Algorithms for NLP

Parsing III

Anjalie Field – CMU

Slides adapted from: Dan Klein – UC Berkeley

Taylor Berg-Kirkpatrick, Yulia Tsvetkov, Maria Ryskina – CMU

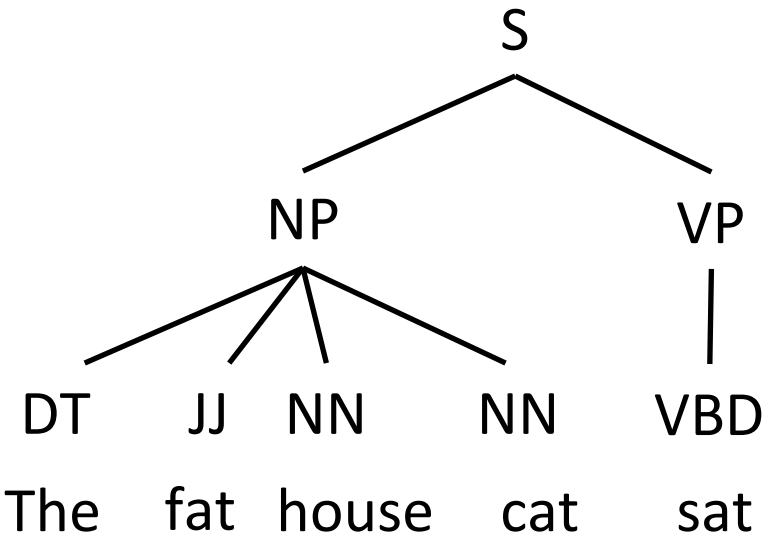
Overview: Improvements to CKY

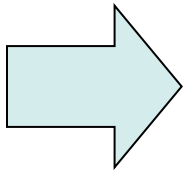
- Tree Binarization
- Relaxing independence assumptions
- Speeding up
- Incorporating word features

Binarization

Treebank PCFGs

- We can take a grammar straight off a tree, using counts to estimate probabilities





$S \rightarrow NP\ VP$ 1

$NP \rightarrow DT\ JJ\ NN\ NN$ 1

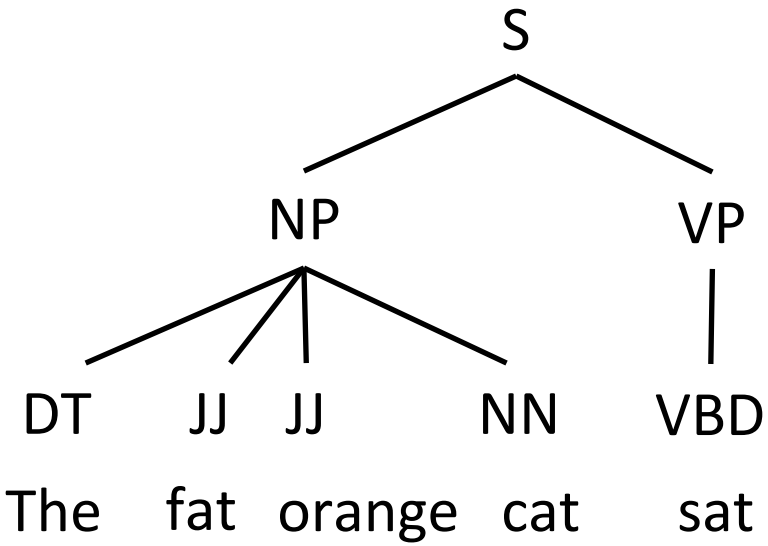
$VP \rightarrow VBD$ 1

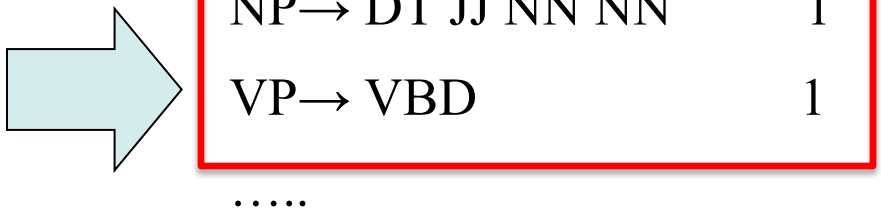
…..

- Can we use CKY to parse sentences according to this grammar?

Treebank PCFGs

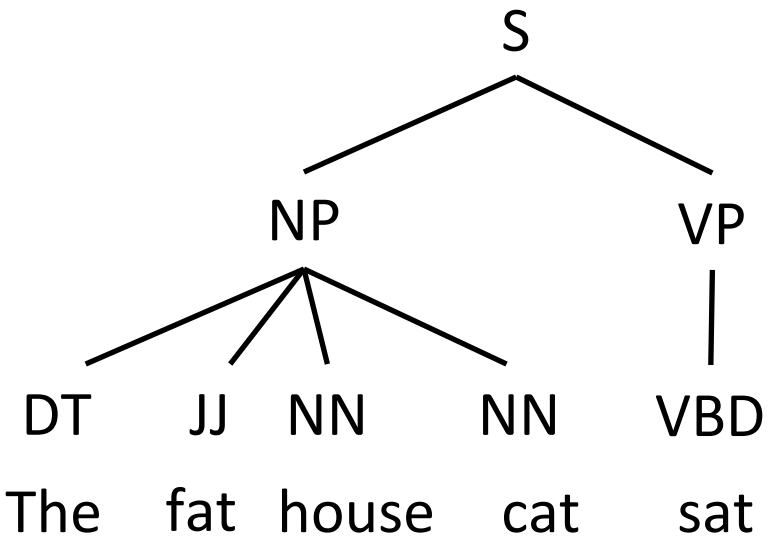
- We can take a grammar straight off a tree, using counts to estimate probabilities





- Vanilla CKY only allows *binary* rules

Option 1: Binarize the Grammar



S→ NP VP

NP→ DT JJ NN NN

VP→ VBD

S→ NP VP

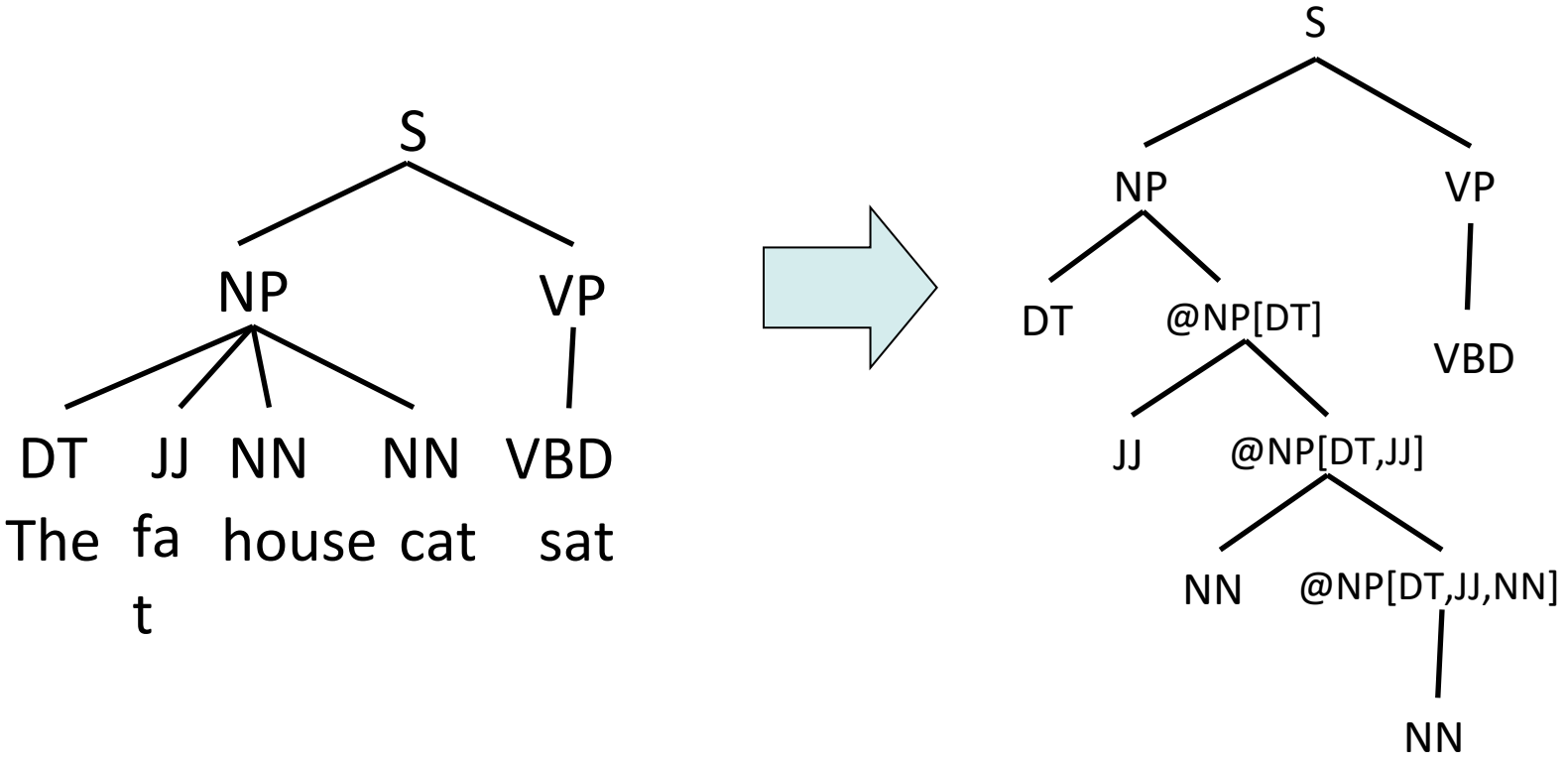
S→ NP VBD

NP→ DT @NP[DT]

@NP[DT]→ JJ @NP[DT JJ]

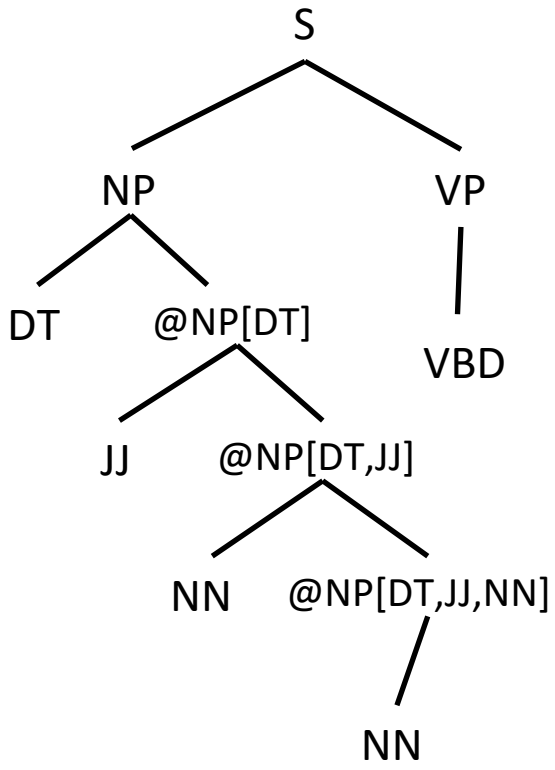
@NP[DT JJ]→ NN NN

Option 2: Binarize the Tree



- Can we use CKY to parse sentences according to the grammar pulled from this tree?

CKY: Modifications for Unary Rules



Binary Rules:

S→ NP VP

NP→ DT @NP[DT]

@NP[DT]→ JJ @NP[DT JJ]

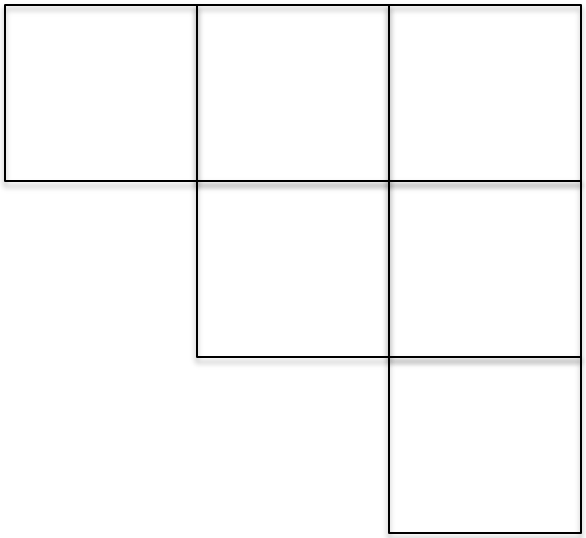
@NP[DT JJ]→ NN @NP[DT,JJ,NN]

Unary Rules:

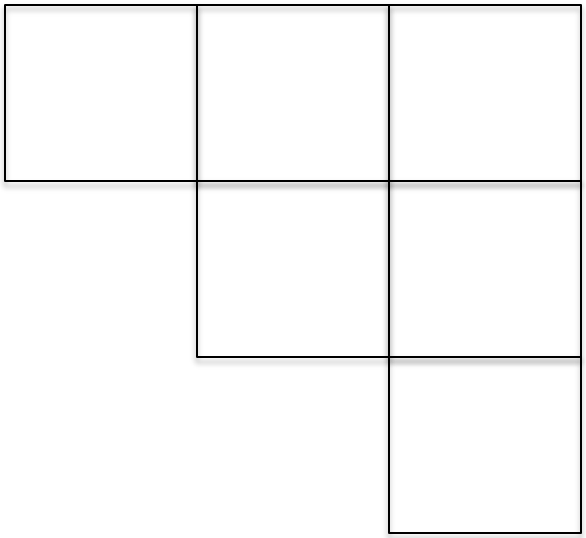
VP→ VBD

@NP[DT,JJ,NN] → NN

CKY: Incorporate Unary Rules



- Binary chart: Store the scores of non-terminals after applying binary rules
- Fill by applying rules to elements of the unary chart



- Unary chart: Store the scores of non-terminals after apply unary rules
- Fill by applying rules to elements of the binary chart

CKY with TreeBank PCFG

[Charniak 96]

- With these modifications, given a treebank we can:
  - Binarize the trees
  - Learn a PCFG from the binarized trees
  - Use the unary-binary chart variant of CKY to obtain parse trees for new sentences
- Does this work?

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.

CKY with TreeBank PCFG

[Charniak 96]

- With these modifications, given a treebank we can:
  - Binarize the trees
  - Learn a PCFG from the binarized trees
  - Use the unary-binary chart variant of CKY to obtain parse trees for new sentences
- Does this work?

Model	*F1*
Baseline	72.0

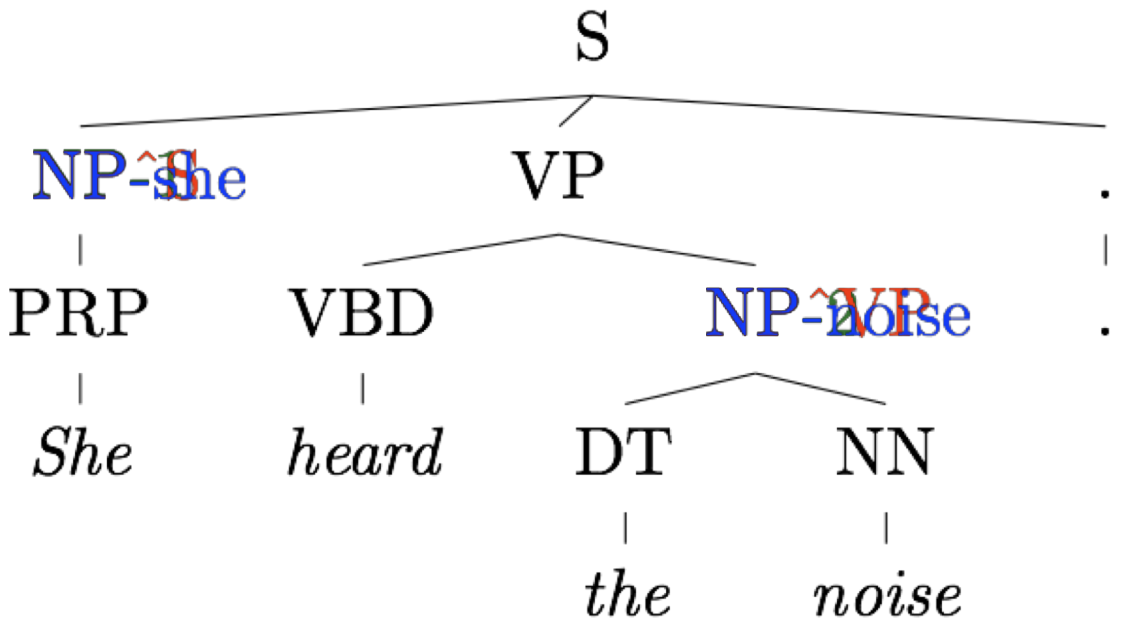
Model Assumptions

- Place Invariance
  - The probability of a subtree does not depend on where in the string the words it dominates are
- Context-free
  - The probability of a subtree does not depend on words not dominated by the subtree
- Ancestor-free
  - The probability of a subtree does not depend on nodes in the derivation outside the tree

Model Assumptions

- We can relax some of these assumptions by *enriching* our grammar
  - We're already doing this in binarization
- Structured Annotation [Johnson '98, Klein&Manning '03]
  - Enrich with features about surrounding nodes
- Lexicalization [Collins '99, Charniak '00]
  - Enrich with word features
- Latent Variable Grammars [Matsuzaki et al. '05, Petrov et al. '06]

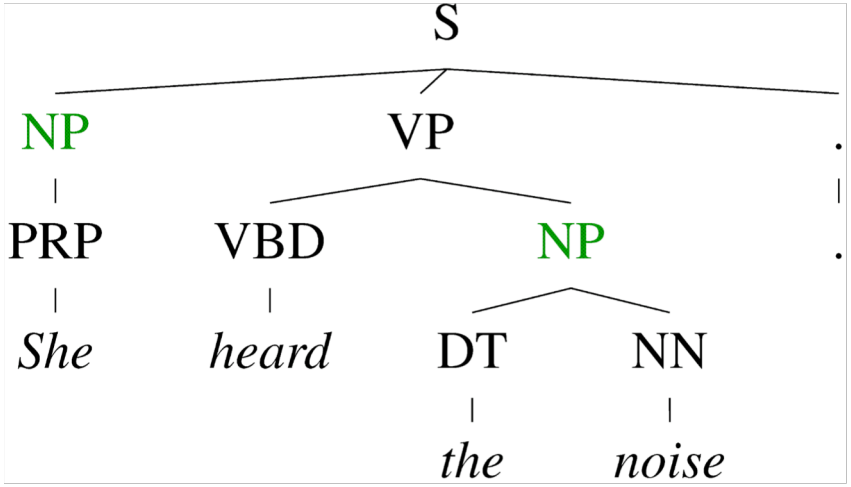
Grammar Refinement



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

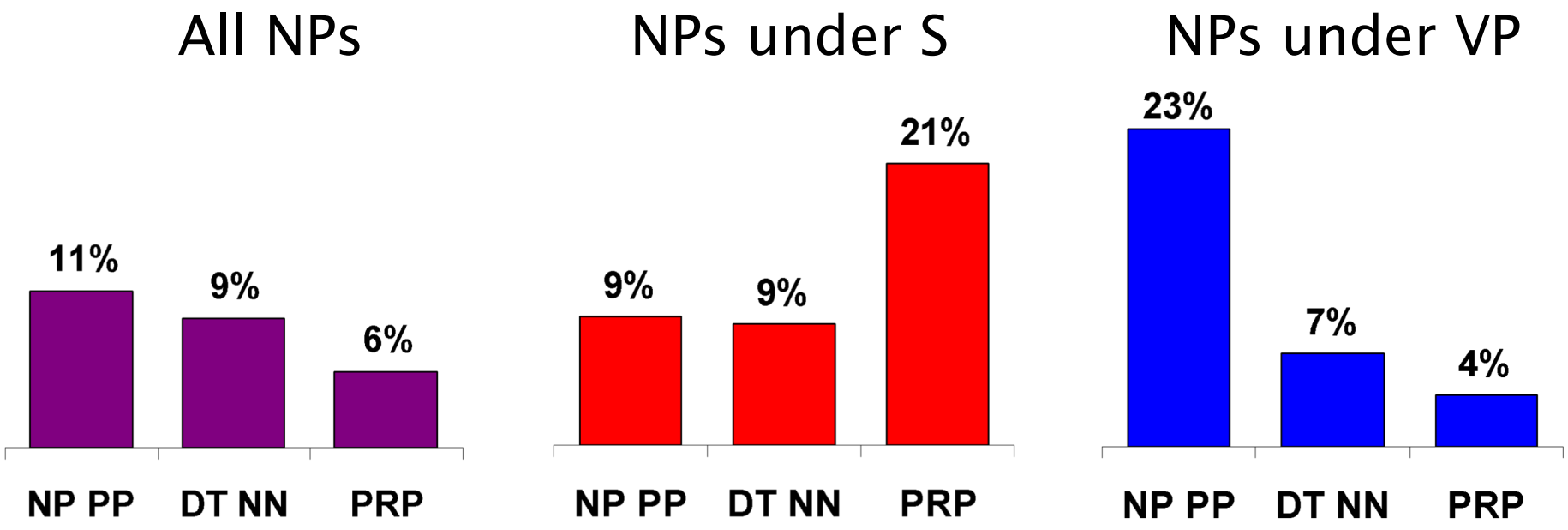
Structural Annotation

Ancestor-free assumption



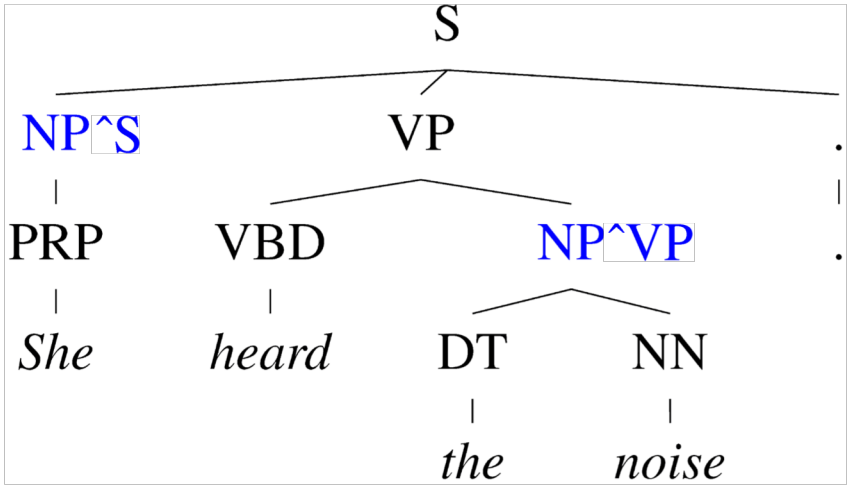
- Not every NP expansion can fill every NP slot

Ancestor-free assumption



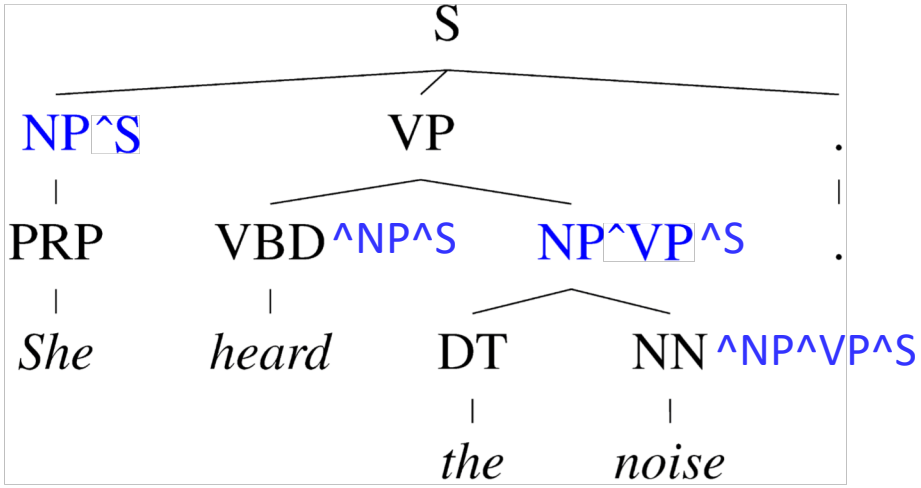
- Example: the expansion of an NP is highly dependent on the parent of the NP (i.e., subjects vs. objects).
- Also: the subject and object expansions are correlated!

Parent Annotation



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

Parent Annotation

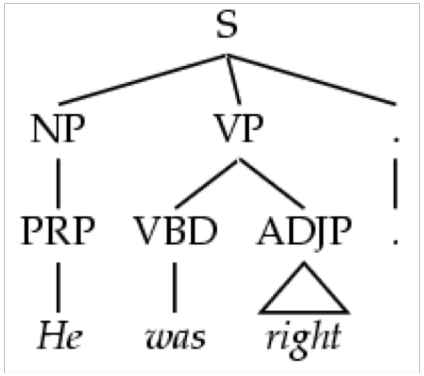


- Why stop at 1 parent?

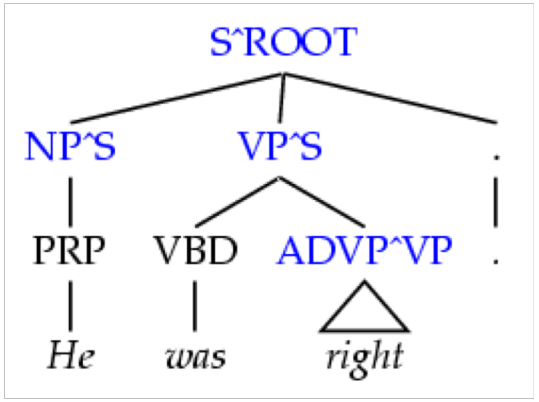
Vertical Markovization

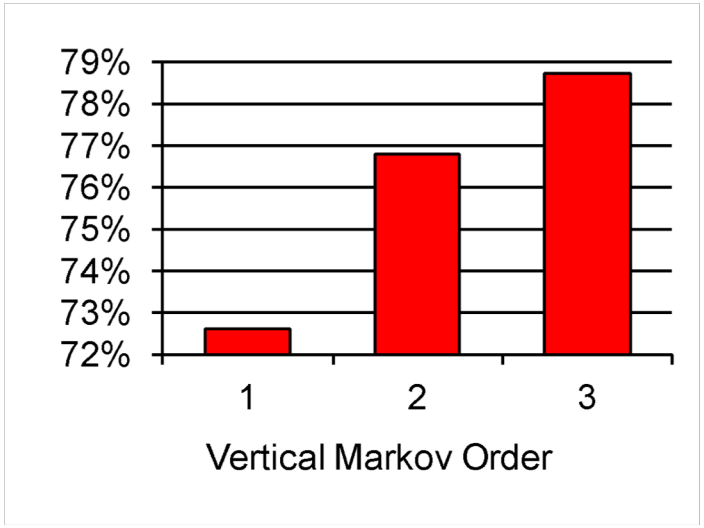
- Vertical Markov order: rewrites depend on past k ancestor nodes.
  (cf. parent annotation)

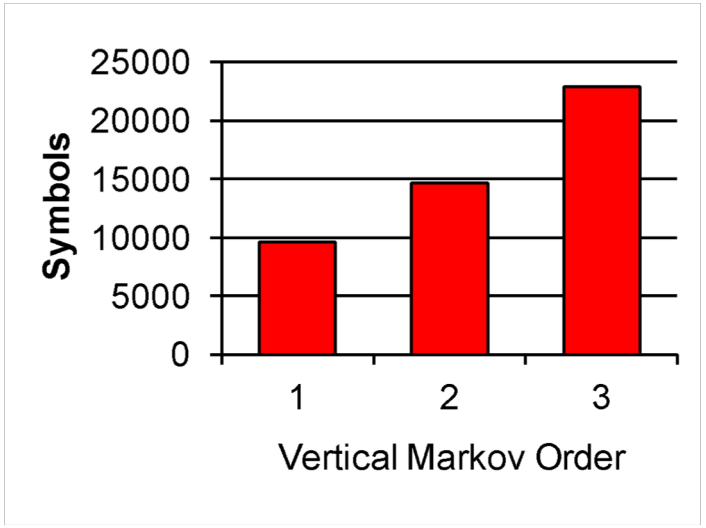
Order 1



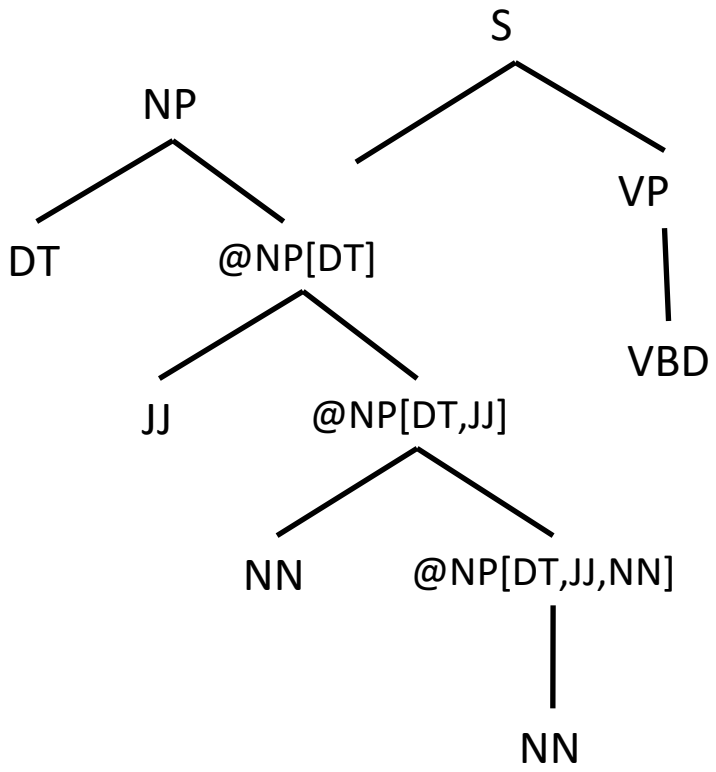
Order 2







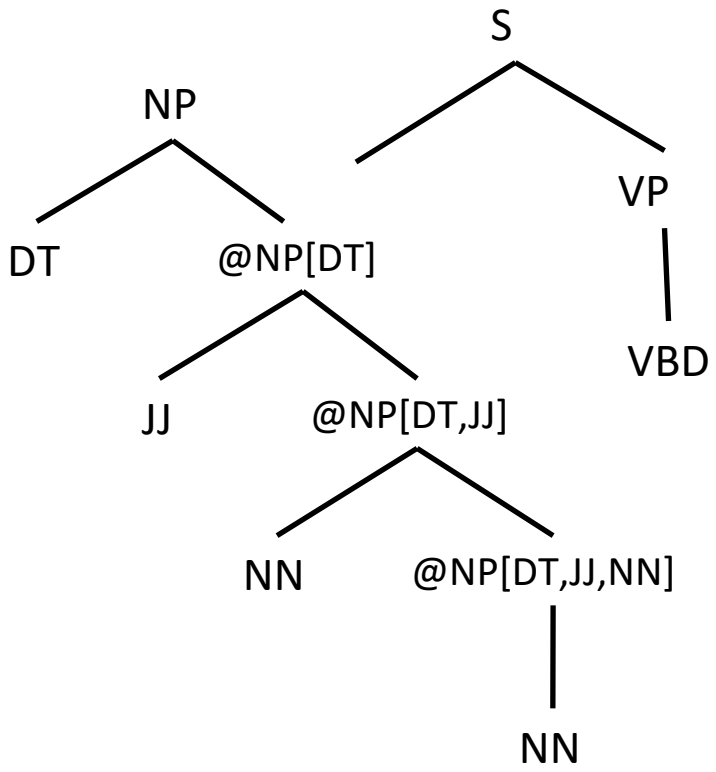
Back to our binarized tree



- How much parent annotating are we doing?

The fat house cat sat

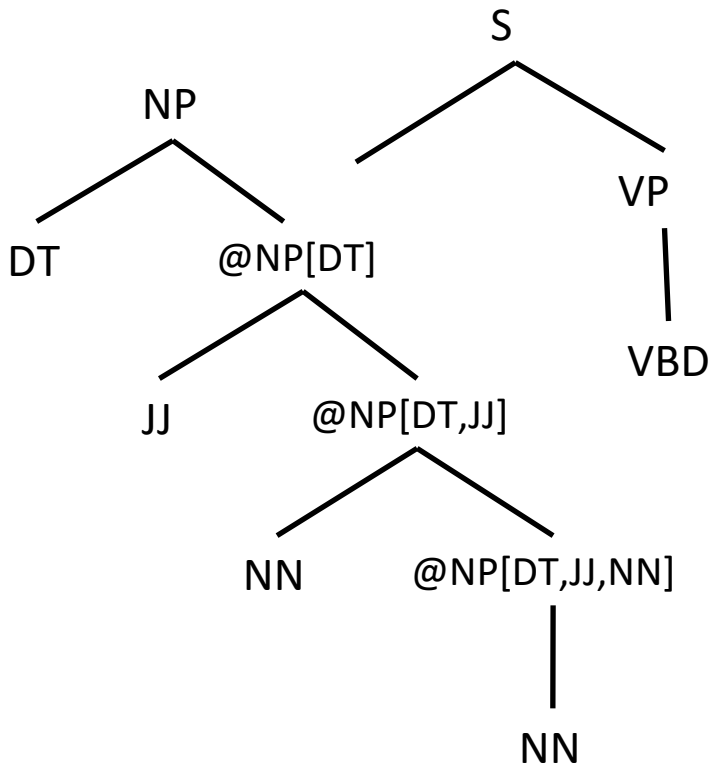
Back to our binarized tree



- Are we doing any other structured annotation?

The fat house cat sat

Back to our binarized tree

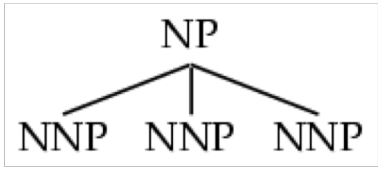


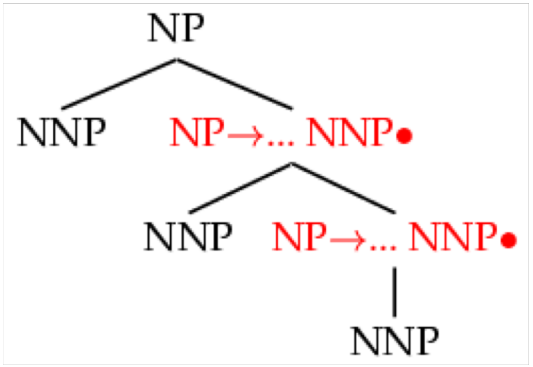
- We're remembering nodes to the left
- If we call parent annotation "vertical" than this is "horizontal"

The fat house cat sat

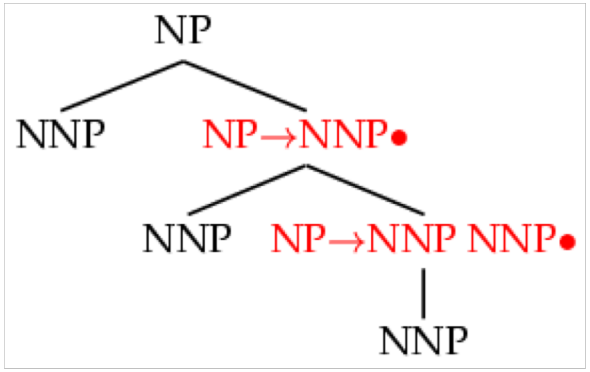
Horizontal Markovization

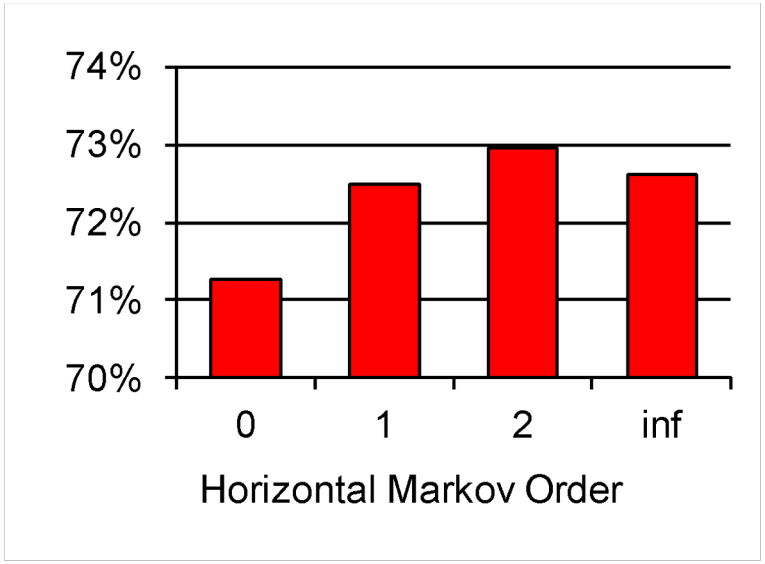
Order 1

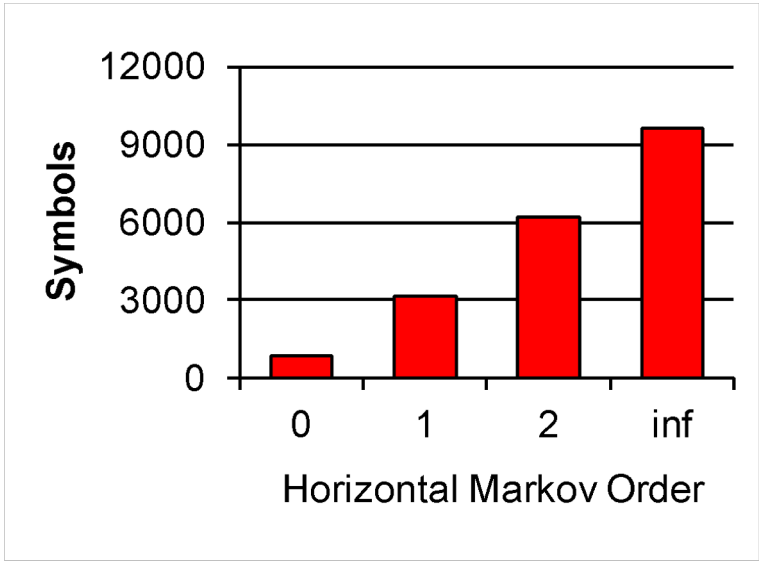




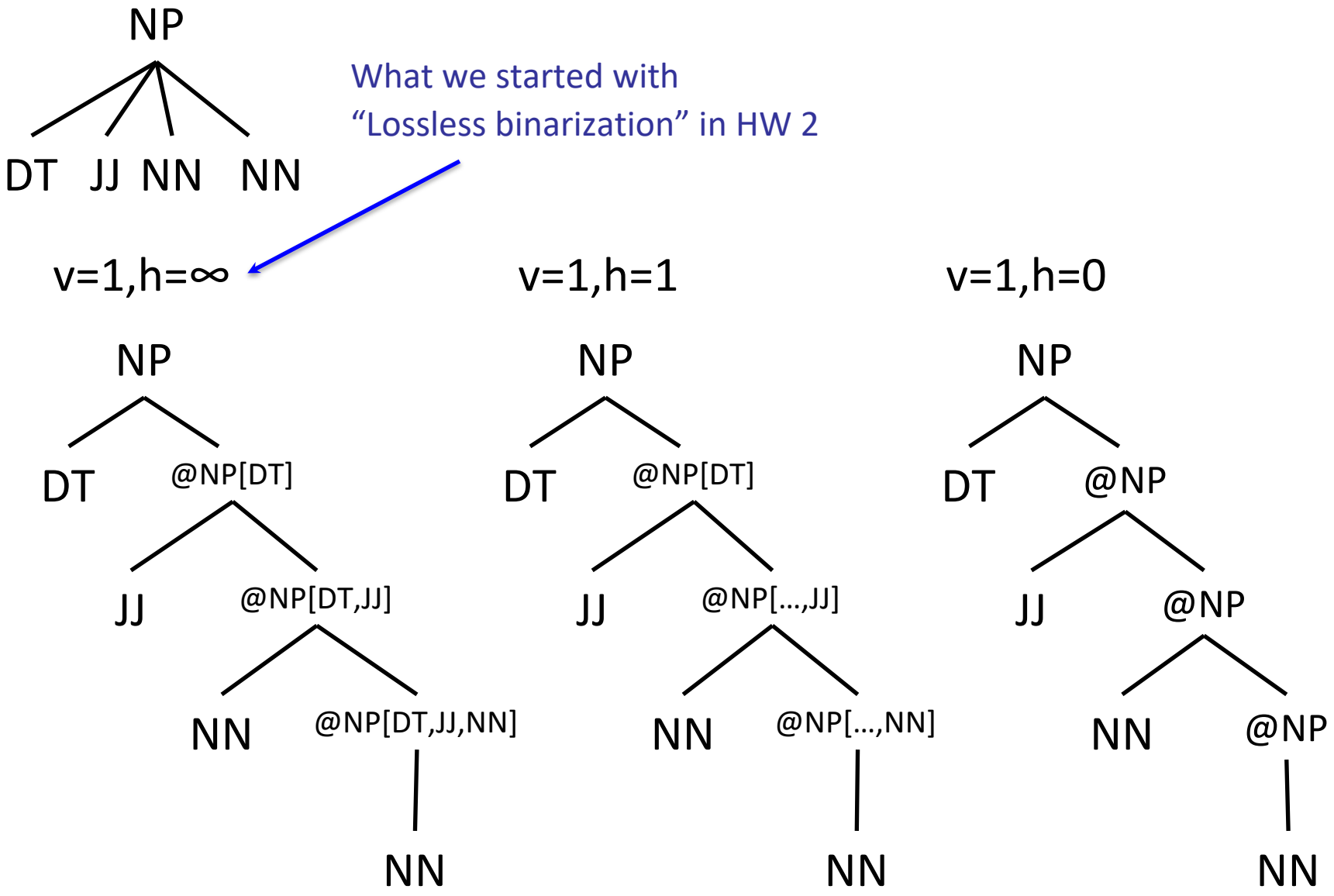
Order ∞



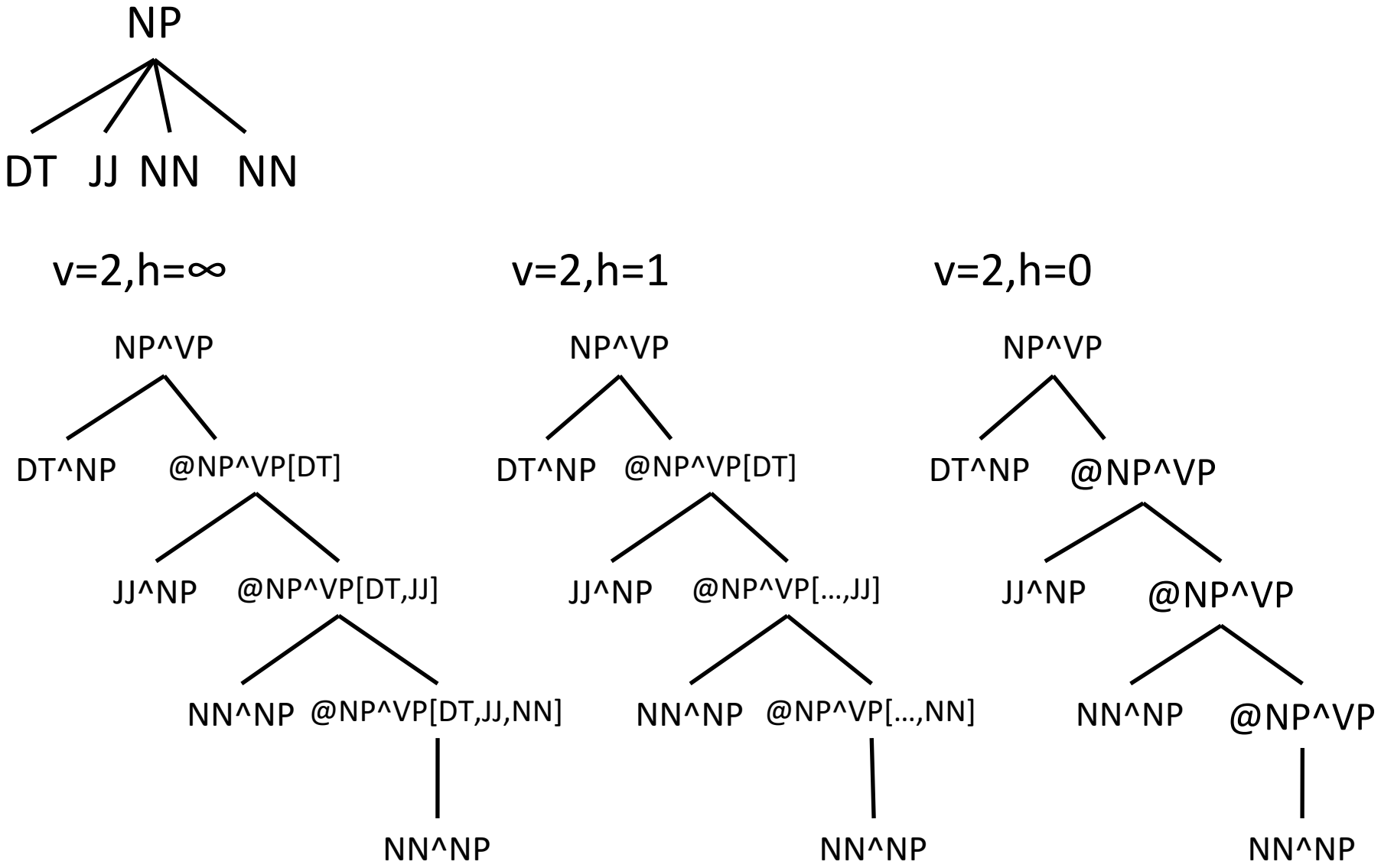




Binarization / Markovization

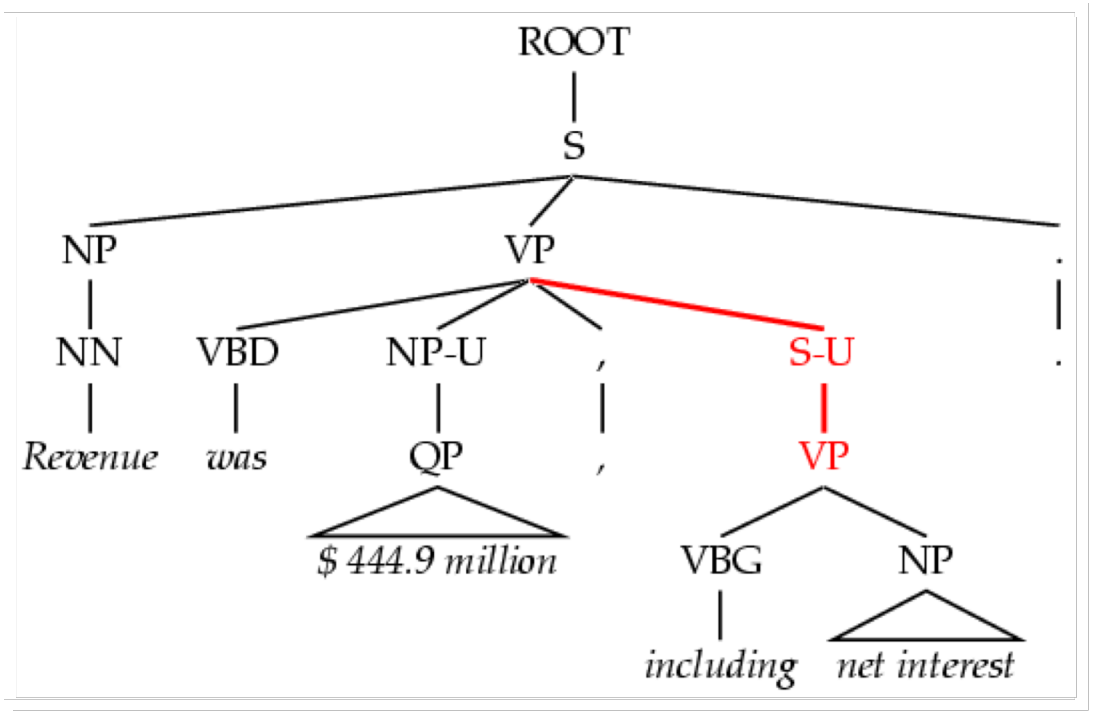


Binarization / Markovization



Unary Splits

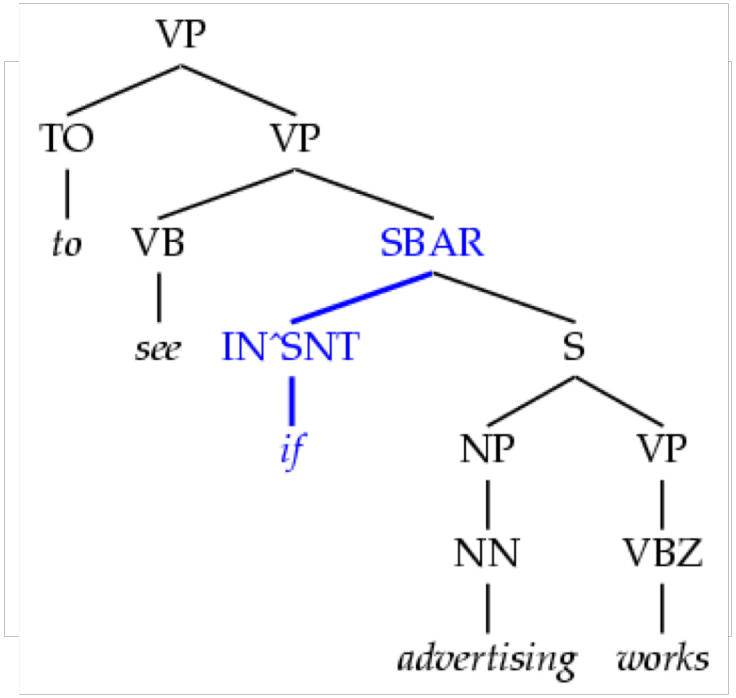
- Problem: unary rewrites used to transmute categories so a high-probability rule can be used.
- Solution: Mark unary rewrite sites with -U



Annotation	F1	Size
Base	77.8	7.5K
UNARY	78.3	8.0K

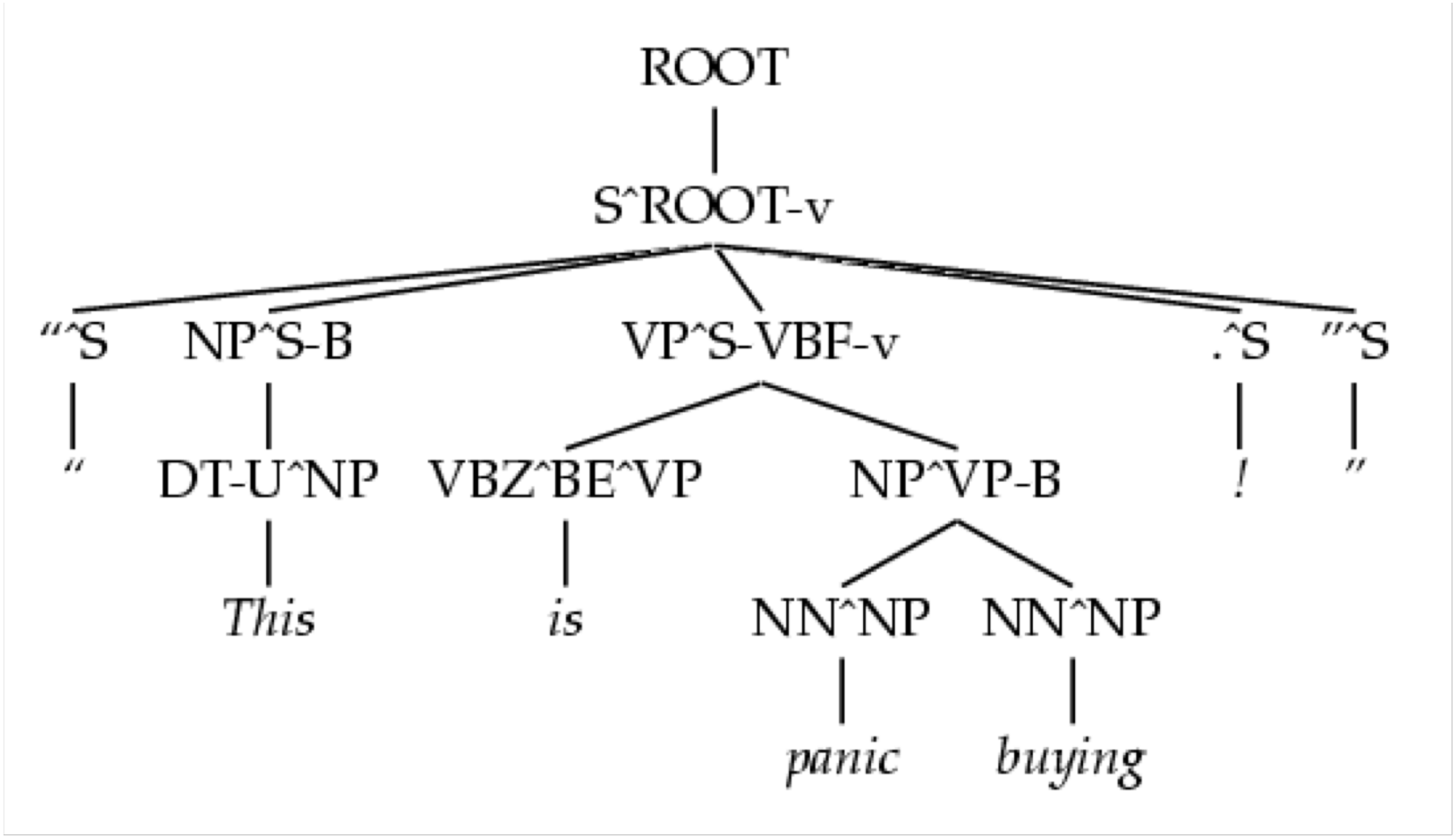
Tag Splits

- 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

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.

Efficient Parsing for Structural Annotation

Overview: Coarse-to-Fine

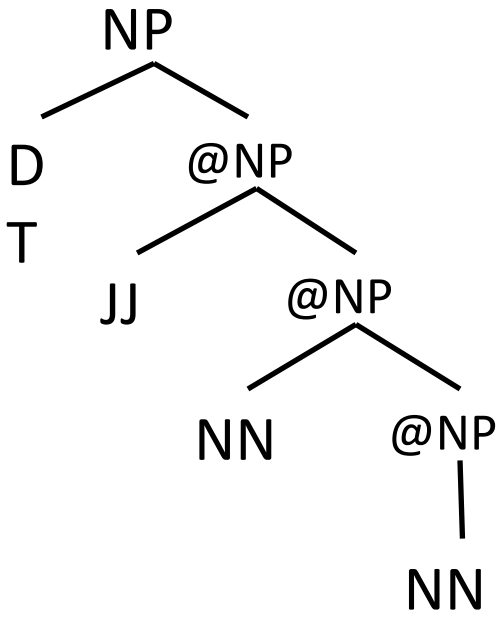
- We've introduce a lot of new symbols in our grammar: do we always need to consider all these symbols?
- Motivation:
  - If any NP is unlikely to span these words, than NP^S[DT], NP^VB[DT], NP^S[JJ], etc. are all unlikely
- High level:
  - First pass: compute probability that a coarse symbol spans these words
  - Second pass: parse as usual, but skip fine symbols that correspond with unprobable coarse symbols

Defining Coarse/Fine Grammars

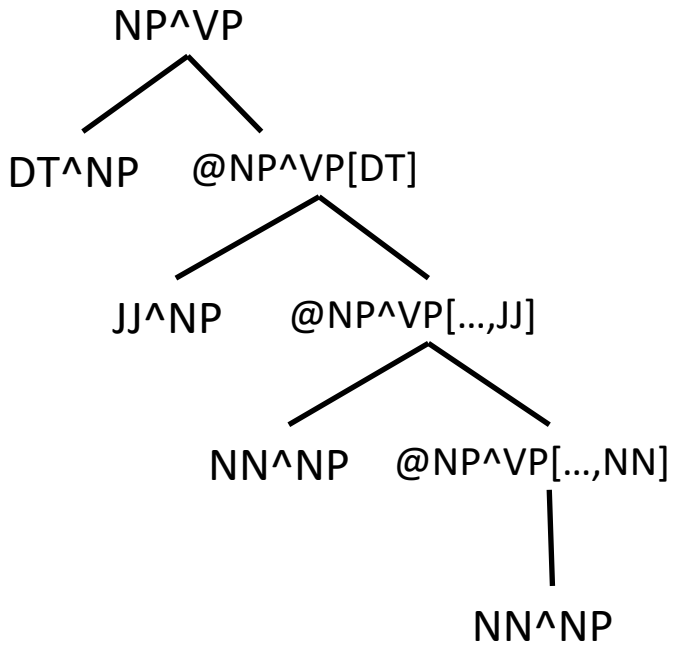
- [Charniak et al. 2006]
  - level 0: ROOT vs. not-ROOT
  - level 1: argument vs. modifier (i.e. two nontrivial nonterminals)
  - level 2: four major phrasal categories (verbal, nominal, adjectival and prepositional phrases)
  - level 3: all standard Penn treebank categories
- Our version: stop at 2 passes

Grammar Projections

Coarse Grammar



Fine Grammar



NP → DT @NP

NP^VP → DT^NP @NP^VP[DT]

Note: X-Bar Grammars are projections with rules like XP → Y @X or XP → @X Y or @X → X

Grammar Projections

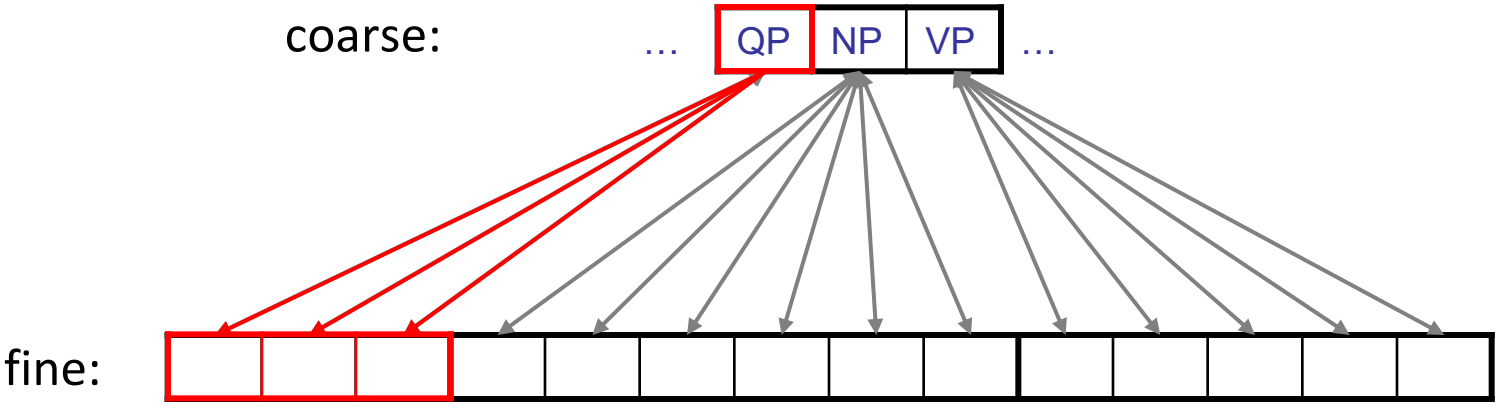
Coarse Symbols	Fine Symbols
NP	NP^VP
	NP^S
@NP	@NP^VP[DT]
	@NP^S[DT]
DT	@NP^VP[…,JJ]
	@NP^S[…,JJ]
	DT^NP

Coarse-to-Fine Pruning

For each coarse chart item $X[i,j]$, compute posterior probability P(*X at [i,j] | sentence)*:

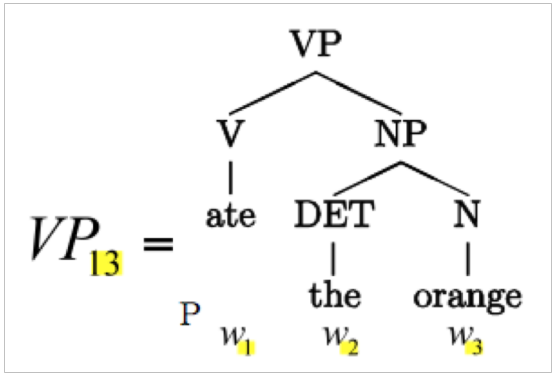
$$\frac{\mathrm{P_{IN}}(X,i,j)\cdot\mathrm{P_{OUT}}(X,i,j)}{\mathrm{P_{IN}}(root,0,n)} < \textit{threshold}$$

E.g. consider the span 5 to 12:



Notation

- Non-terminal symbols (latent variables): $\{N^1, \ldots, N^n\}$
- Sentence (observed data): $\{w_1, \ldots, w_m\} = w_{1m}$
- N^j_{pq} denotes that N^j spans w_{pq} in the sentence



Inside probability

Definition (compare with backward prob for HMMs):

$$\beta_j(p,q) = P(w_p, \ldots, w_q | N^j_{pq}, G) = P(N^j_{pq} \rightarrow w_{pq} | G)$$

Computed recursively

Base case: $\beta_j(k,k) = P(w_k | N^j_{kk}, G) = P(N_j \rightarrow w_k | G)$

Induction:

$$\beta_j(p,q) = \sum_{rs} \sum_{d=p}^{q-1} P(N^j \rightarrow N^r N^s) \beta_r(p,d) \beta_s(d+1,q)$$

The grammar is binarized

Implementation: PCFG parsing

```
for each max from 2 to n
  for each min from max - 2 down to 0
    for each syntactic category C
      double best = undefined
      for each binary rule C -> C1 C2
        for each mid from min + 1 to max - 1
          double t1 = chart[min][mid][C1]
          double t2 = chart[mid][max][C2]
          double candidate = t1 * t2 * p(C -> C1 C2)
          if candidate > best then
            best = candidate
        chart[min][max][C] = best
```

Implementation: inside

```
for each max from 2 to n
  for each min from max - 2 down to 0
    for each syntactic category C
      double total = 0.0
      for each binary rule C -> C1 C2
        for each mid from min + 1 to max - 1
          double t1 = chart[min][mid][C1]
          double t2 = chart[mid][max][C2]
          double candidate = t1 * t2 * p(C -> C1 C2)
          total = total + candidate
      chart[min][max][C] = best
```

Implementation: inside

```
for each max from 2 to n
  for each min from max - 2 down to 0
    for each syntactic category C
      double total = 0.0
      for each binary rule C -> C1 C2
        for each mid from min + 1 to max - 1
          double t1 = chart[min][mid][C1]
          double t2 = chart[mid][max][C2]
          double candidate = t1 * t2 * p(C -> C1 C2)
          total = total + candidate
      chart[min][max][C] = best
```

$$\beta_j(p,q) = \sum_{rs} \sum_{d=p}^{q-1} P(N^j \rightarrow N^r N^s) \beta_r(p,d) \beta_s(d+1,q)$$

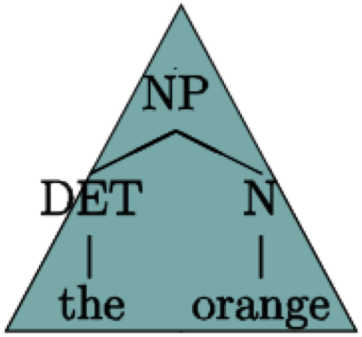
Implementation: inside

```
for each max from 2 to n
  for each min from max - 2 down to 0
    for each syntactic category C
      double total = 0.0
      for each binary rule C -> C1 C2
        for each mid from min + 1 to max - 1
          double t1 = chart[min][mid][C1]
          double t2 = chart[mid][max][C2]
          double candidate = t1 * t2 * p(C -> C1 C2)
          total = total + candidate
      chart[min][max][C] = best
```

$$\beta_j(p,q) = \sum_{rs} \sum_{d=p}^{q-1} P(N^j \rightarrow N^r N^s) \beta_r(p,d) \beta_s(d+1,q)$$

Inside probability: example

NP→DET N	0.8	NP→N	0.2
DET→a	0.6	DET→the	0.4
N→apple	0.8	N→orange	0.2



$\beta_{DET}(1,1)$

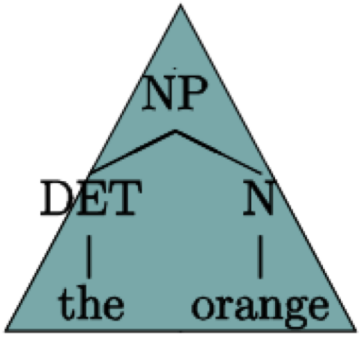
$\beta_{N}(2,2)$ =

$\beta_{NP}(1,2)$

$\beta_{NP}(1,2)$

Inside probability: example

NP→DET N	0.8	NP→N	0.2
DET→a	0.6	DET→the	0.4
N→apple	0.8	N→orange	0.2



$$\beta_{DET}(1,1) = P(the \mid DET_{11}, G) = P(DET \rightarrow the \mid G) = 0.4$$

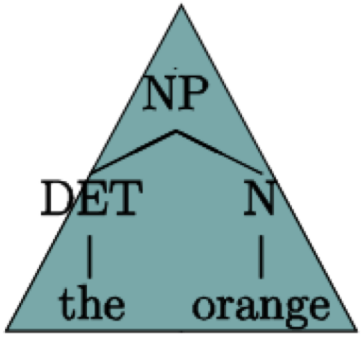
$$\beta_{N}(2,2) = P(N \rightarrow orange \mid G) = 0.2$$

$$\beta_{NP}(1,2)$$

$$\beta_{NP}(1,2)$$

Inside probability: example

NP→DET N	0.8	NP→N	0.2
DET→a	0.6	DET→the	0.4
N→apple	0.8	N→orange	0.2



$$\beta_{DET}(1,1) = P(the \mid DET_{11}, G) = P(DET \rightarrow the \mid G) = 0.4$$

$$\beta_{N}(2,2) = P(N \rightarrow orange \mid G) = 0.2$$

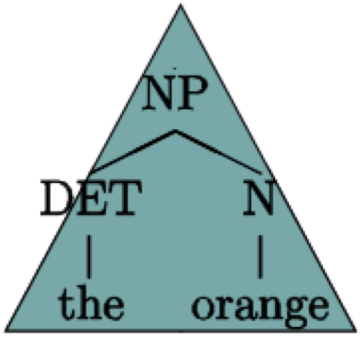
$$\beta_{NP}(1,2) = P(NP \rightarrow DET \cdot N)\beta_{DET}(1,1)\beta_{N}(2,2)$$

$$= 0.8 \qquad \times 0.4 \qquad \times 0.2$$

$$\beta_{NP}(1,2)$$

Inside probability: example

NP→DET N	0.8	NP→N	0.2
DET→a	0.6	DET→the	0.4
N→apple	0.8	N→orange	0.2



$$\beta_{DET}(1,1) = P(the \mid DET_{11}, G) = P(DET \rightarrow the \mid G) = 0.4$$

$$\beta_{N}(2,2) = P(N \rightarrow orange \mid G) = 0.2$$

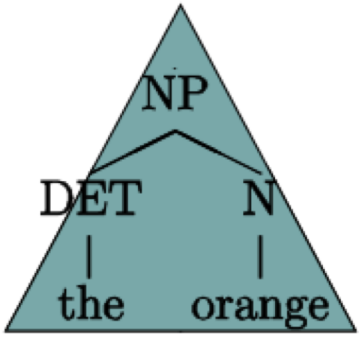
$$\beta_{NP}(1,2) = P(NP \rightarrow DET \cdot N)\beta_{DET}(1,1)\beta_{N}(2,2)$$

$$= 0.8 \quad \times 0.4 \quad \times 0.2$$

$$\beta_{NP}(1,2) = 0.064$$

Inside probability: example

NP→DET N	0.8	NP→N	0.2
DET→a	0.6	DET→the	0.4
N→apple	0.8	N→orange	0.2



$$\beta_{DET}(1,1) = P(the \mid DET_{11}, G) = P(DET \rightarrow the \mid G) = 0.4$$

$$\beta_{N}(2,2) = P(N \rightarrow orange \mid G) = 0.2$$

$$\beta_{NP}(1,2) = P(NP \rightarrow DET \cdot N)\beta_{DET}(1,1)\beta_{N}(2,2)$$

$$= 0.8 \quad \times 0.4 \quad \times 0.2$$

$$\beta_{NP}(1,2) = 0.064$$

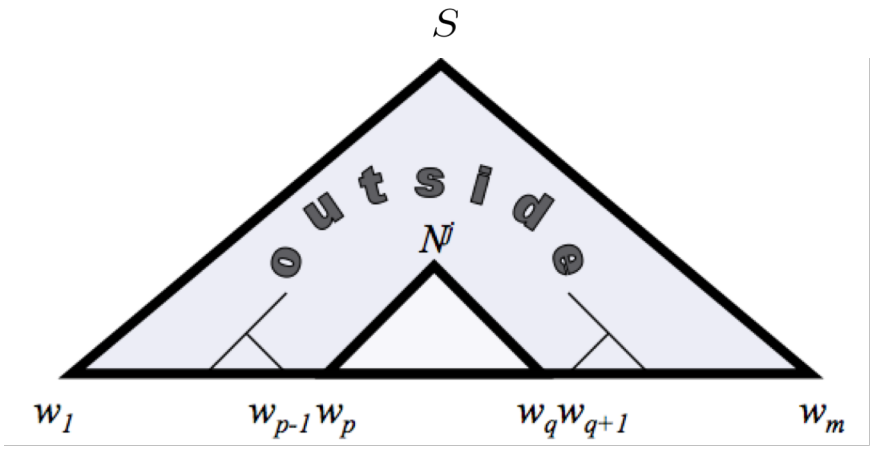
$$\beta_S(1,m) = P(S \rightarrow w_1, \ldots, w_m | G)$$

Outside probability

Definition (compare with forward prob for HMMs):

$$\alpha_j(p,q) = P(w_{1(p-1)}, N^j_{pq}, w_{(q+1)m}|G)$$

The joint probability of starting with *S*, generating words $w_1, \ldots, w_{p-1}$, the non terminal N^j and words $w_{q+1}, \ldots, w_m$



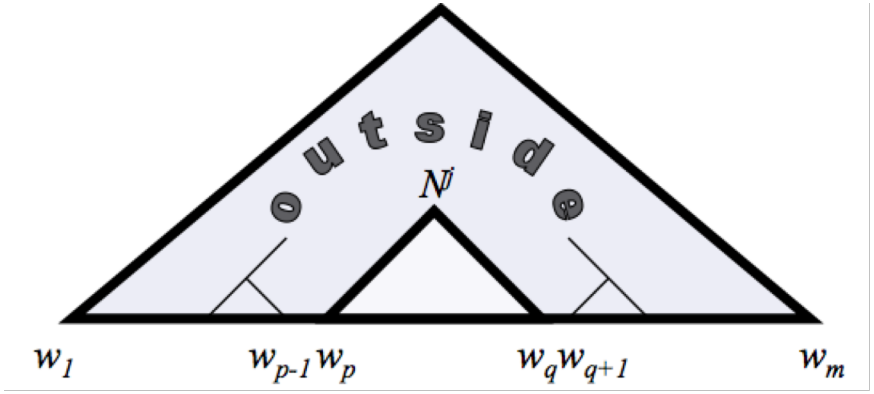
Calculating outside probability

Computed recursively, base case

$\alpha_1(1,m) = \alpha_S(1,m) = 1$ $\alpha_{j\neq 1}(1,m) = 0$

Induction?

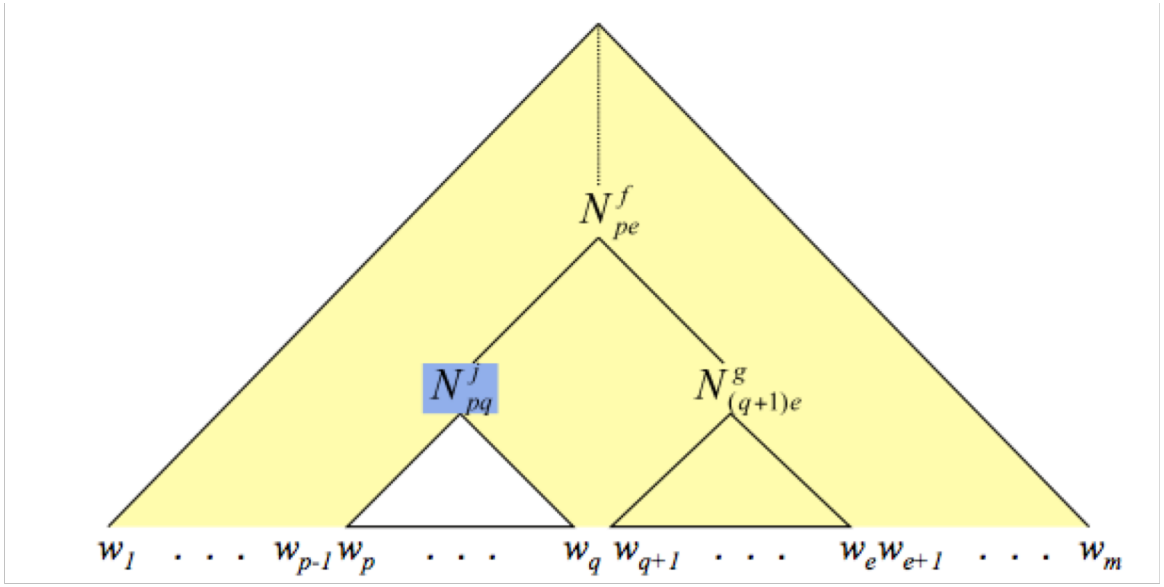
Intuition: N^j_{pq} must be either the L or R child of a parent node. We first consider the case when it is the L child.



Calculating outside probability

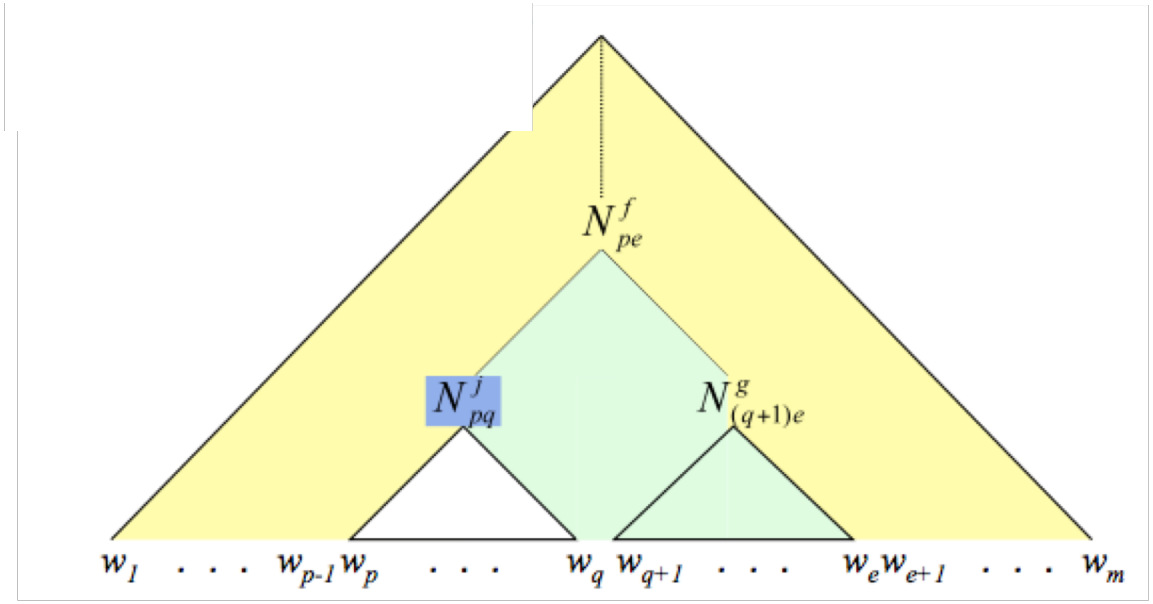
The yellow area is the probability we would like to calculate

How do we decompose it?



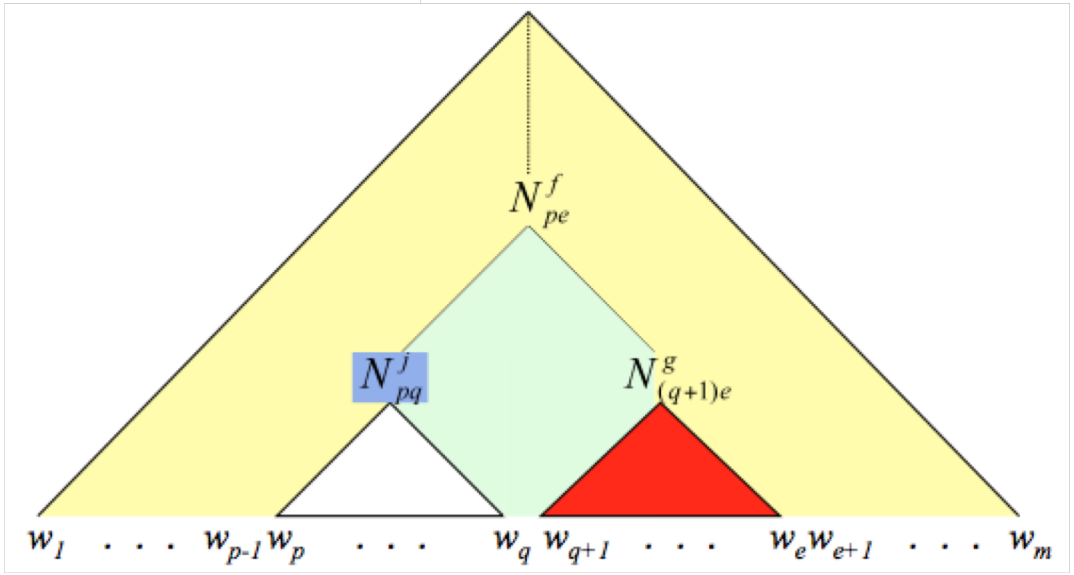
Calculating outside probability

Step 1: We assume that N^f_{pe} is the parent of N^j_{pq}. Its outside probability, $\alpha_f(p,e)$, (represented by the yellow shading) is available recursively. But how do we compute the green part?



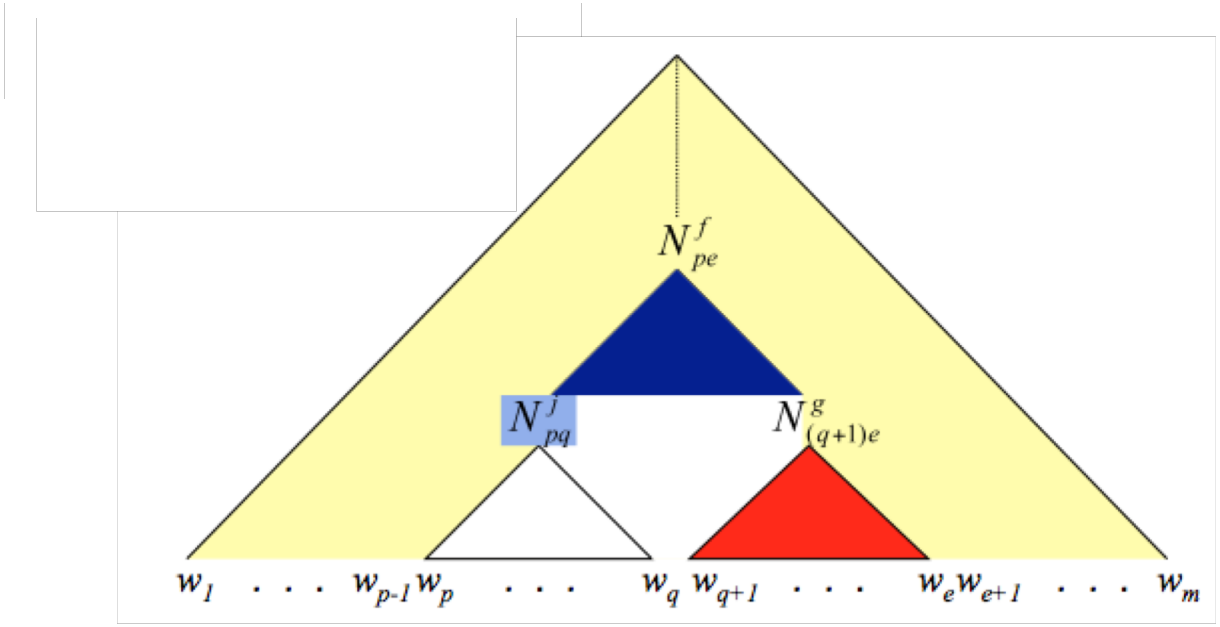
Calculating outside probability

Step 1: The red shaded area is the inside probability for $N^g_{(q+1)e}$, i.e. $\beta_g(q+1, e)$



Calculating outside probability

Step 3: The blue shaded area is just the production $N^f \rightarrow N^j N^g$, the corresponding probability $P(N^f \rightarrow N^j N^g | N^f, G)$

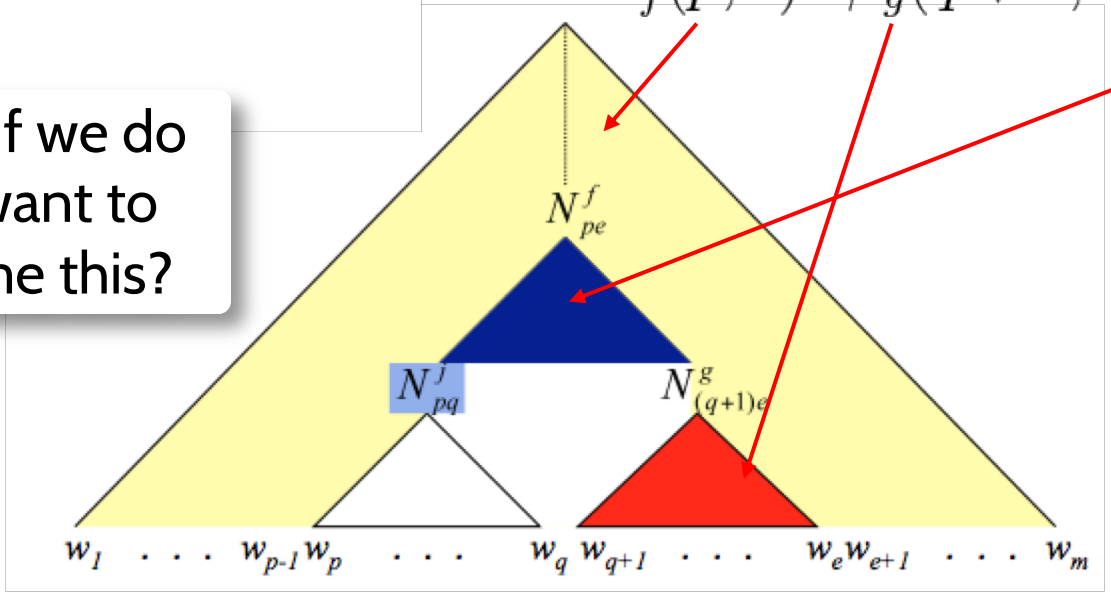


Calculating outside probability

If we multiply the terms together, we have the joint probability corresponding to the yellow, red and blue areas, assuming N^j was the L child of N^f, and give fixed non-terminals *f* and *g*, as well as a fixed partition *e*

$$\alpha_f(p,e) \cdot \beta_g(q+1,e) \cdot P(N^f \rightarrow N^j N^g)$$

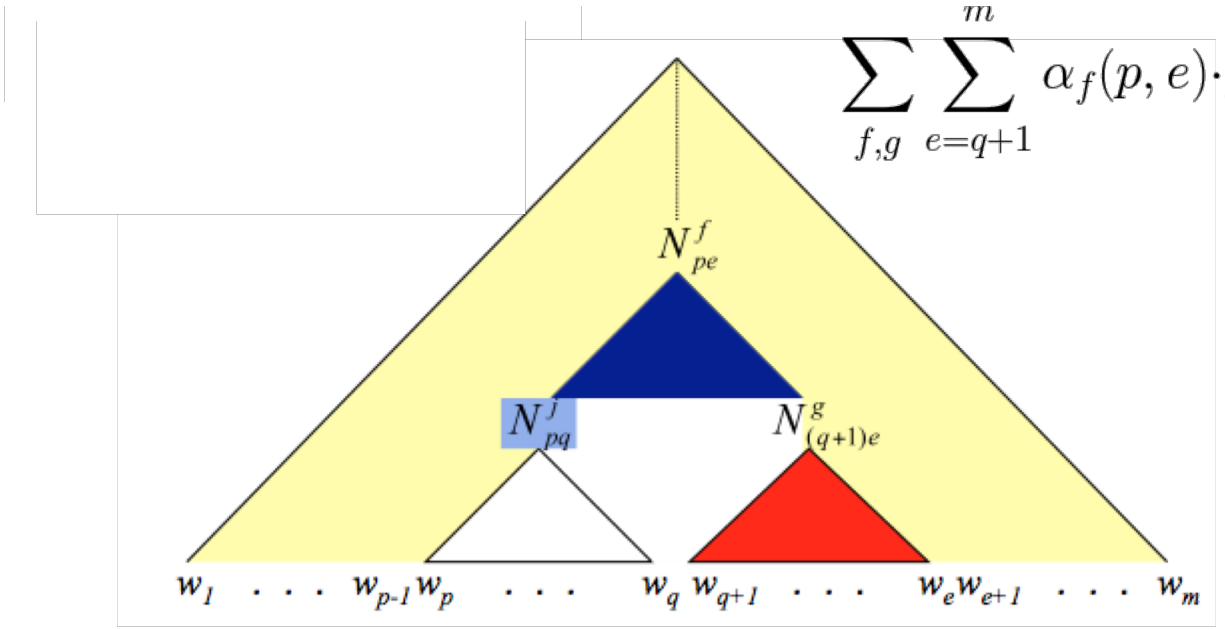
What if we do not want to assume this?



Calculating outside probability

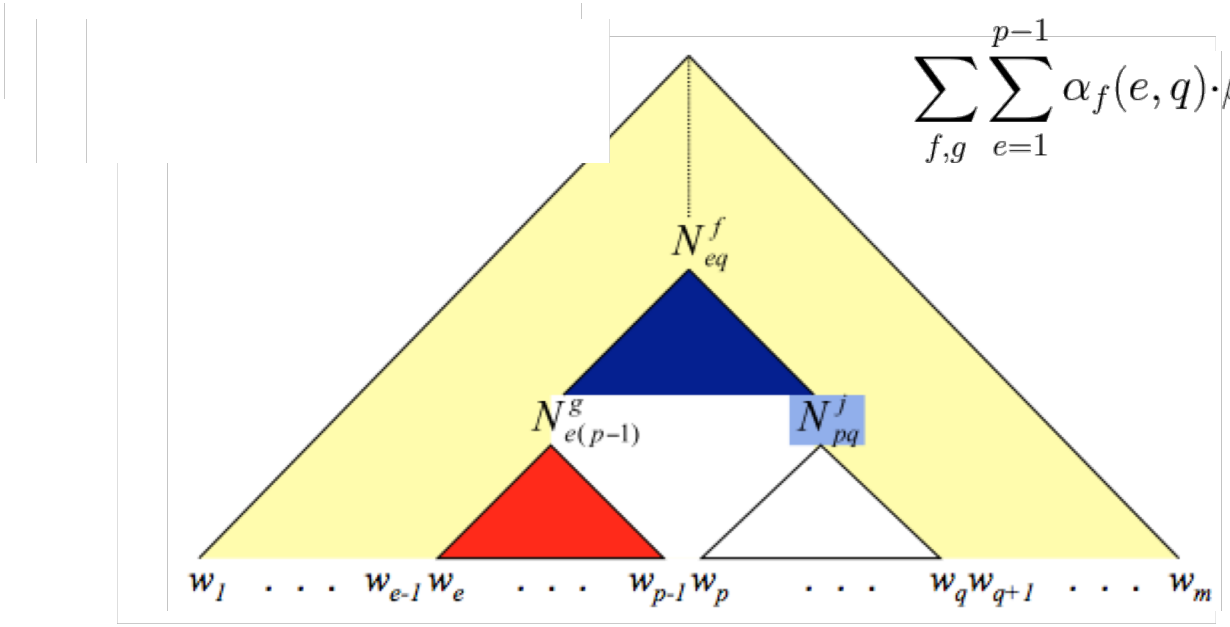
The joint probability corresponding to the yellow, red and blue areas, assuming N^j was the L child of some non-terminal:

$$\sum_{f,g} \sum_{e=q+1}^{m} \alpha_f(p,e) \cdot \beta_g(q+1,e) \cdot P(N^f \rightarrow N^j N^g)$$



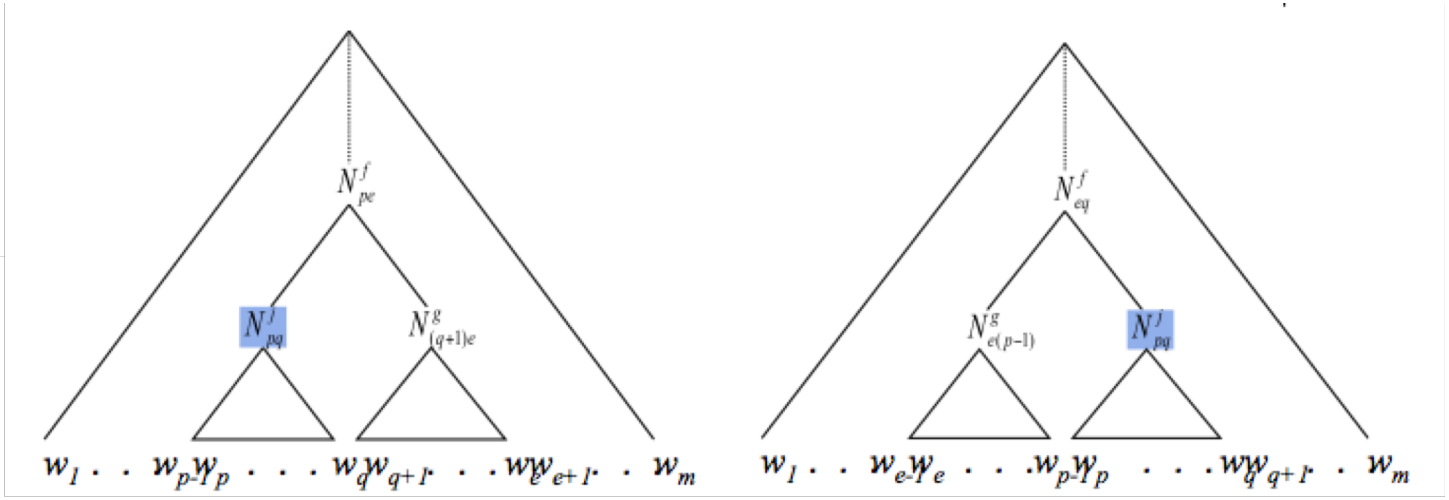
Calculating outside probability

The joint probability corresponding to the yellow, red and blue areas, assuming N^j was the R child of some non-terminal:



Calculating outside probability

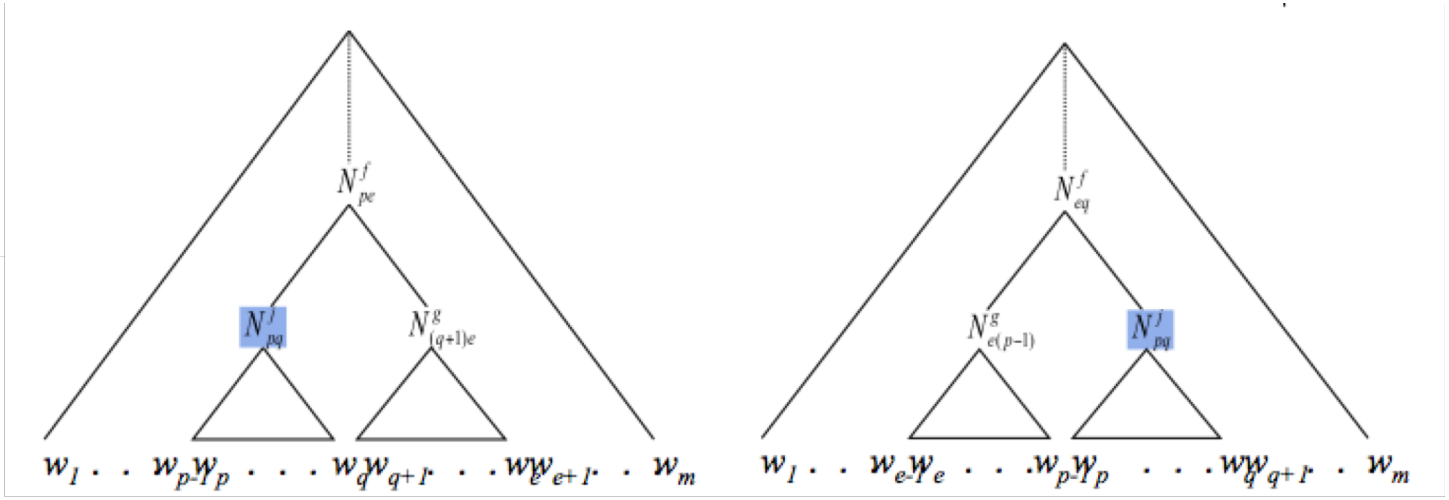
The joint final joint probability (the sum over the L and R cases):



$$\alpha_j(p,q) = \sum_{f,g} \sum_{e=q+1}^{m} \alpha_f(p,e) \cdot \beta_g(q+1,e) \cdot P(N^f \rightarrow N^j N^g) + \sum_{f,g} \sum_{e=1}^{p-1} \alpha_f(e,q) \cdot \beta_g(e,p-1) \cdot P(N^f \rightarrow N^g N^j)$$

Calculating outside probability

The joint final joint probability (the sum over the L and R cases):



$$\alpha_j(p,q) = \sum_{f,g\neq j} \sum_{e=q+1}^{m} \alpha_f(p,e) \cdot \beta_g(q+1,e) \cdot P(N^f \rightarrow N^j N^g) + \sum_{f,g} \sum_{e=1}^{p-1} \alpha_f(e,q) \cdot \beta_g(e,p-1) \cdot P(N^f \rightarrow N^g N^j)$$

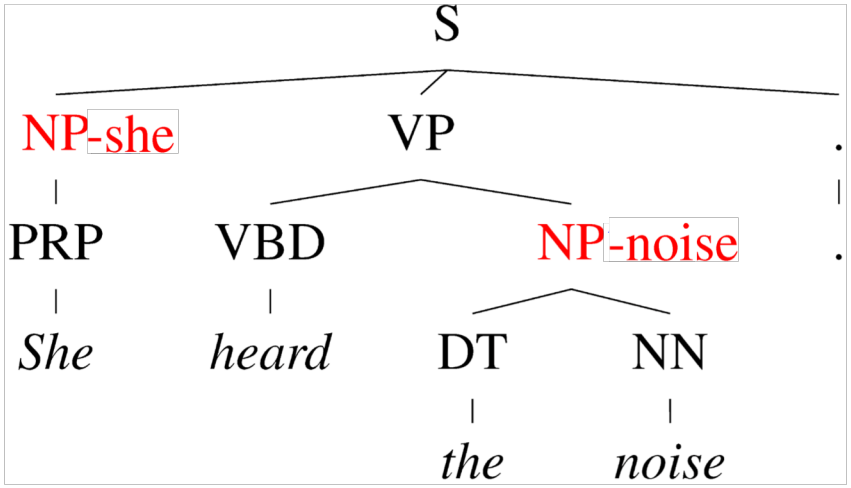
Is C2F an Improvement?

$$\frac{\mathrm{P}_{\mathrm{IN}}(X,i,j)\cdot\mathrm{P}_{\mathrm{OUT}}(X,i,j)}{\mathrm{P}_{\mathrm{IN}}(root,0,n)} < \textit{threshold}$$

- Does coarse-to-fine pruning improve accuracy?
  - If your threshold is too high, it might throw away correct parses
- Does coarse-to-fine pruning improve speed?
  - Maybe, if your threshold is too low pruning might not be very useful

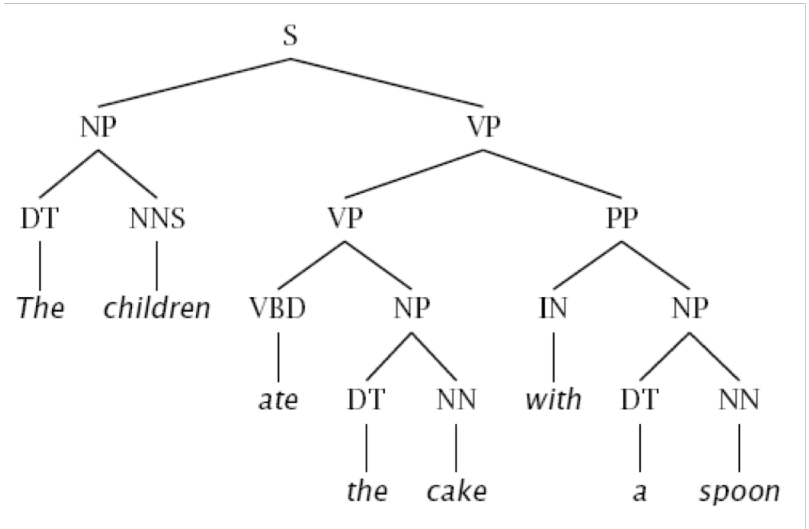
Beyond Structured Annotation: Lexicalization and Latent Variable Grammars

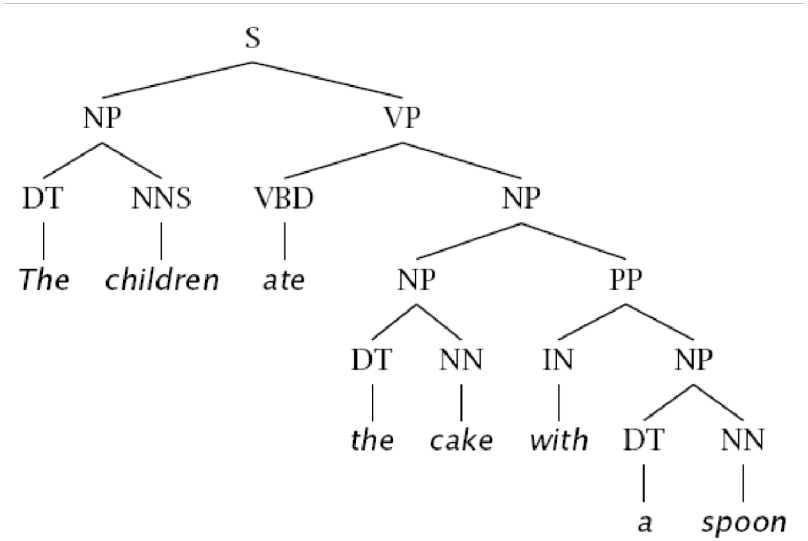
The Game of Designing a Grammar



- 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



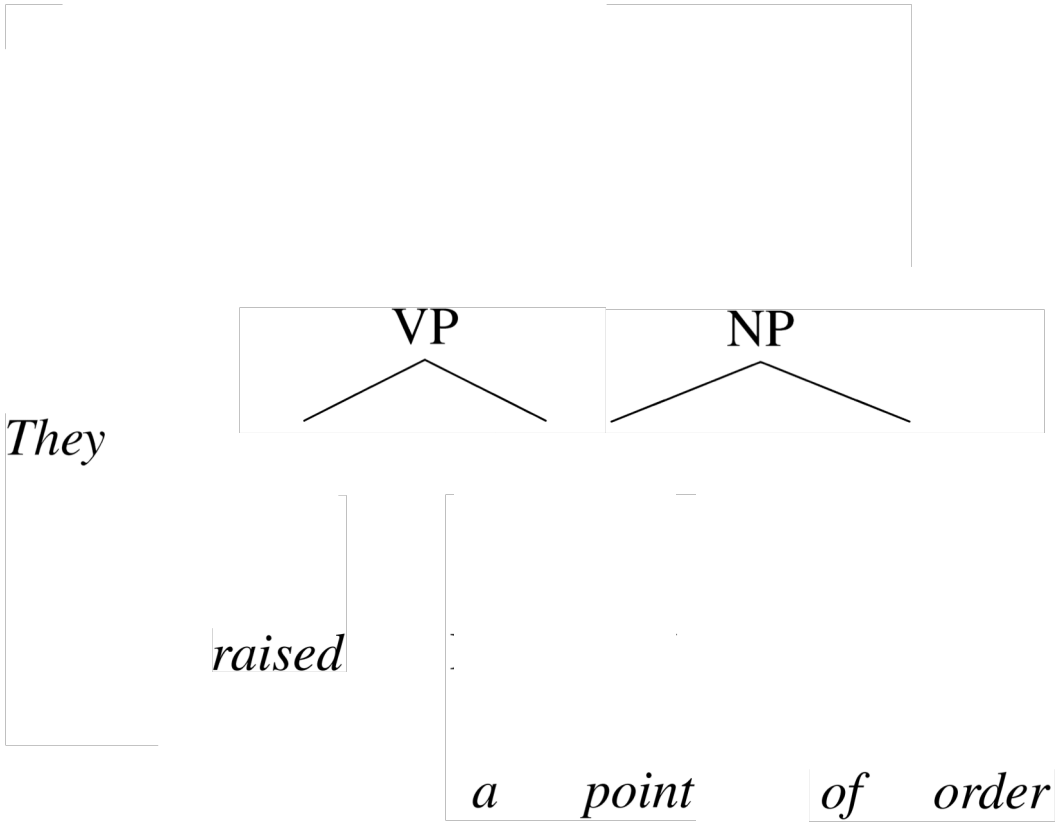


- If we do no annotation, these trees differ only in one rule:
  - VP → VP PP
  - NP → NP PP
- Parse will go one way or the other, regardless of words
- We addressed this in one way with unlexicalized grammars (how?)
- Lexicalization allows us to be sensitive to specific words

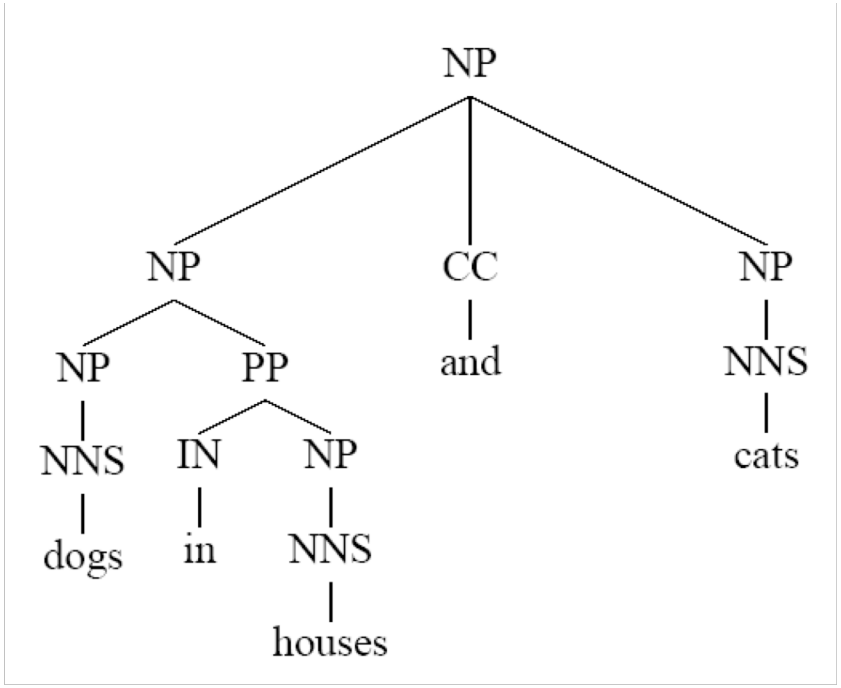
Local tree	Verb *come*	*take*	*think*	*want*
VP → V	9.5%	2.6%	4.6%	5.7%
VP → V NP	1.1%	32.1%	0.2%	13.9%
VP → V PP	34.5%	3.1%	7.1%	0.3%
VP → V SBAR	6.6%	0.3%	73.0%	0.2%
VP → V S	2.2%	1.3%	4.8%	70.8%
VP → V NP S	0.1%	5.7%	0.0%	0.3%
VP → V PRT NP	0.3%	5.8%	0.0%	0.0%
VP → V PRT PP	6.1%	1.5%	0.2%	0.0%

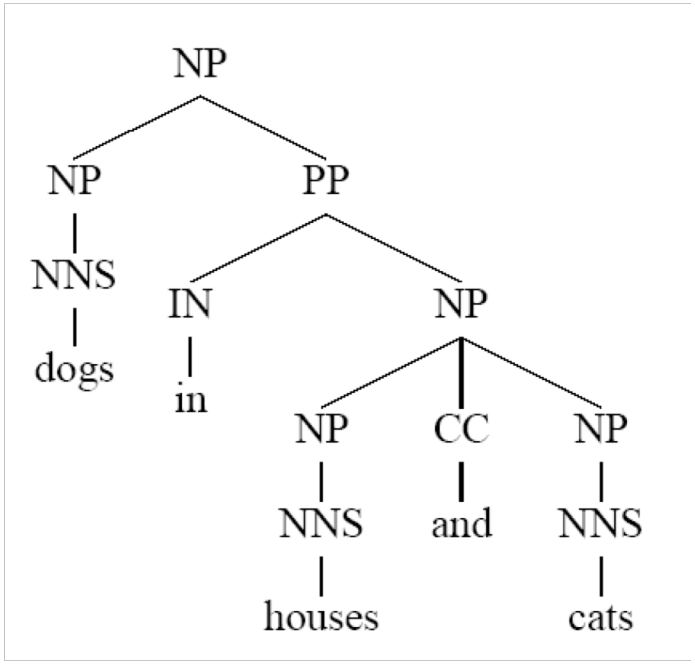
Grammar Refinement

- Example: PP attachment



Problems with PCFGs

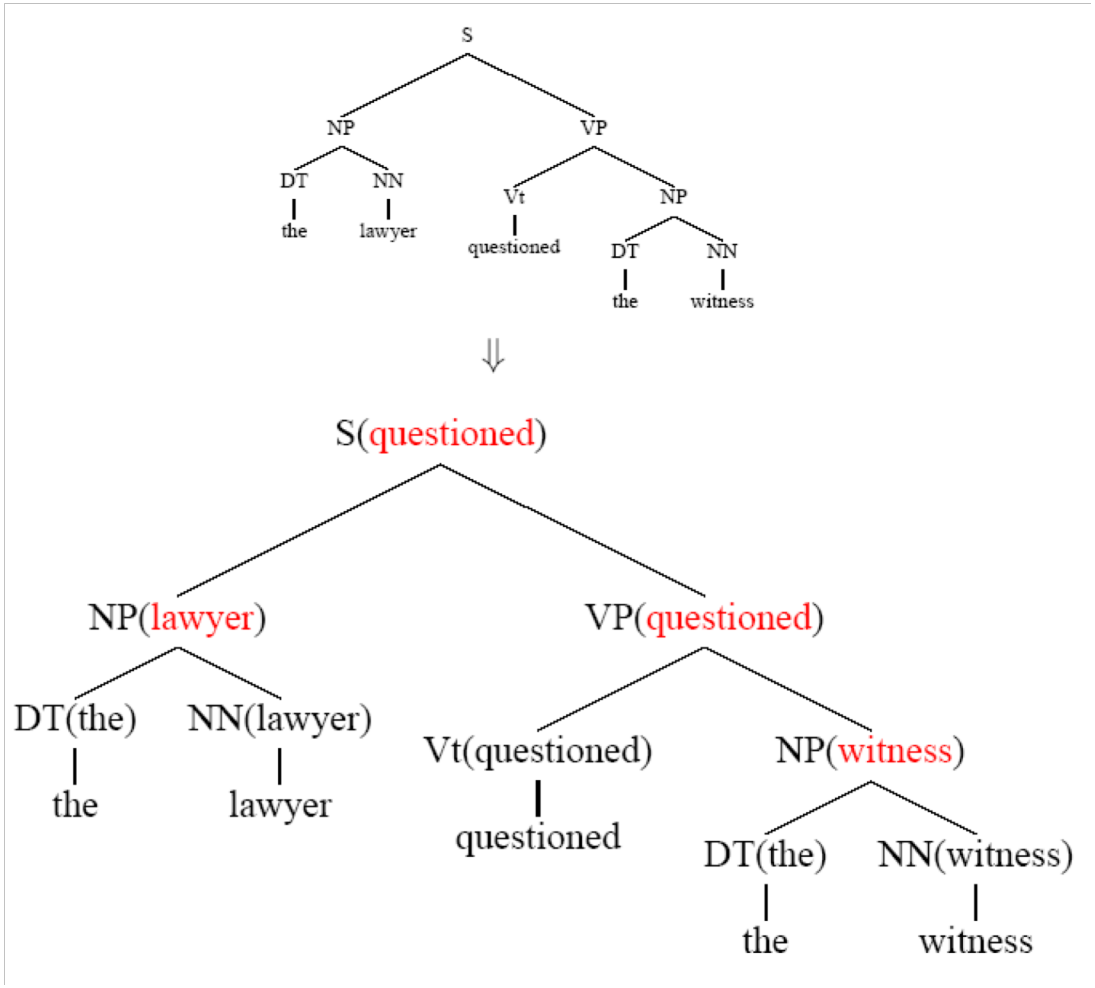




- What's different between basic PCFG scores here?
- What (lexical) correlations need to be scored?

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

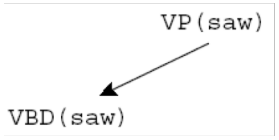


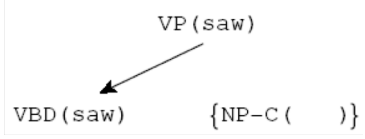
Lexicalized PCFGs?

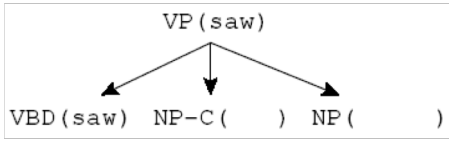
- Problem: we now have to estimate probabilities like

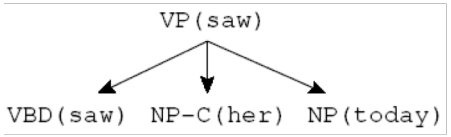
```
VP(saw) -> VBD(saw) NP-C(her) NP(today)
```

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







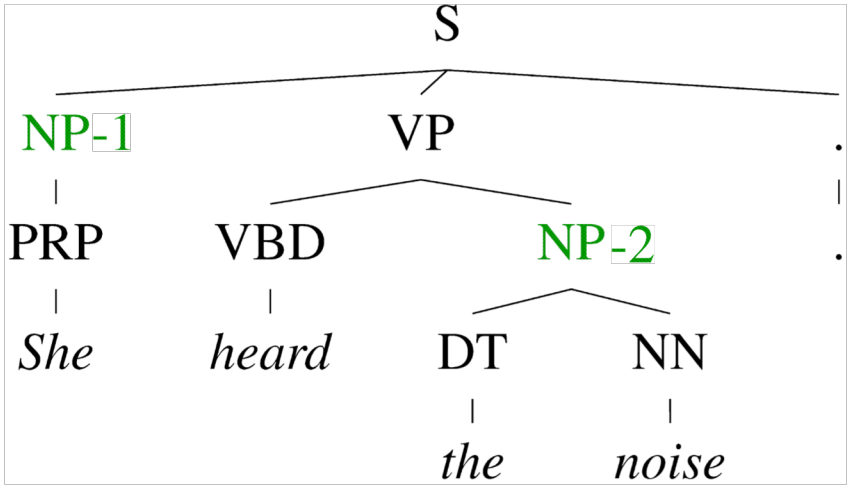


Some Test Set Results

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

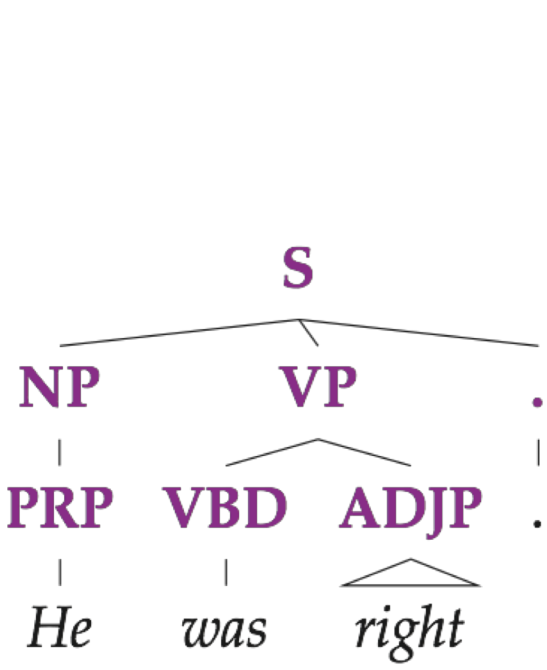
- Beats “first generation” lexicalized parsers.
- Lots of room to improve – more complex models next.

The Game of Designing a Grammar

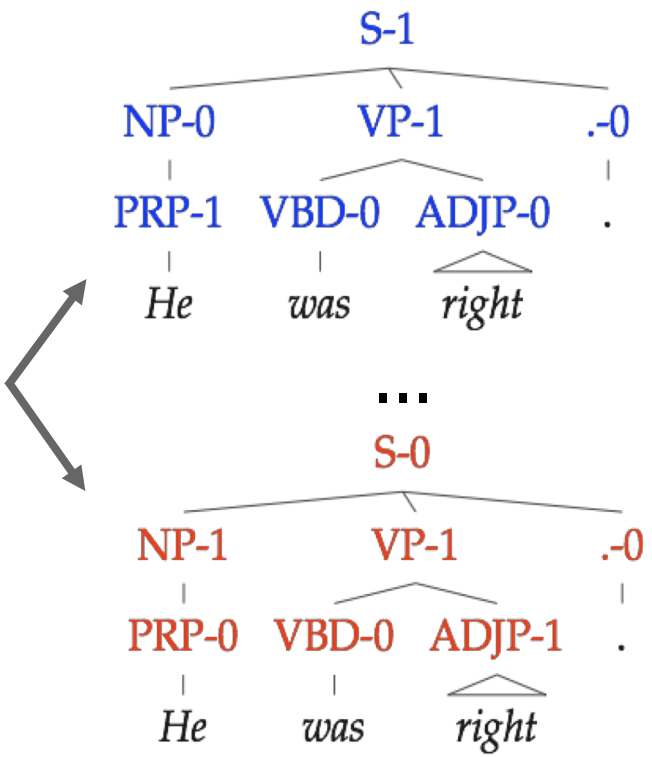


- Annotation refines base treebank symbols to improve statistical fit of the grammar
  - Parent annotation [Johnson '98]
  - Head lexicalization [Collins '99, Charniak '00]
  - Automatic clustering?

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

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

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

Higher Level: What have we done?

- Starting point: CKY with lossless binarization
- How can we relax model assumptions?
  - Lexicalization: reminiscent of transition from Word2Vec → ELMo/BERT
- How can we improve efficiency? (Maybe at the cost of accuracy)
  - Pretraining?
- How can we reduce language-dependency?