Algorithms for NLP

Parsing III

Maria Ryskina – CMU

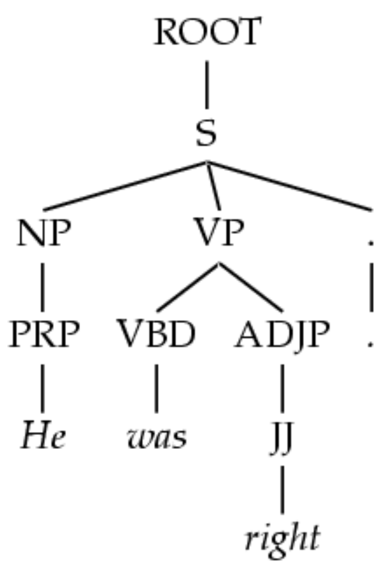
Slides adapted from: Dan Klein – UC Berkeley

Taylor Berg-Kirkpatrick, Yulia Tsvetkov – CMU

Learning PCFGs

Treebank PCFGs [Charniak 96]

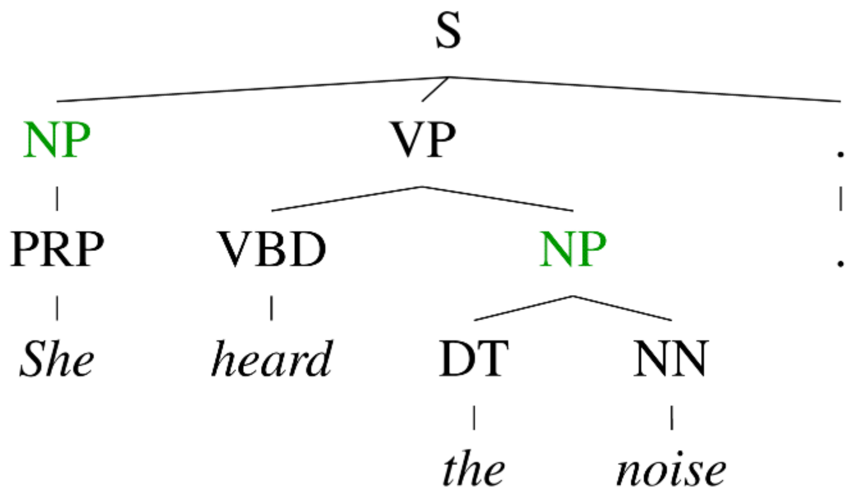
- Use PCFGs for broad coverage parsing
- Can take a grammar right off the trees (doesn't work well):



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

Model	*F1*
Baseline	72.0

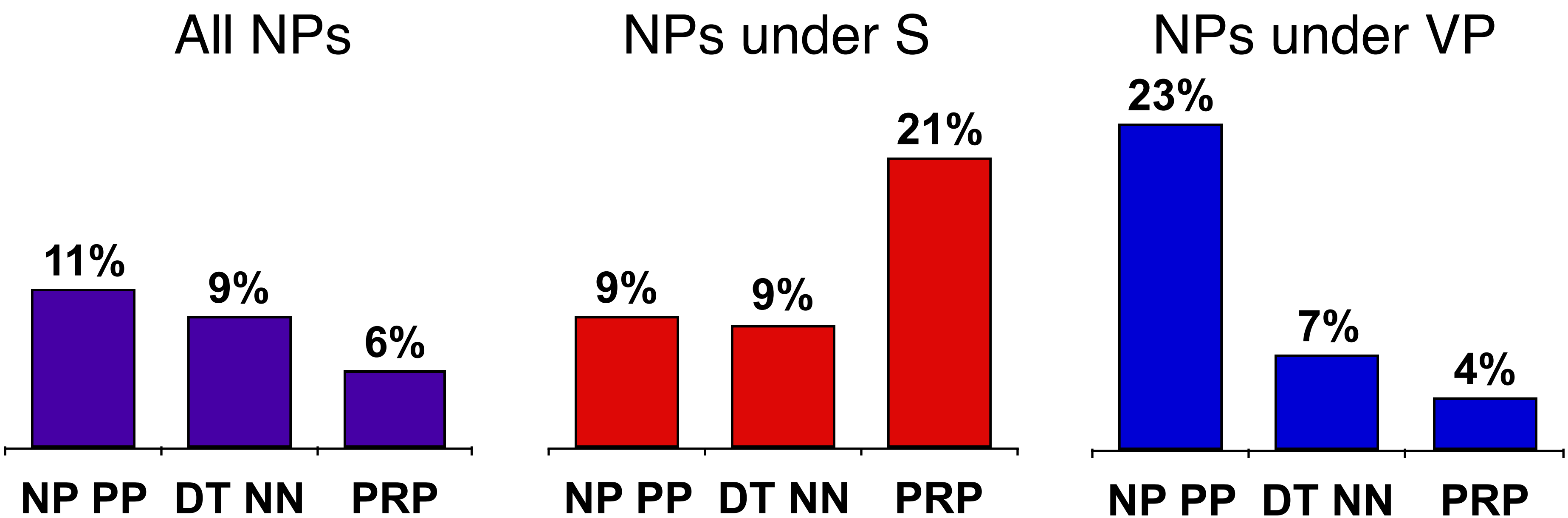
Conditional Independence?



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

Non-Independence

- Independence assumptions are often too strong.



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

Grammar Refinement

- Example: PP attachment

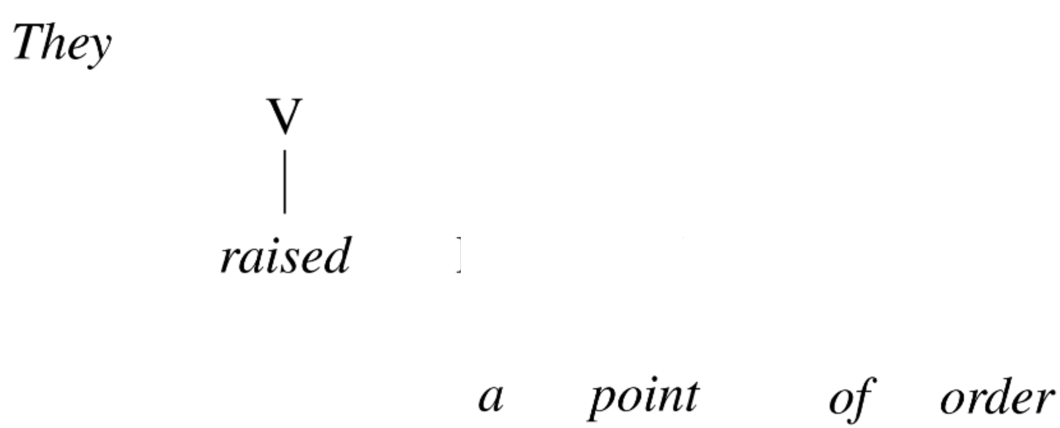
They

raised

a *point* *of* *order*

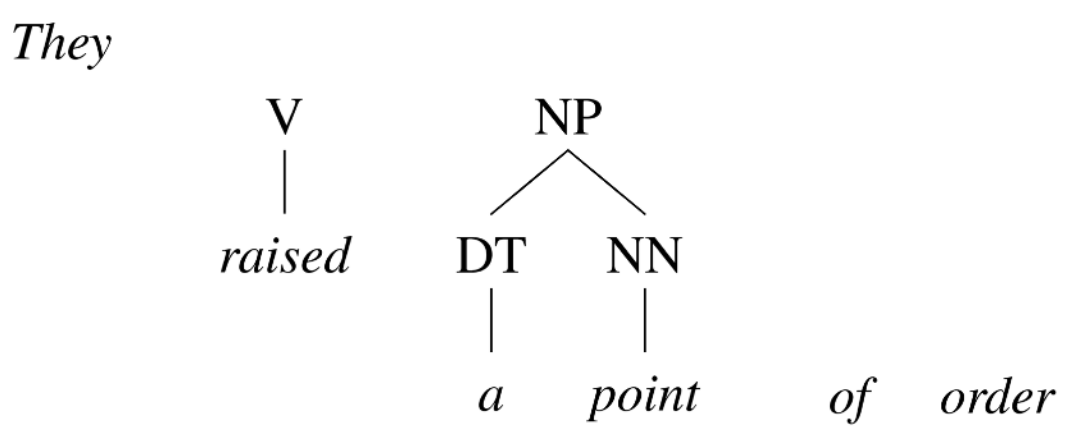
Grammar Refinement

- Example: PP attachment



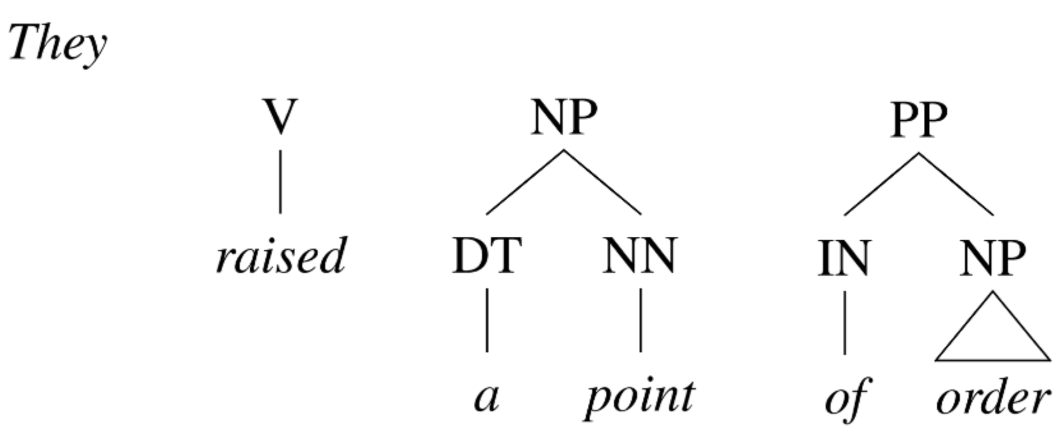
Grammar Refinement

- Example: PP attachment



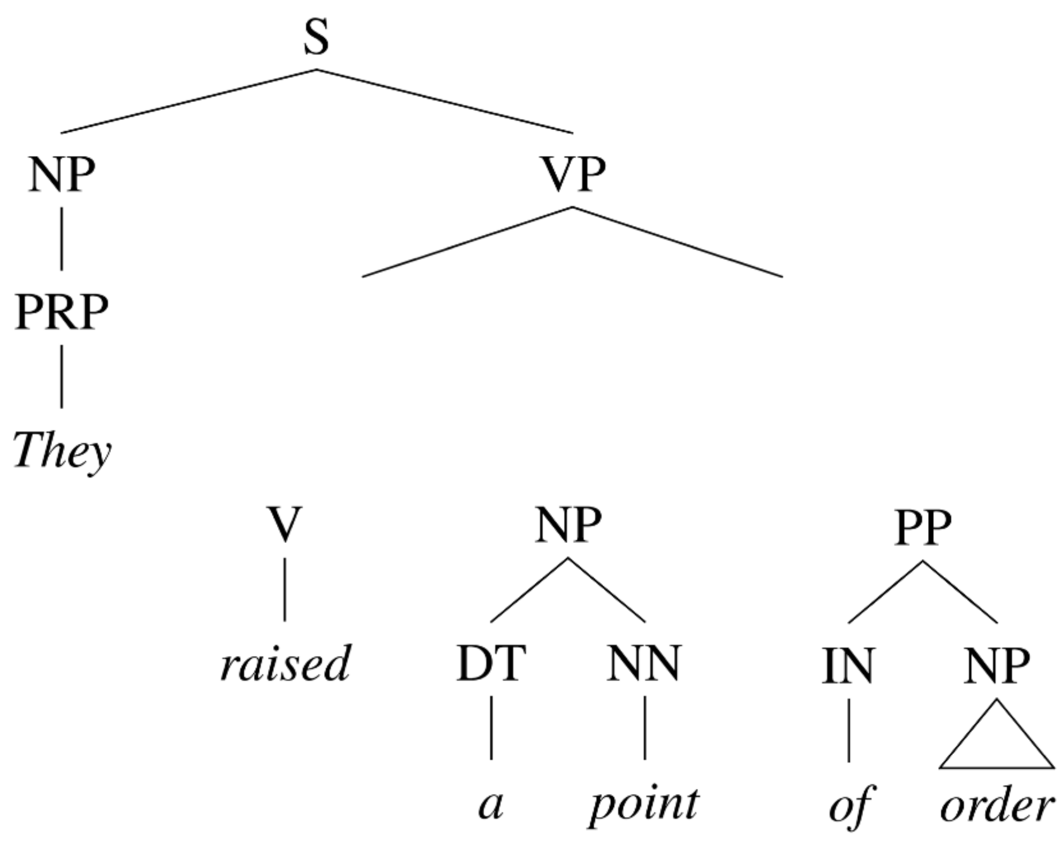
Grammar Refinement

- Example: PP attachment



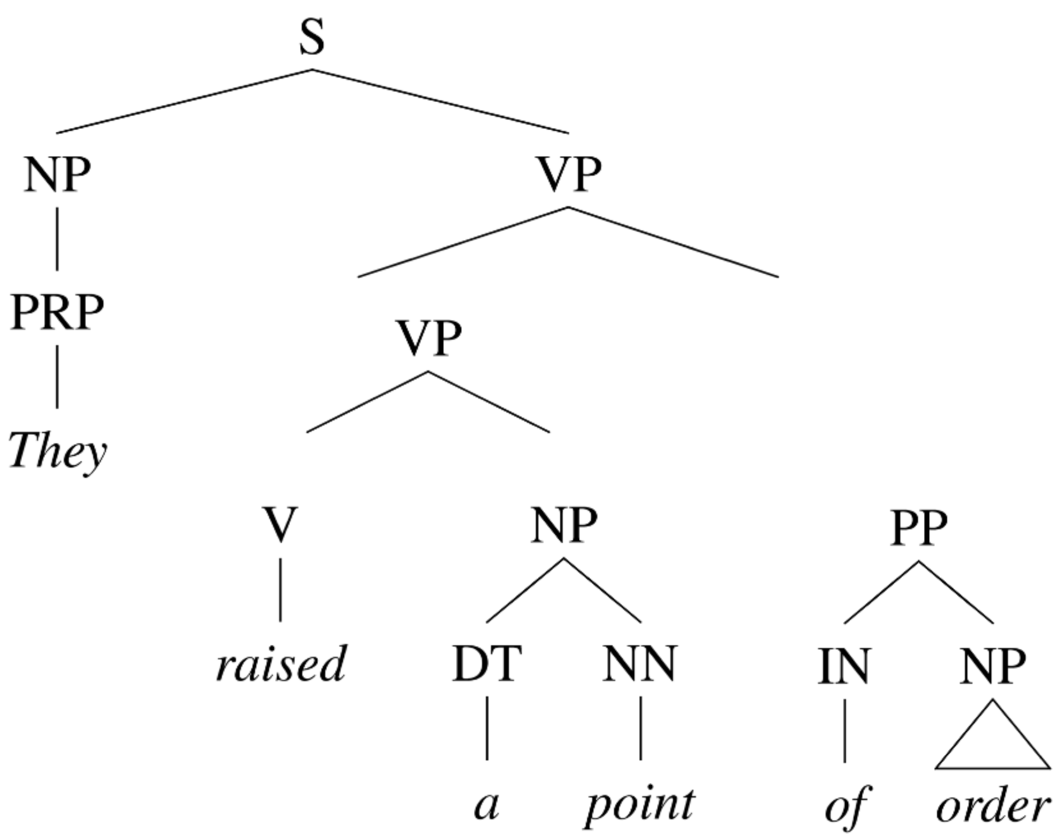
Grammar Refinement

- Example: PP attachment



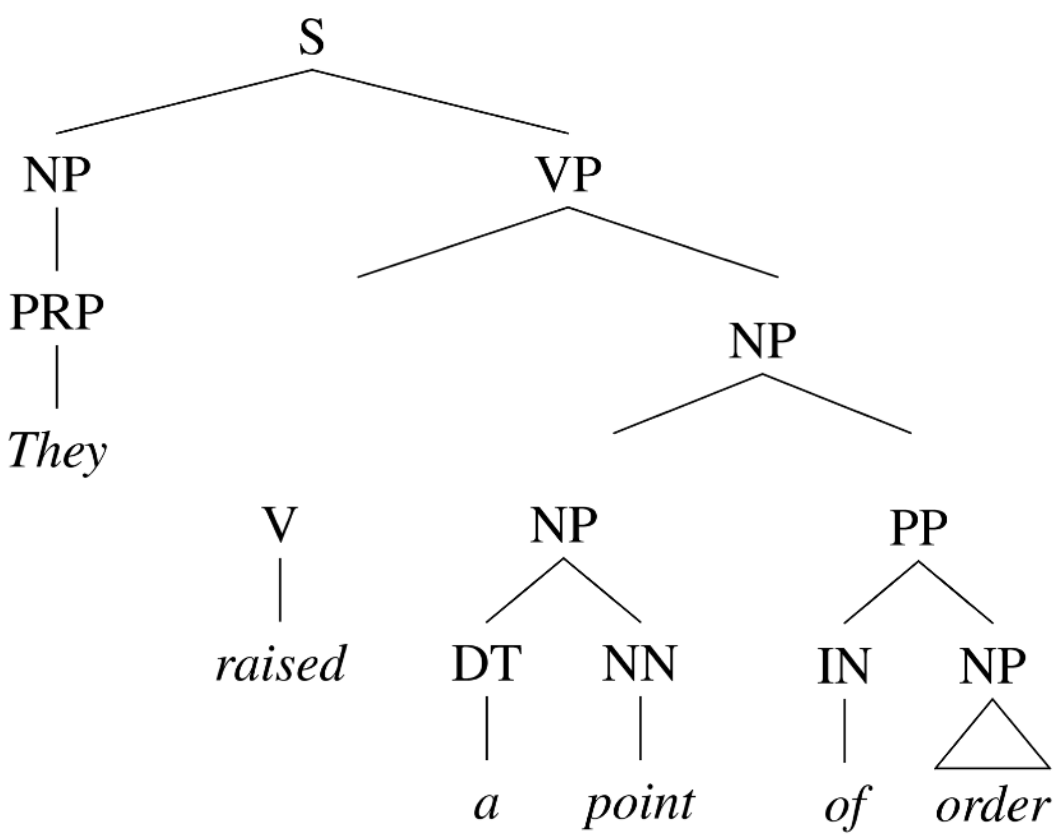
Grammar Refinement

- Example: PP attachment

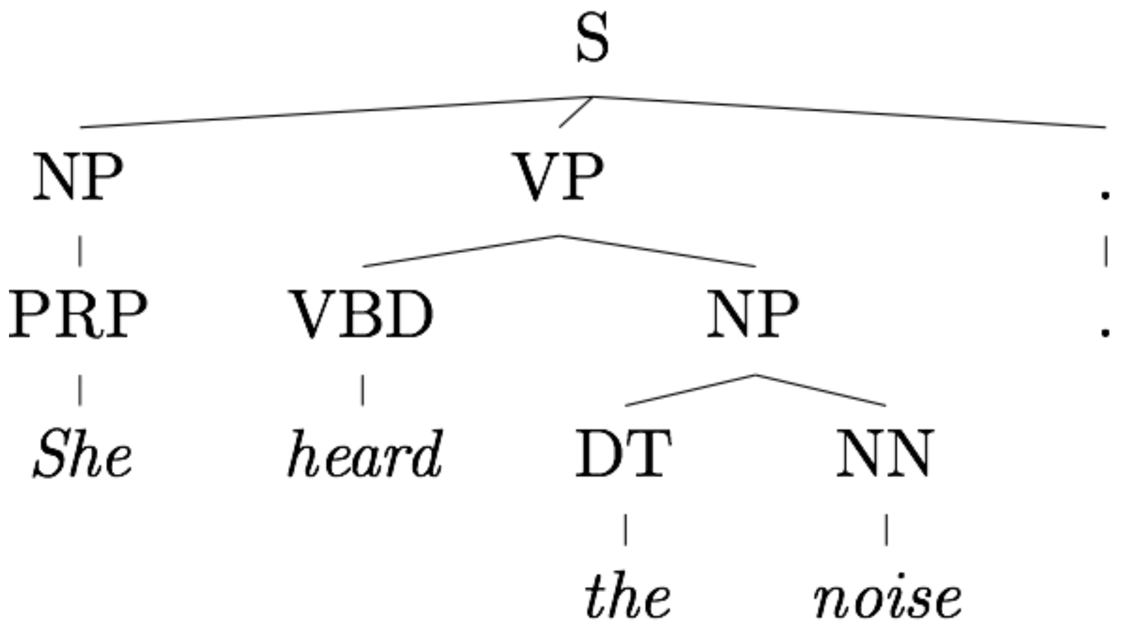


Grammar Refinement

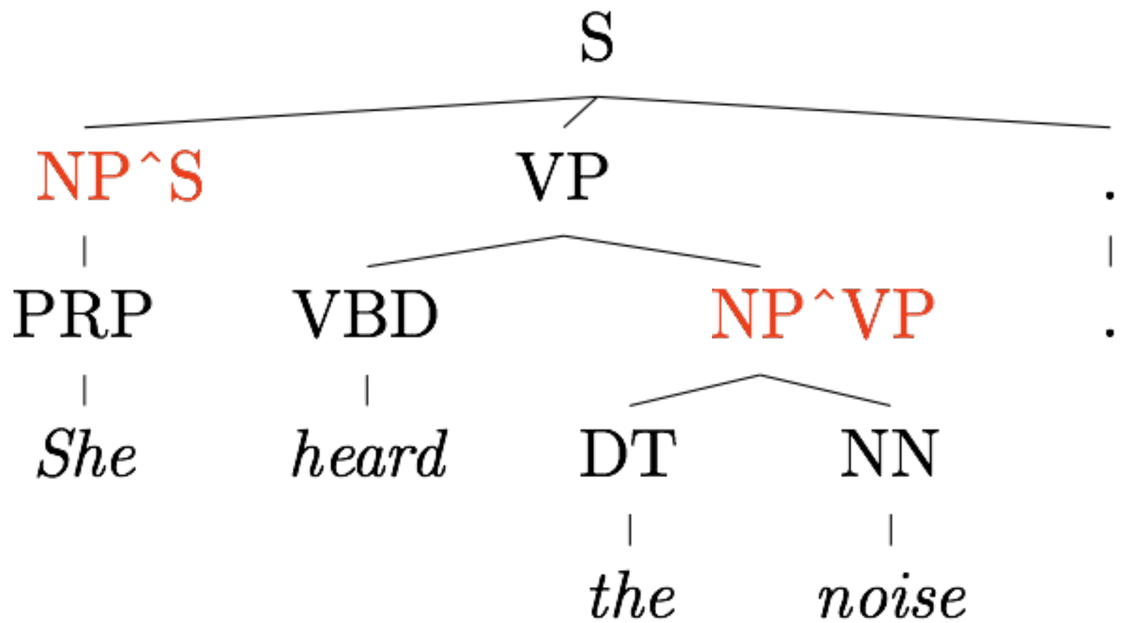
- Example: PP attachment



Grammar Refinement

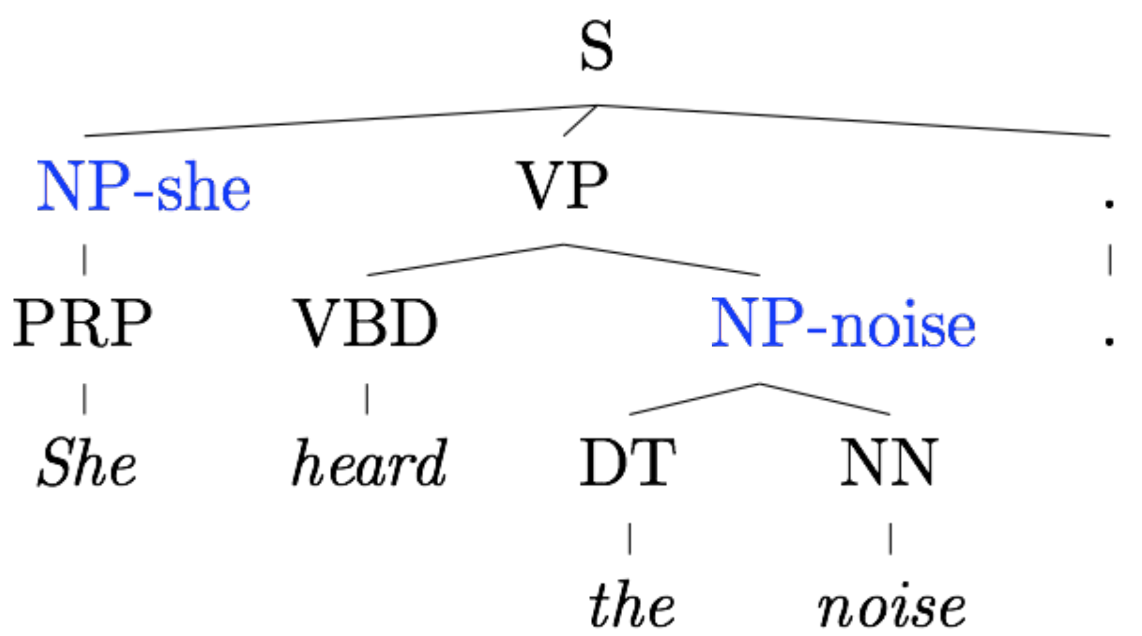


Grammar Refinement



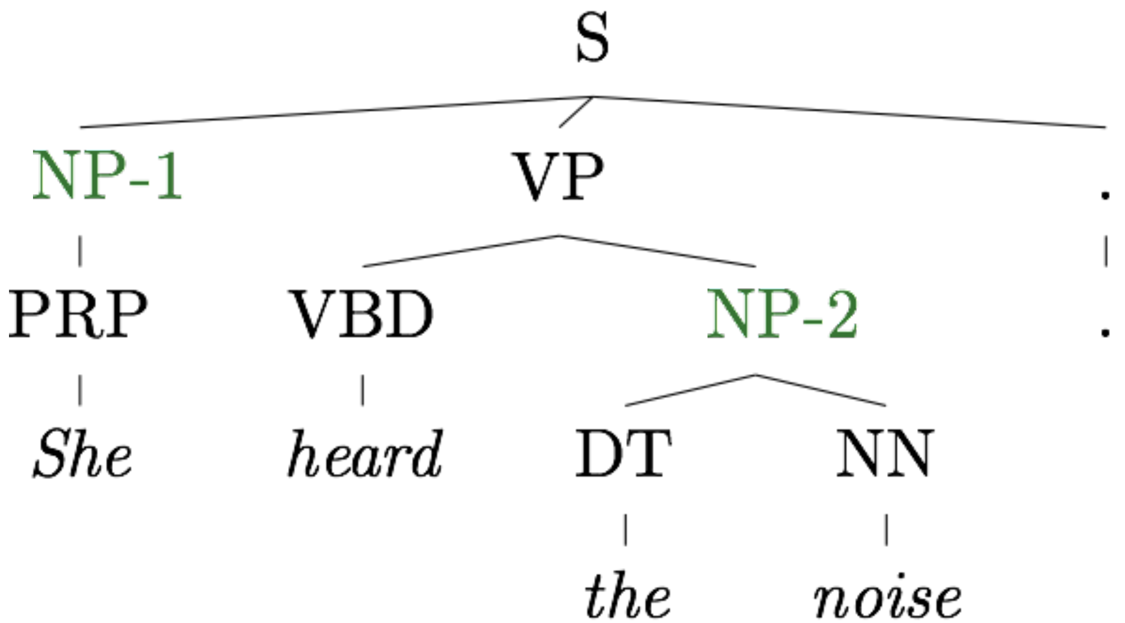
- Structural Annotation [Johnson '98, Klein&Manning '03]

Grammar Refinement



- Structural Annotation [Johnson '98, Klein&Manning '03]
- Lexicalization [Collins '99, Charniak '00]

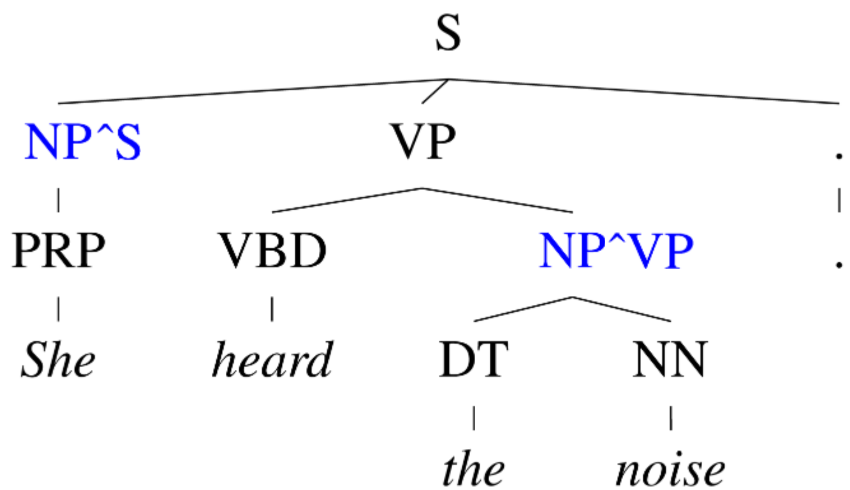
Grammar Refinement



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

Structural Annotation

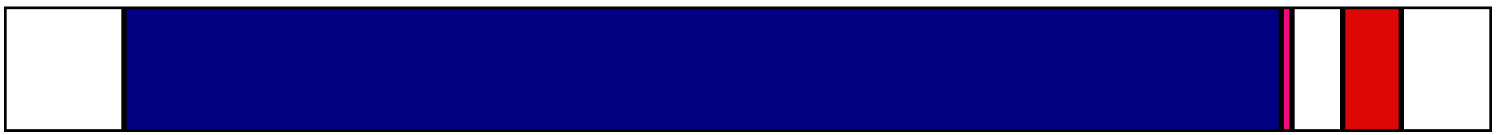
The Game of Designing a Grammar



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

Typical Experimental Setup

- Corpus: Penn Treebank, WSJ



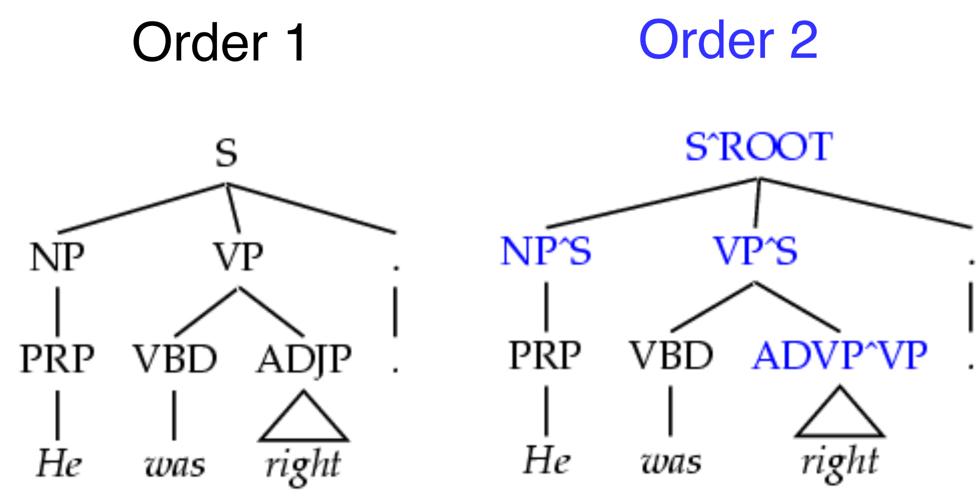
Training:	sections	02-21
Development:	section	22 (here, first 20 files)
Test:	section	23

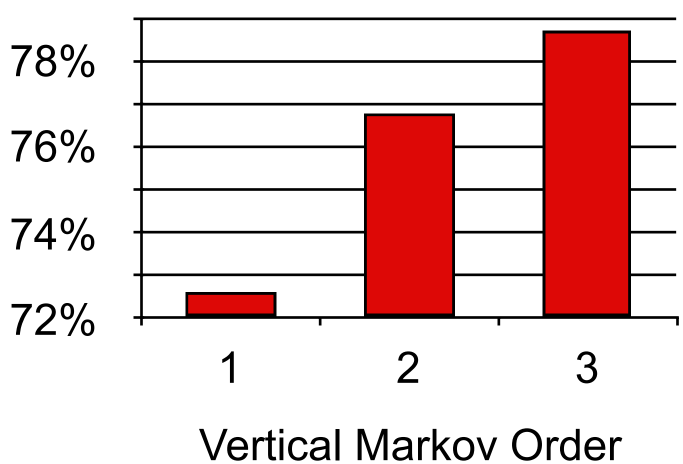
- Accuracy – F1: harmonic mean of per-node labeled precision and recall.
- Here: also size – number of symbols in grammar.

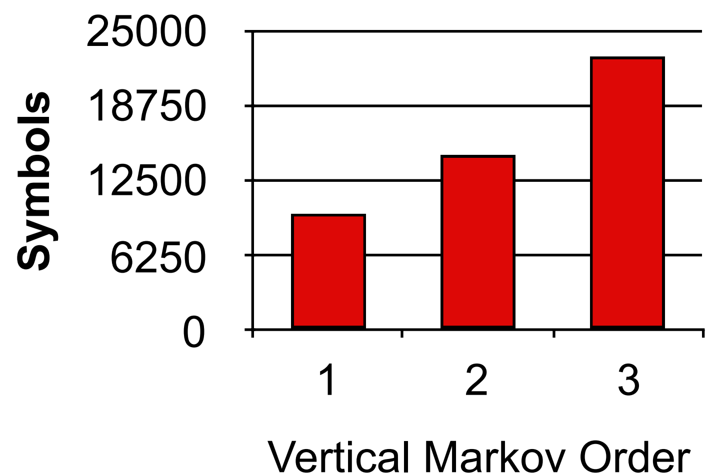
Vertical Markovization

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

  (cf. parent annotation)



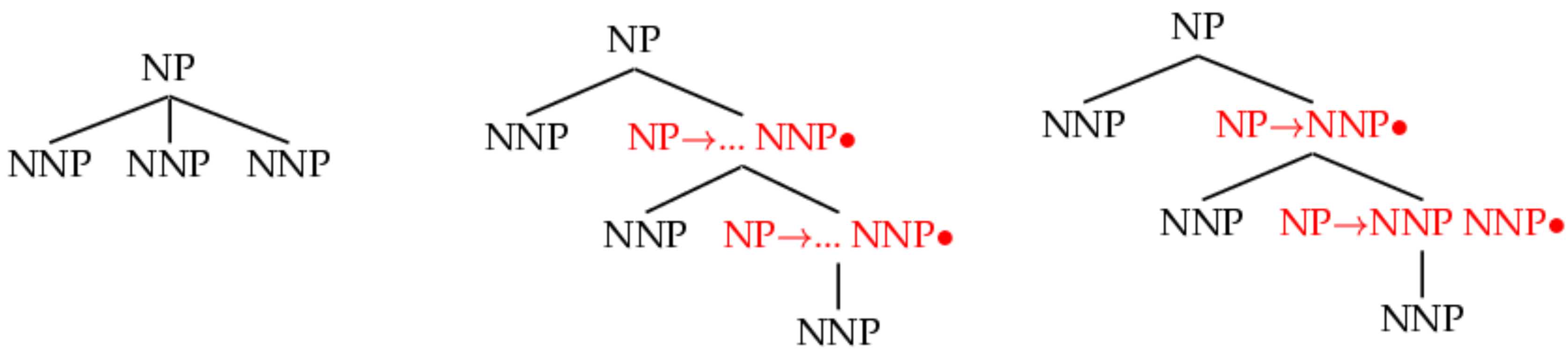


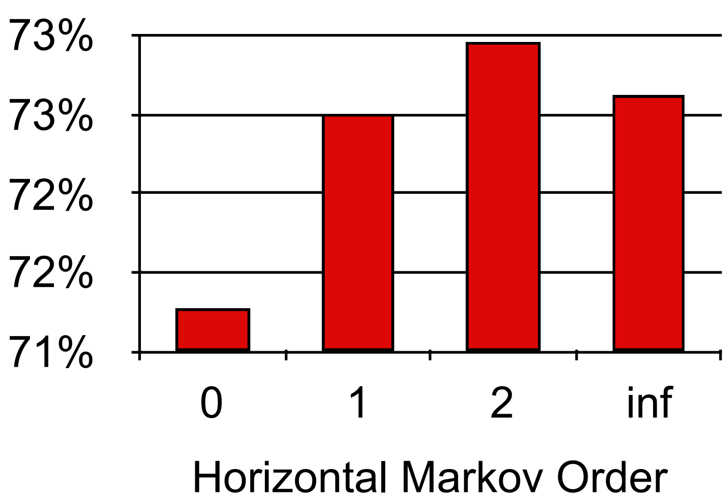


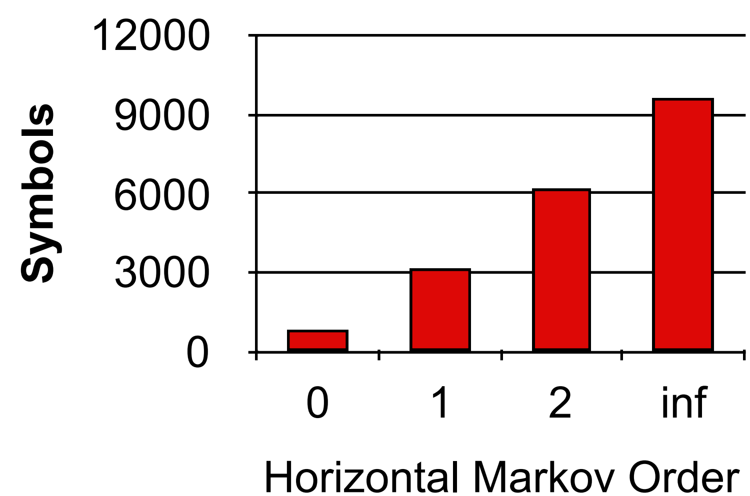
Horizontal Markovization

Order 1

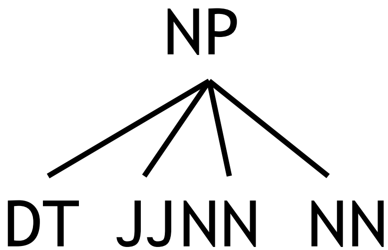
Order ∞



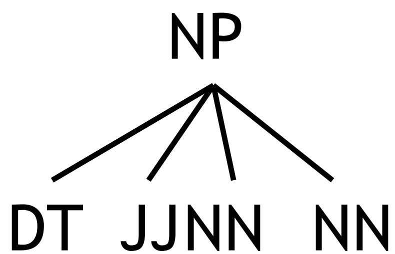




Binarization / Markovization

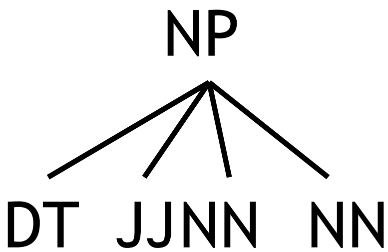


Binarization / Markovization

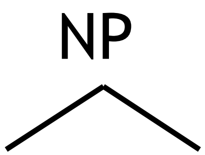


v=1,h=∞

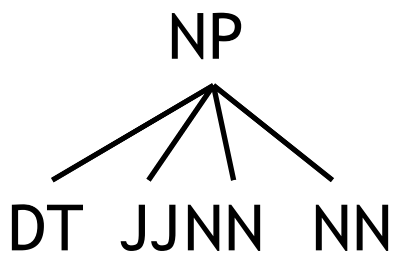
Binarization / Markovization



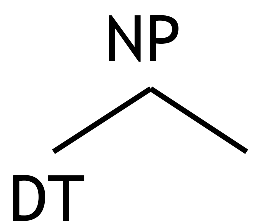
v=1,h=∞



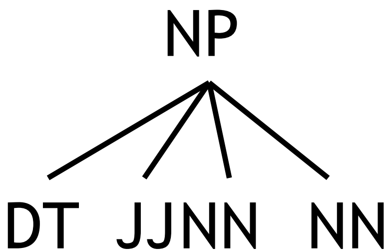
Binarization / Markovization



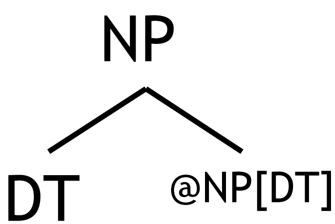
v=1,h=∞



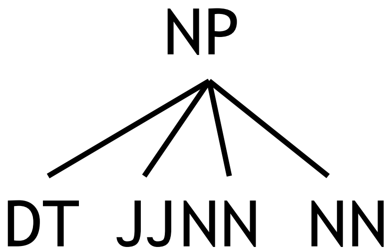
Binarization / Markovization



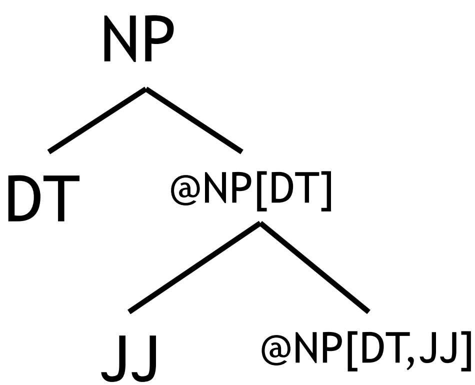
v=1,h=∞



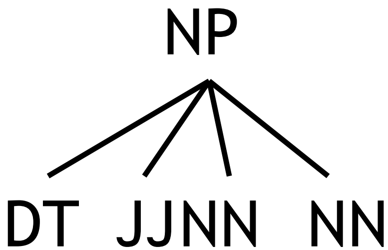
Binarization / Markovization



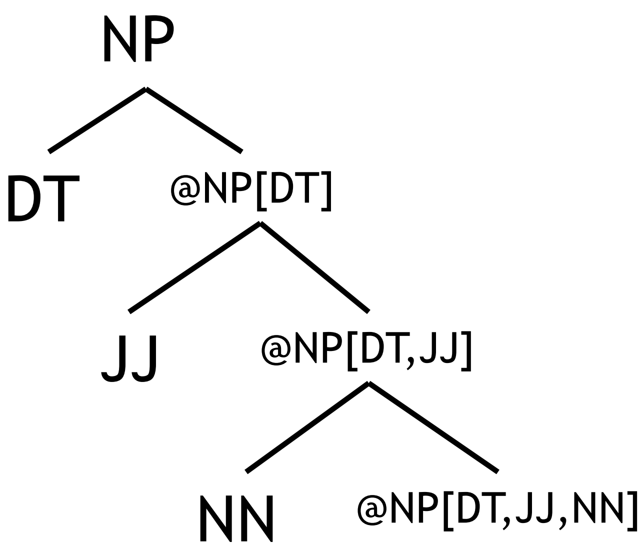
v=1,h=∞



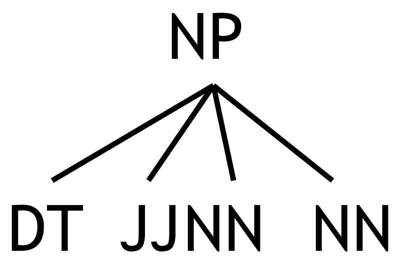
Binarization / Markovization



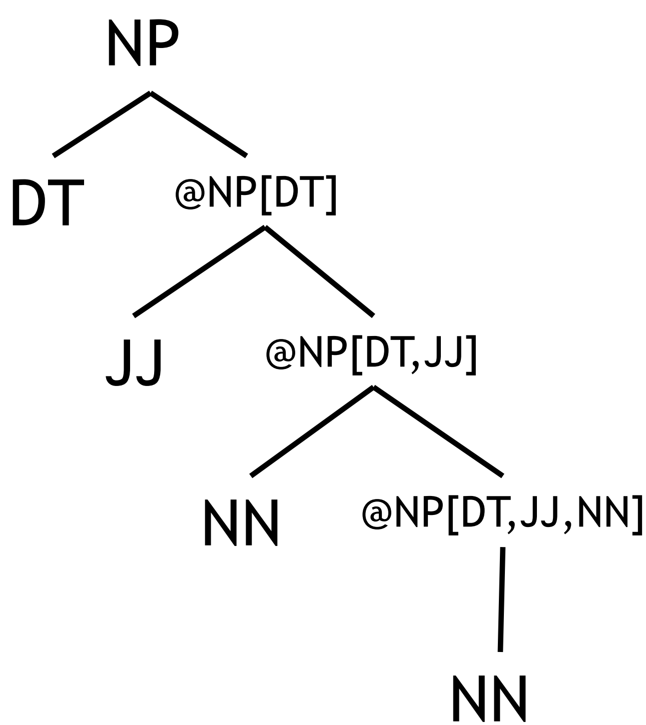
v=1,h=∞



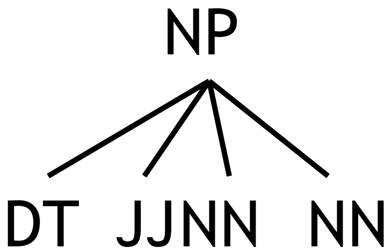
Binarization / Markovization



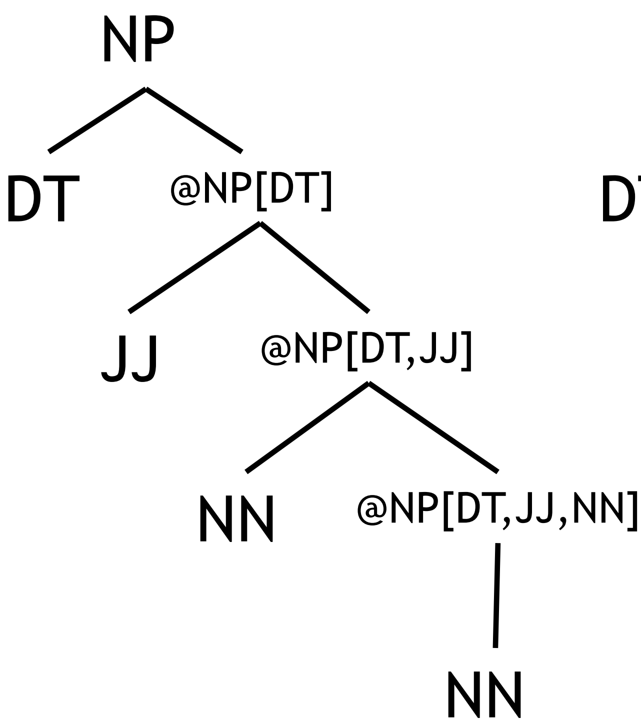
v=1,h=∞



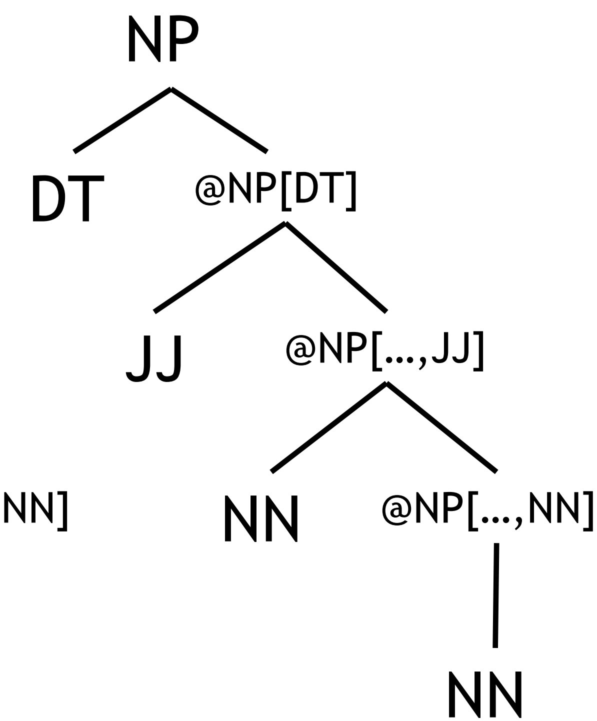
Binarization / Markovization



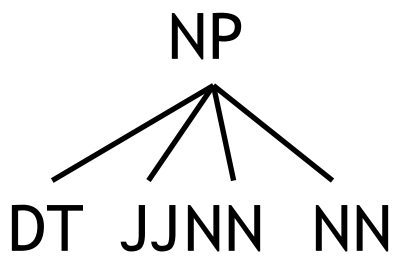
v=1,h=∞



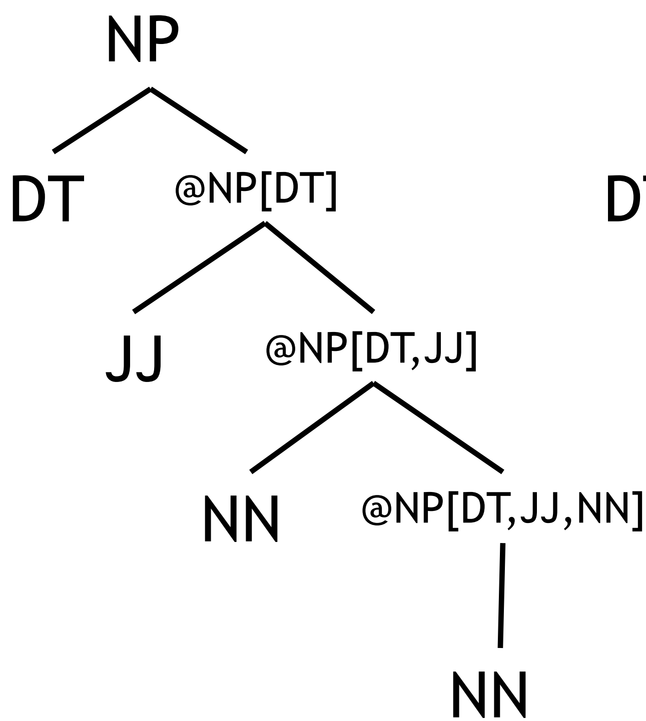
v=1,h=1



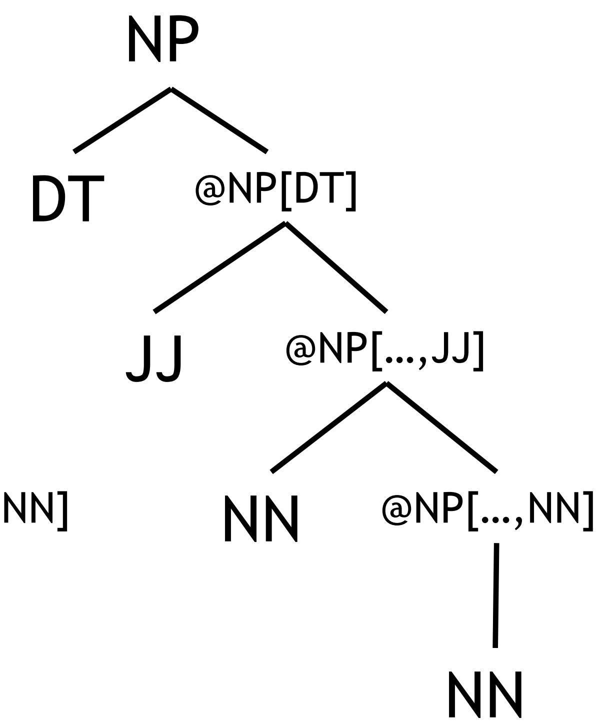
Binarization / Markovization



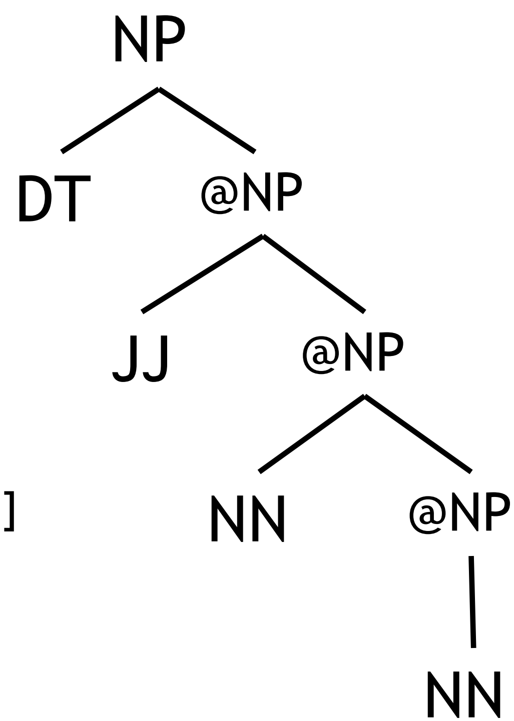
v=1,h=∞



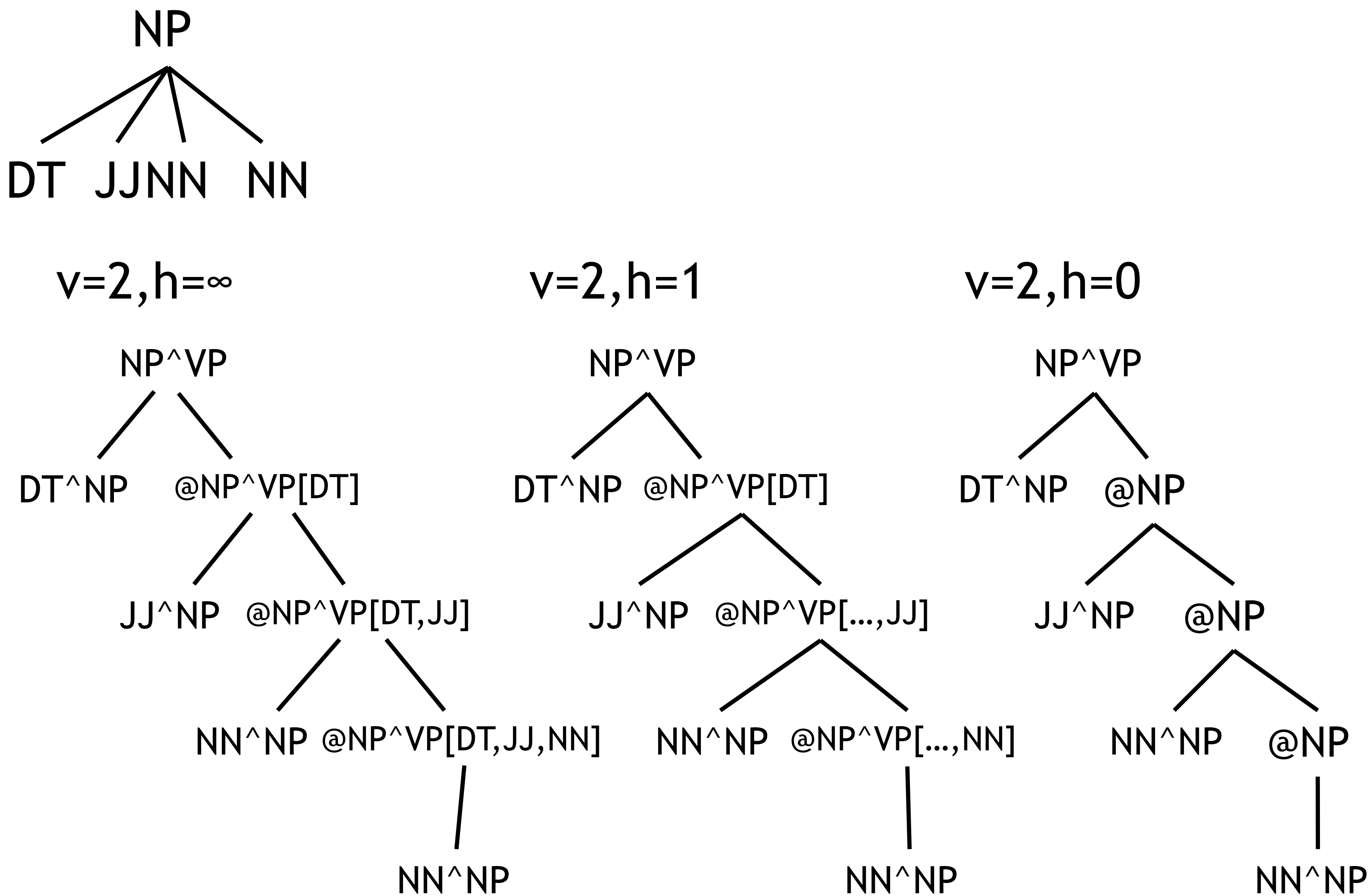
v=1,h=1



v=1,h=0

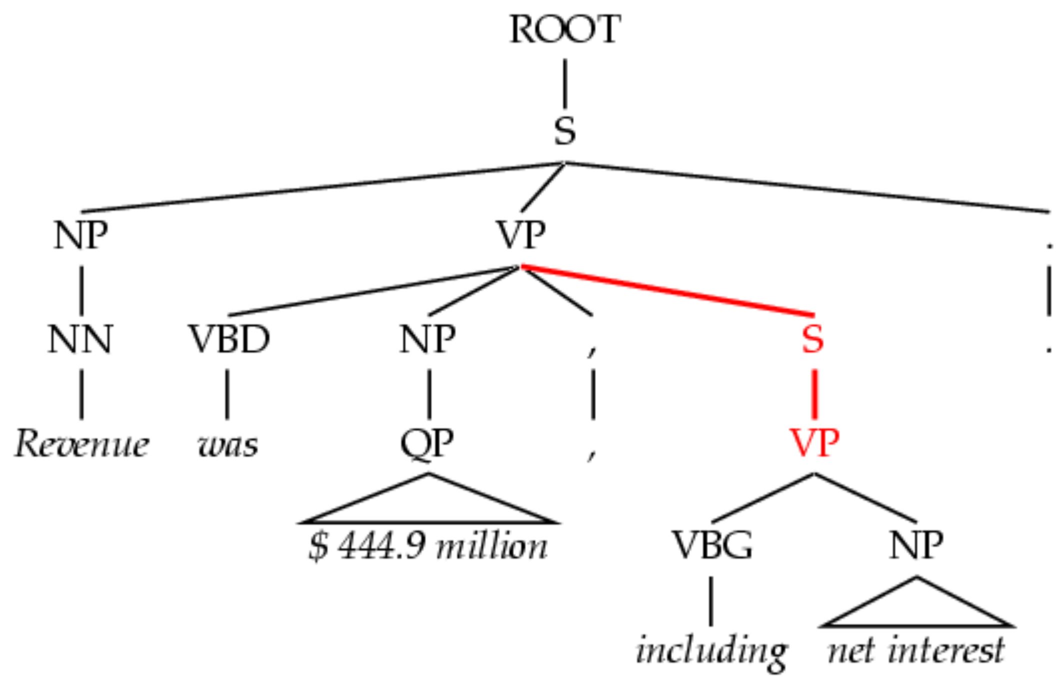


Binarization / Markovization



Unary Splits

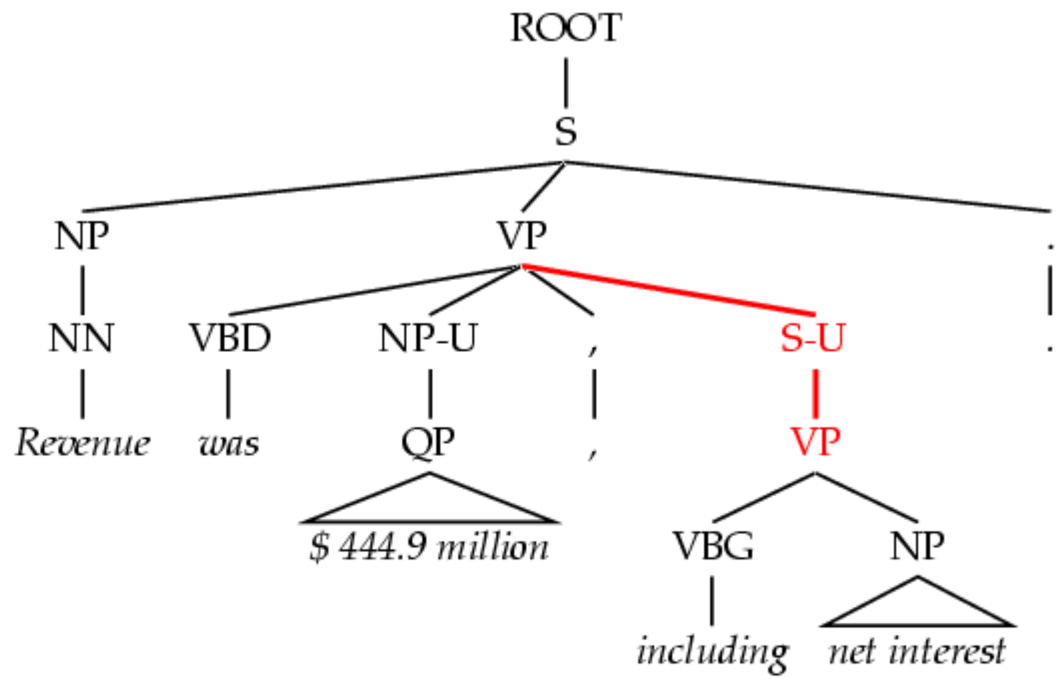
- Problem: unary rewrites used to transmute categories so a high-probability rule can be used.



Annotation	F1	Size
Base	77.8	7.5K
UNARY	78.3	8.0K

Unary Splits

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

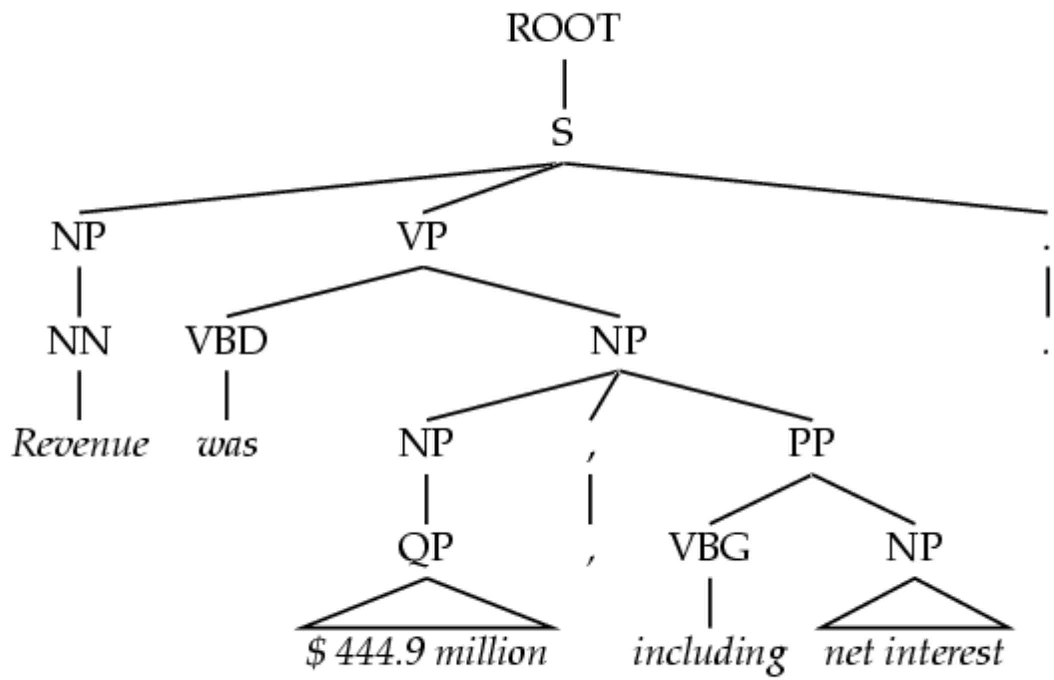


- Solution: Mark unary rewrite sites with -U

Annotation	F1	Size
Base	77.8	7.5K
UNARY	78.3	8.0K

Unary Splits

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

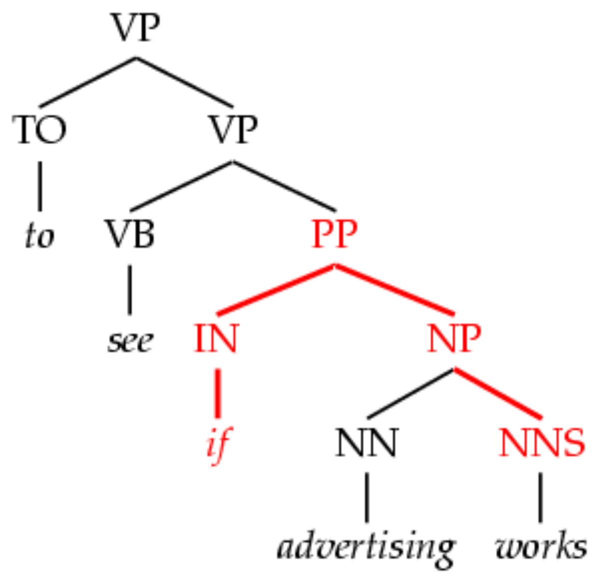


- Solution: Mark unary rewrite sites with -U

Annotation	F1	Size
Base	77.8	7.5K
UNARY	78.3	8.0K

Tag Splits

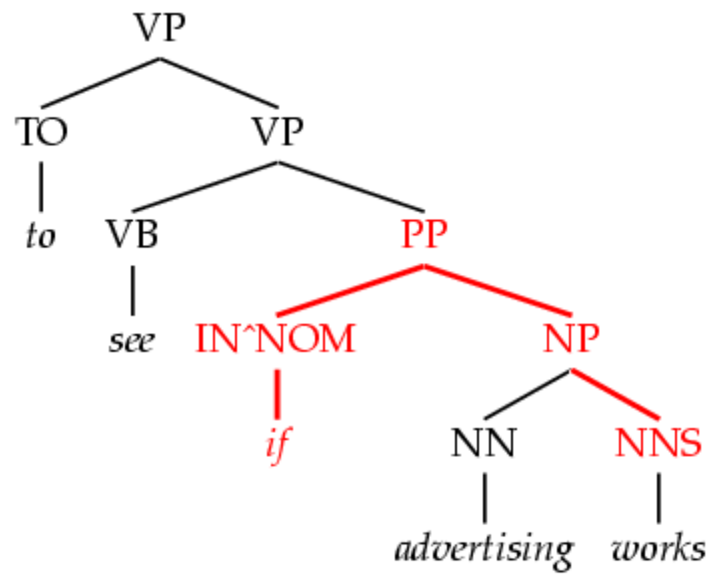
- Problem: Treebank tags are too coarse.
- Example: Sentential, PP, and other prepositions are all marked IN.
- Partial Solution:
  - Subdivide the IN tag.



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

Tag Splits

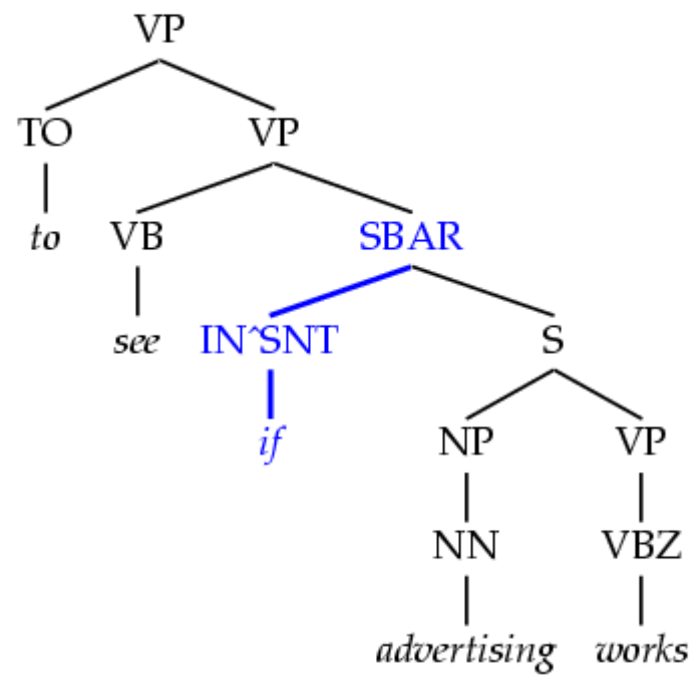
- Problem: Treebank tags are too coarse.
- Example: Sentential, PP, and other prepositions are all marked IN.
- Partial Solution:
  - Subdivide the IN tag.



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

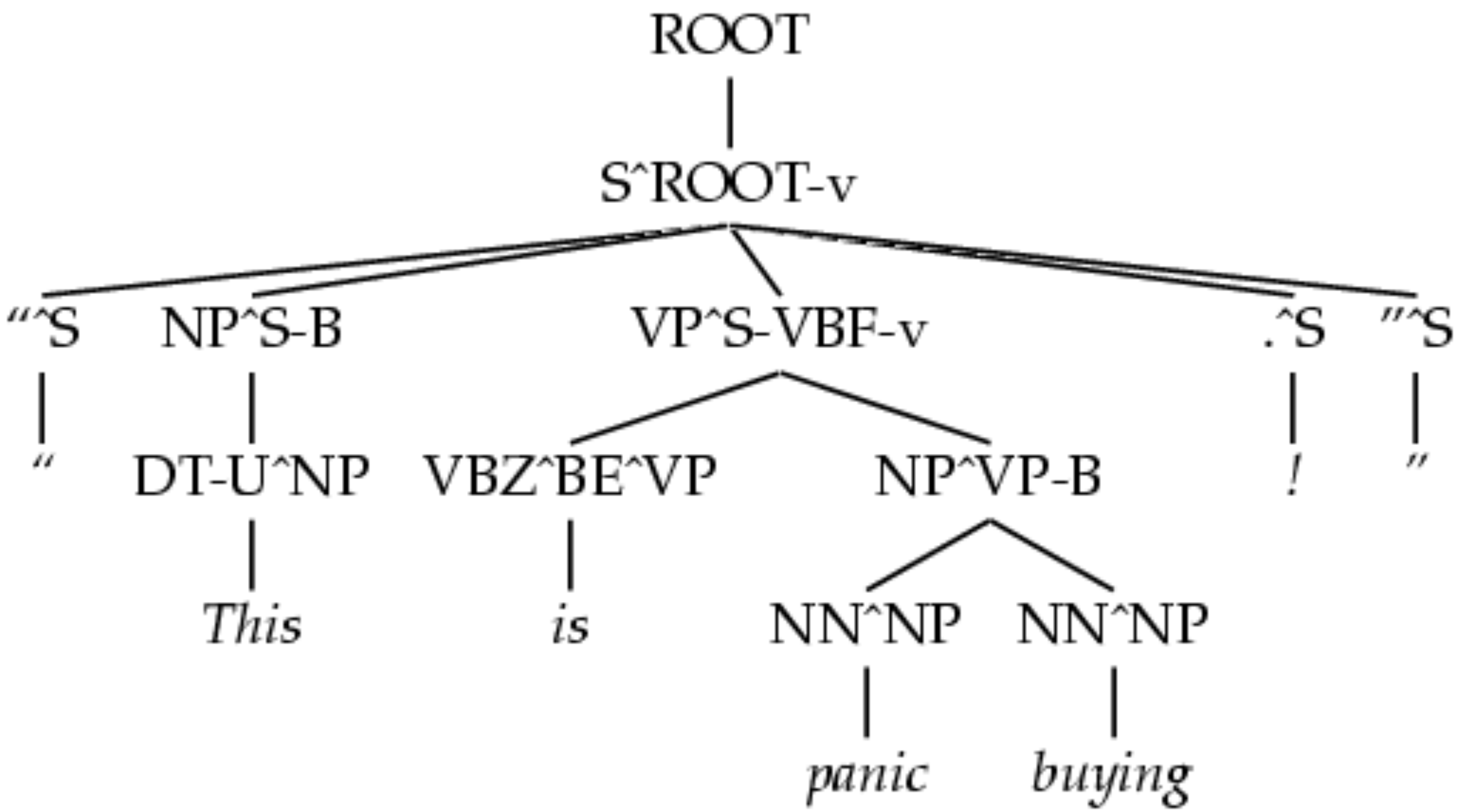
Tag Splits

- 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

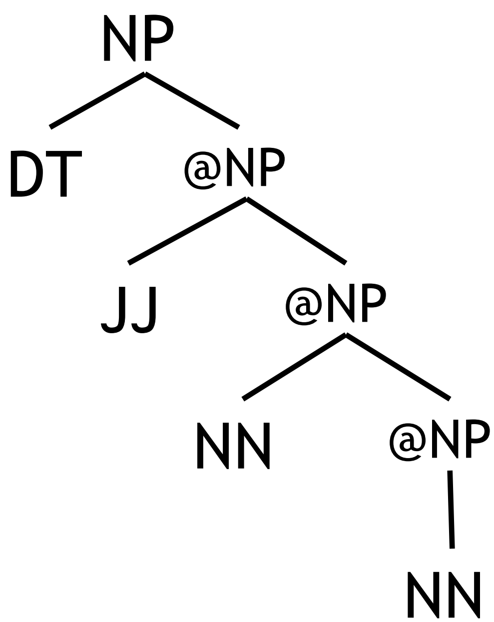
Parser	LP	LR	**F1**	CB	0 CB
Magerman 95	84.9	84.6	**84.7**	1.26	56.6
Collins 96	86.3	85.8	**86.0**	1.14	59.9
Unlexicalized	86.9	85.7	**86.3**	1.10	60.3
Charniak 97	87.4	87.5	**87.4**	1.00	62.1
Collins 99	88.7	88.6	**88.6**	0.90	67.1

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

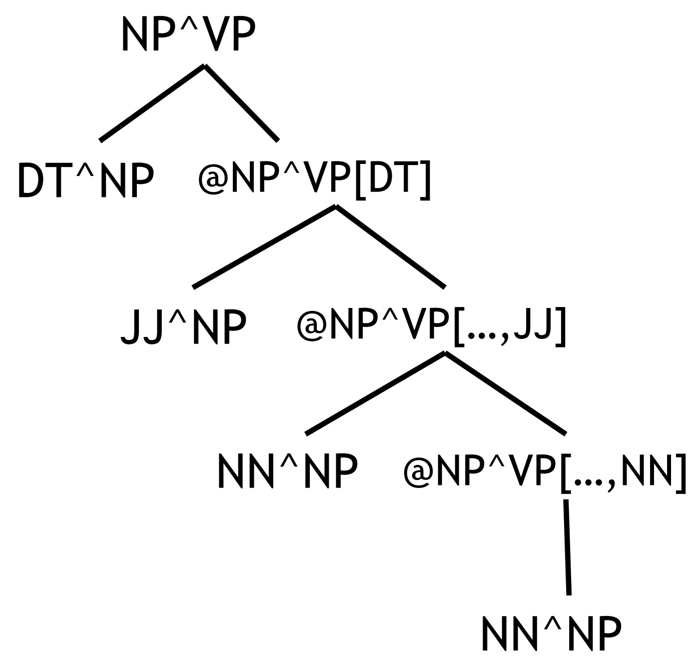
Efficient Parsing for Structural Annotation

Grammar Projections

Coarse Grammar

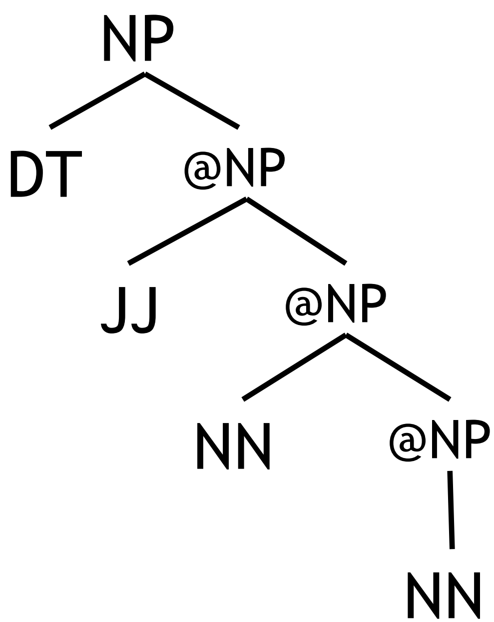


Fine Grammar

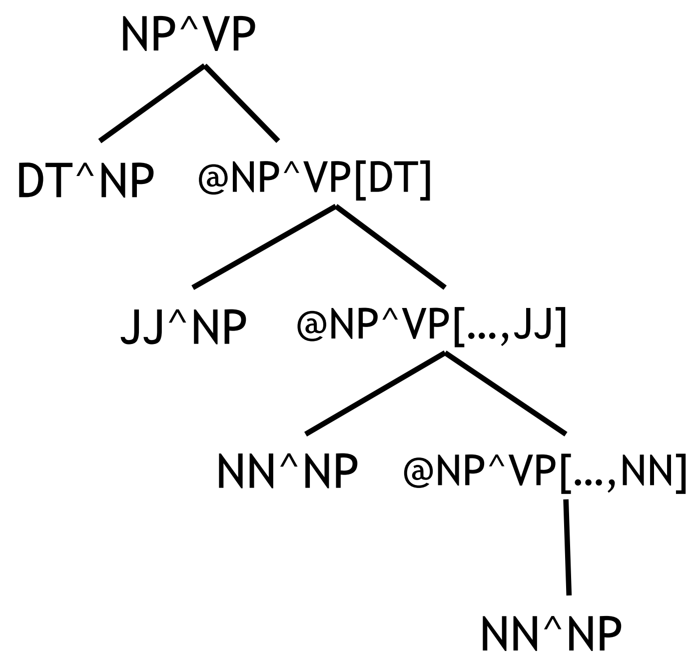


Grammar Projections

Coarse Grammar



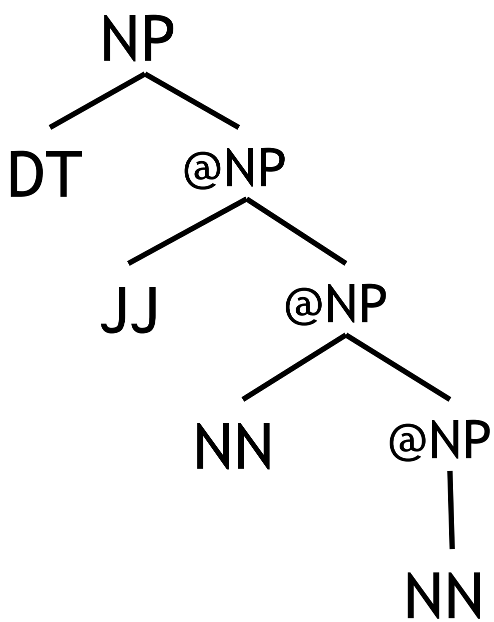
Fine Grammar



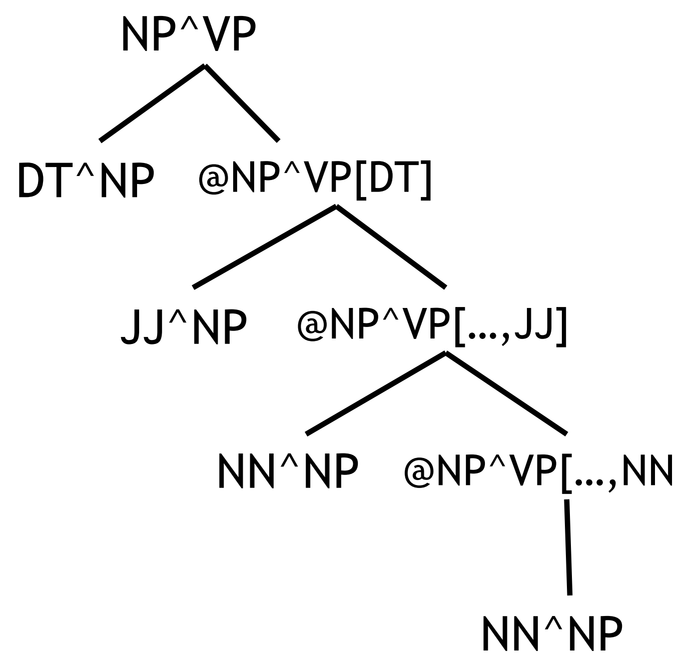
NP → DT @NP

Grammar Projections

Coarse Grammar



Fine Grammar

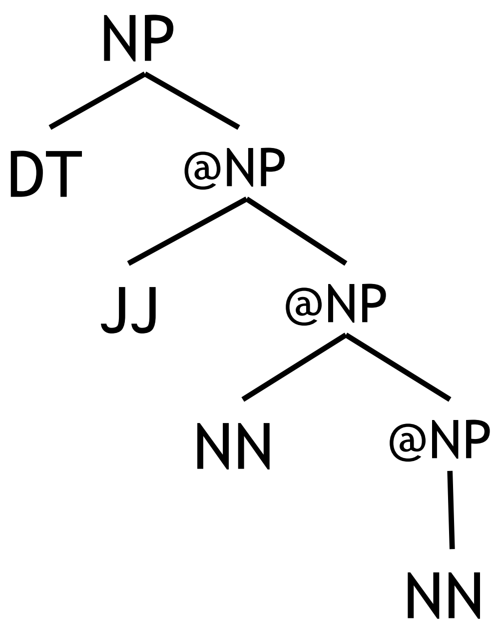


NP → DT @NP

NP^VP → DT^NP @NP^VP[DT]

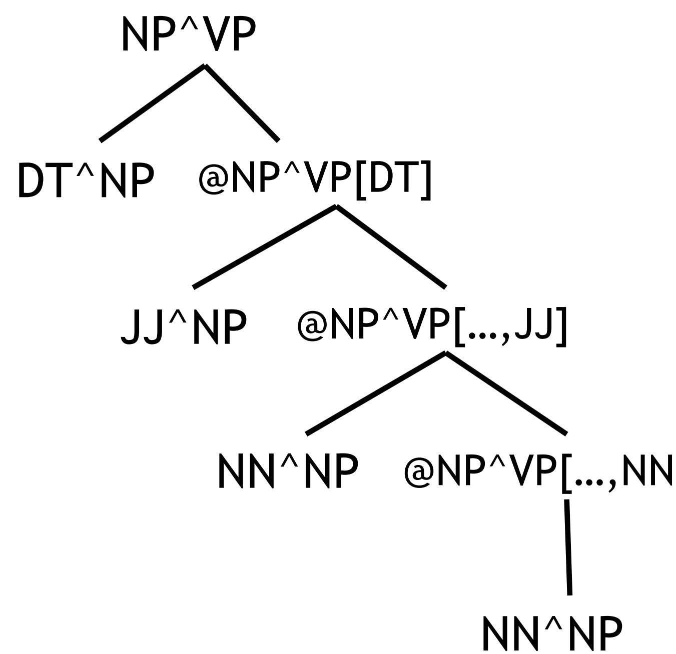
Grammar Projections

Coarse Grammar



$NP \rightarrow DT\ @NP$

Fine Grammar



$NP\text{^}VP \rightarrow DT\text{^}NP\ @NP\text{^}VP[DT]$

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] @NP^VP[…,JJ] @NP^S[…,JJ]
DT	DT^NP

Coarse-to-Fine Pruning

For each coarse chart item $X[i,j]$, compute posterior probability:

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

E.g. consider the span 5 to 12:

coarse: … | QP | NP | VP | …

Coarse-to-Fine Pruning

For each coarse chart item $X[i,j]$, compute posterior probability:

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

E.g. consider the span 5 to 12:

coarse: … $\boxed{\text{QP}}$ | NP | VP …

Coarse-to-Fine Pruning

For each coarse chart item $X[i,j]$, compute posterior probability:

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

E.g. consider the span 5 to 12:

coarse: … $\boxed{\text{QP}}$ NP VP …

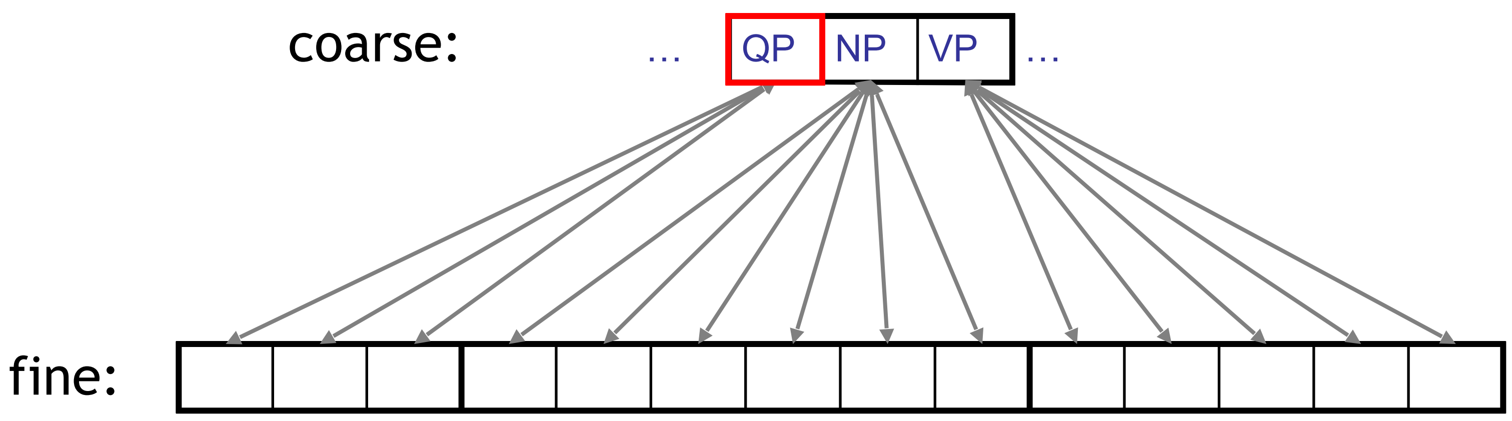
fine:

Coarse-to-Fine Pruning

For each coarse chart item $X[i,j]$, compute posterior probability:

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

E.g. consider the span 5 to 12:

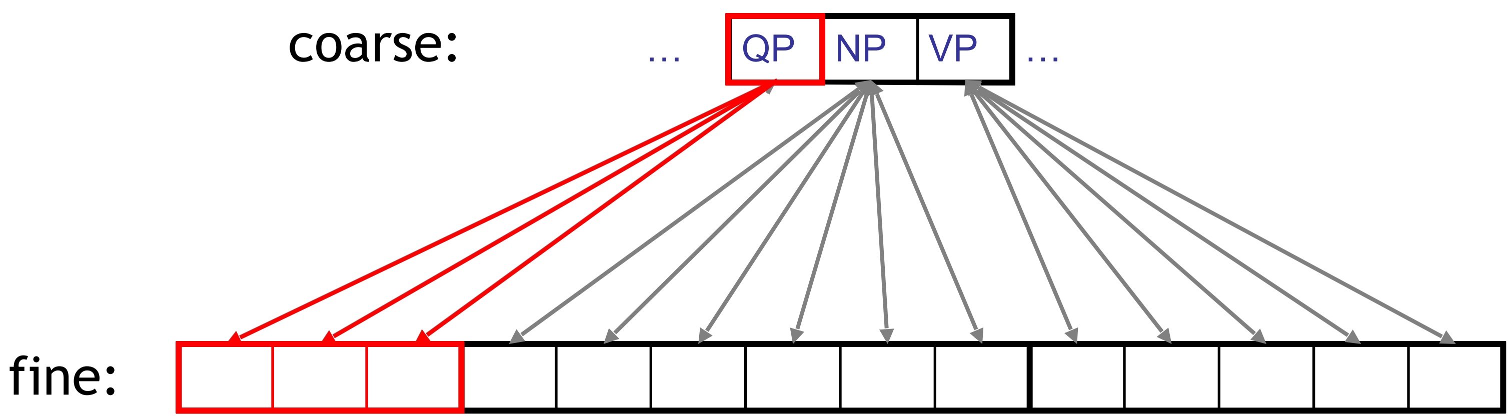


Coarse-to-Fine Pruning

For each coarse chart item $X[i,j]$, compute posterior probability:

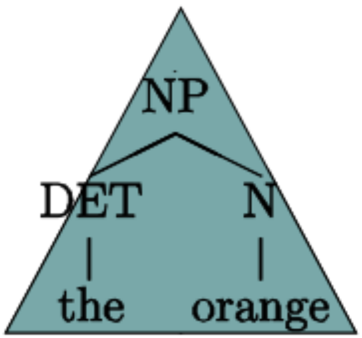
$$\frac{P_{\mathrm{IN}}(X,i,j) \cdot P_{\mathrm{OUT}}(X,i,j)}{P_{\mathrm{IN}}(root,0,n)} \quad < \quad \textbf{\textit{threshold}}$$

E.g. consider the span 5 to 12:



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



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

$$\beta_N(2,2) = P(N \rightarrow orange \mid G) = 0.2$$

$$\begin{aligned}\beta_{NP}(1,2) &= P(NP \rightarrow DET \cdot N)\beta_{DET}(1,1)\beta_N(2,2) \\ &= 0.8 \qquad \times 0.4 \qquad \times 0.2\end{aligned}$$

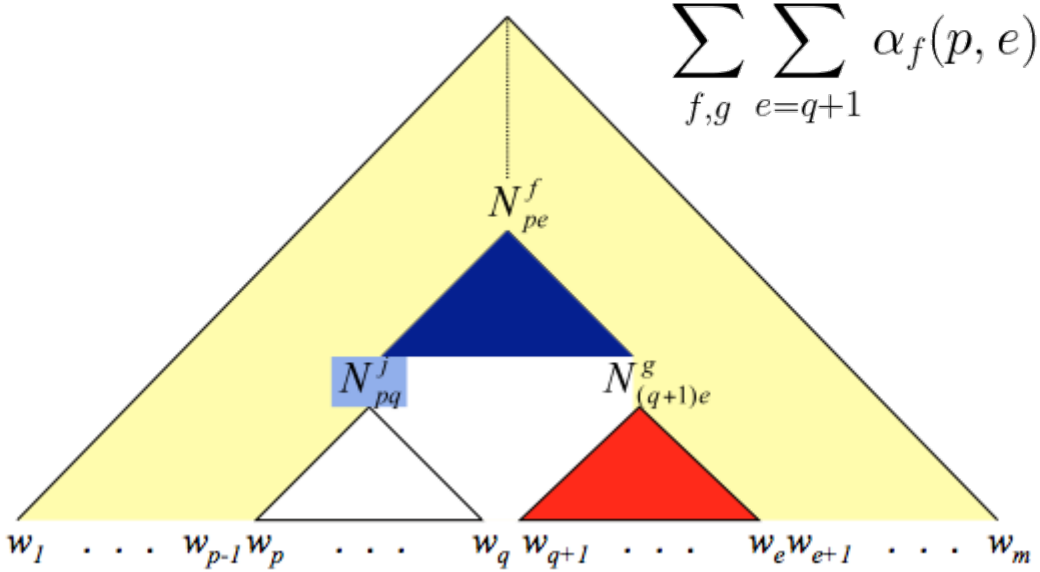
$$\beta_{NP}(1,2) = 0.064$$

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

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:

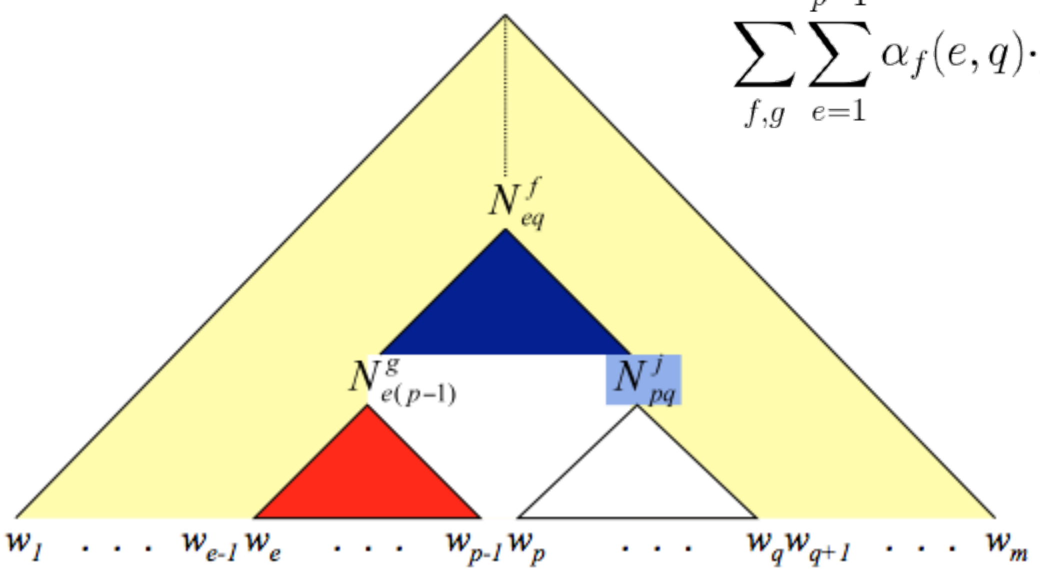
$$\sum_{f,g} \sum_{e=q+1}^{m} \alpha_f(p,e) \cdot \beta_g(q+1,e) \cdot P(N^f \rightarrow N^j N^g)$$



Calculating outside probability

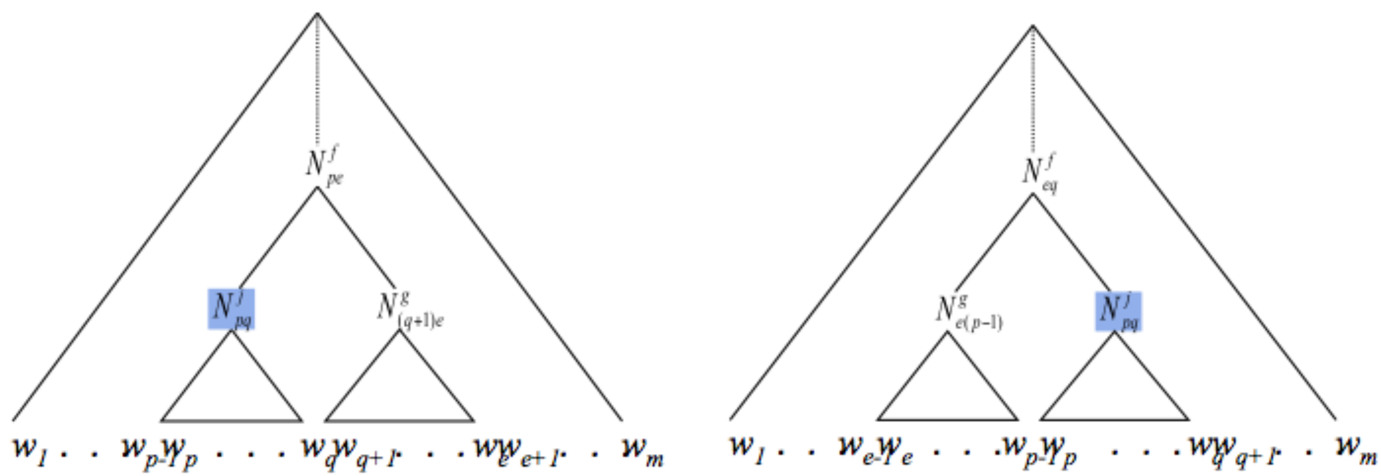
- The joint probability corresponding to the yellow, red and blue areas, assuming N^j was the R child of some non-terminal:

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



Calculating outside probability

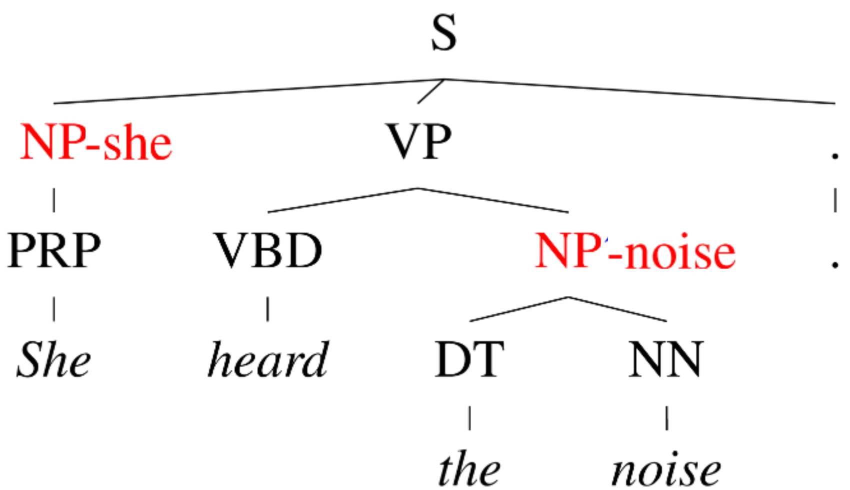
- The joint final joint probability (the sum over the L and R cases):



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

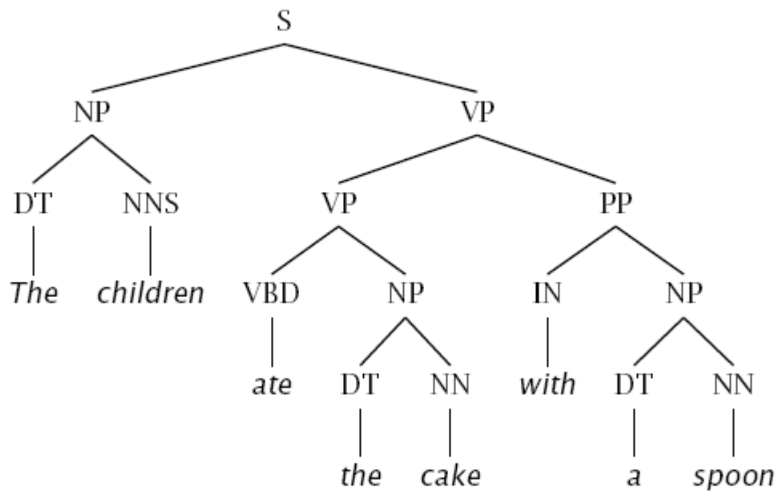
Lexicalization

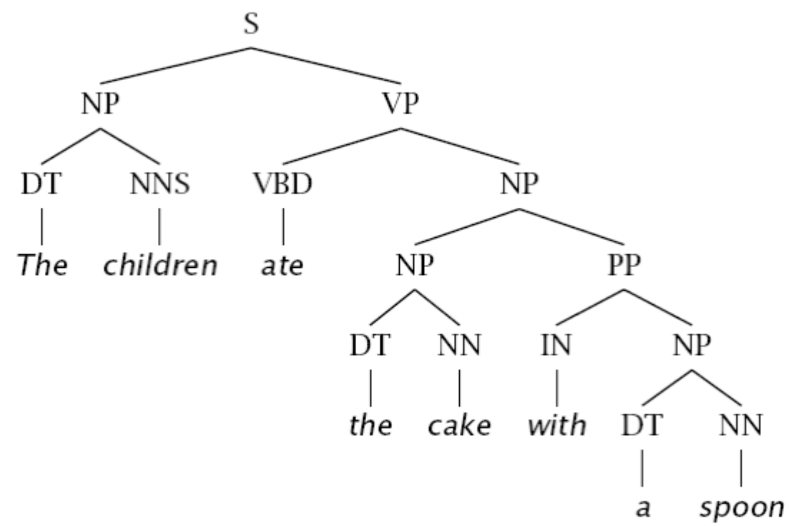
The Game of Designing a Grammar



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

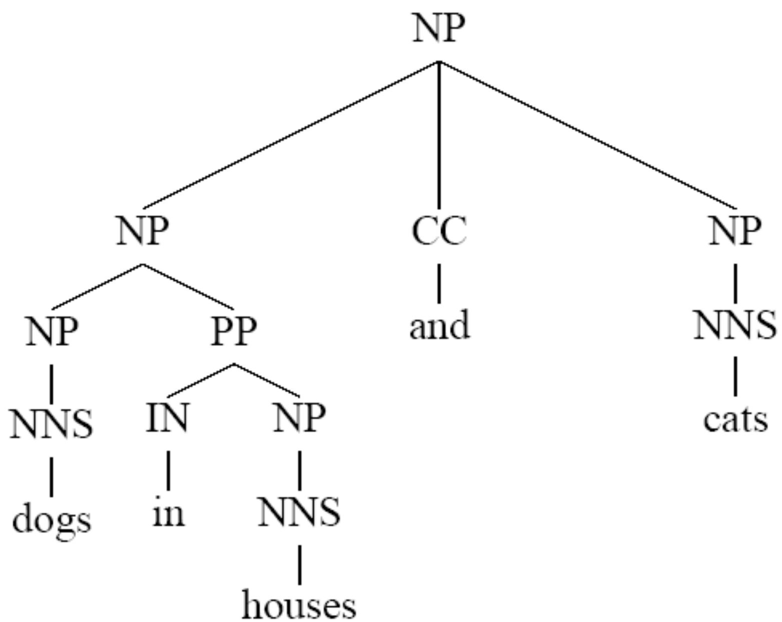
Problems with PCFGs

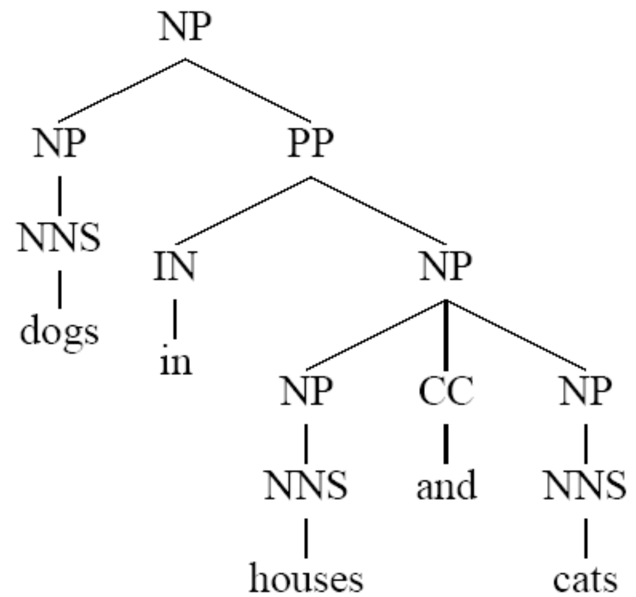




- If we do no annotation, these trees differ only in one rule:
  - VP → VP PP
  - NP → NP PP
- Parse will go one way or the other, regardless of words
- We addressed this in one way with unlexicalized grammars (how?)
- Lexicalization allows us to be sensitive to specific words

Problems with PCFGs

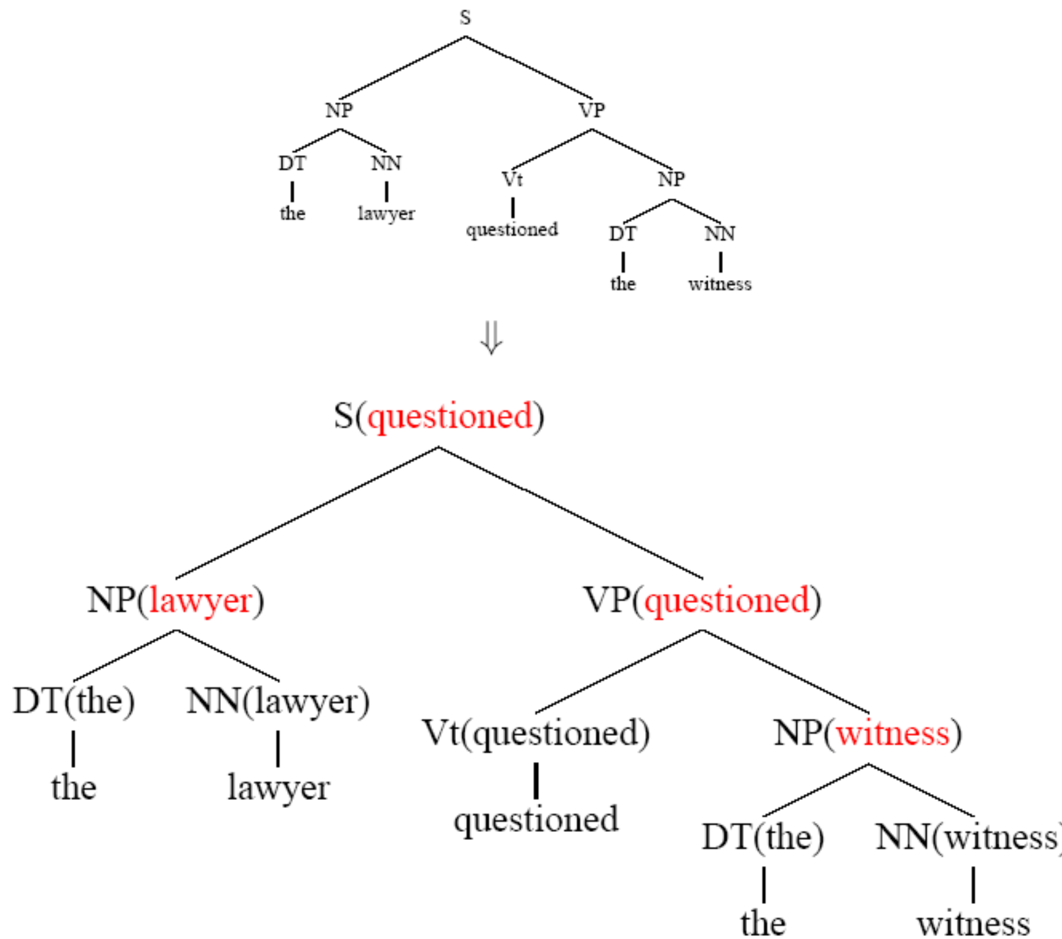




- What's different between basic PCFG scores here?
- What (lexical) correlations need to be scored?

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

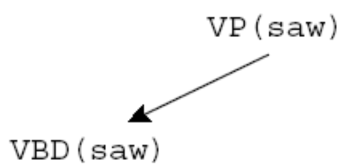
```
VP(saw) -> VBD(saw) NP-C(her) NP(today)
```

- Never going to get these atomically off of a treebank

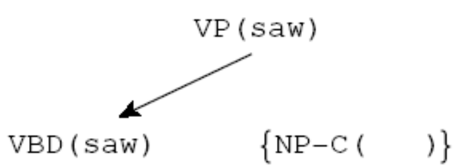
- Solution: break up derivation into smaller steps

Lexical Derivation Steps

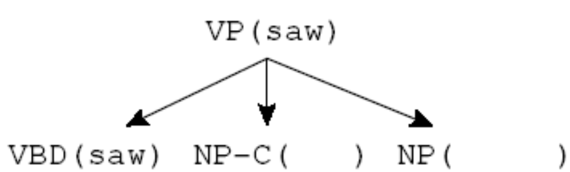
- A derivation of a local tree [Collins 99]



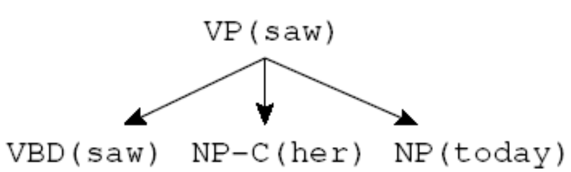
Choose a head tag and word



Choose a complement bag

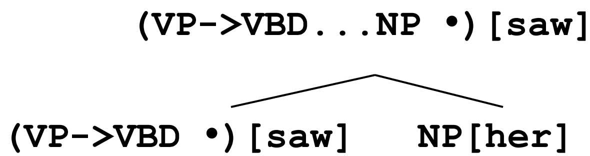


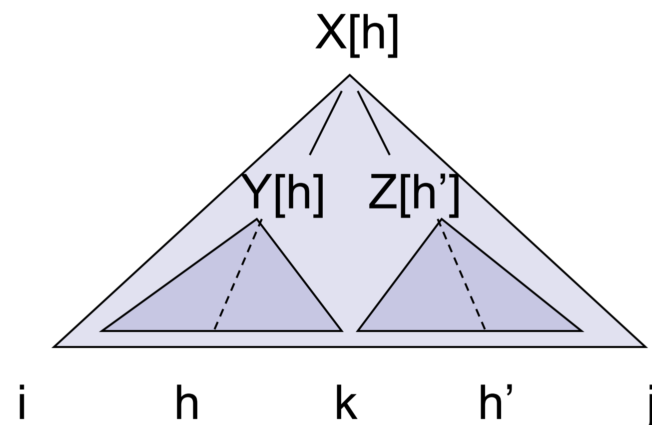
Generate children (incl. adjuncts)



Recursively derive children

Lexicalized CKY



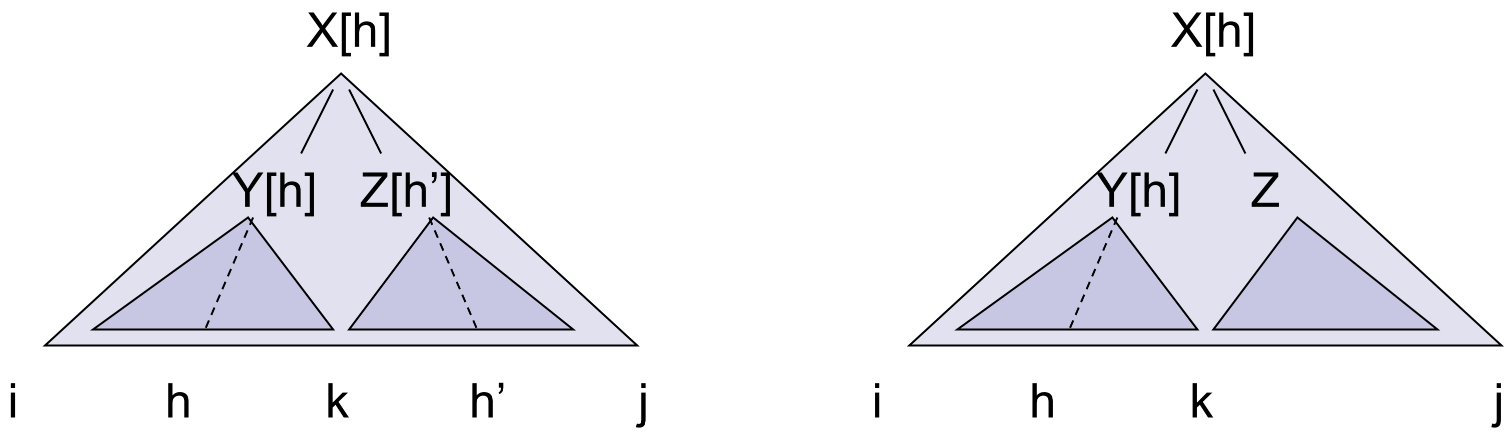



```
bestScore(X,i,j,h)
  if (j = i+1)
    return tagScore(X,s[i])
  else
    return
      max max score(X[h]->Y[h] Z[h']) *
  k,h',X->YZ  bestScore(Y,i,k,h) *
              bestScore(Z,k,j,h')
          max score(X[h]->Y[h'] Z[h]) *
  k,h',X->YZ  bestScore(Y,i,k,h') *
              bestScore(Z,k,j,h)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

Efficient Parsing for Lexical Grammars

Quartic Parsing

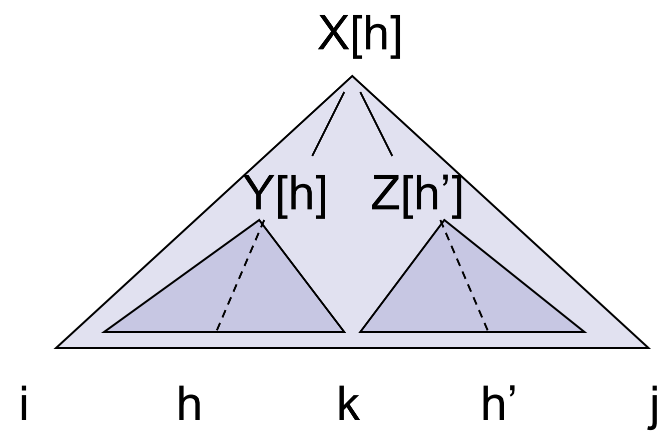
- Turns out, you can do (a little) better [Eisner 99]



- Gives an $O(n^4)$ algorithm
- Still prohibitive in practice if not pruned

Pruning with Beams

- The Collins parser prunes with per-cell beams [Collins 99]
  - Essentially, run the $O(n^5)$ CKY
  - Remember only a few hypotheses for each span <i,j>.
  - If we keep K hypotheses at each span, then we do at most $O(nK^2)$ work per span (why?)
  - Keeps things more or less cubic (and in practice is more like linear!)



- Also: certain spans are forbidden entirely on the basis of punctuation (crucial for speed)

Pruning with a PCFG

- The Charniak parser prunes using a two-pass, coarse-to-fine approach [Charniak 97+]
  - First, parse with the base grammar
  - For each X:[i,j] calculate P(X|i,j,s)
    - This isn't trivial, and there are clever speed ups
  - Second, do the full $O(n^5)$ CKY
    - Skip any X :[i,j] which had low (say, < 0.0001) posterior
  - Avoids almost all work in the second phase!

- Charniak et al 06: can use more passes
- Petrov et al 07: can use many more passes

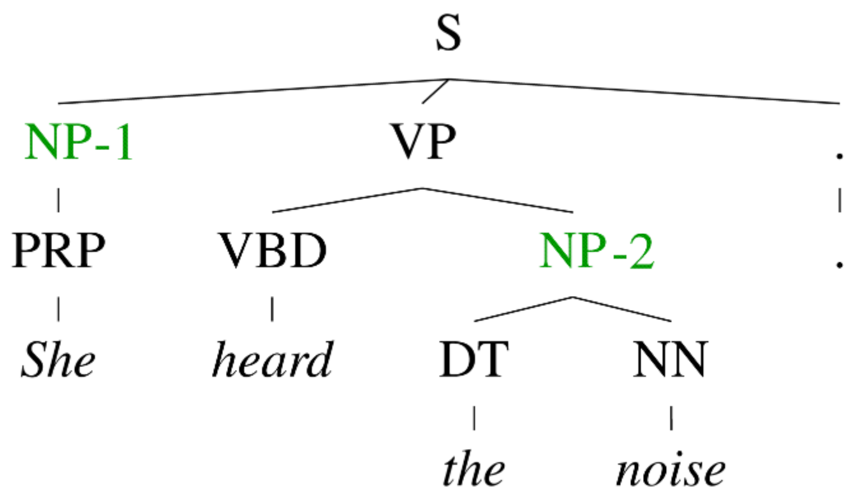
Results

- 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

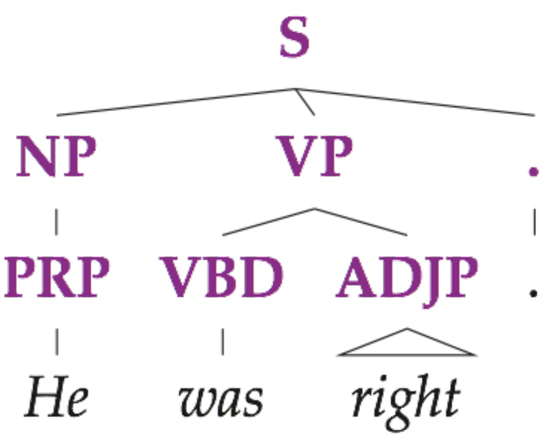
Latent Variable PCFGs

The Game of Designing a Grammar



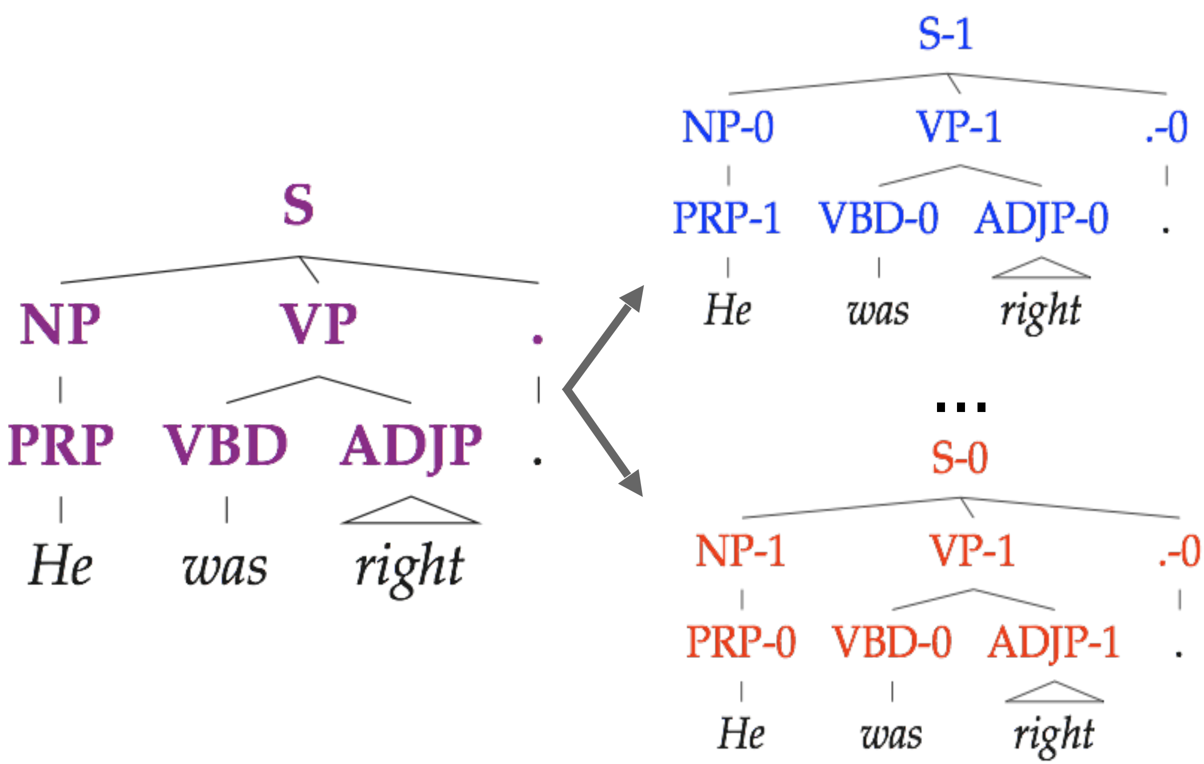
- Annotation refines base treebank symbols to improve statistical fit of the grammar
  - Parent annotation [Johnson '98]
  - Head lexicalization [Collins '99, Charniak '00]
  - Automatic clustering?

Latent Variable Grammars



Parse Tree T
Sentence w

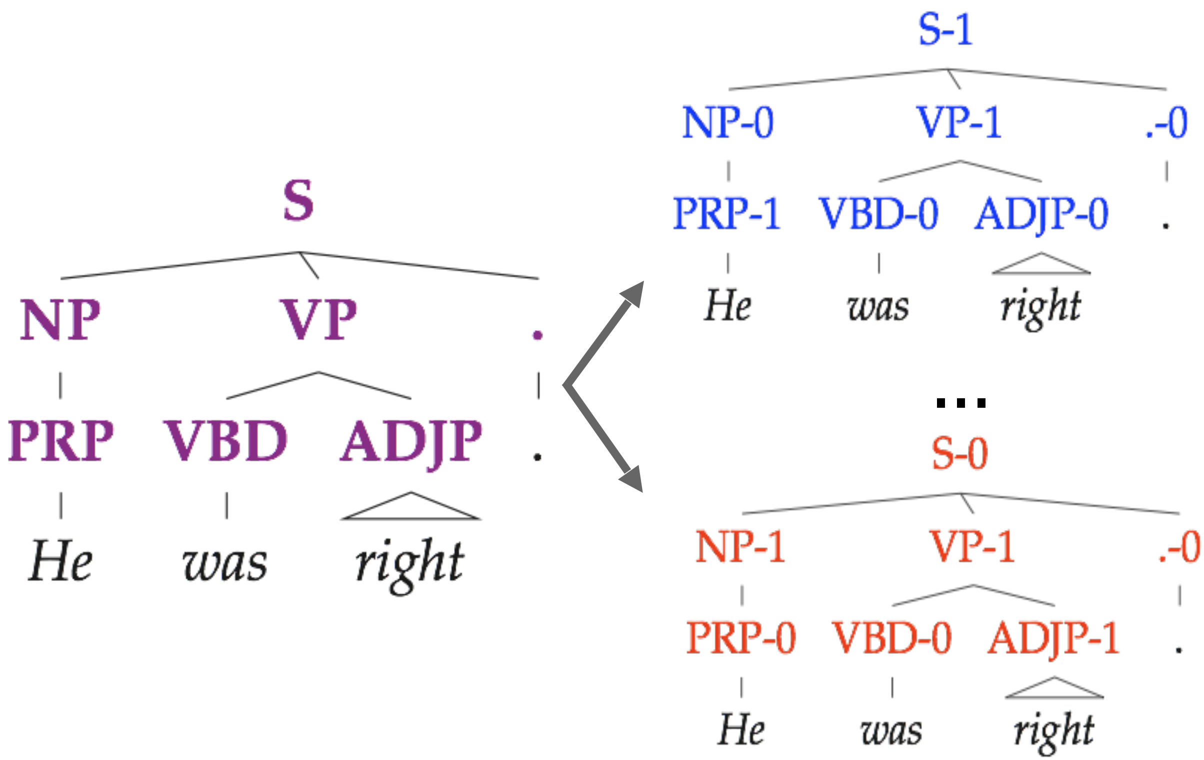
Latent Variable Grammars



Parse Tree T
Sentence w

Derivations $t : T$

Latent Variable Grammars



Parse Tree T

Sentence w

Derivations $t : T$

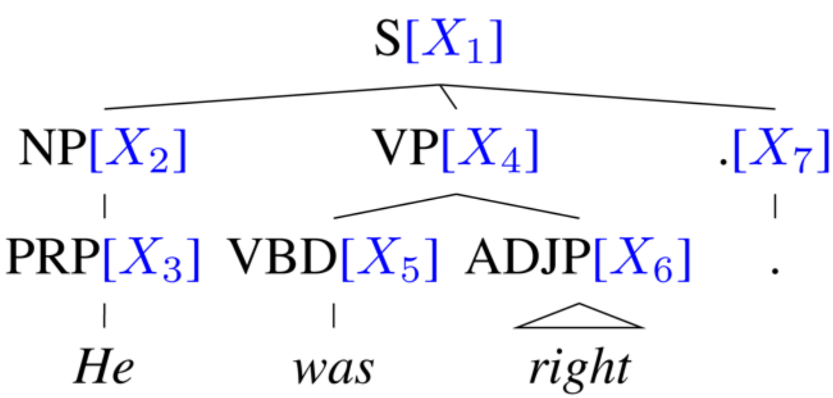
Grammar G	
$S_0 \rightarrow NP_0\ VP_0$	?
$S_0 \rightarrow NP_1\ VP_0$	?
$S_0 \rightarrow NP_0\ VP_1$	?
$S_0 \rightarrow NP_1\ VP_1$	?
$S_1 \rightarrow NP_0\ VP_0$	?
...	
$S_1 \rightarrow NP_1\ VP_1$	?
...	
$NP_0 \rightarrow PRP_0$	?
$NP_0 \rightarrow PRP_1$	?
...	

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

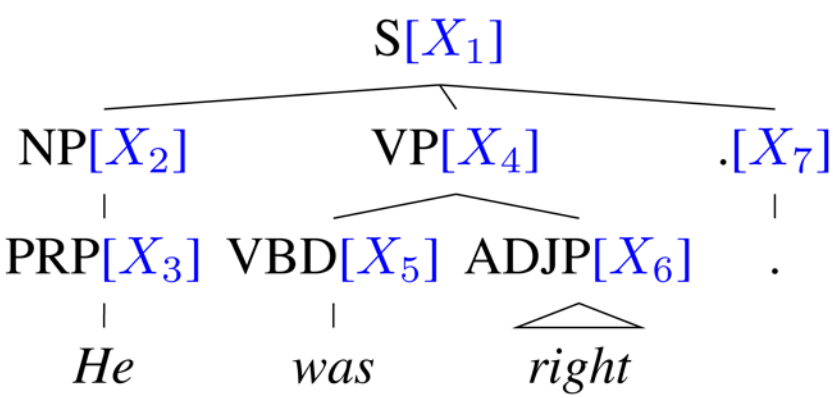
EM algorithm:



Learning Latent Annotations

EM algorithm:

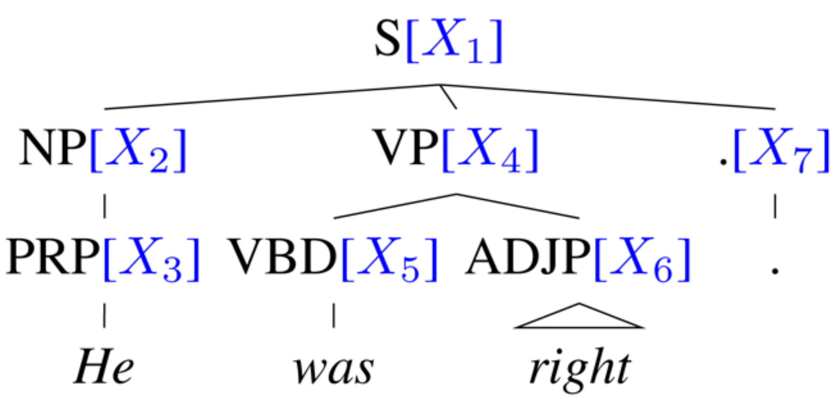
- Brackets are known
- Base categories are known
- Only induce subcategories

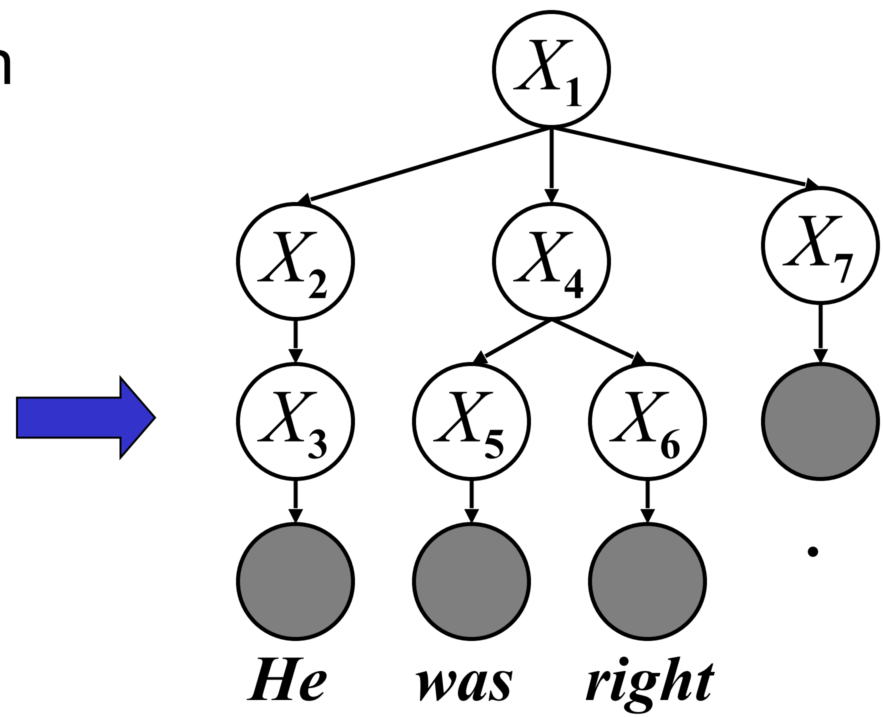


Learning Latent Annotations

EM algorithm:

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

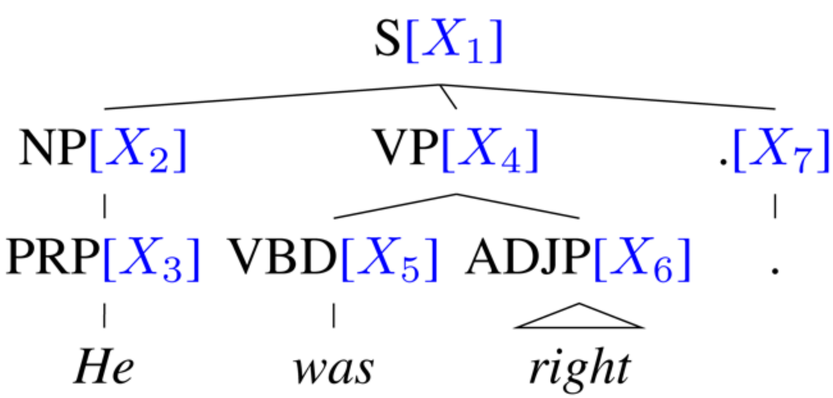


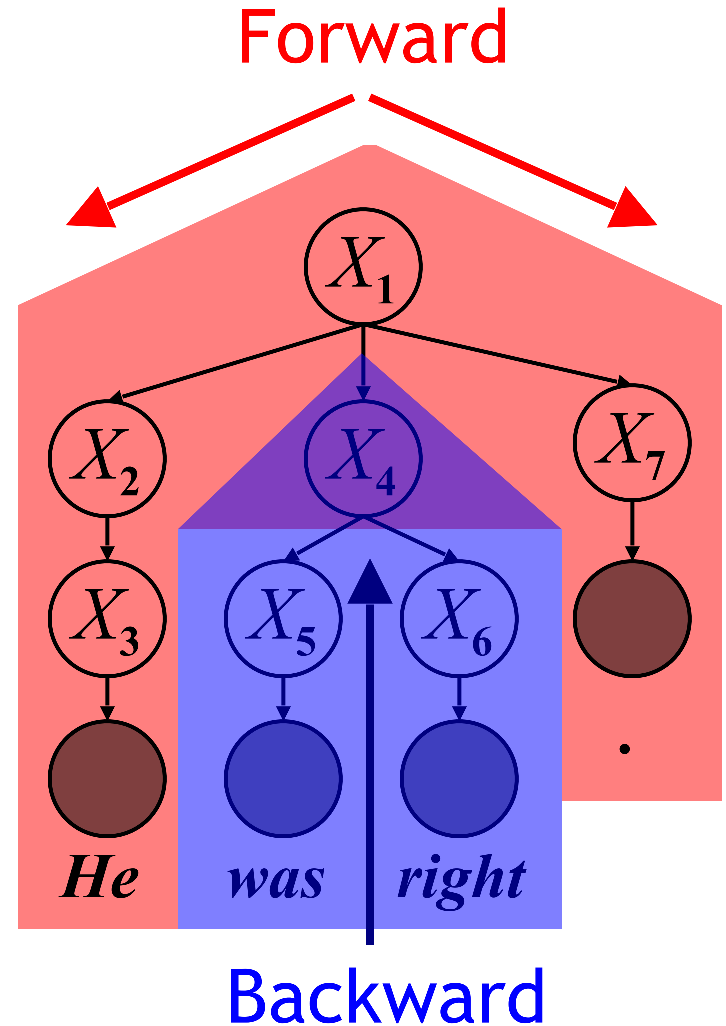


Learning Latent Annotations

EM algorithm:

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





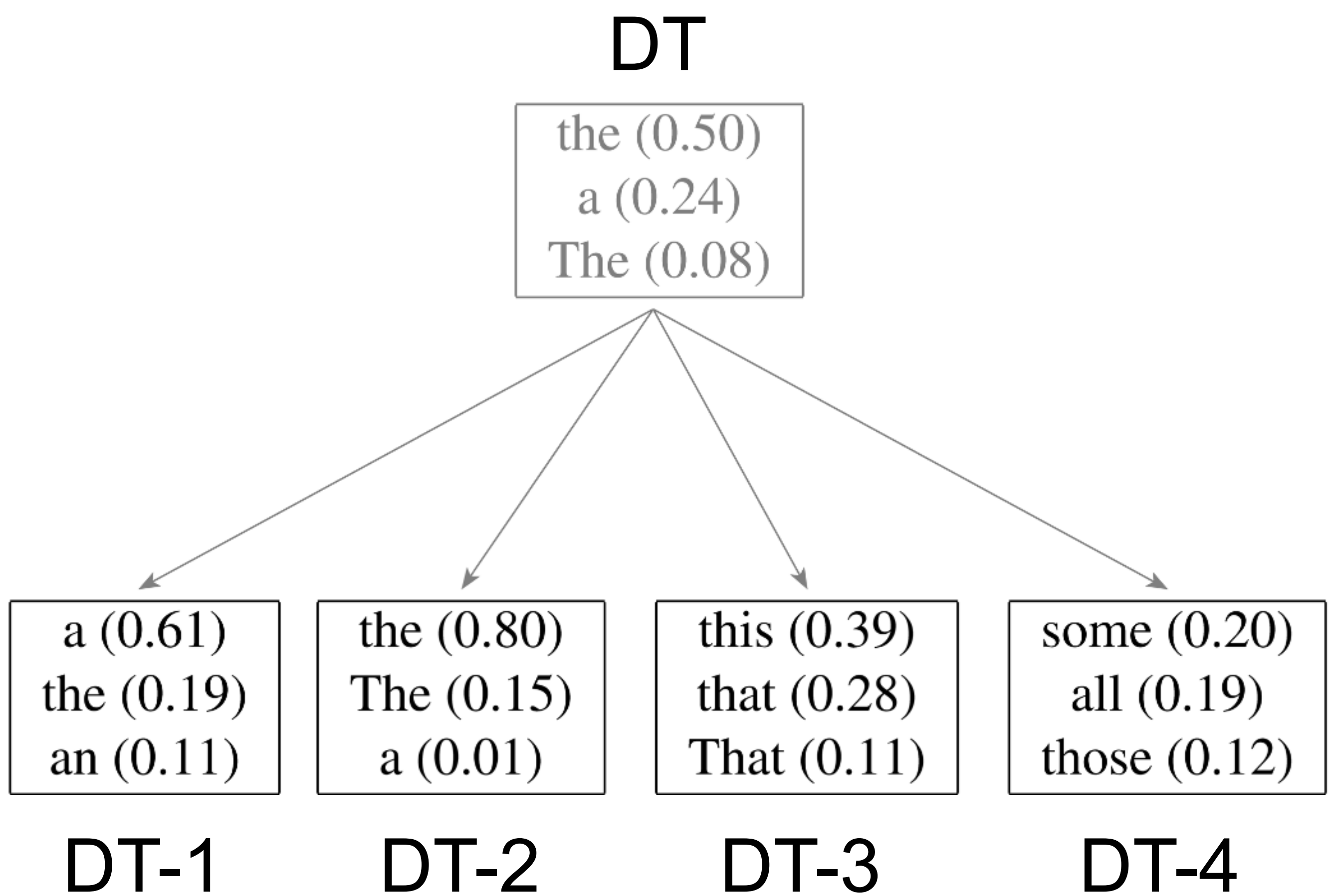
Just like Forward-Backward for HMMs.

Refinement of the DT tag

DT

the (0.50)
a (0.24)
The (0.08)

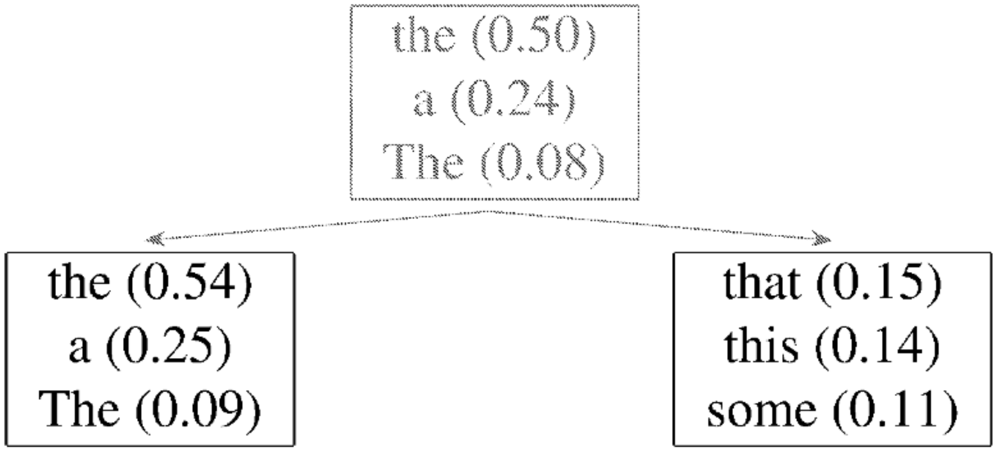
Refinement of the DT tag



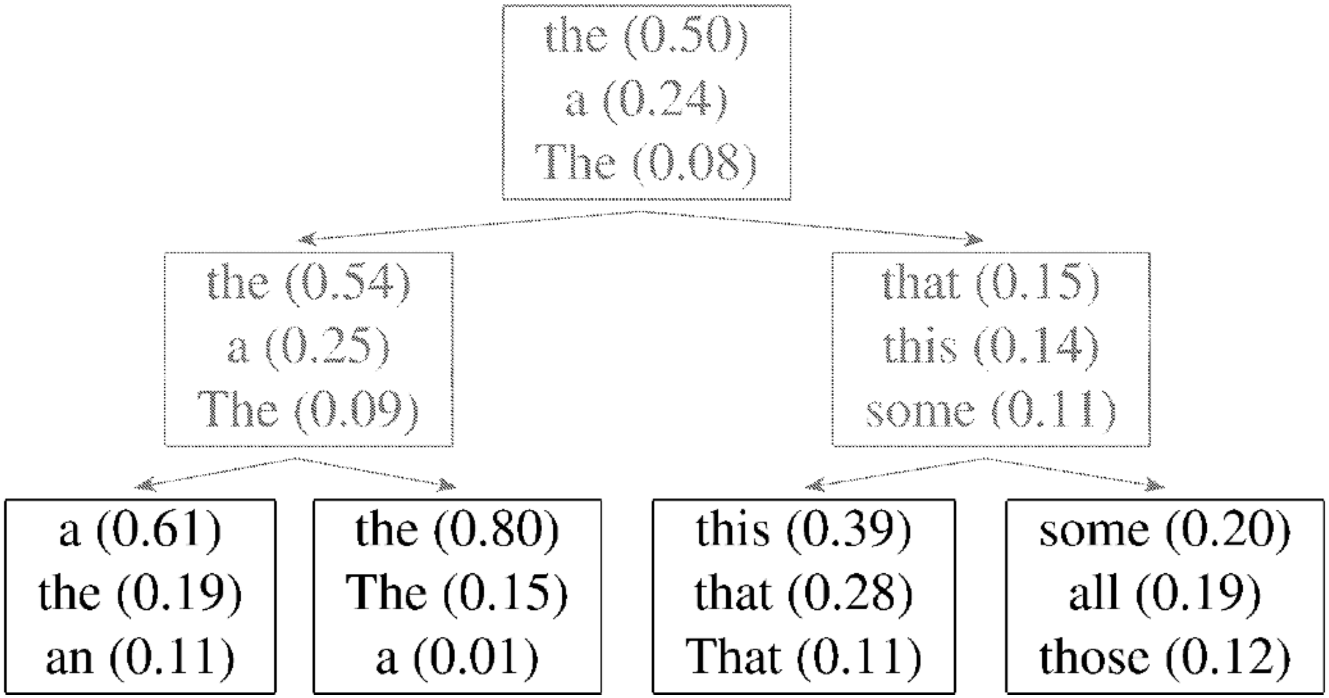
Hierarchical refinement

the (0.50)
a (0.24)
The (0.08)

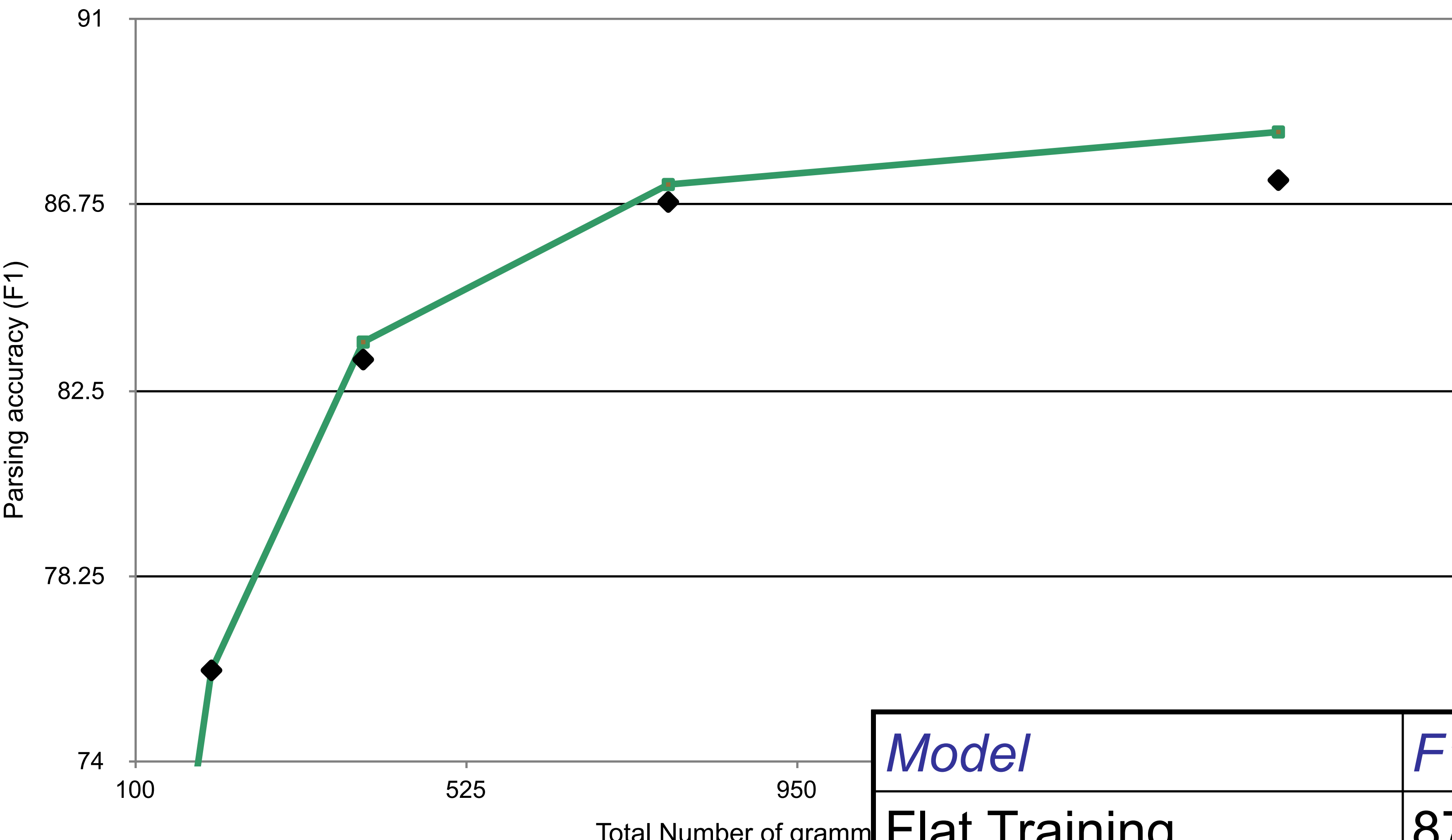
Hierarchical refinement



Hierarchical refinement



Hierarchical Estimation Results



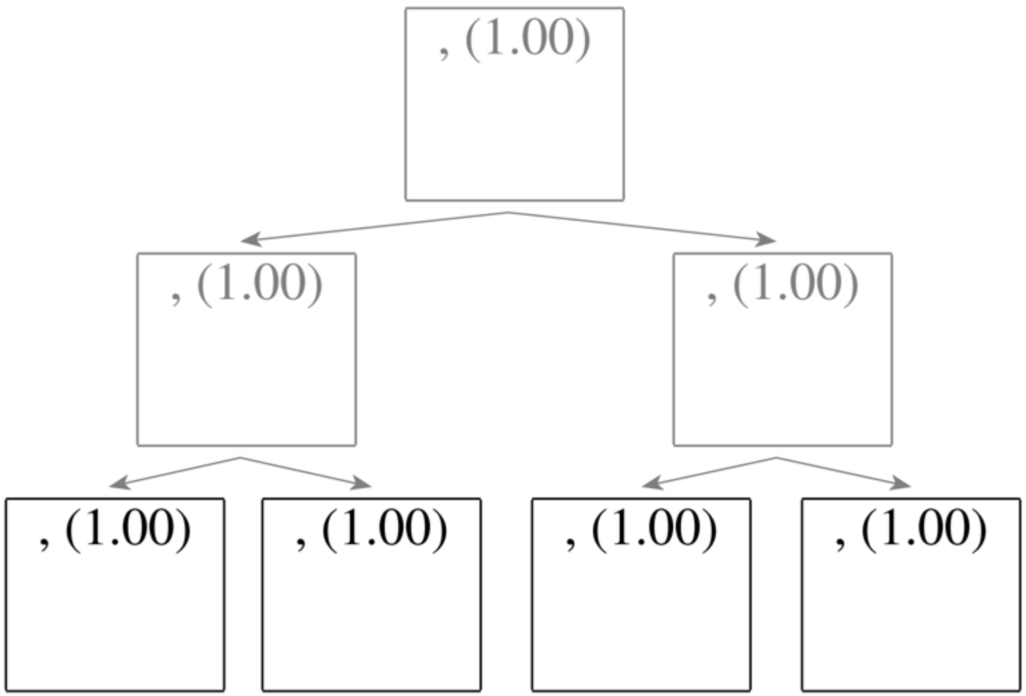
Model	*F1*
Flat Training	87.3
Hierarchical Training	88.4

Refinement of the , tag

- Splitting all categories equally is wasteful:

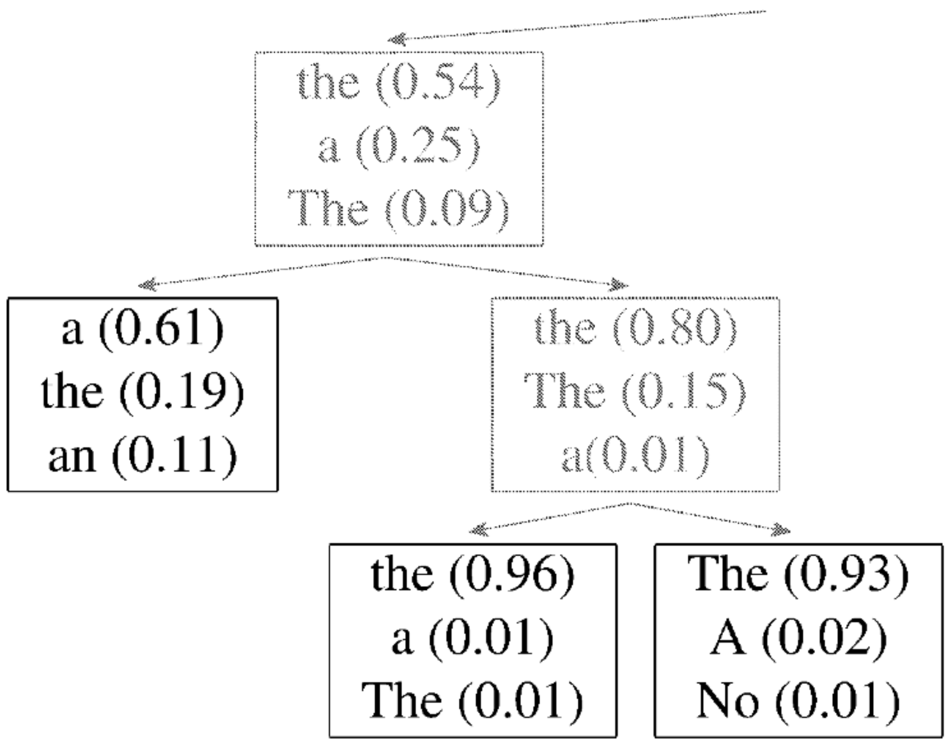
Refinement of the , tag

- Splitting all categories equally is wasteful:

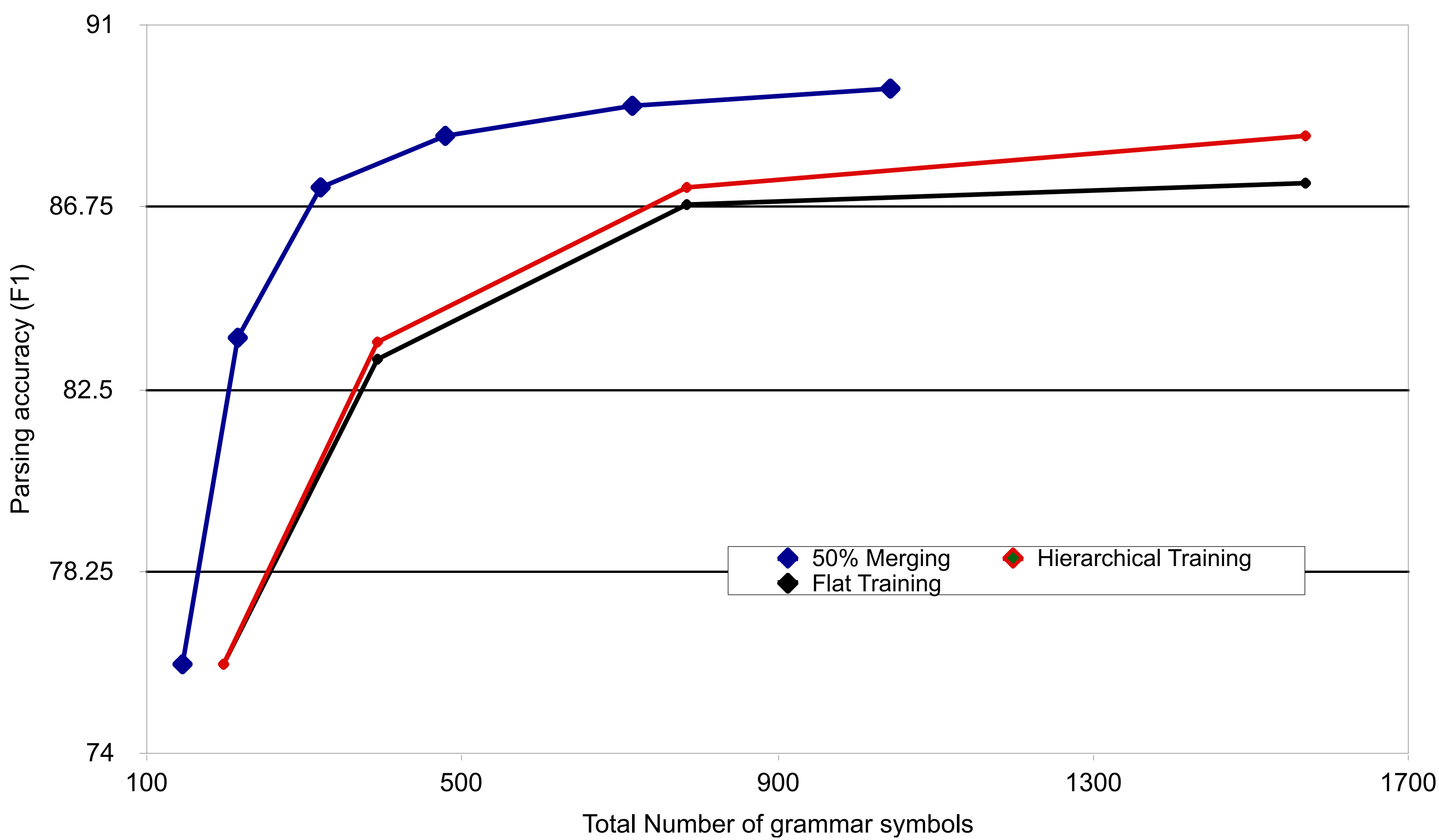


Adaptive Splitting

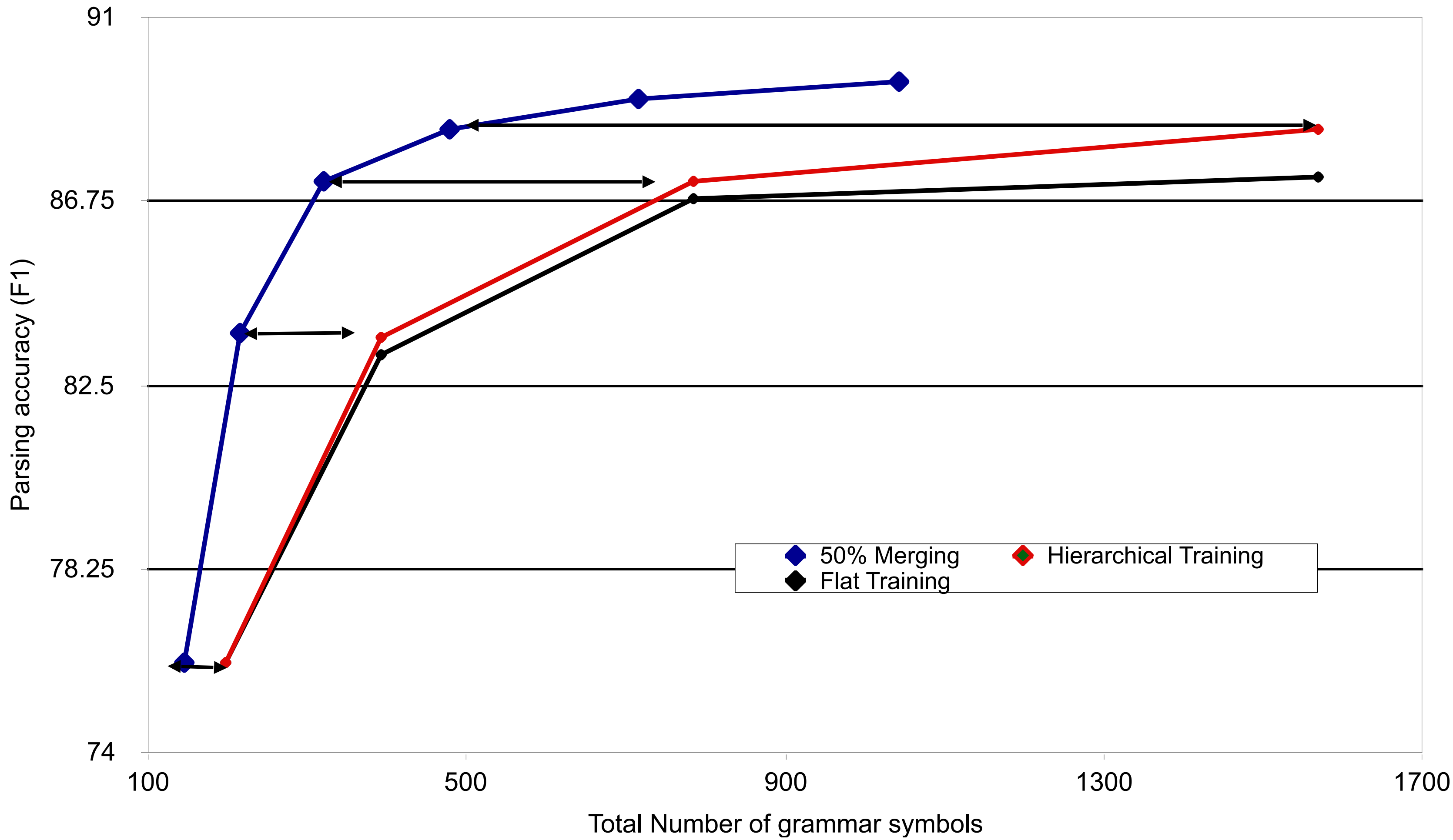
- Want to split complex categories more
- Idea: split everything, roll back splits which were least useful



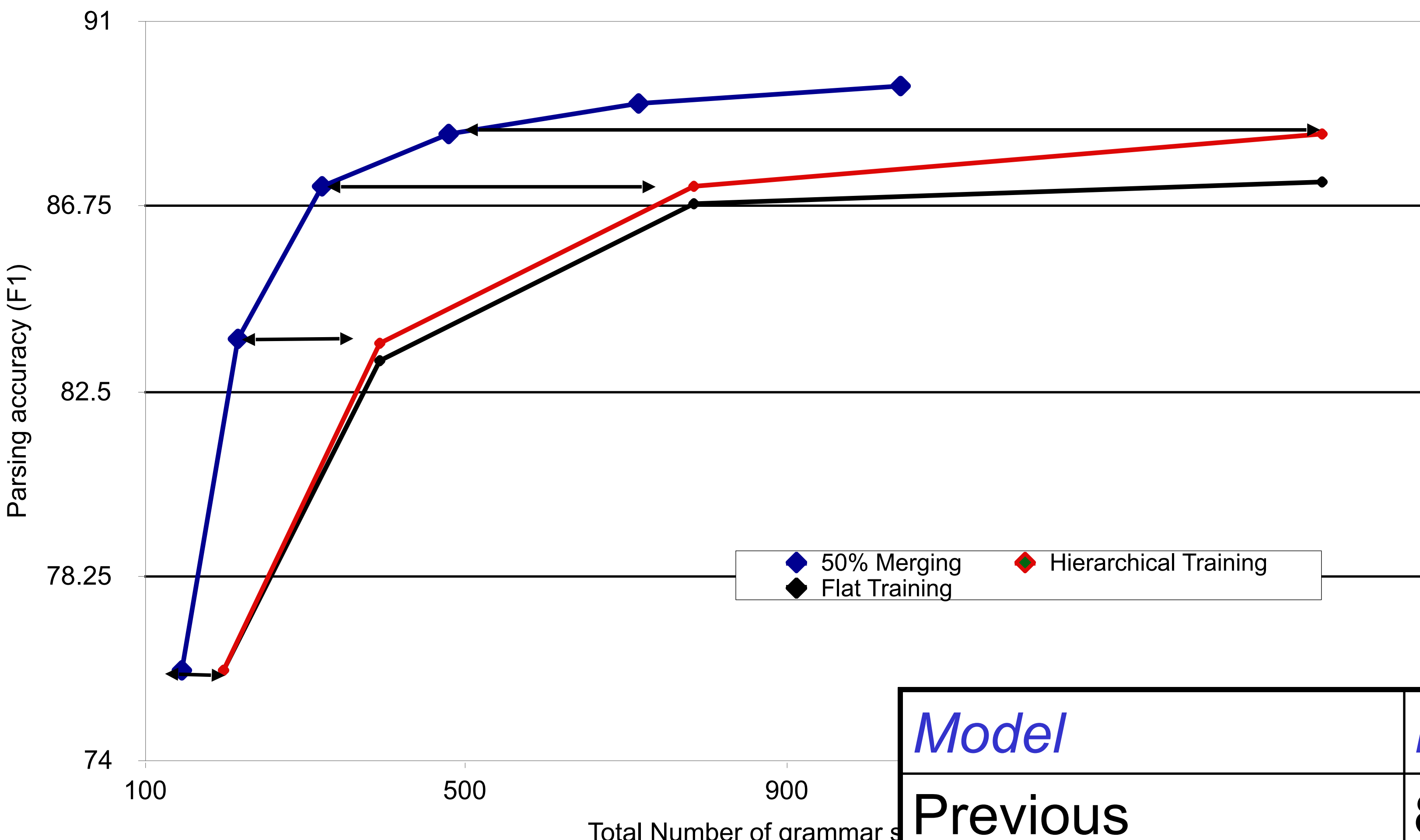
Adaptive Splitting Results



Adaptive Splitting Results

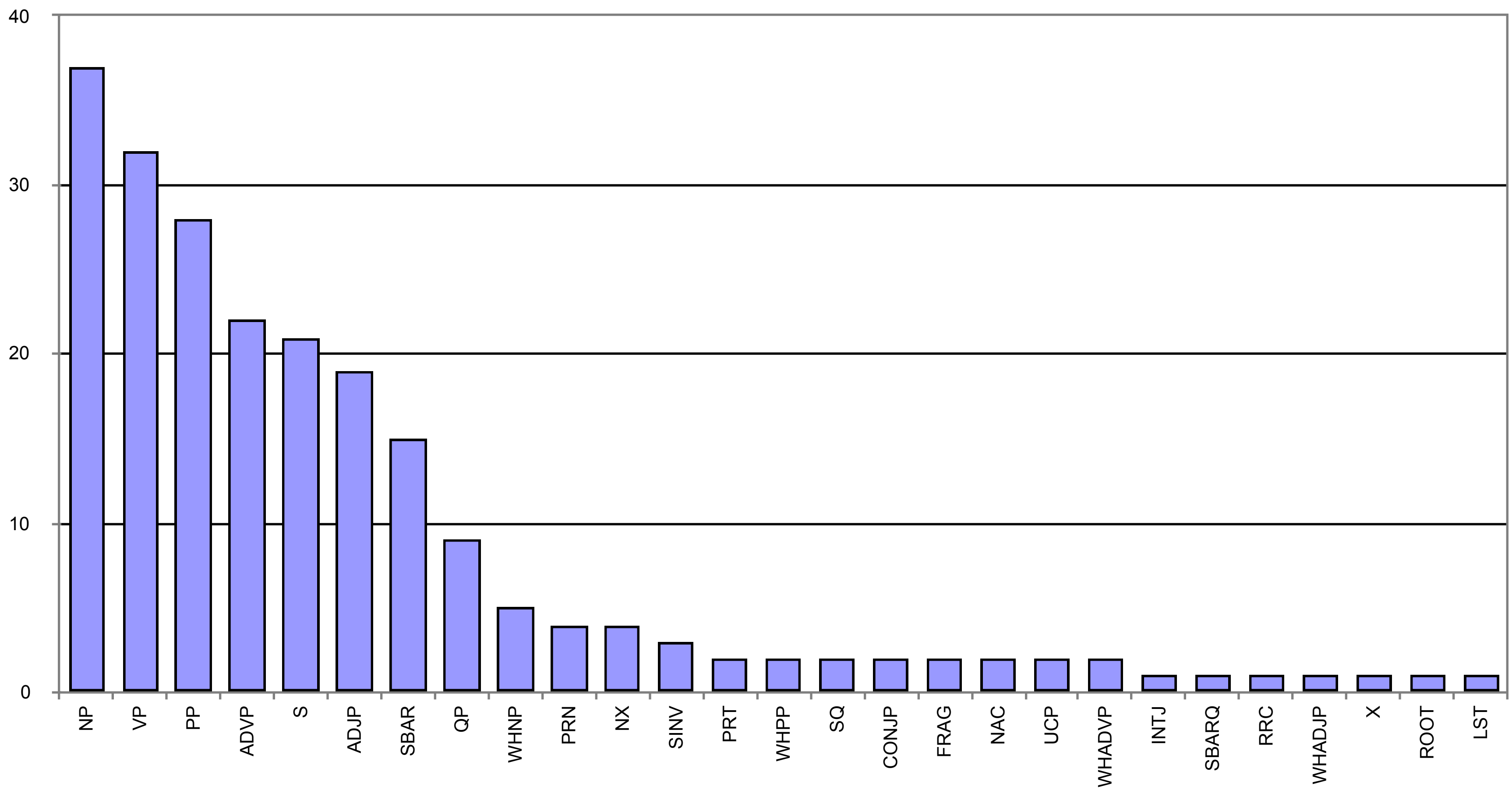


Adaptive Splitting Results

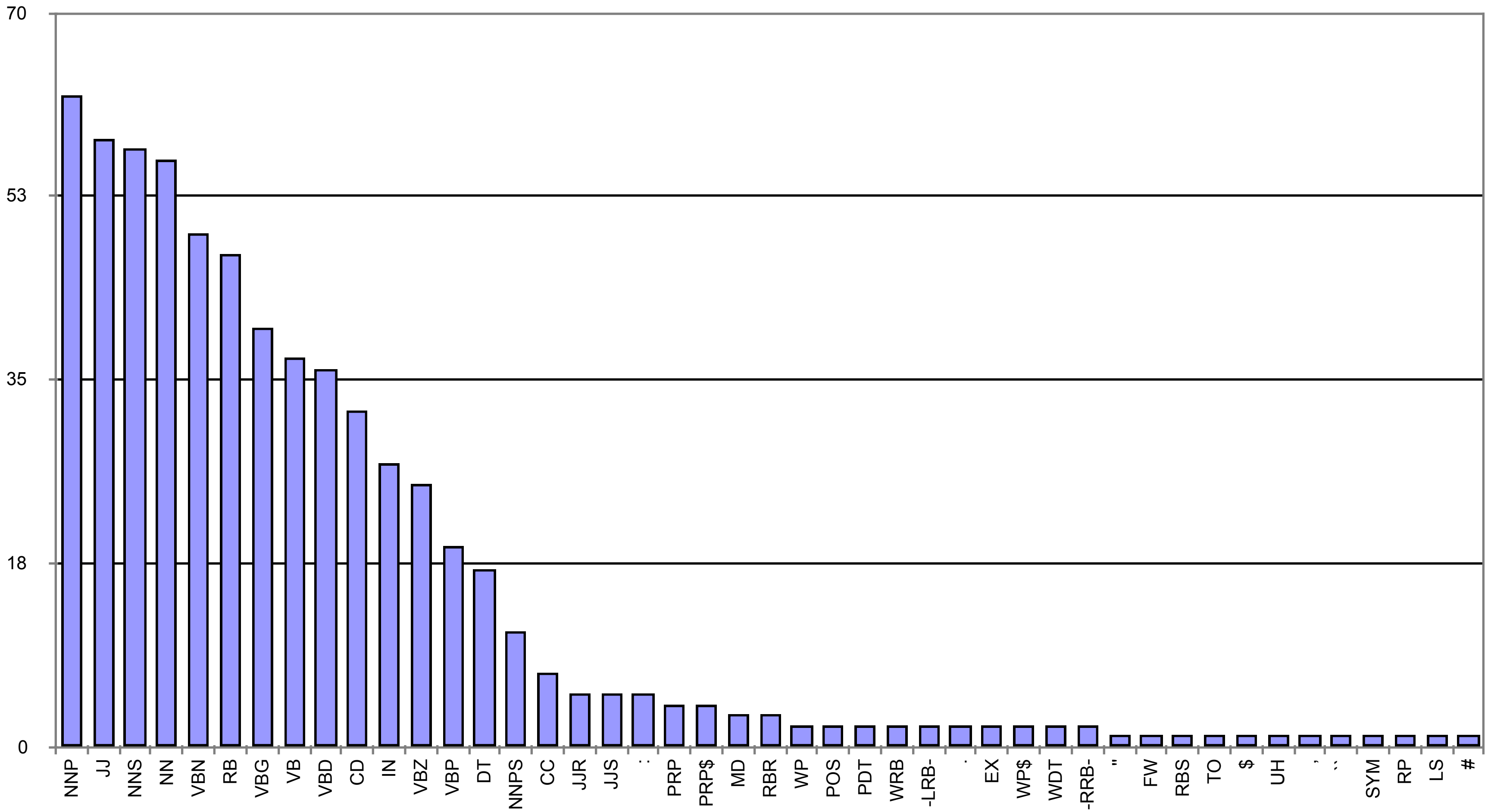


Model	*F1*
Previous	88.4
With 50% Merging	89.5

Number of Phrasal Subcategories



Number of Lexical Subcategories



Learned Splits

- Proper Nouns (NNP):

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

- 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)

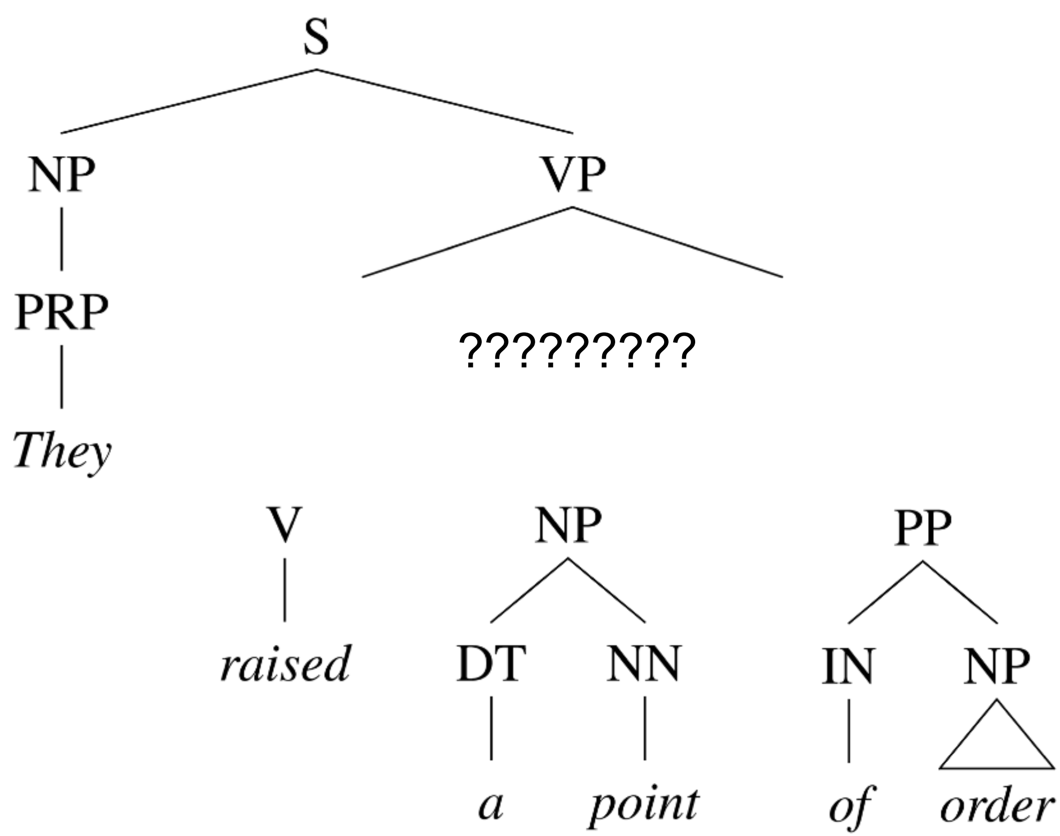
		≤ 40 words F1	all F1
ENG	Charniak&Johnson '05 (generative)	90.1	89.6
	Split / Merge	**90.6**	**90.1**
GER	Dubey '05	76.3	-
	Split / Merge	**80.8**	**80.1**
CHN	Chiang et al. '02	80.0	76.6
	Split / Merge	**86.3**	**83.4**

Still higher numbers from reranking / self-training methods

Efficient Parsing for Hierarchical Grammars

Coarse-to-Fine Inference

- Example: PP attachment



Hierarchical Pruning

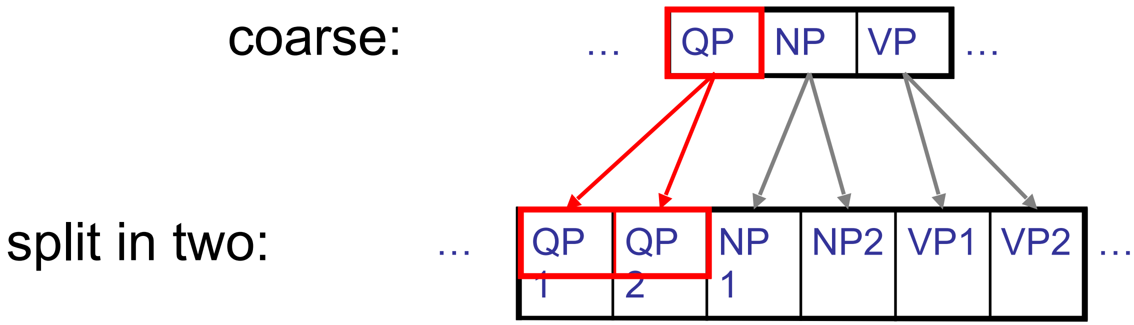
Hierarchical Pruning

coarse: … QP NP VP …

Hierarchical Pruning

coarse: … | QP | NP | VP | …

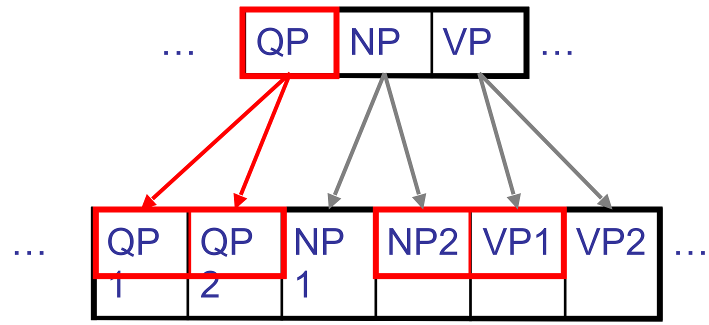
Hierarchical Pruning



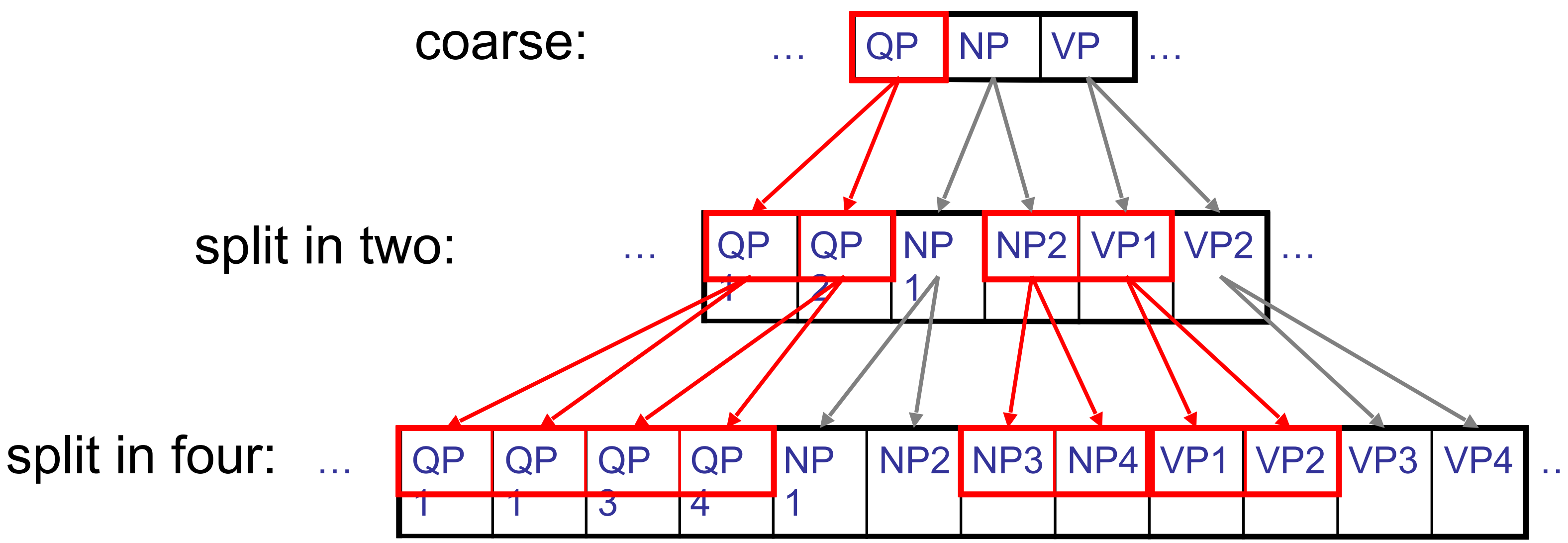
Hierarchical Pruning

coarse:

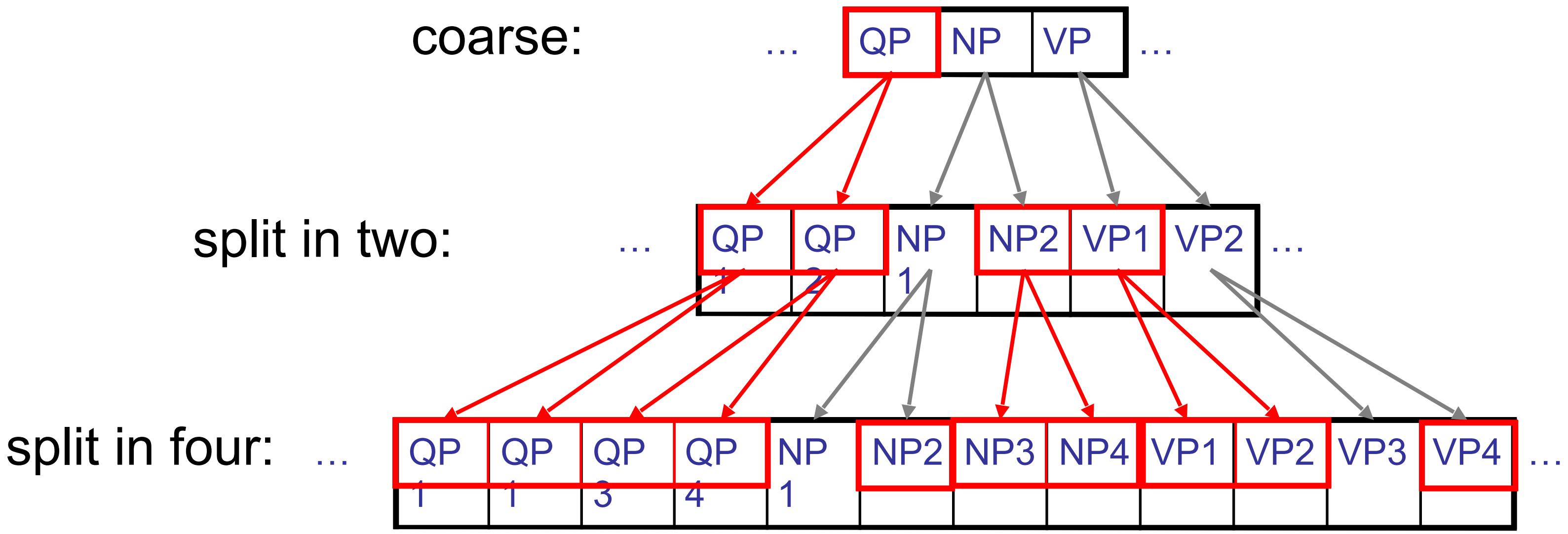
split in two:



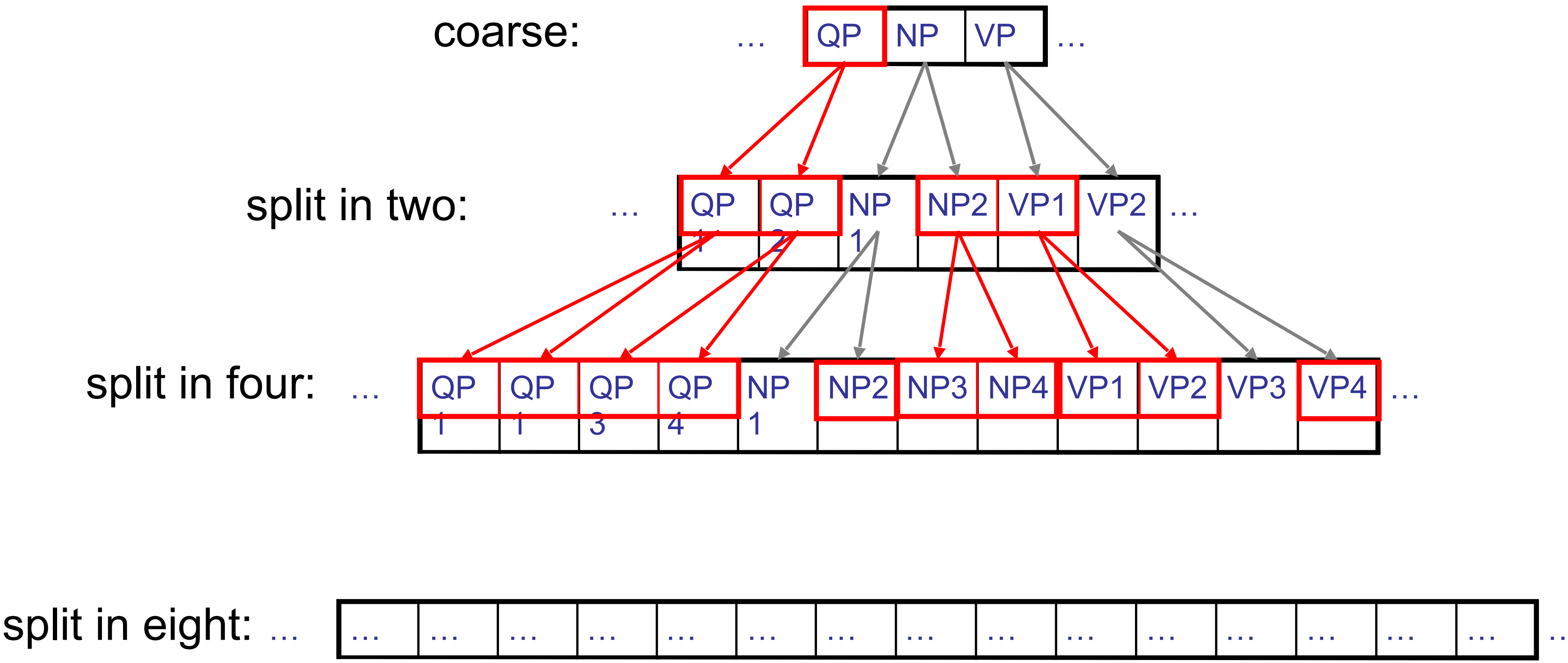
Hierarchical Pruning



Hierarchical Pruning

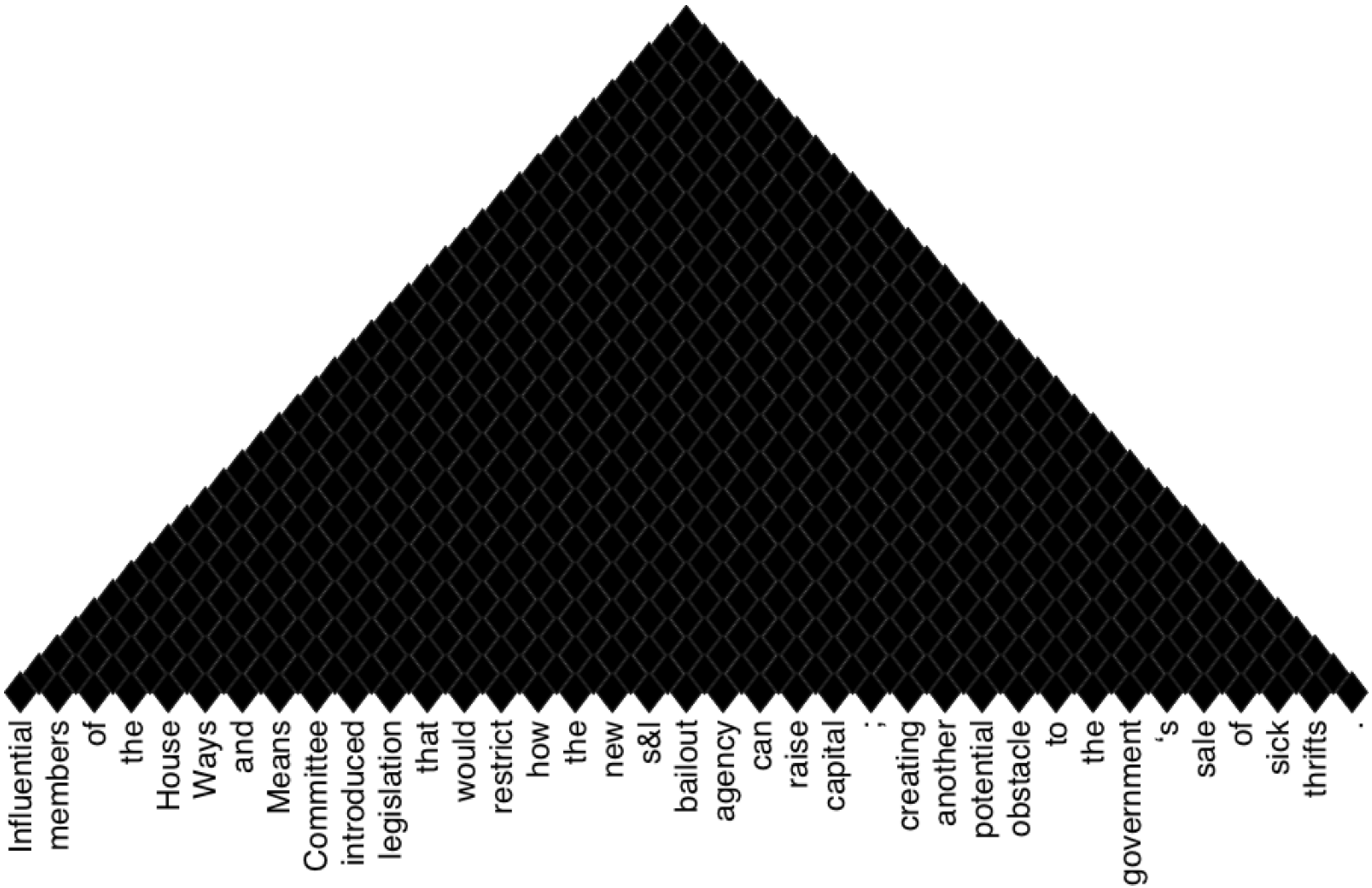


Hierarchical Pruning

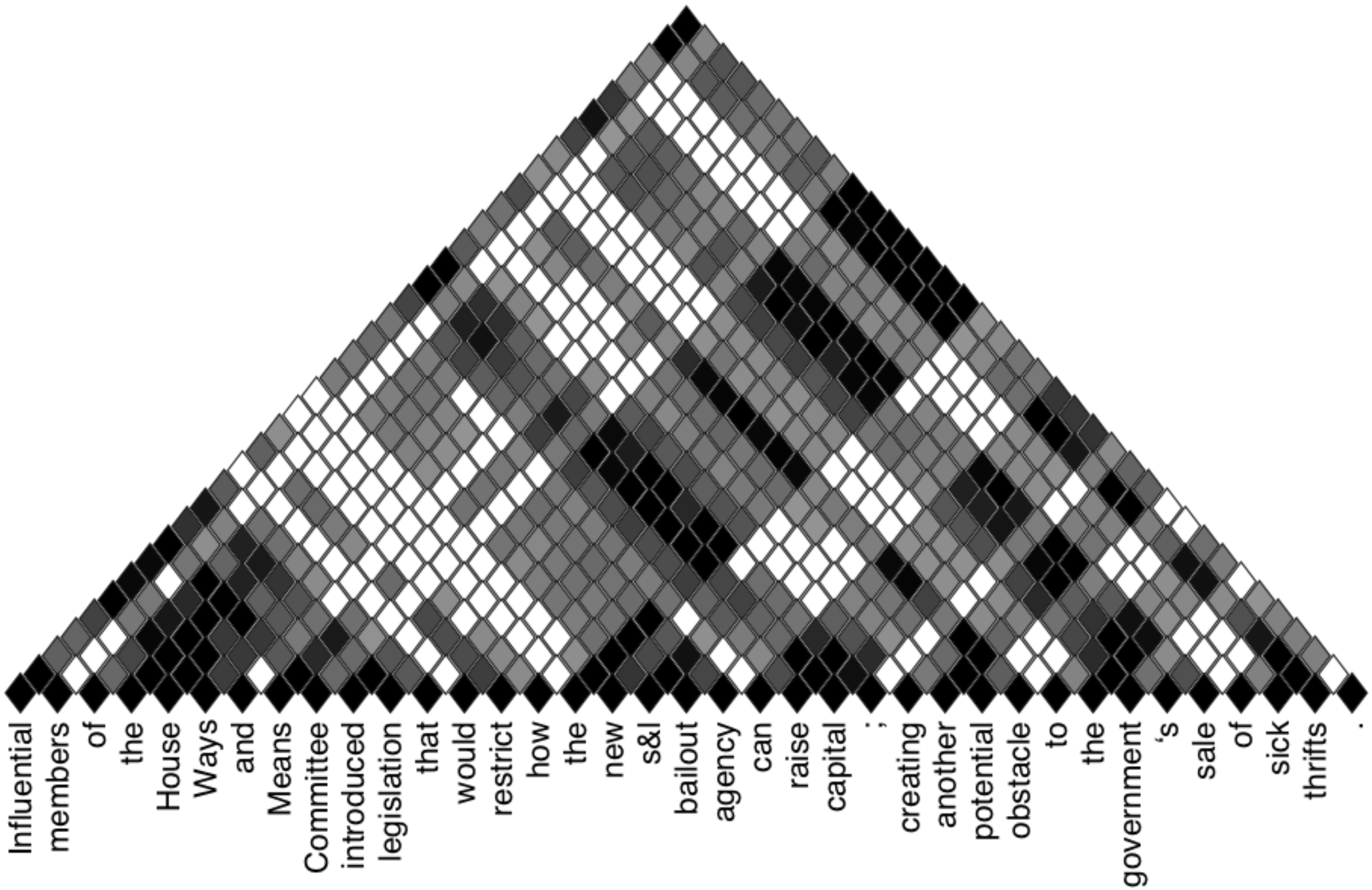


Bracket Posteriors

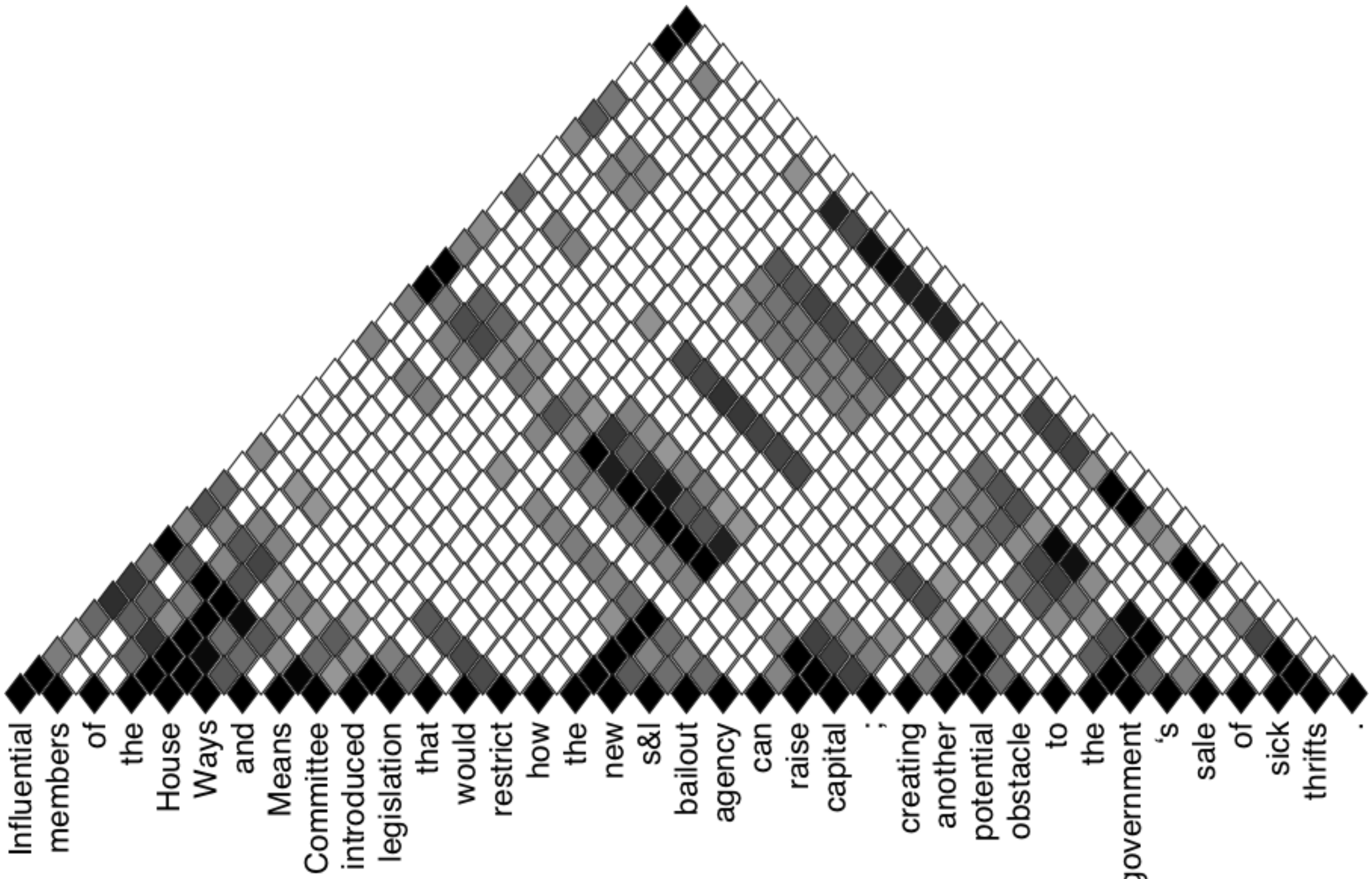
Bracket Posteriors



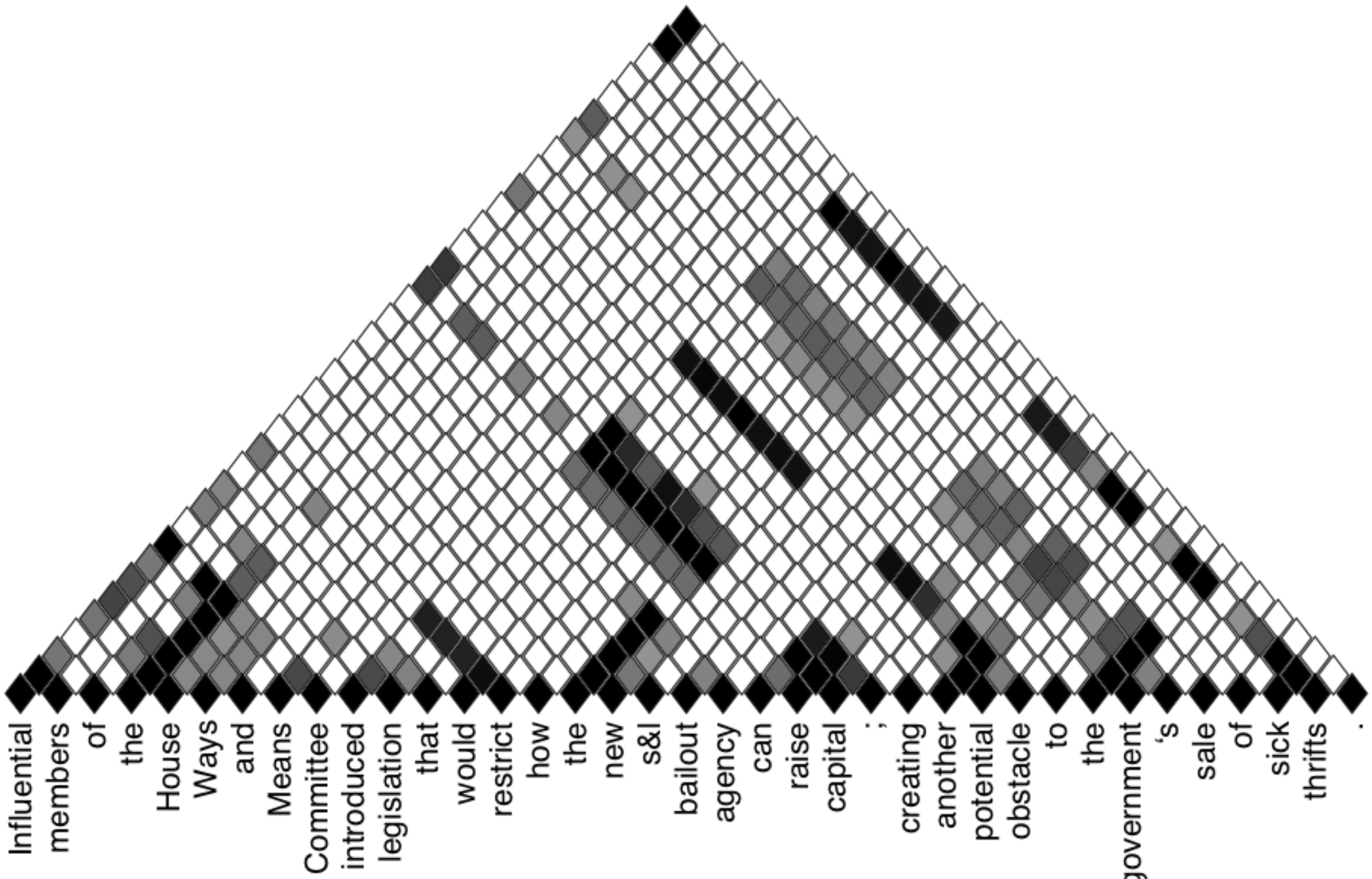
Bracket Posteriors



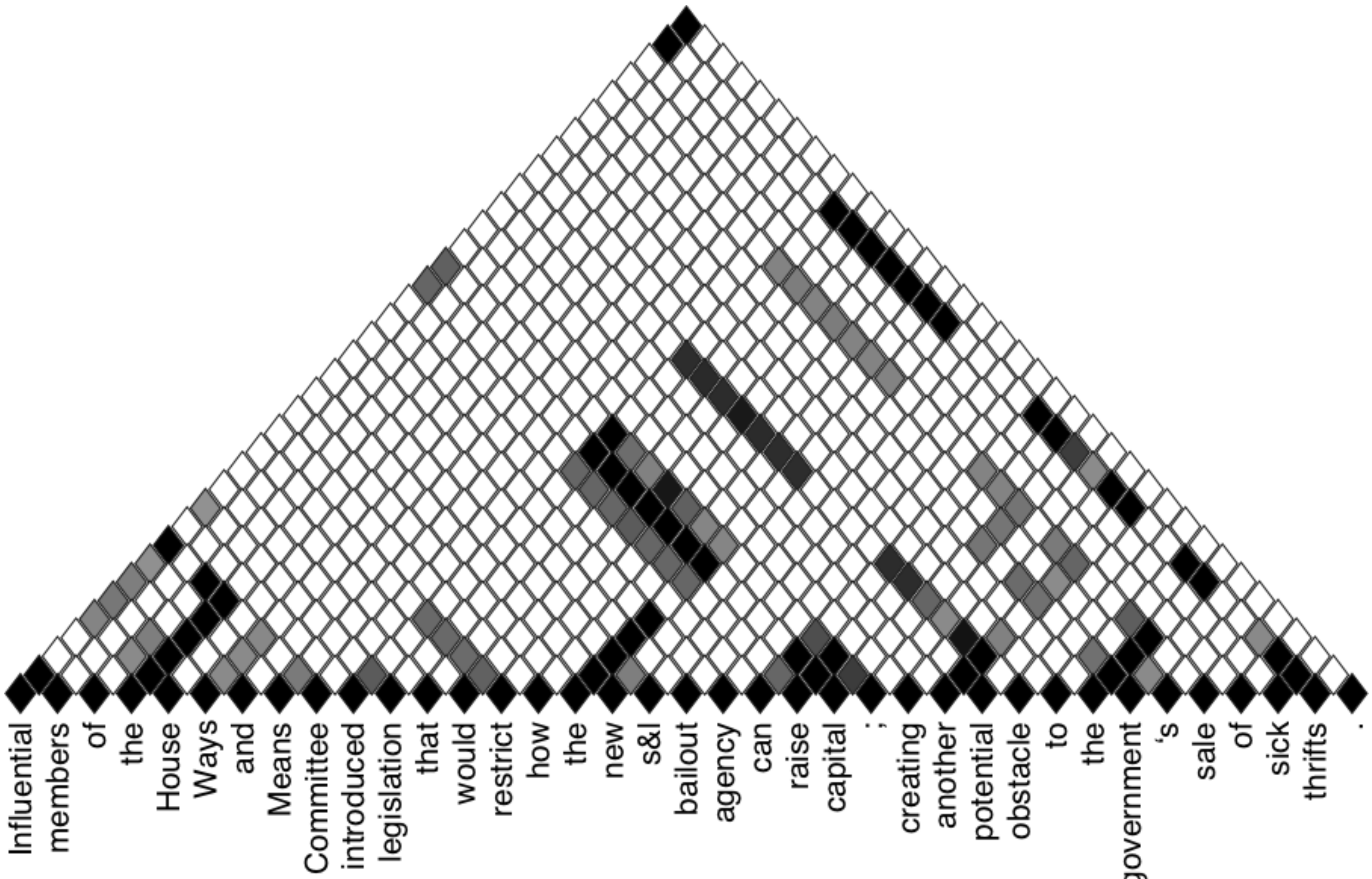
Bracket Posteriors



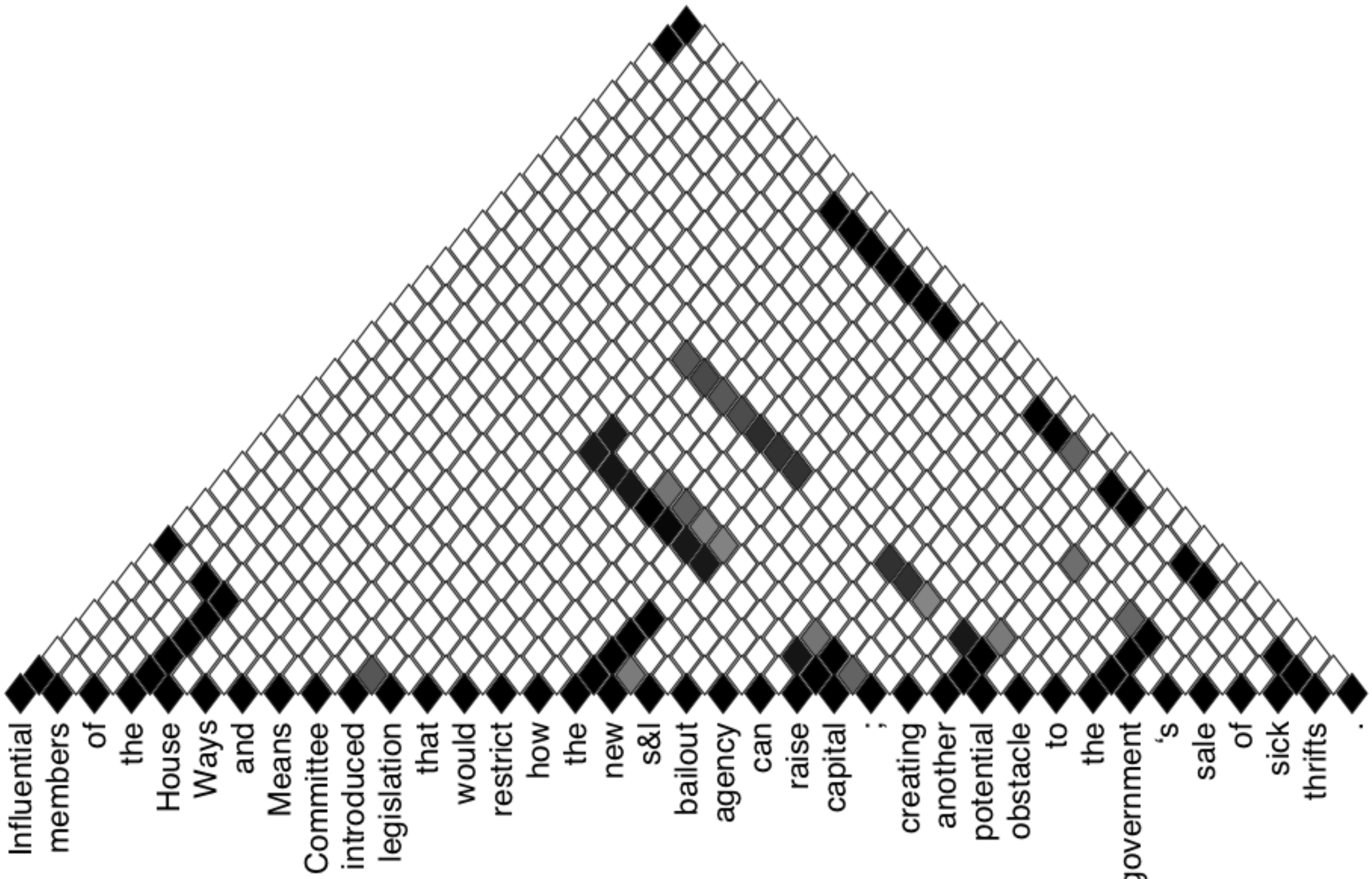
Bracket Posteriors



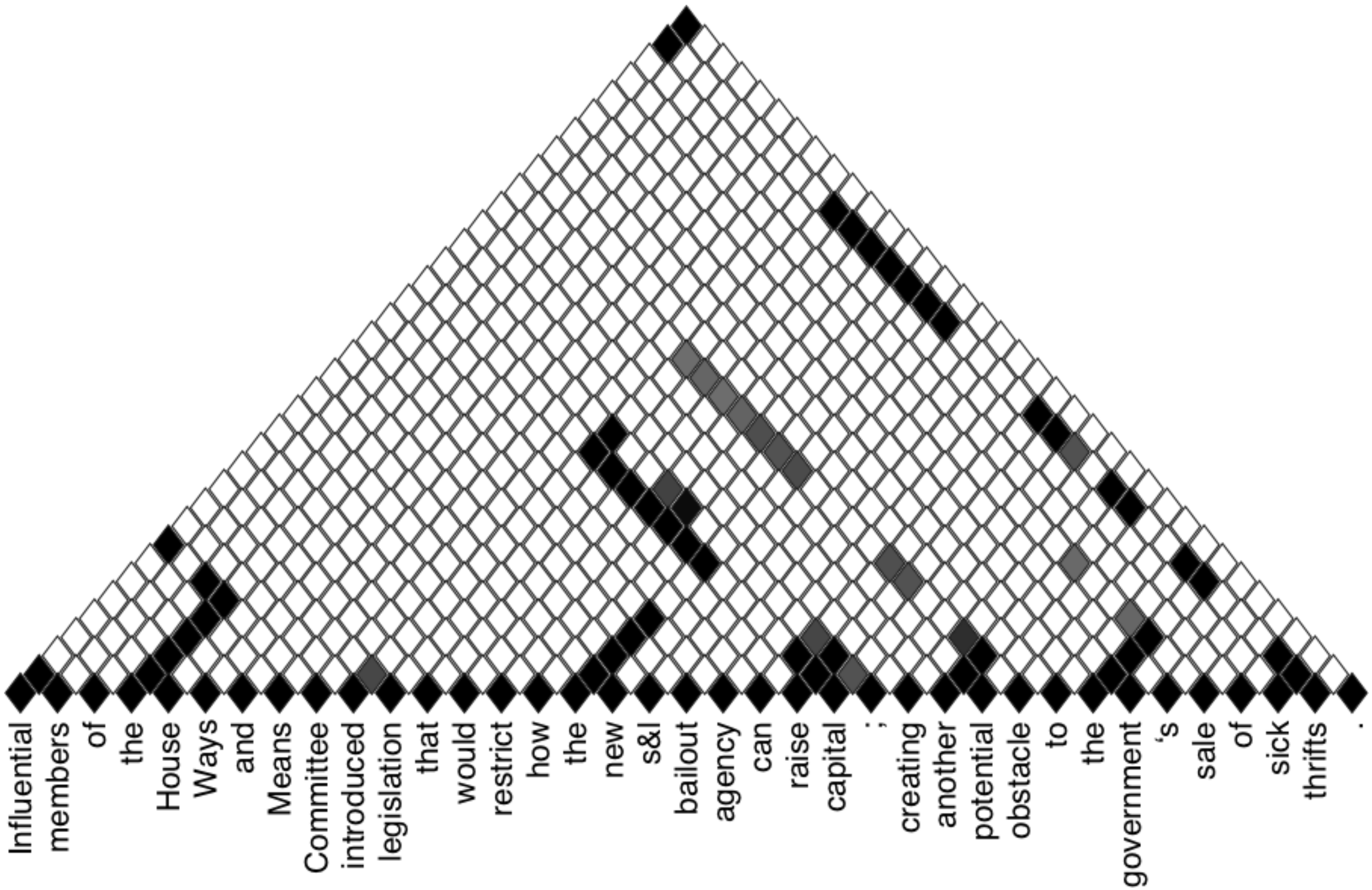
Bracket Posteriors



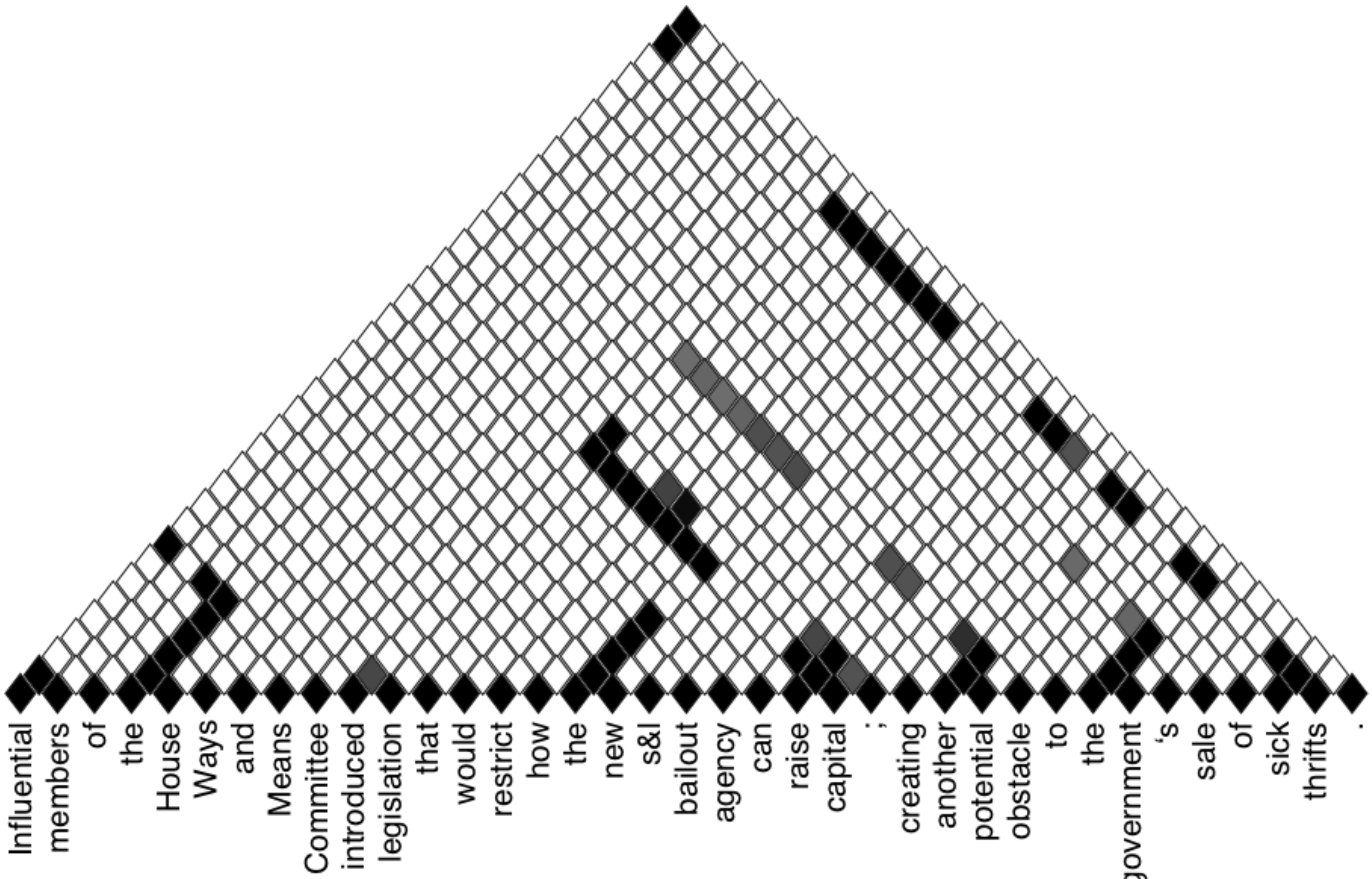
Bracket Posteriors



Bracket Posteriors



Bracket Posteriors



1621 min

111 min

35 min

15 min

(no search error)