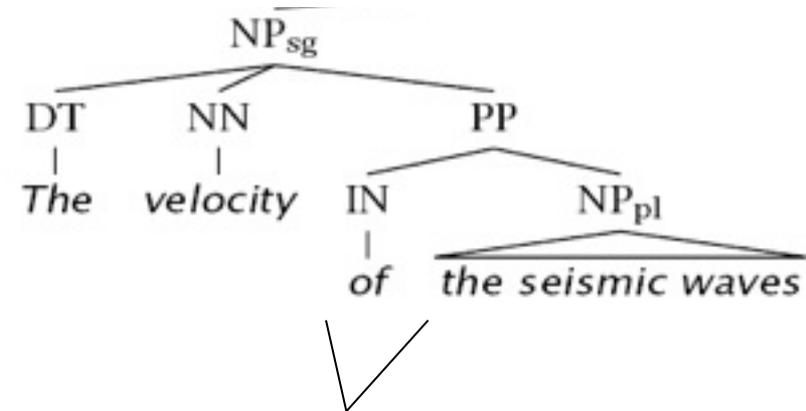
- Constituency isn't always clear

- 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.



La vitesse des ondes sismiques

Non-Local Phenomena

- Dislocation / gapping

- Why did the postman think that the neighbors were home?
- A debate arose which continued until the election.

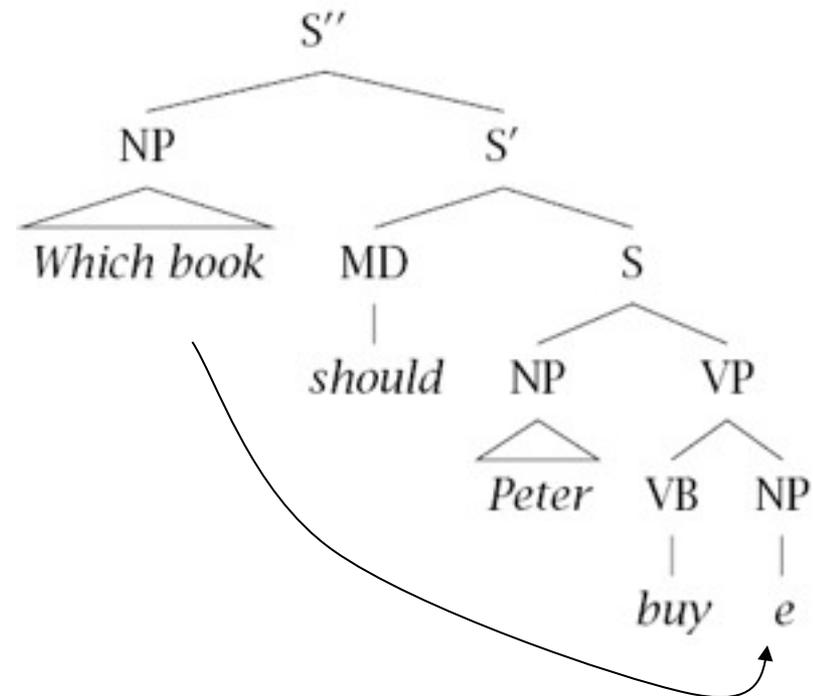
- Binding

- 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)

ROOT \rightarrow S

S \rightarrow NP VP

NP \rightarrow DT NN

NP \rightarrow NN NNS

NP \rightarrow NP PP

VP \rightarrow VBP NP

VP \rightarrow VBP NP PP

PP \rightarrow IN NP

Lexicon

NN \rightarrow interest

NNS \rightarrow raises

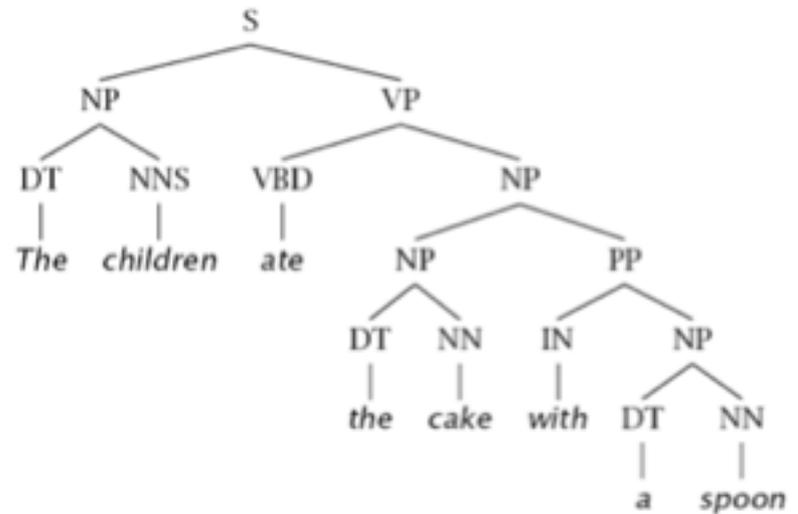
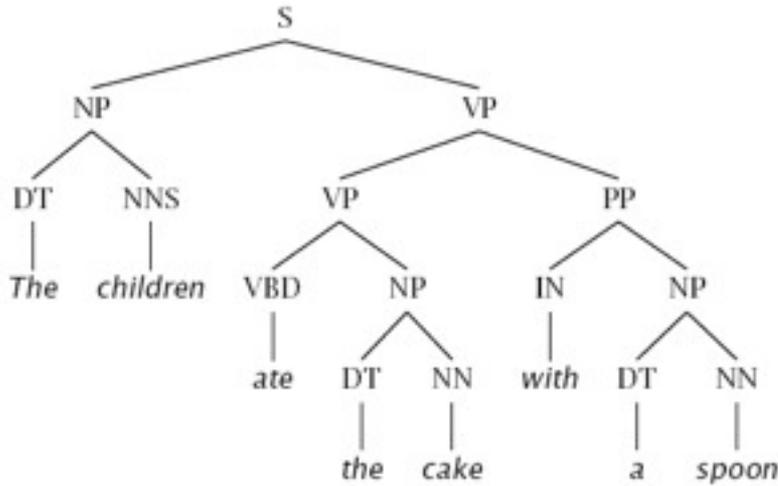
VBP \rightarrow interest

VBZ \rightarrow 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

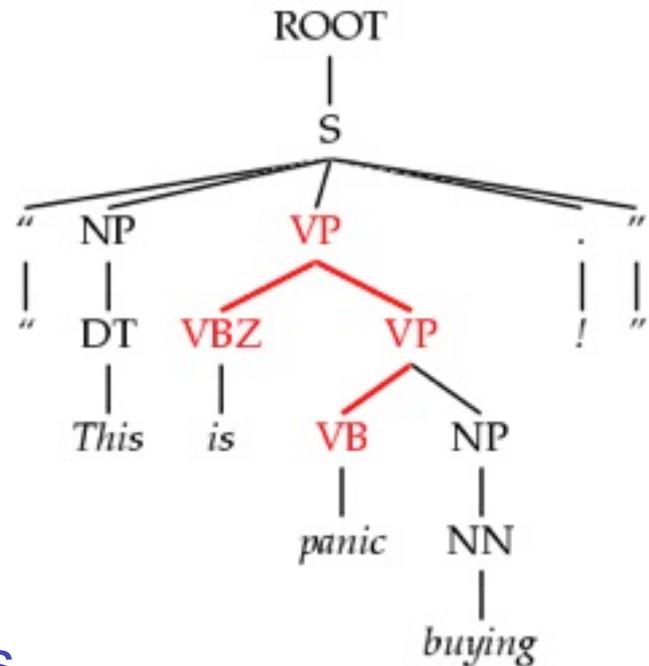
- **Modifier scope within NPs**
impractical design requirements
plastic cup holder
- **Multiple gap constructions**
The chicken is ready to eat.
The contractors are rich enough to sue.
- **Coordination scope:**
Small rats and mice can squeeze into holes or cracks in the wall.

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

Probabilistic Context-Free Grammars

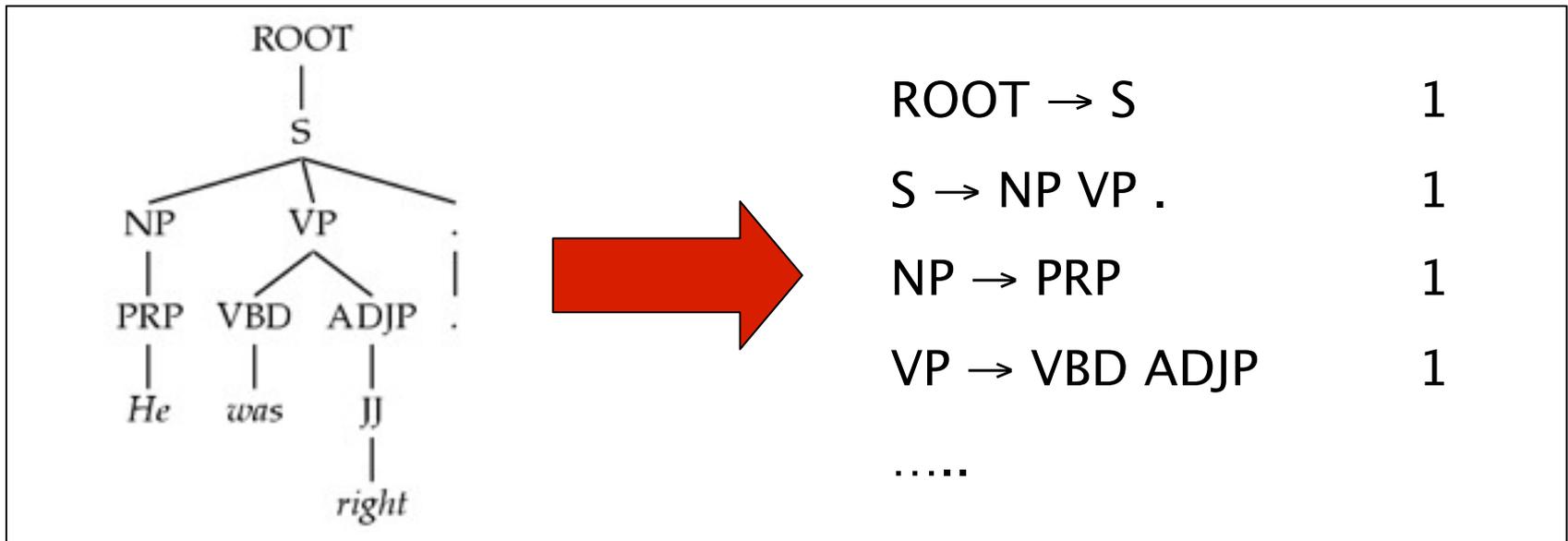
- A context-free grammar is a tuple $\langle N, T, S, R \rangle$
 - N : the set of non-terminals
 - Phrasal categories: S, NP, VP, ADJP, etc.
 - Parts-of-speech (pre-terminals): NN, JJ, DT, VB
 - T : the set of terminals (the words)
 - S : the start symbol
 - Often written as ROOT or TOP
 - *Not* usually 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 $P(X \rightarrow Y_1 Y_2 \dots Y_k)$

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):

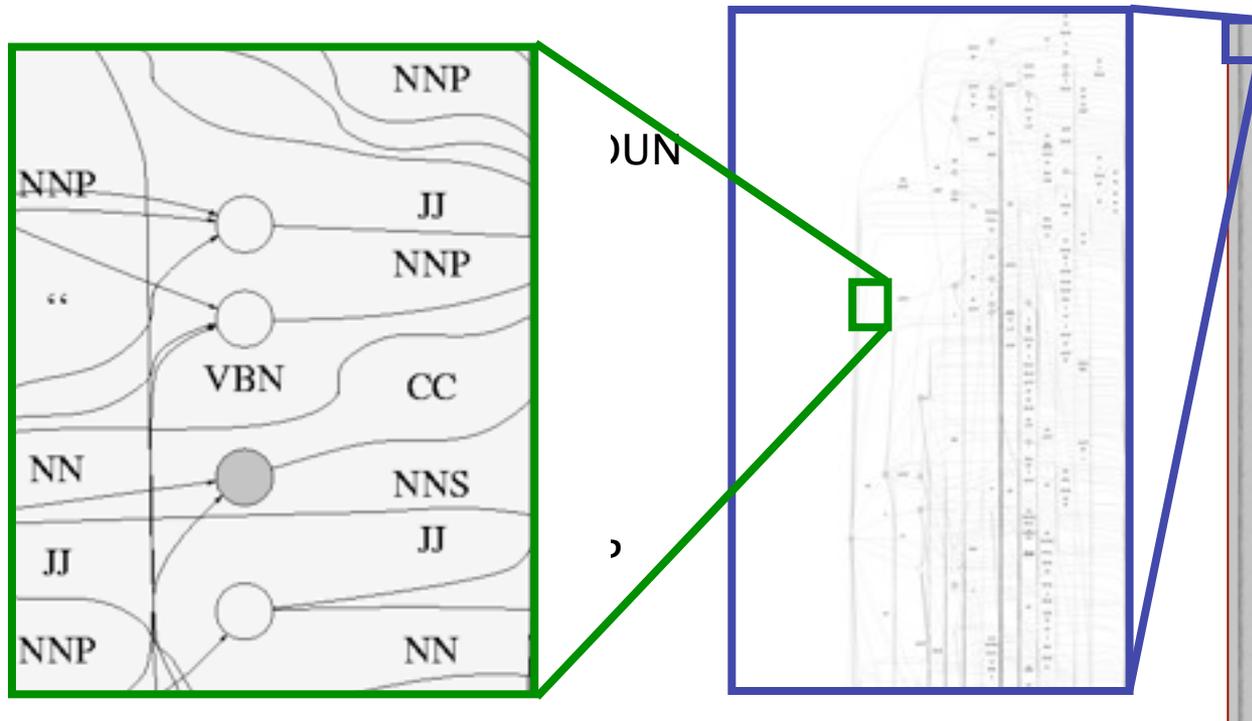


- Better results by enriching the grammar (e.g., lexicalization).
- Can also get reasonable parsers without lexicalization.

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

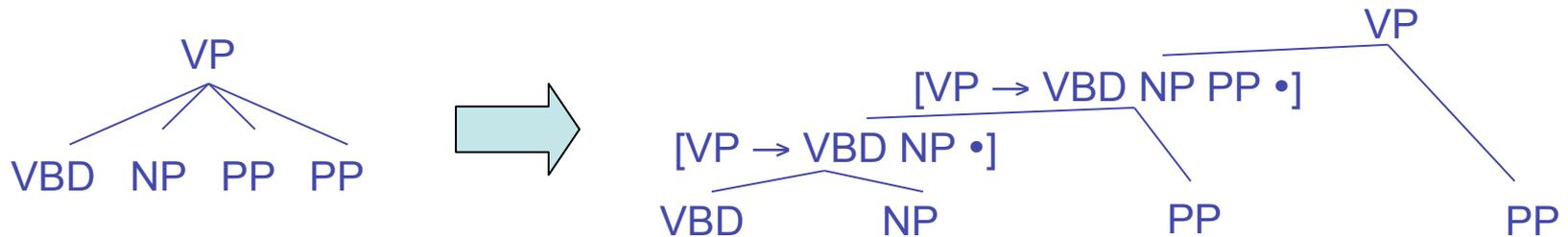
NP:



Chomsky Normal Form

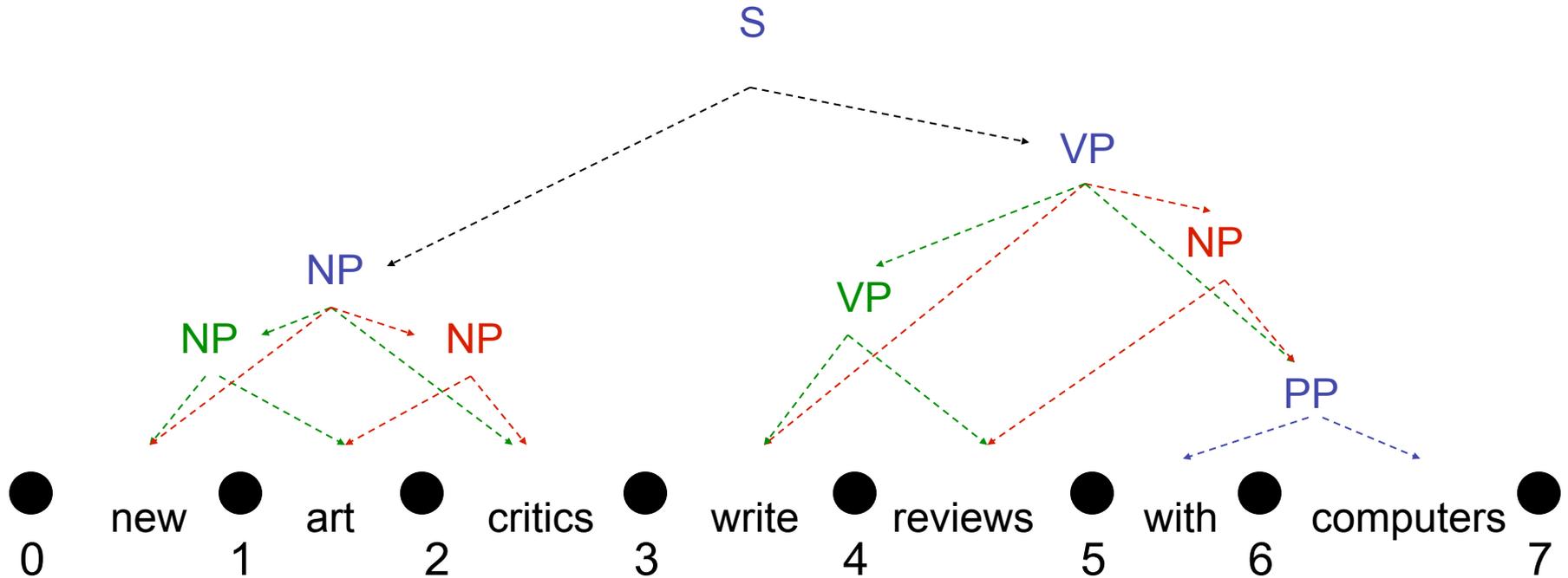
- Chomsky normal form:

- All rules of the form $X \rightarrow Y Z$ 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
 - The straightforward transformations don’t preserve tree scores
- Makes parsing algorithms simpler!

The Parsing Problem



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)
```

- 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?

A Memoized Parser

- One small change:

```
bestScore(X,i,j,s)
  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,s) *
                bestScore(Z,k,j,s)
    scores[X][i][j] = score
  return scores[X][i][j]
```

A Bottom-Up Parser (CKY)

- Can also organize things bottom-up

```
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)
```

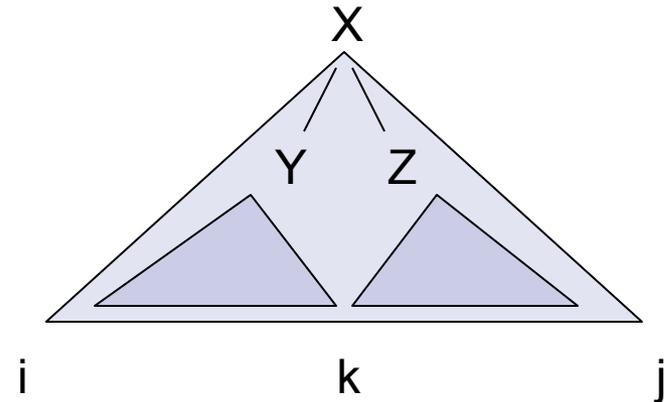
```
        for (k : [i+1, j-1])
```

```
          score[X][i][j] = max score[X][i][j],
```

```
                                score(X->YZ) *
```

```
                                score[Y][i][k] *
```

```
                                score[Z][k][j]
```



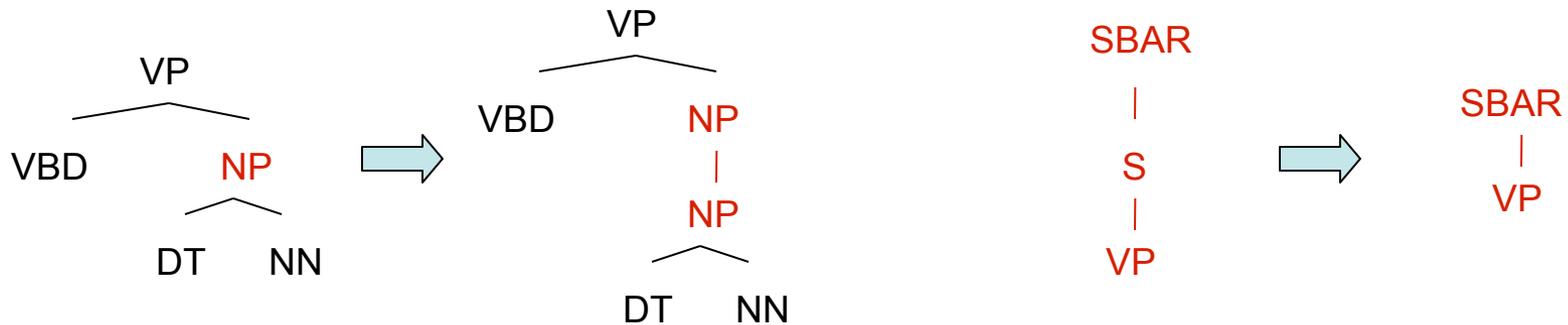
Unary Rules

- 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)
```

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)
```

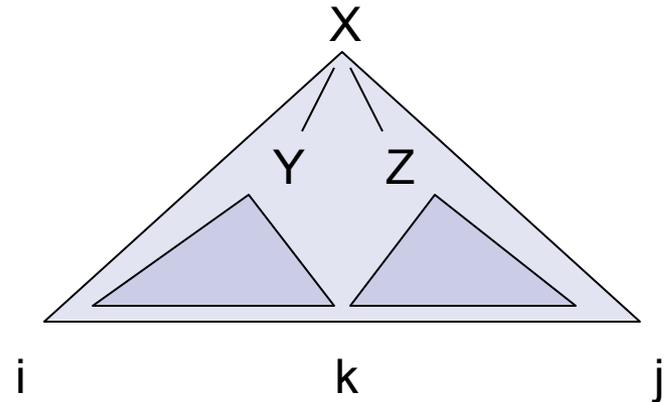
Memory

- How much memory does this require?
 - Have to store the score cache
 - Cache size: $|\text{symbols}| * n^2$ doubles
 - For the plain treebank grammar:
 - $X \sim 20K$, $n = 40$, double ~ 8 bytes = $\sim 256MB$
 - Big, but workable.
- Pruning: Beams
 - $\text{score}[X][i][j]$ can get too large (when?)
 - Can keep beams (truncated maps $\text{score}[i][j]$) 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?

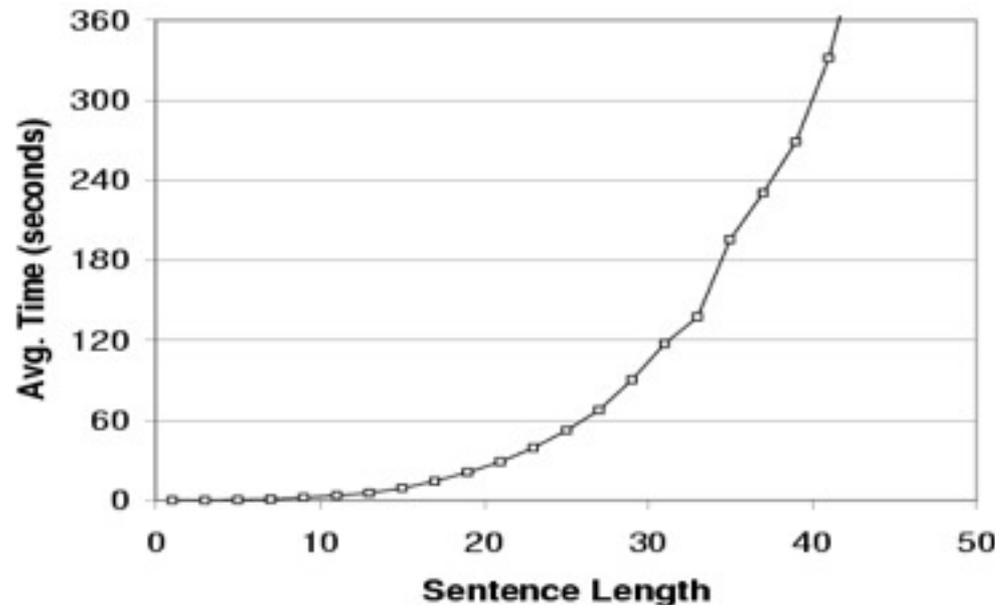
- For each diff ($\leq n$)
 - For each i ($\leq n$)
 - For each rule $X \rightarrow Y Z$
 - For each split point k
Do constant work



- Total time: $|\text{rules}| * n^3$
- 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

Efficient CKY

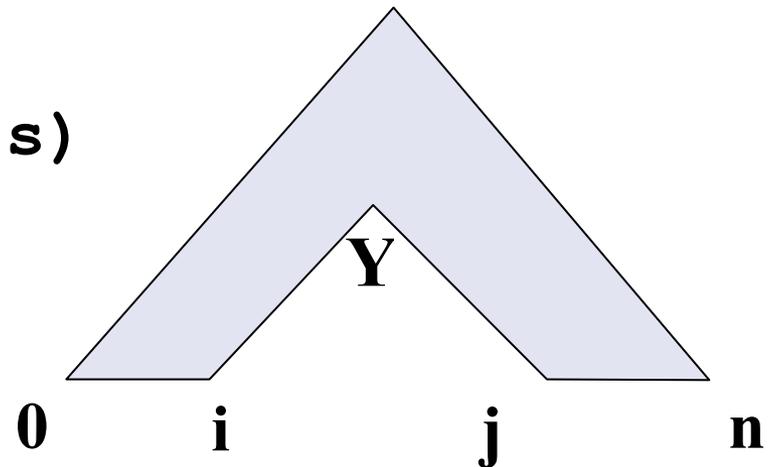
- Lots of tricks to make CKY efficient
 - Most of them are little engineering details:
 - E.g., first choose k , then enumerate through the $Y:[i,k]$ which are non-zero, then loop through rules by left child.
 - Optimal layout of the dynamic program depends on grammar, input, even system details.
 - Another kind is more critical:
 - Many $X:[i,j]$ can be suppressed on the basis of the input string
 - We'll see this next class as figures-of-merit or A^* heuristics

Best Outside Scores

Want to compute the best parse missing a specific word span:

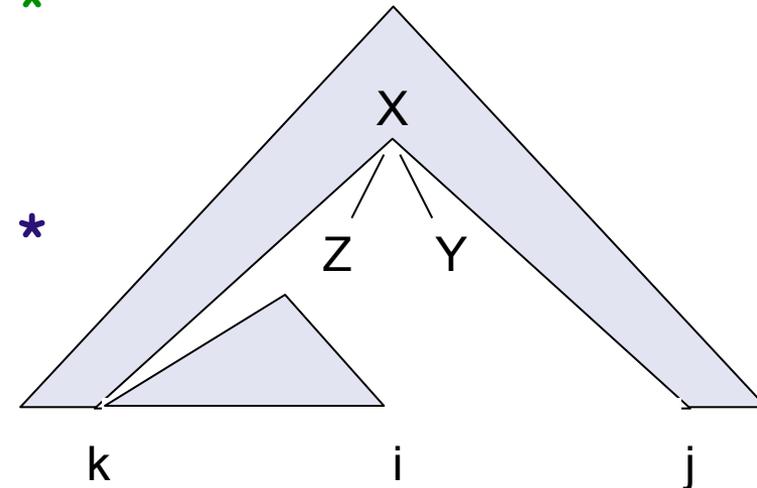
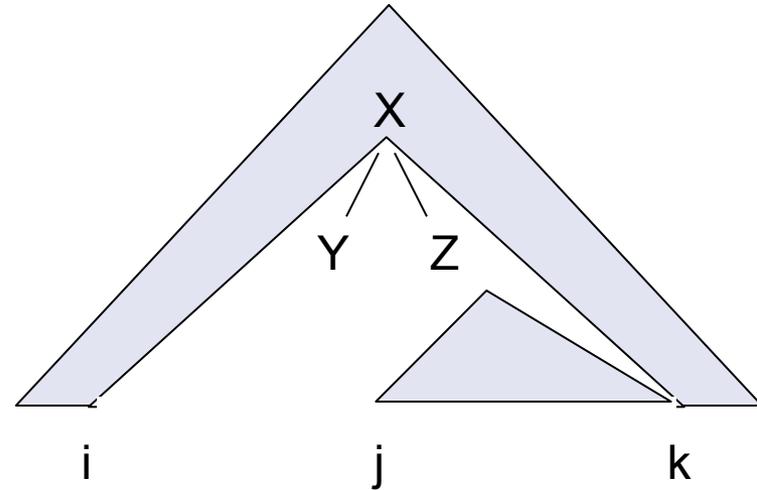
- Tree rooted at Y from words $s[i:j]$ is left unspecified
- this is the “opposite” of the `bestScore` / inside score

`bestOutside(Y, i, j, s)`



Best Outside Scores

```
bestOutside(Y,i,j,s)
  if (i==0 && j==n)
    return 1.0
  else
    return max
      max score(X->YZ) *
          bestOutside(X,i,k,s) *
          bestScore(Z,j,k,s)
      max score(X->ZY) *
          bestOutside(X,k,j,s) *
          bestScore(Z,k,i,s)
```



Agenda-Based Parsing

- Agenda-based parsing is like graph search (but over a hypergraph)
- Concepts:
 - Numbering: we number fenceposts between words
 - “Edges” or items: spans with labels, e.g. PP[3,5], represent the **sets** of trees over those words rooted at that label (cf. search states)
 - A chart: records edges we’ve expanded (cf. closed set)
 - An agenda: a queue which holds edges (cf. a fringe or open set)



Word Items

- Building an item for the first time is called discovery. Items go into the agenda on discovery.
- To initialize, we discover all word items (with score 1.0).

AGENDA

critics[0,1], write[1,2], reviews[2,3], with[3,4], computers[4,5]

CHART [EMPTY]



Unary Projection

- When we pop a word item, the lexicon tells us the tag item successors (and scores) which go on the agenda

critics[0,1]	write[1,2]	reviews[2,3]	with[3,4]	computers[4,5]
NNS[0,1]	VBP[1,2]	NNS[2,3]	IN[3,4]	NNS[4,5]



critics write reviews with computers

Item Successors

- When we pop items off of the agenda:
 - Graph successors: unary projections (NNS \rightarrow critics, NP \rightarrow NNS)

$Y[i,j]$ with $X \rightarrow Y$ forms $X[i,j]$

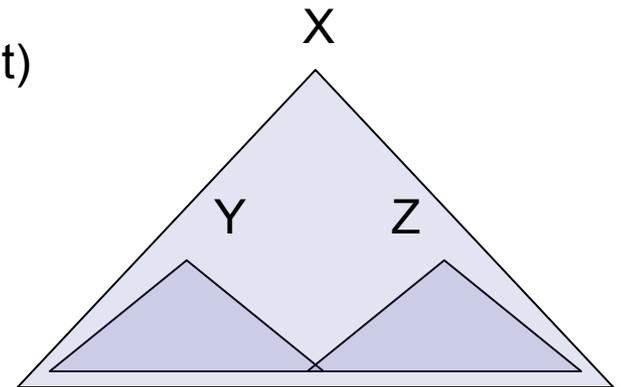
- Hypergraph successors: combine with items already in our chart

$Y[i,j]$ and $Z[j,k]$ with $X \rightarrow Y Z$ form $X[i,k]$

- Enqueue / promote resulting items (if not in chart already)
- Record backtraces as appropriate
- Stick the popped edge in the chart (closed set)

- Queries a chart must support:

- Is edge $X[i,j]$ in the chart? (What score?)
- What edges with label Y end at position j ?
- What edges with label Z start at position i ?



An Example

NNS[0,1] VBP[1,2] NNS[2,3] IN[3,4] NNS[3,4] NP[0,1] VP[1,2] NP[2,3] NP[4,5] S[0,2]
VP[1,3] PP[3,5] ROOT[0,2] S[0,3] VP[1,5] NP[2,5] ROOT[0,3] S[0,5] ROOT[0,5]

ROOT[0-5]

S[0-5]

ROOT[0-3]
S[0-3]

VP[1,5]

ROOT[0-2]

NP[2-5]

S[0-2]

VP[1-3]

PP[3-5]

NP[0-1]

VP[1-2]

NP[2-3]

NP[4-5]

NNS[0-1]

VBP[1-2]

NNS[2-3]

IN[3-4]

NNS[4-5]



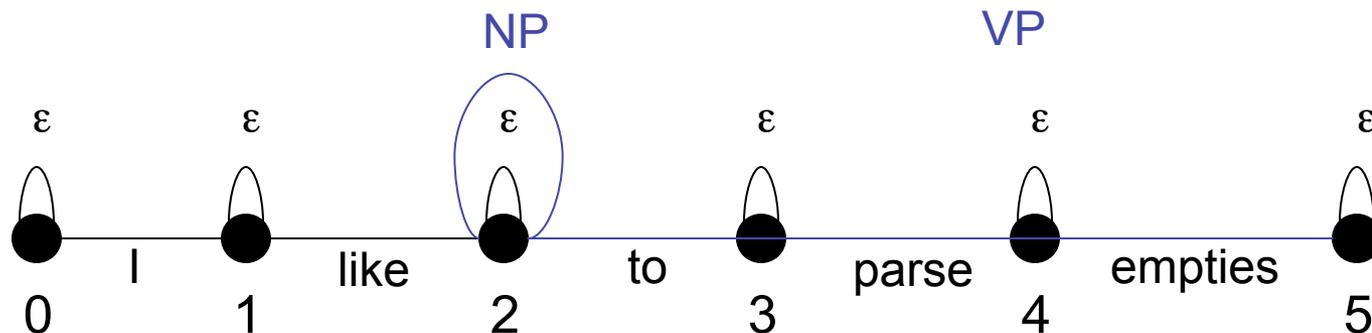
Empty Elements

- Sometimes we want to posit nodes in a parse tree that don't contain any pronounced words:

I want you to parse this sentence

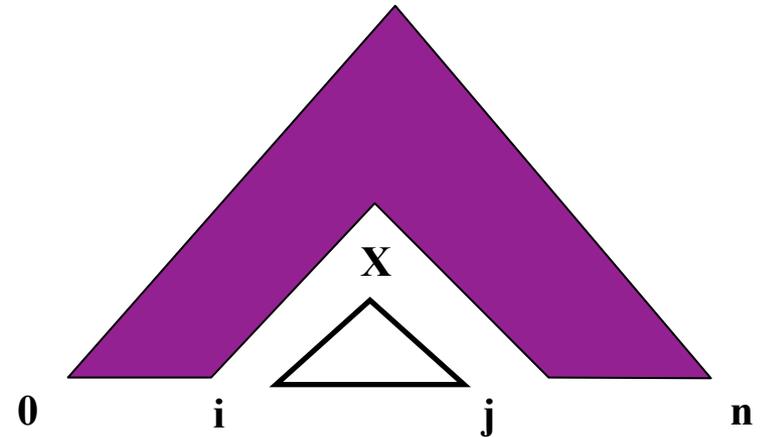
I want [] to parse this sentence

- These are easy to add to a chart parser!
 - For each position i , add the “word” edge $\varepsilon:[i,i]$
 - Add rules like $NP \rightarrow \varepsilon$ to the grammar
 - That's it!



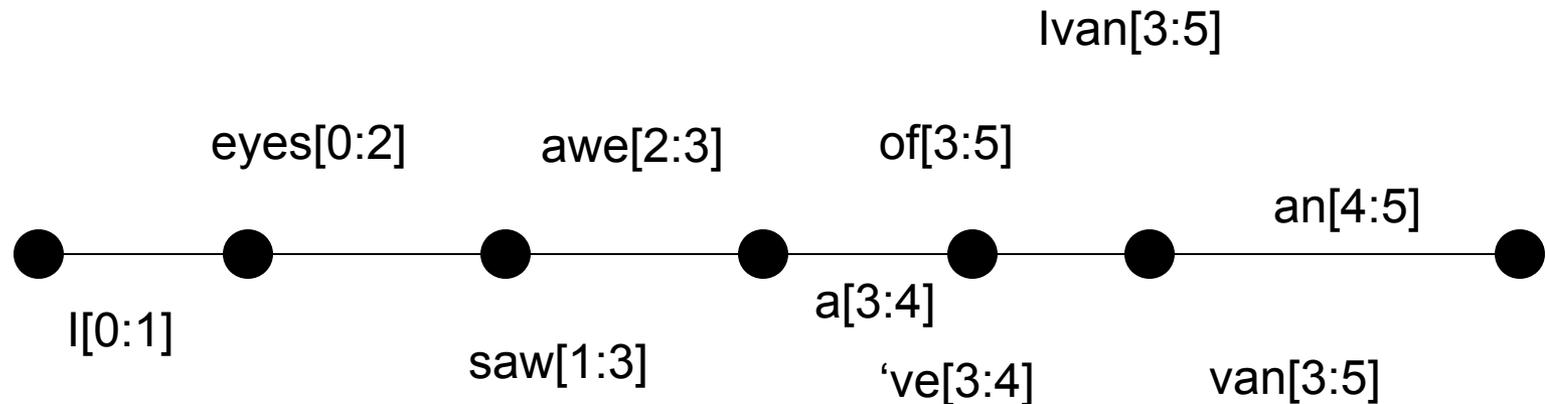
UCS / A*

- With weighted edges, order matters
 - Must expand optimal parse from bottom up (subparses first)
 - CKY does this by processing smaller spans before larger ones
 - UCS pops items off the agenda in order of decreasing Viterbi score
 - A* search also well defined
- You can also speed up the search without sacrificing optimality
 - Can select which items to process first
 - Can do with any “figure of merit” [Charniak 98]
 - If your figure-of-merit is a valid A* heuristic, no loss of optimality [Klein and Manning 03]



(Speech) Lattices

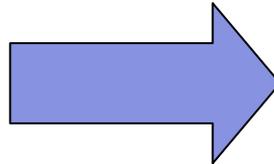
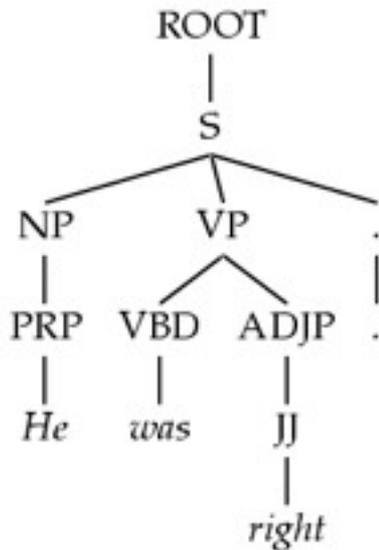
- There was nothing magical about words spanning exactly one position.
- When working with speech, we generally don't know how many words there are, or where they break.
- We can represent the possibilities as a lattice and parse these just as easily.



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

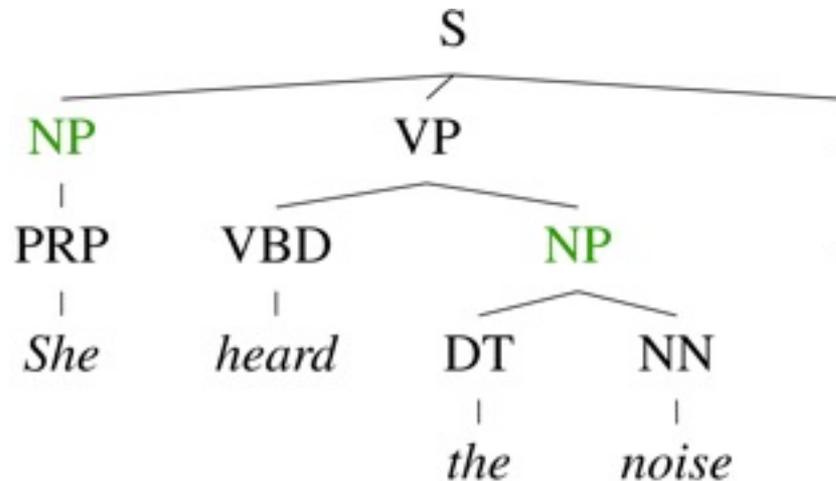
NP → PRP 1

VP → VBD ADJP 1

.....

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

Conditional Independence?

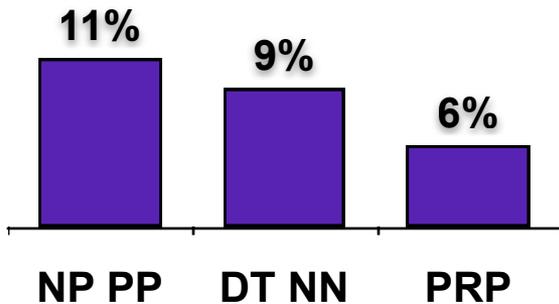


- Not every NP expansion can fill every NP slot
 - A grammar with symbols like “NP” won’t be context-free
 - Statistically, conditional independence too strong

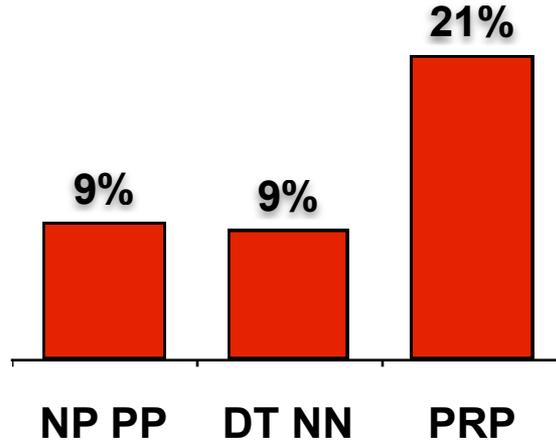
Non-Independence

- Independence assumptions are often too strong.

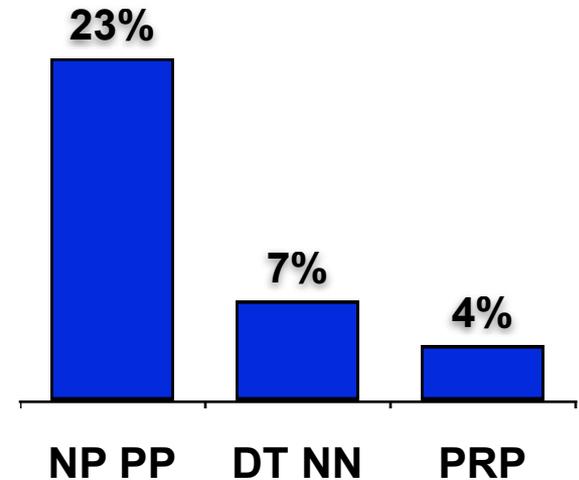
All NPs



NPs under S



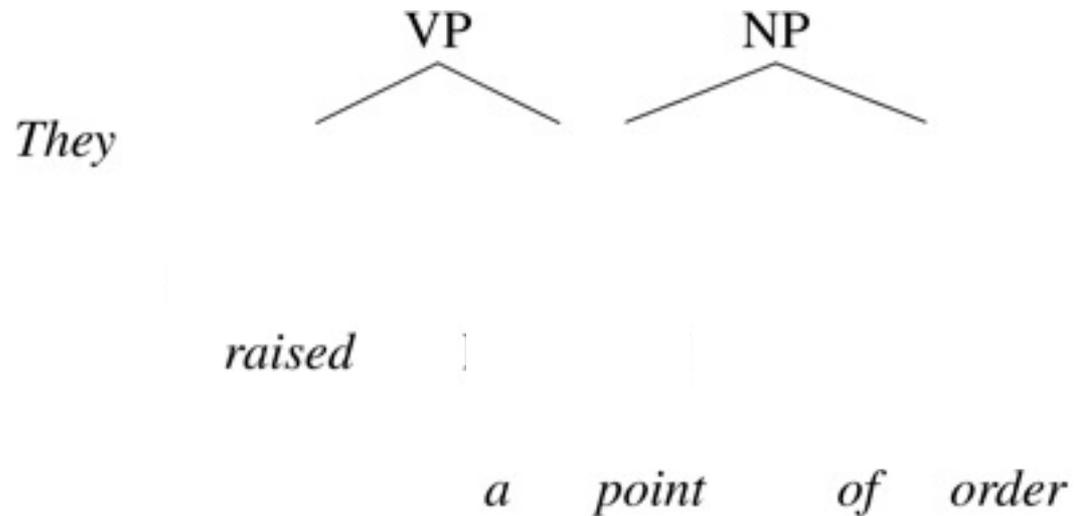
NPs under VP



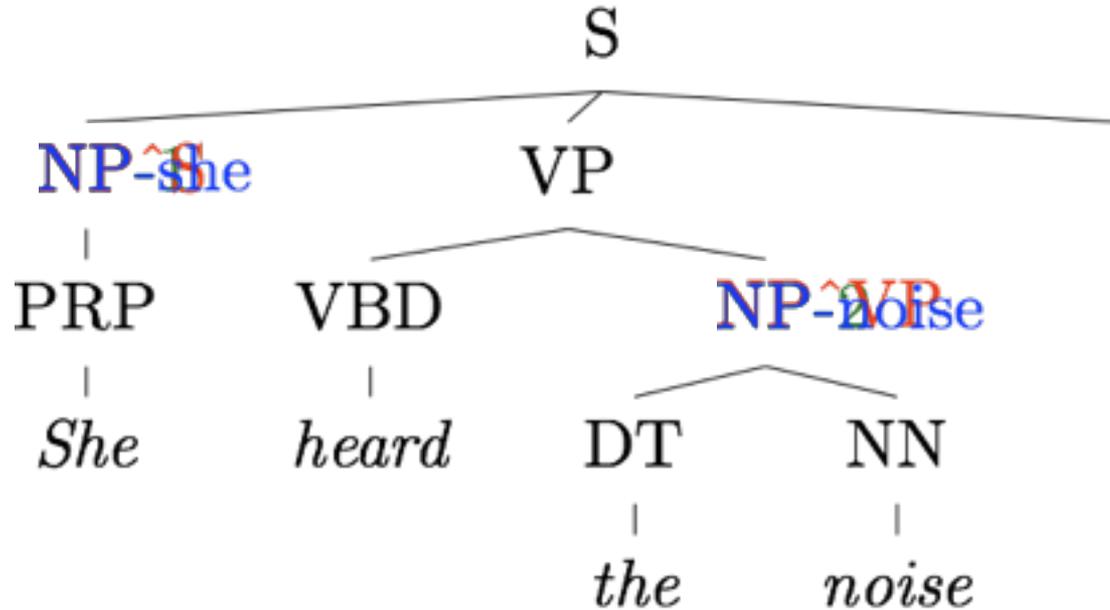
- Example: the expansion of an NP is highly dependent on the parent of the NP (i.e., subjects vs. objects).
- Also: the subject and object expansions are correlated!

Grammar Refinement

- Example: PP attachment

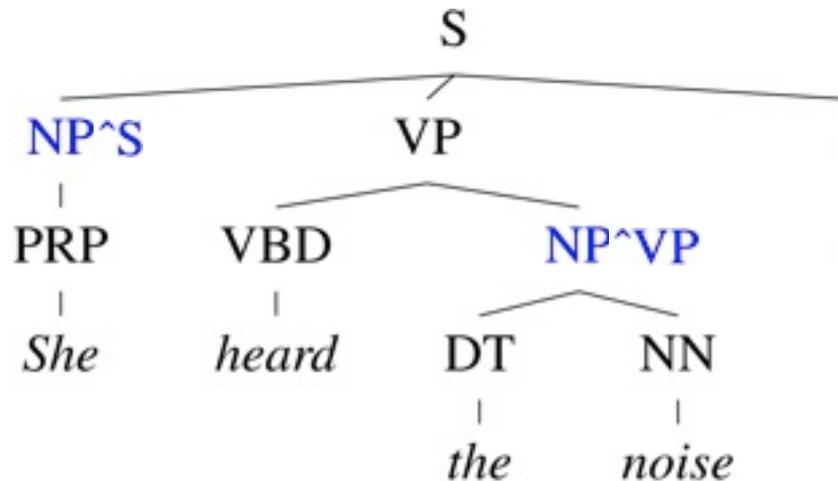


Grammar Refinement



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

The Game of Designing a Grammar



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

Typical Experimental Setup

- Corpus: Penn Treebank, WSJ

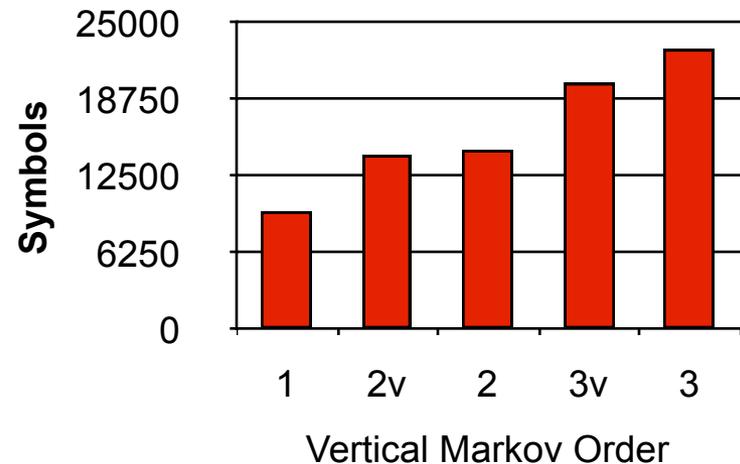
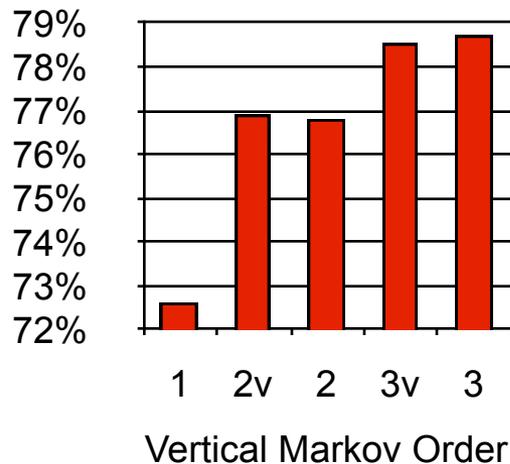
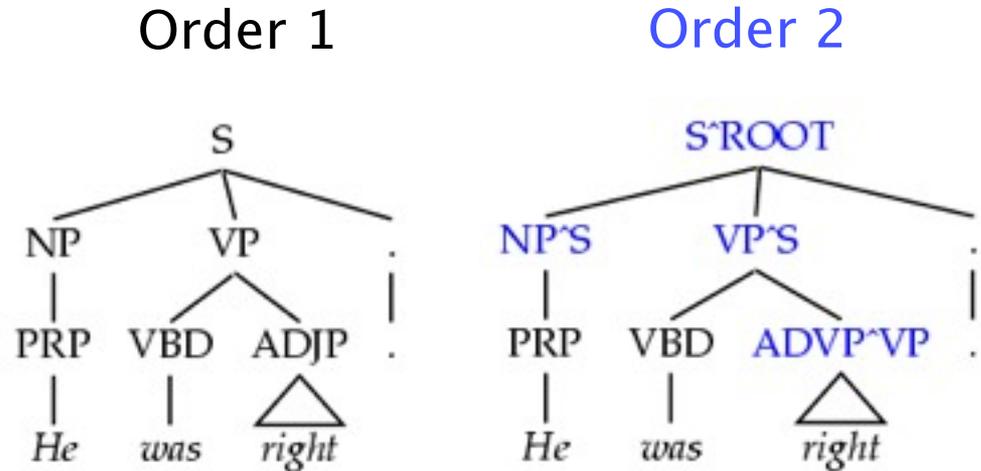


Training:	sections	02-21
Development:	section	22 (here, first 20 files)
Test:	section	23

- 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 → NP CC •

Vertical Markovization

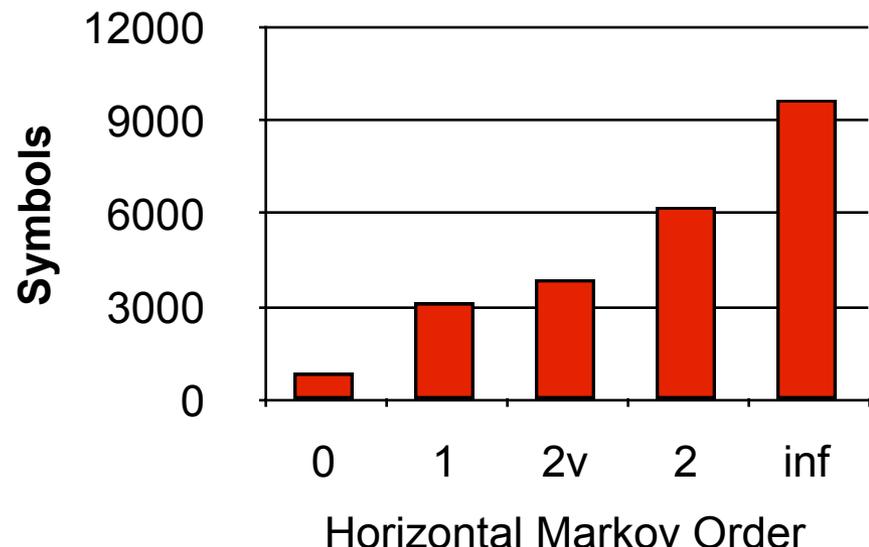
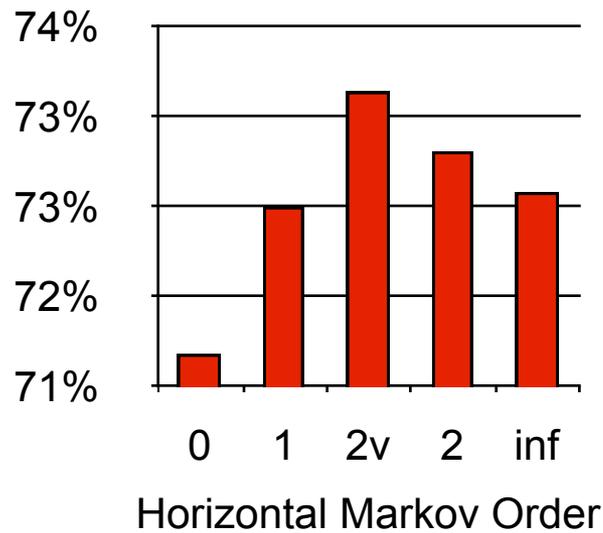
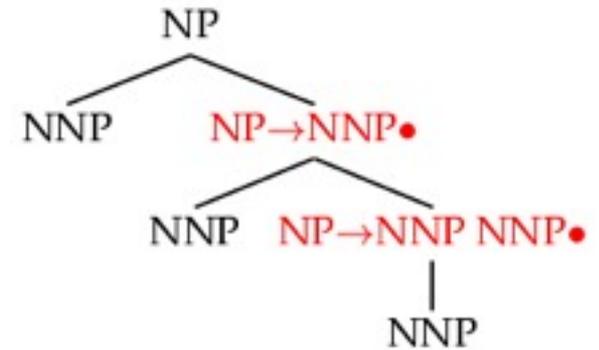
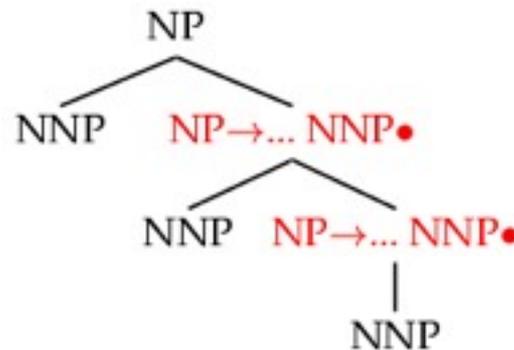
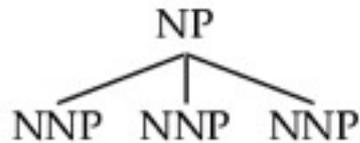
- Vertical Markov order: rewrites depend on past k ancestor nodes. (cf. parent annotation)



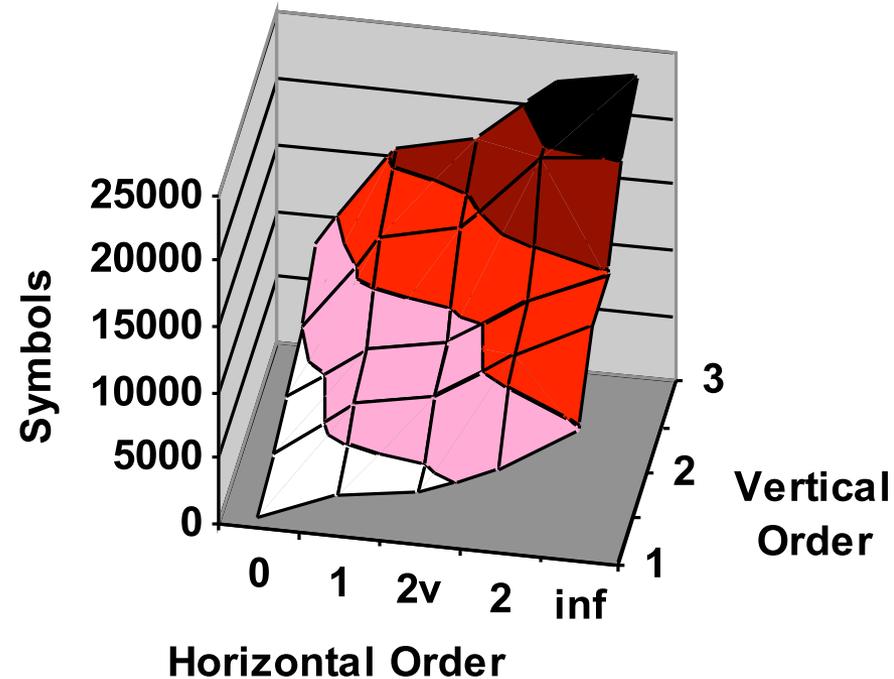
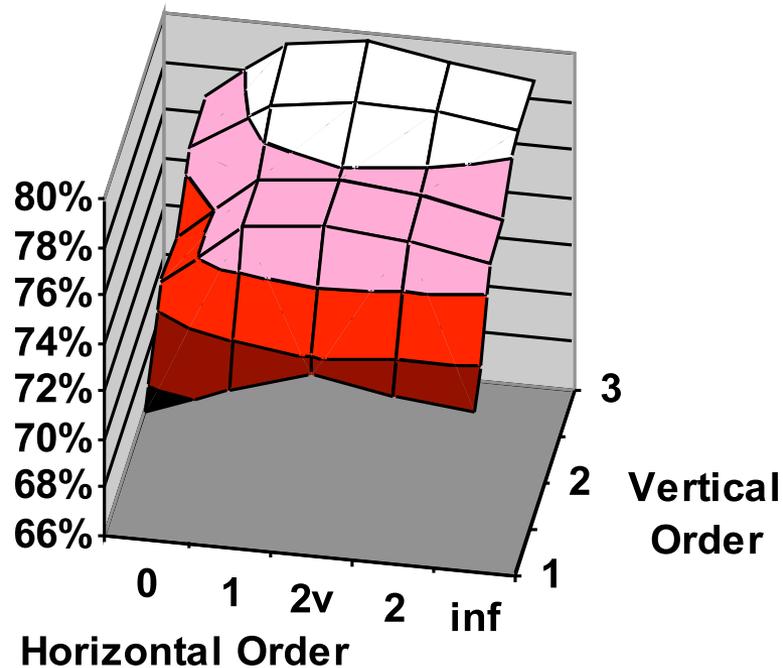
Horizontal Markovization

Order 1

Order ∞



Vertical and Horizontal



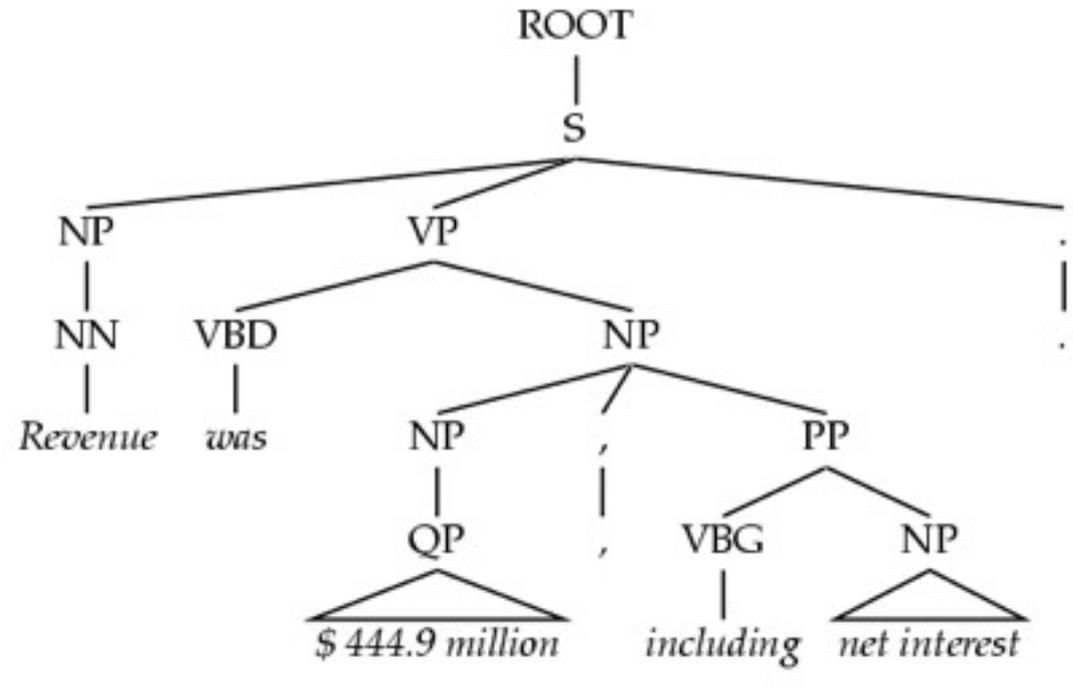
Examples:

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

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

Unary Splits

- Problem: unary rewrites used to transmute categories so a high-probability rule can be used.

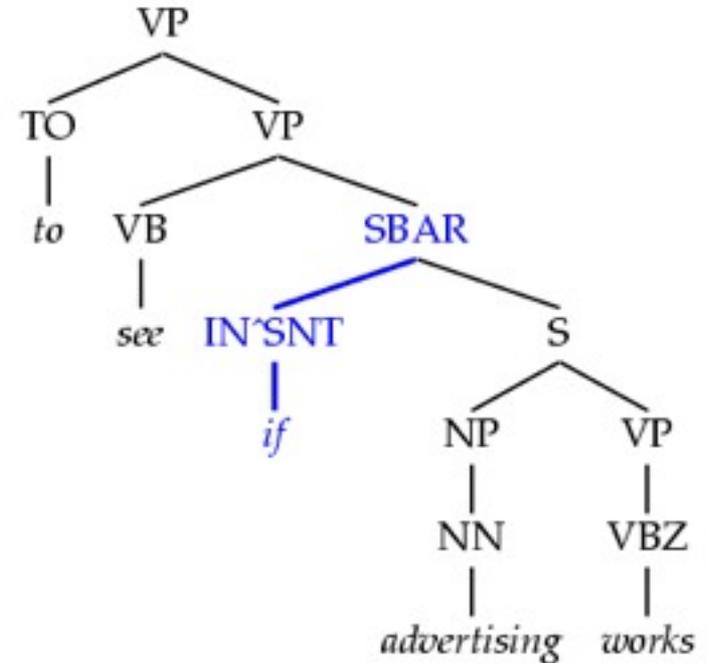


- Solution: Mark unary rewrite sites with -U

Annotation	F1	Size
Base	77.8	7.5K
UNARY	78.3	8.0K

Tag Splits

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



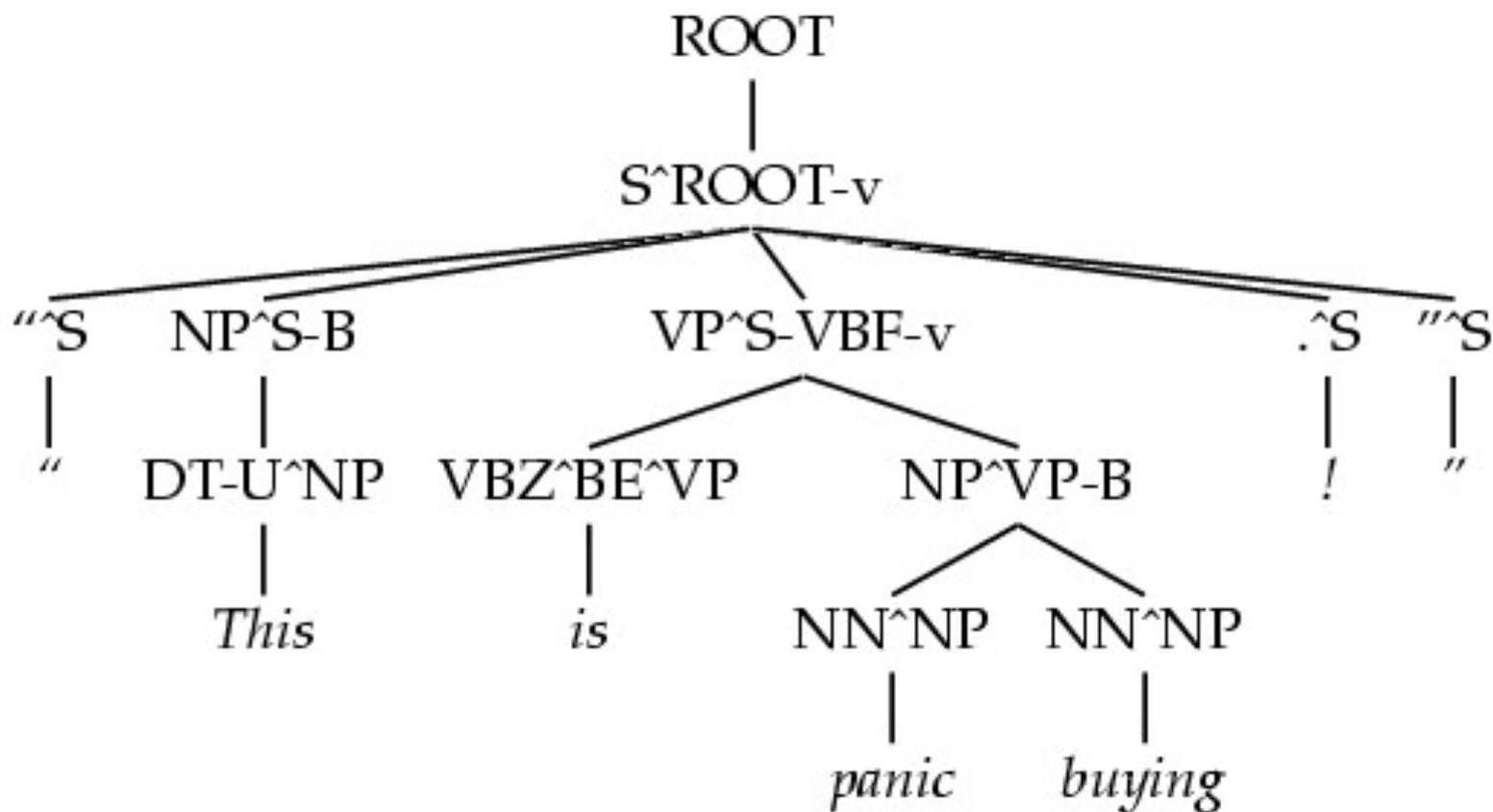
Annotation	F1	Size
Previous	78.3	8.0K
SPLIT-IN	80.3	8.1K

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	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
Unlexicalized	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.
- Lots of room to improve – more complex models next.