Syntax and Parsing

Slav Petrov – Google

Thanks to:

Dan Klein, Ryan McDonald, Alexander Rush, Joakim Nivre

Lisbon Machine Learning School

Analyzing Natural Language

They solved the problem with statistics .

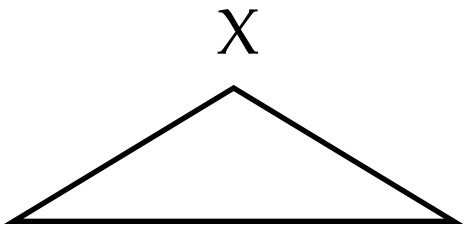
Analyzing Natural Language

They [solved the problem] with statistics .

Analyzing Natural Language

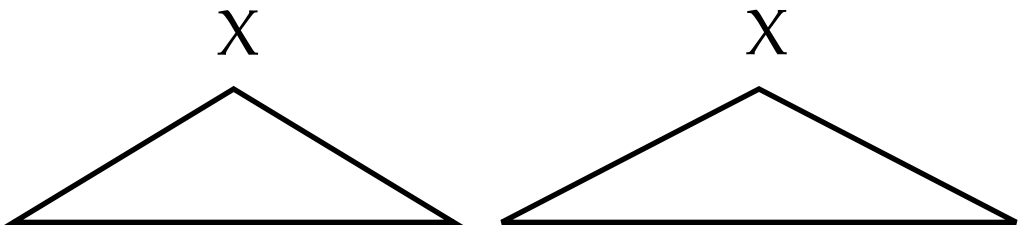
They solved [the problem with statistics] .

Analyzing Natural Language



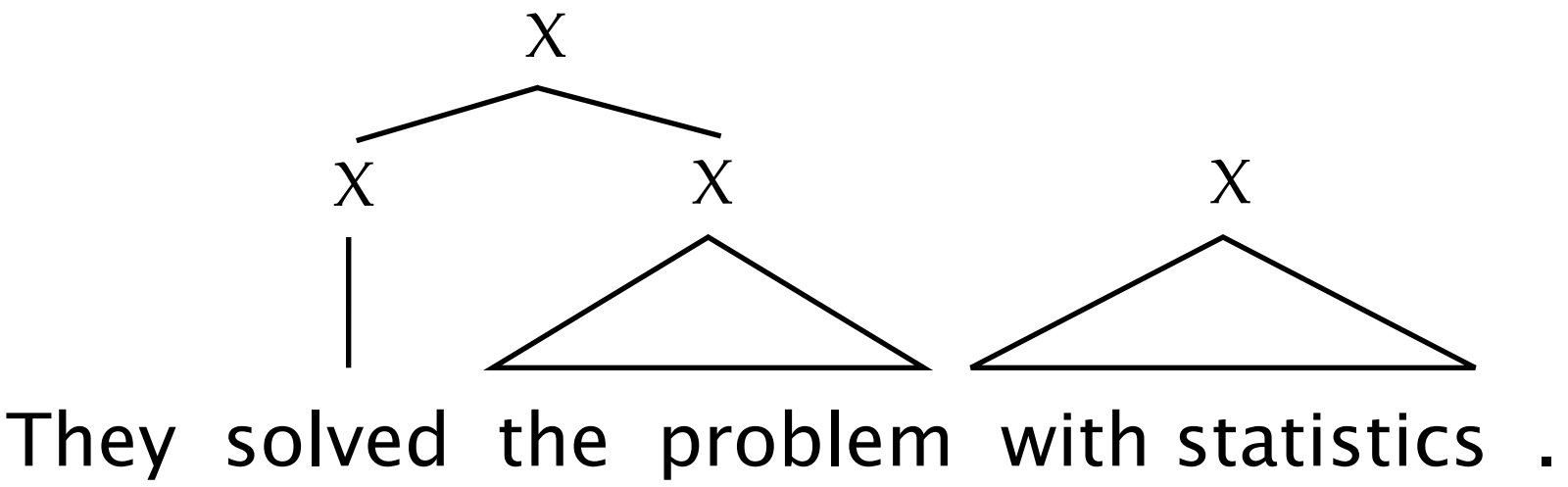
They solved the problem with statistics .

Analyzing Natural Language

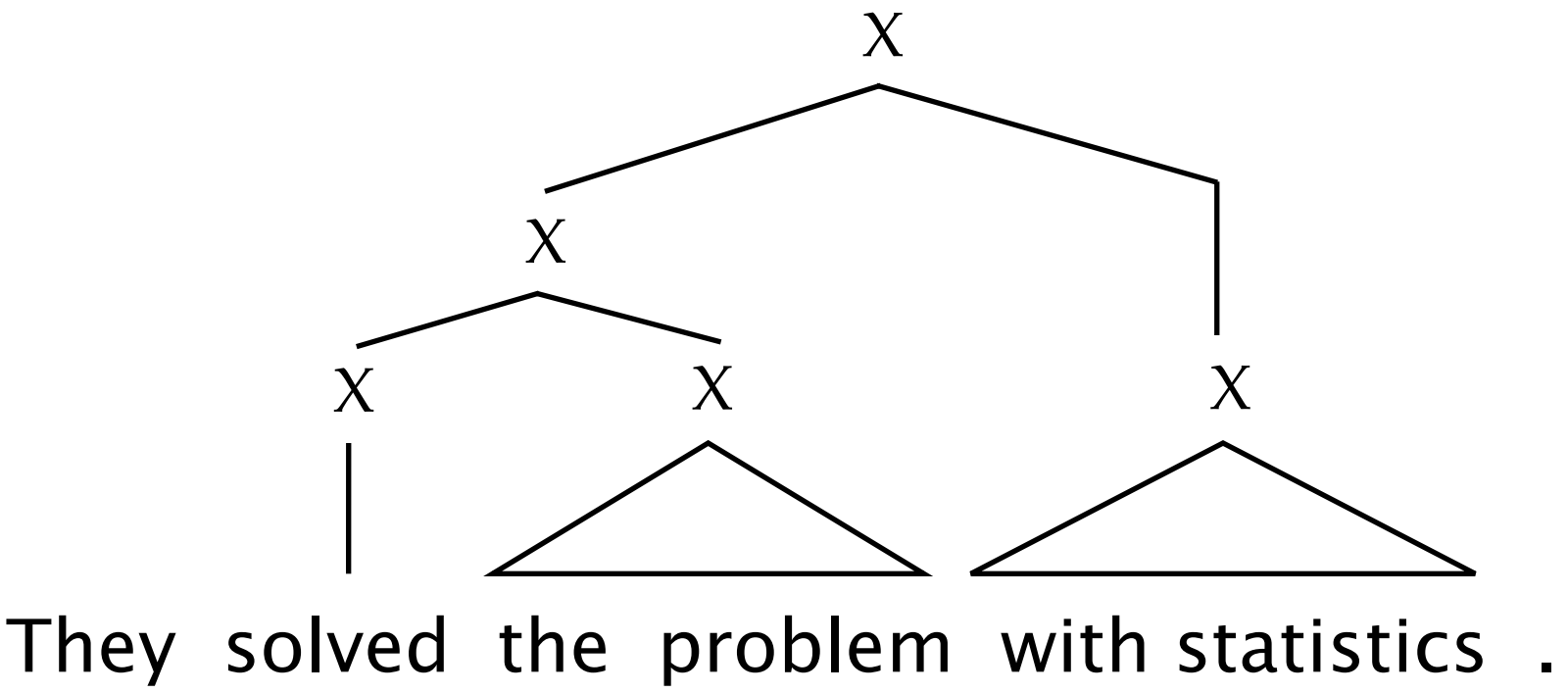


They solved the problem with statistics .

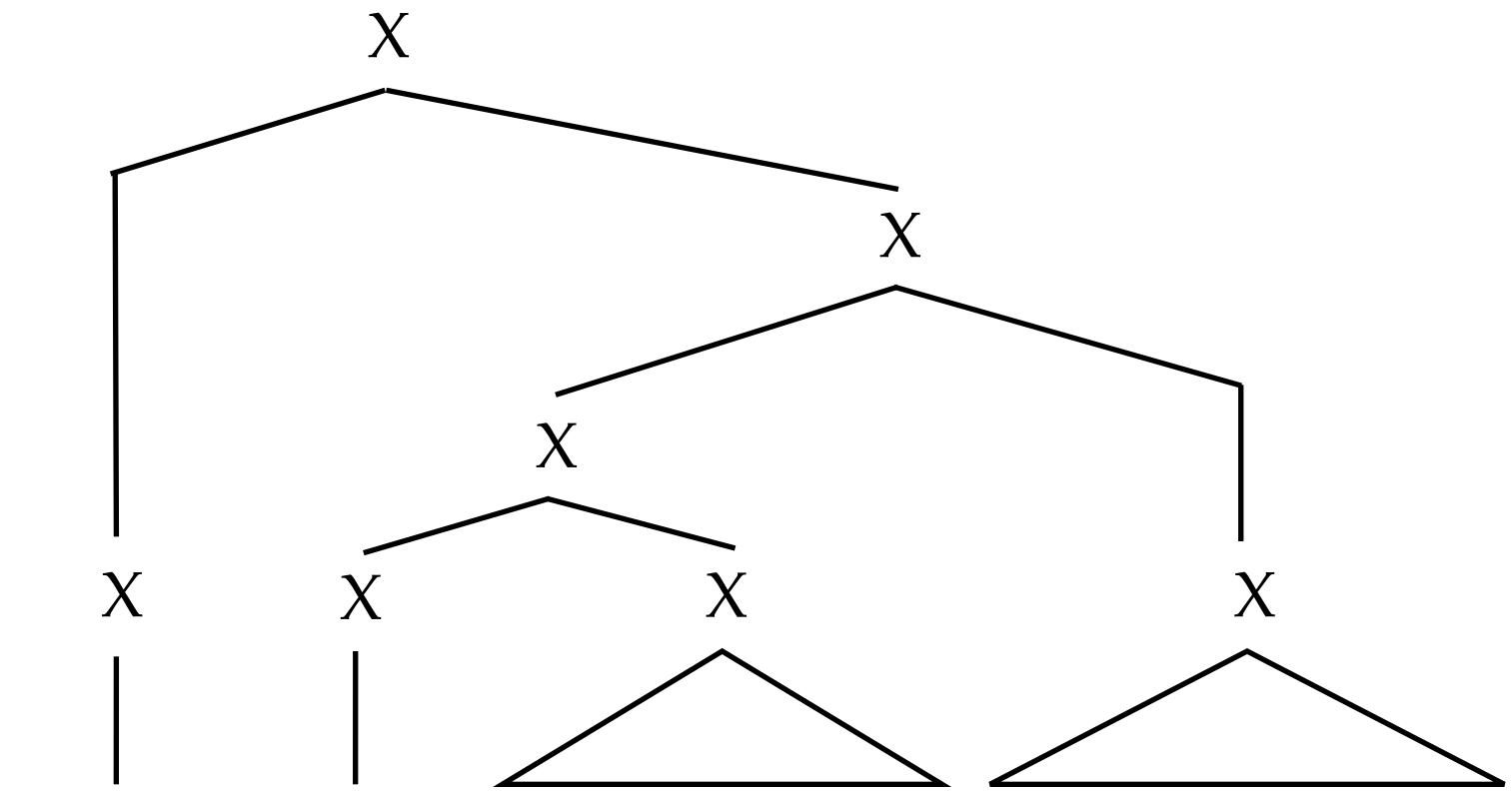
Analyzing Natural Language



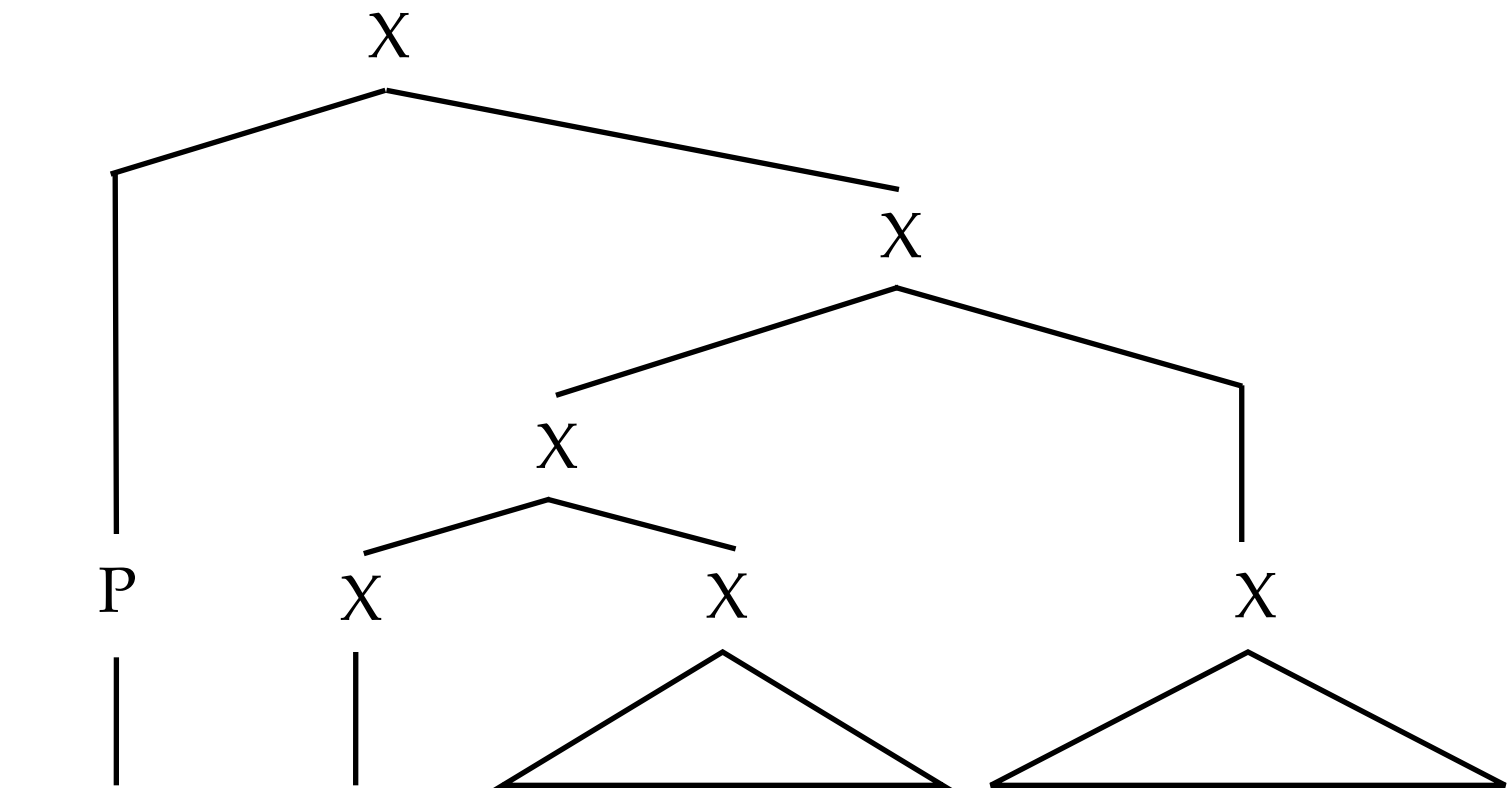
Analyzing Natural Language



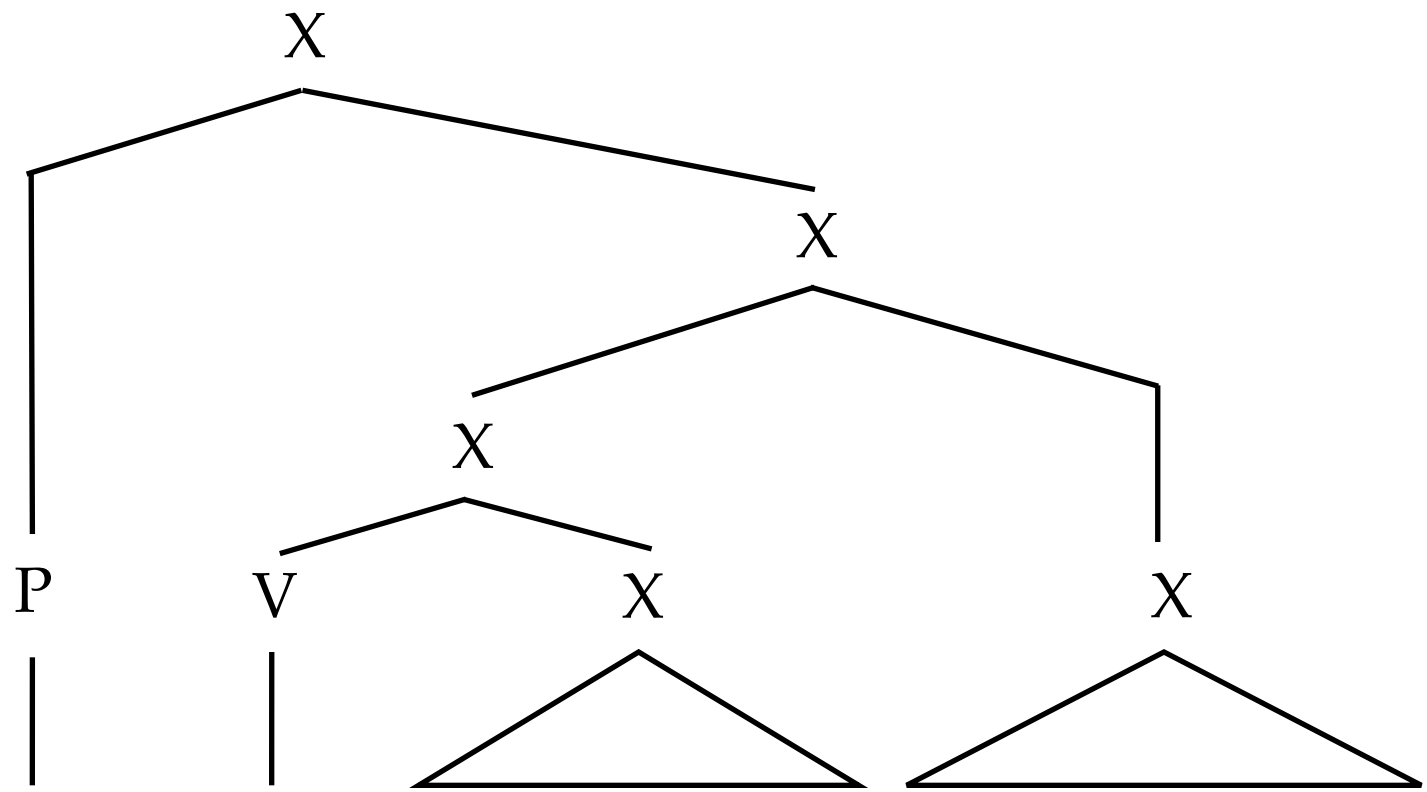
Analyzing Natural Language



Analyzing Natural Language

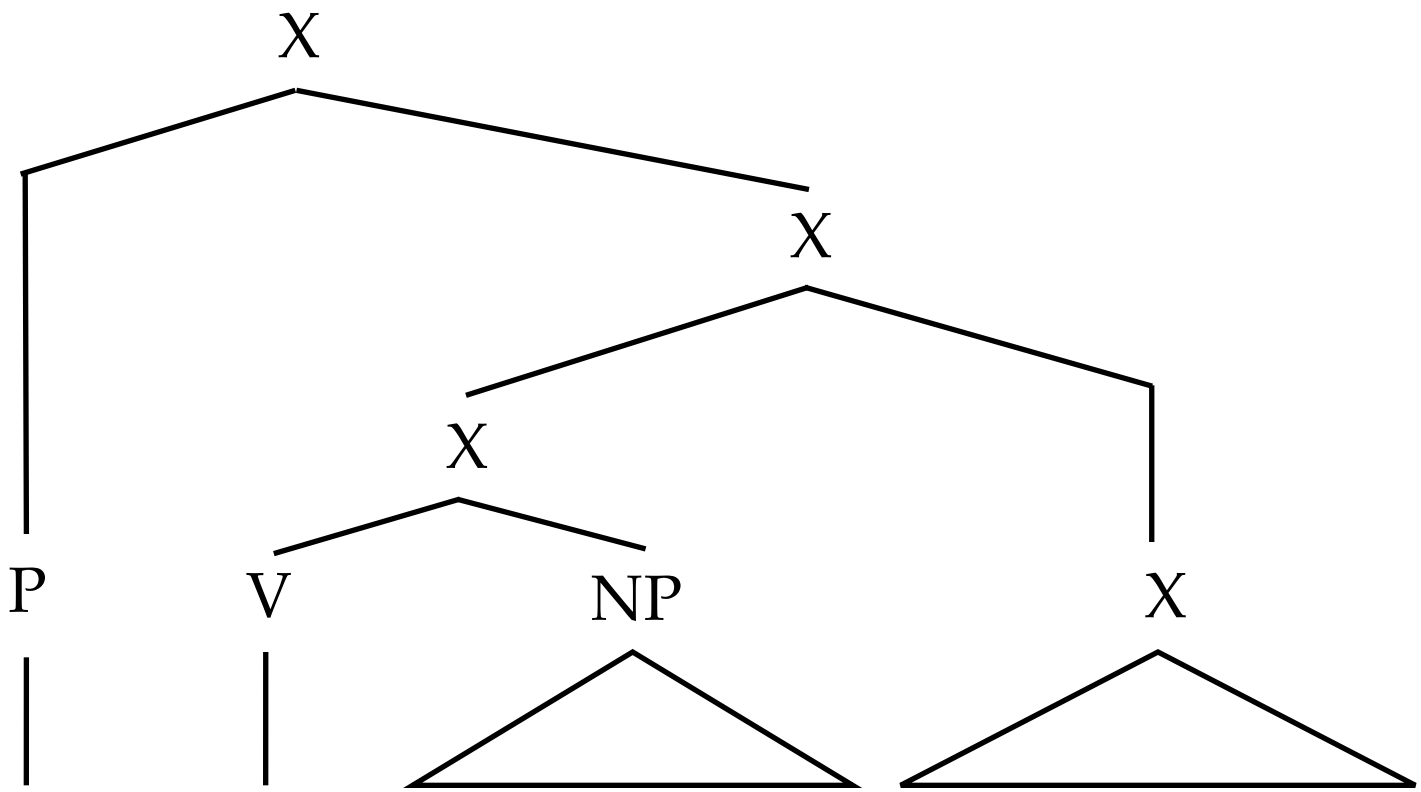


Analyzing Natural Language

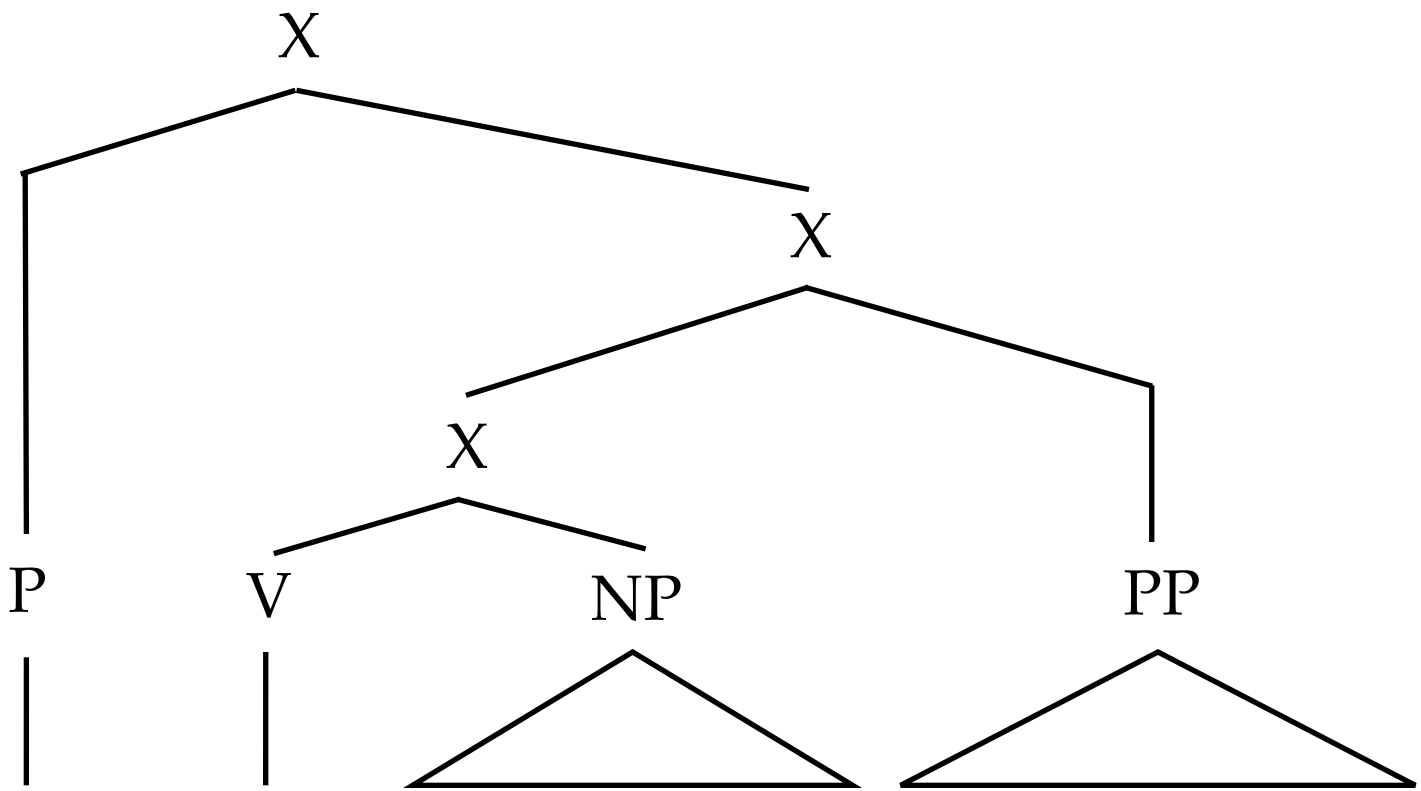


They solved the problem with statistics .

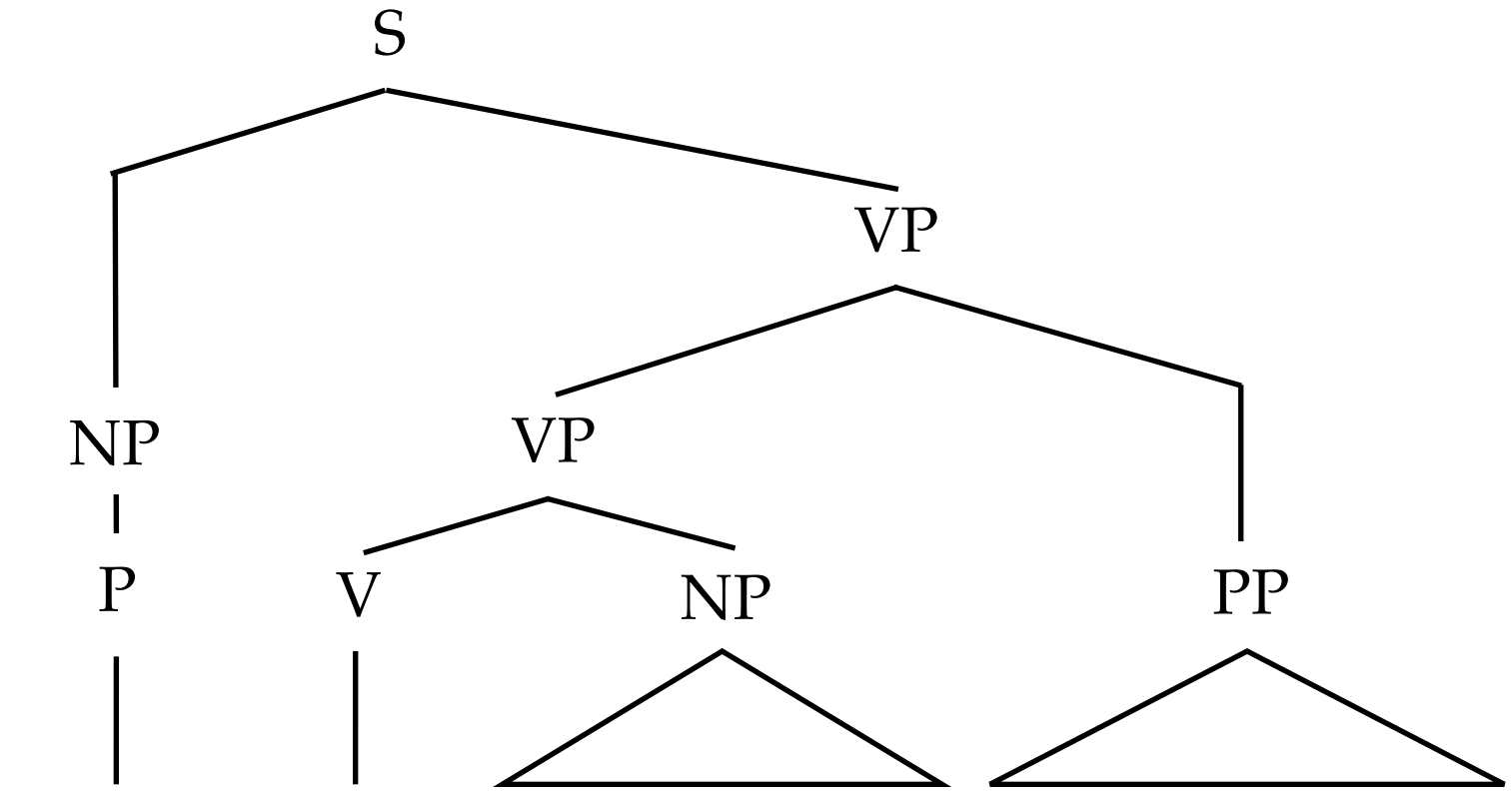
Analyzing Natural Language



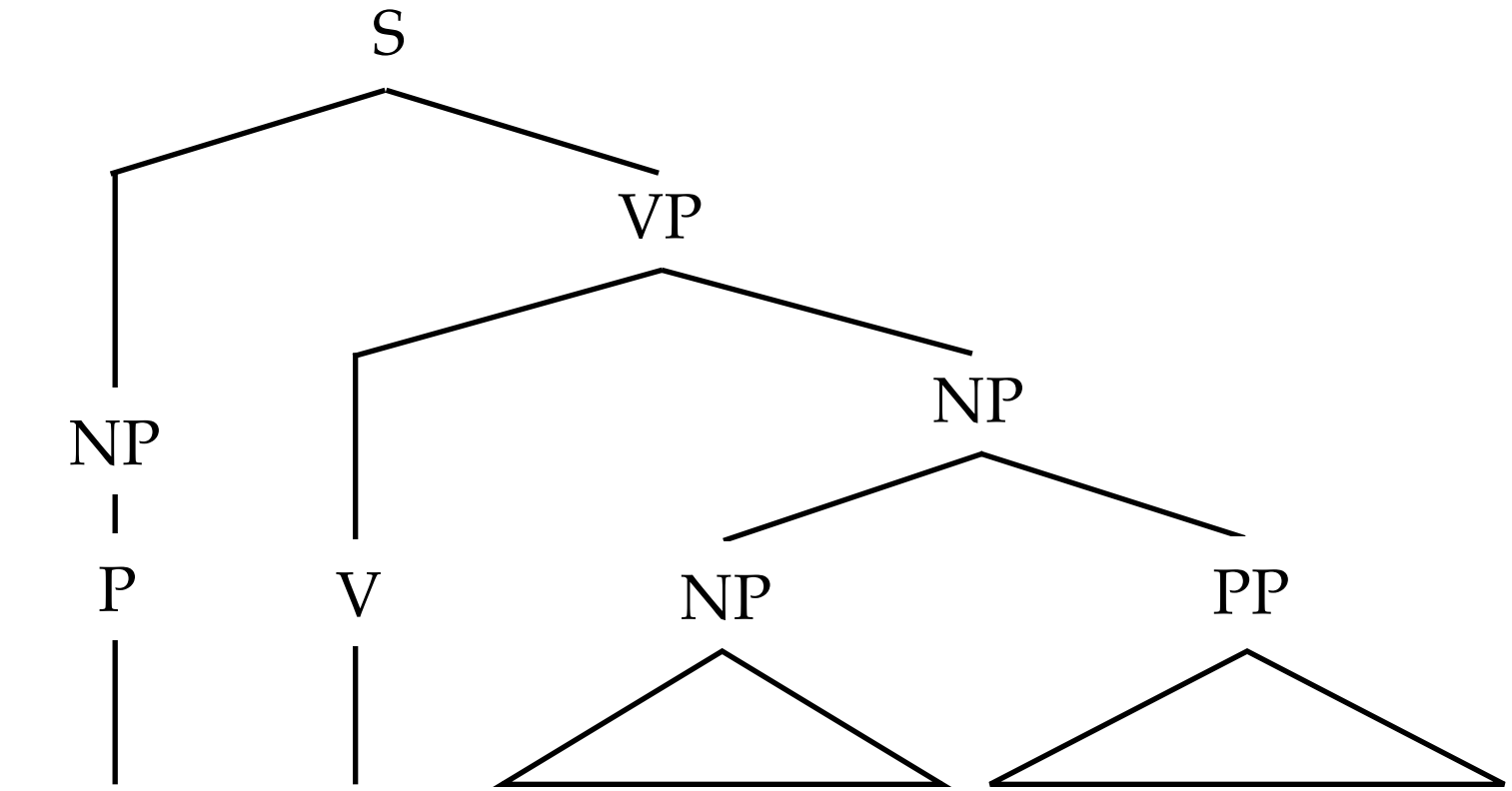
Analyzing Natural Language



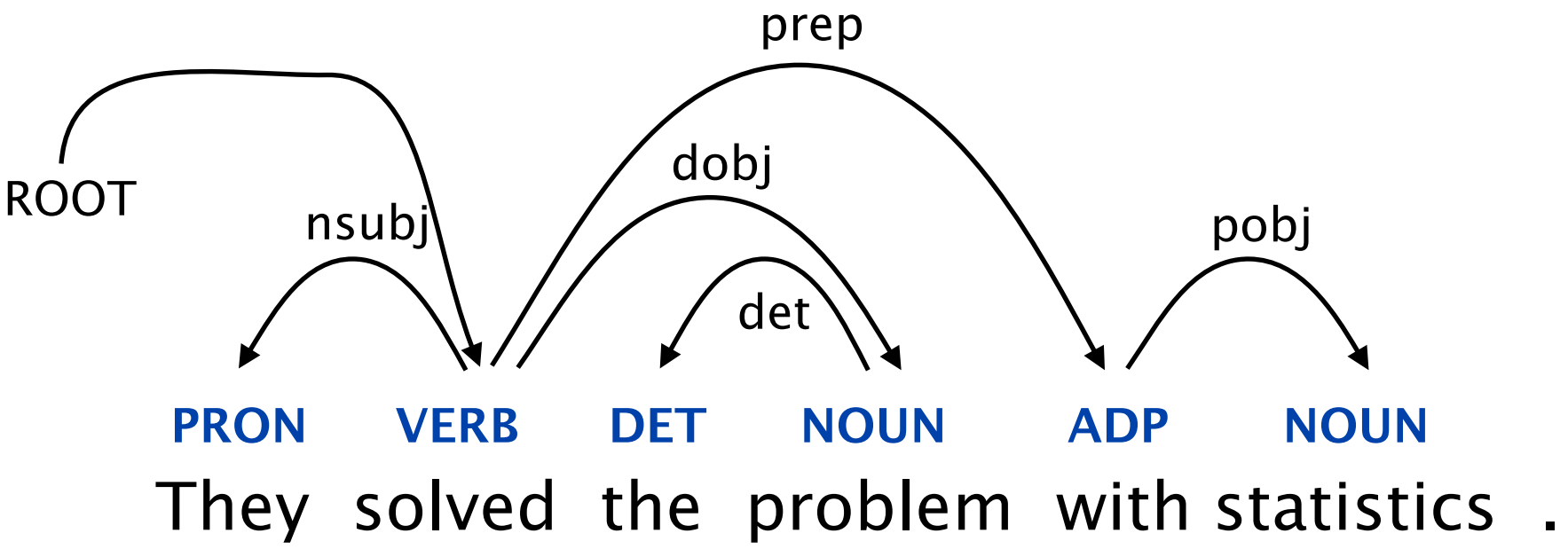
Analyzing Natural Language



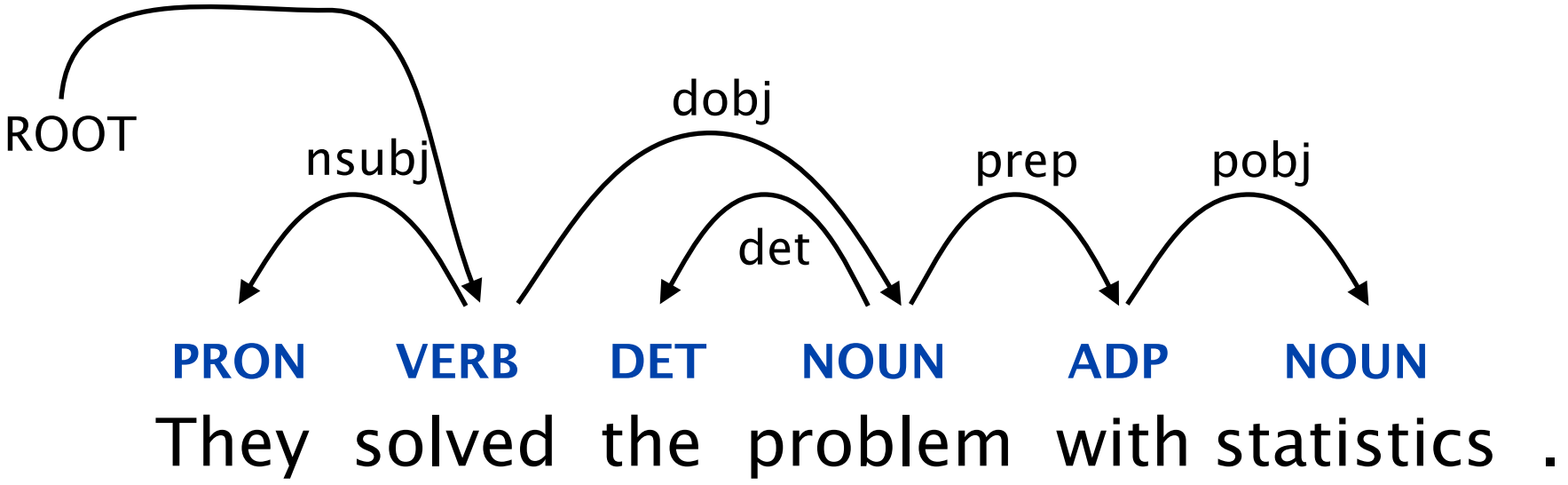
Syntax and Semantics



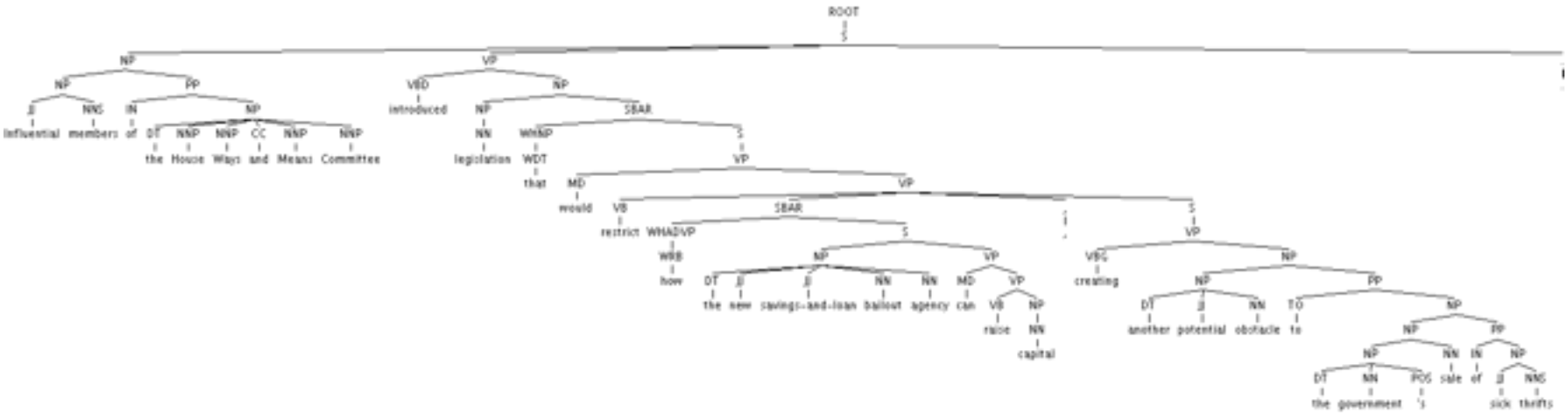
Constituency and Dependency



Constituency and Dependency



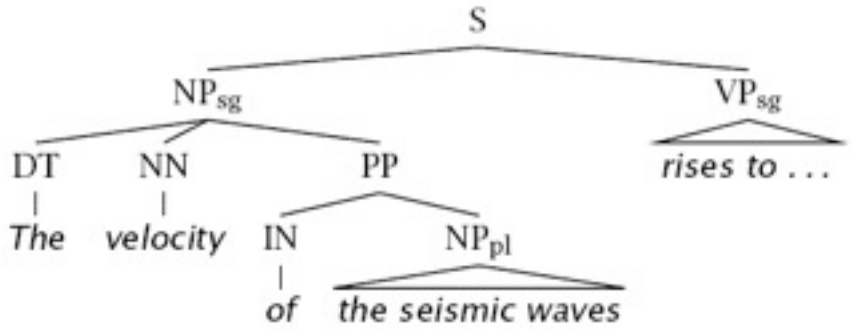
A “real” Sentence



Influential members of the House Ways and Means Committee introduced legislation that would restrict how the new savings-and-loan bailout agency can raise capital, creating another potential obstacle to the government's sale of sick thrifts.

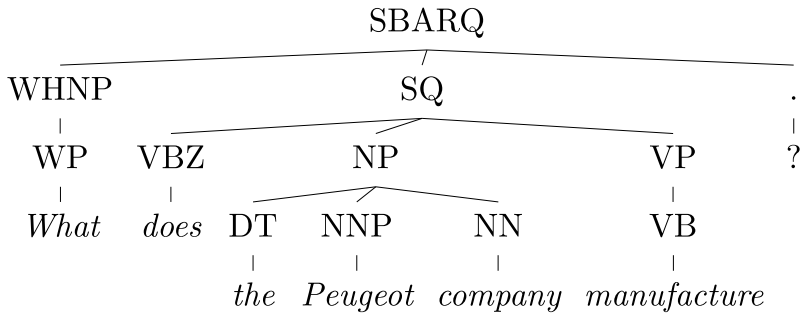
Phrase Structure Parsing

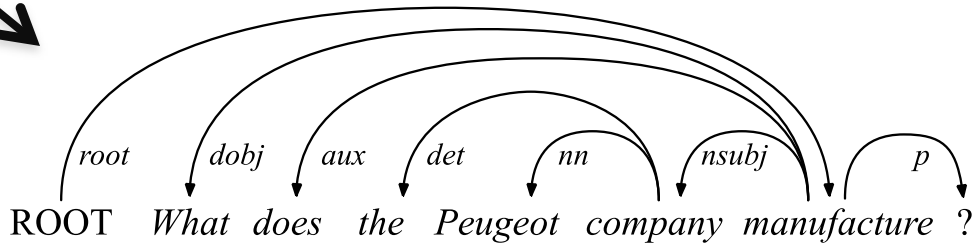
- Phrase structure parsing organizes syntax into constituents or brackets
- In general, this involves nested trees
- Linguists can, and do, argue about details
- Lots of ambiguity
- Not the only kind of syntax…
- First part of today's lecture



new art critics write reviews with computers

Dependency Parsing





- Directed edges between pairs of word (head, dependent)
- Can handle free word-order languages
- Very efficient decoding algorithms exist
- Second part of today's lecture

Classical NLP: Parsing

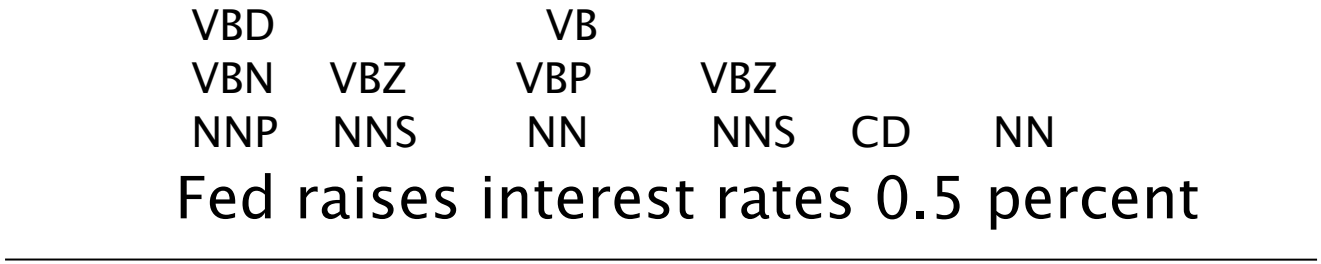
- Write symbolic or logical rules:

Fed raises interest rates 0.5 percent

- Use deduction systems to prove parses from words
  - Minimal grammar on "Fed raises" sentence: 36 parses
  - Real-size grammar: many millions of parses
- This scaled very badly, didn't yield broad-coverage tools

Classical NLP: Parsing

- Write symbolic or logical rules:



- Use deduction systems to prove parses from words
  - Minimal grammar on "Fed raises" sentence: 36 parses
  - Real-size grammar: many millions of parses
- This scaled very badly, didn't yield broad-coverage tools

Classical NLP: Parsing

- Write symbolic or logical rules:

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

Fed raises interest rates 0.5 percent

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

- Use deduction systems to prove parses from words
  - Minimal grammar on "Fed raises" sentence: 36 parses
  - Real-size grammar: many millions of parses
- This scaled very badly, didn't yield broad-coverage tools

Attachments

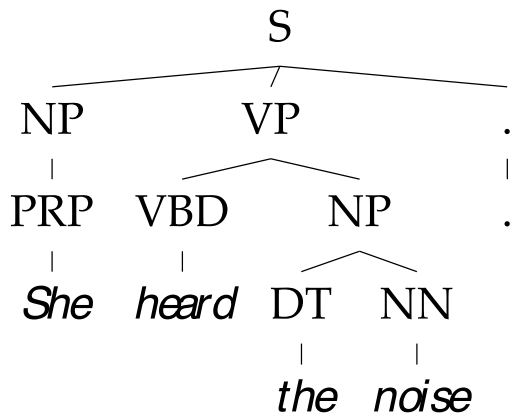
- I cleaned the dishes from dinner
- I cleaned the dishes with detergent
- I cleaned the dishes in my pajamas
- I cleaned the dishes in the sink

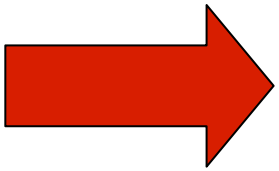
Probabilistic Context-Free Grammars

- A context-free grammar is a tuple <N, T, S, R>
  - N : the set of non-terminals
    - Phrasal categories: S, NP, VP, ADJP, etc.
    - Parts-of-speech (pre-terminals): NN, JJ, DT, VB
  - T : the set of terminals (the words)
  - S : the start symbol
    - Often written as ROOT or TOP
    - Not usually the sentence non-terminal S
  - R : the set of rules
    - Of the form $X \rightarrow Y1\ Y2\ \ldots\ Yk$, with $X, Yi \in N$
    - 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(Y1 Y2 … Yk | X)

Treebank Grammars

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



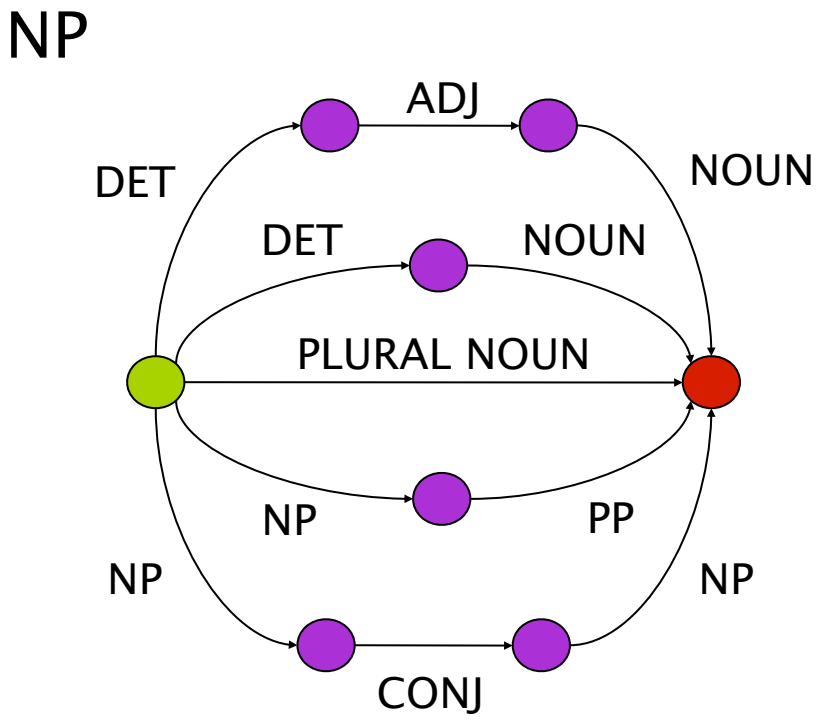


S → NP VP .	1
NP → PRP	0.5
NP → DT NN	0.5
VP → VBD NP	1
PRP → She	1
…	

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

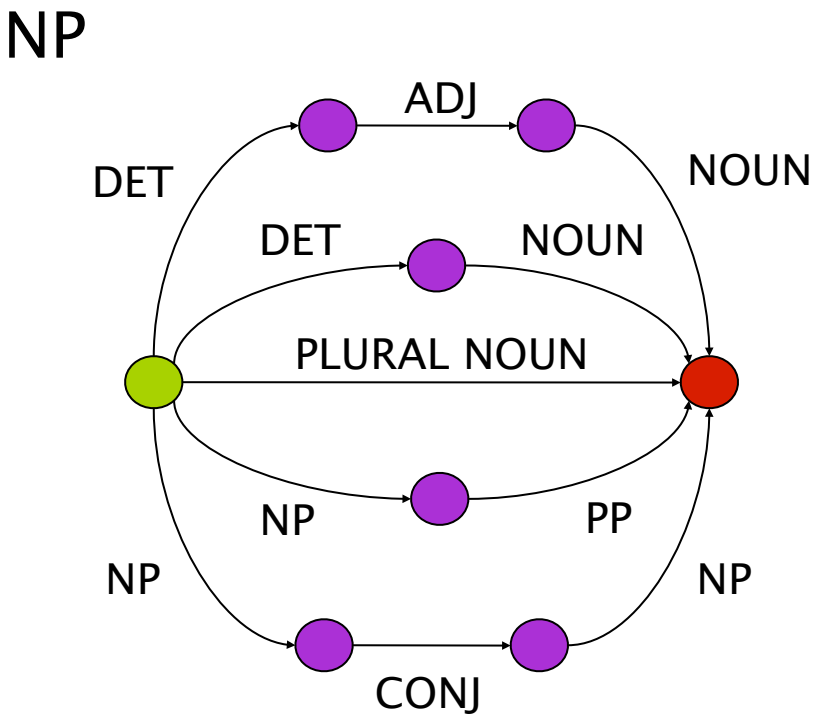
Treebank Grammar Scale

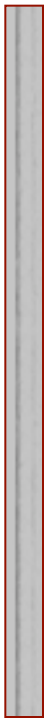
- Treebank grammars can be enormous
  - As FSAs, the raw grammar has ~10K states, excluding the lexicon
  - Better parsers usually make the grammars larger, not smaller



Treebank Grammar Scale

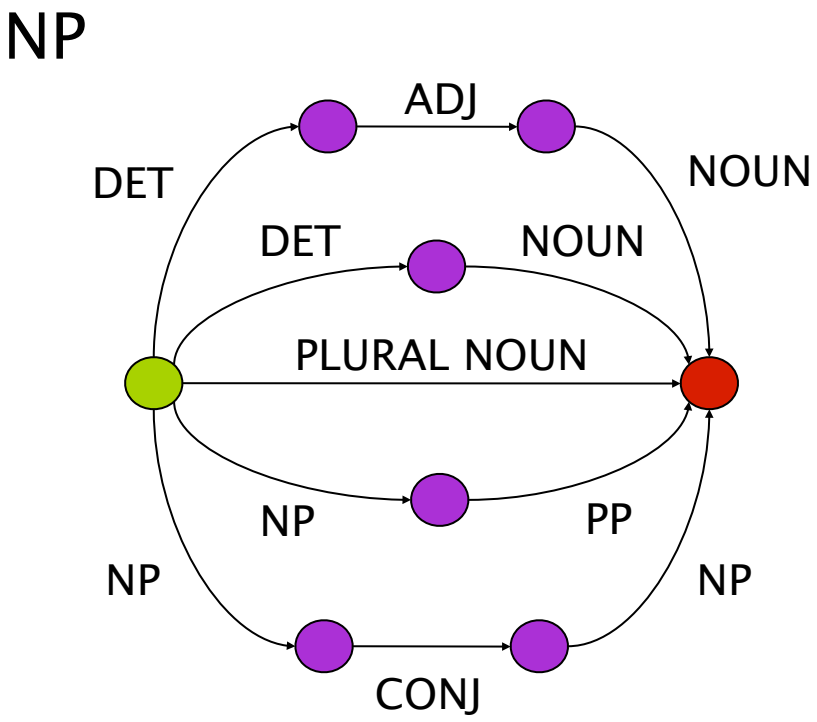
- Treebank grammars can be enormous
  - As FSAs, the raw grammar has ~10K states, excluding the lexicon
  - Better parsers usually make the grammars larger, not smaller

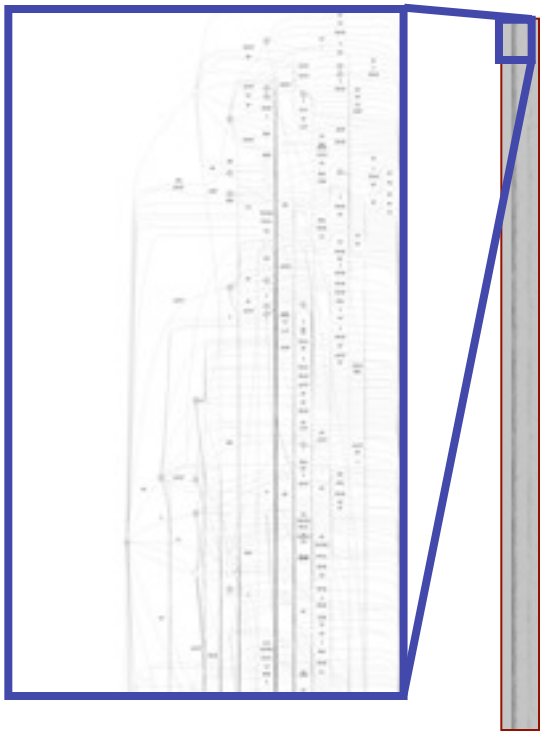




Treebank Grammar Scale

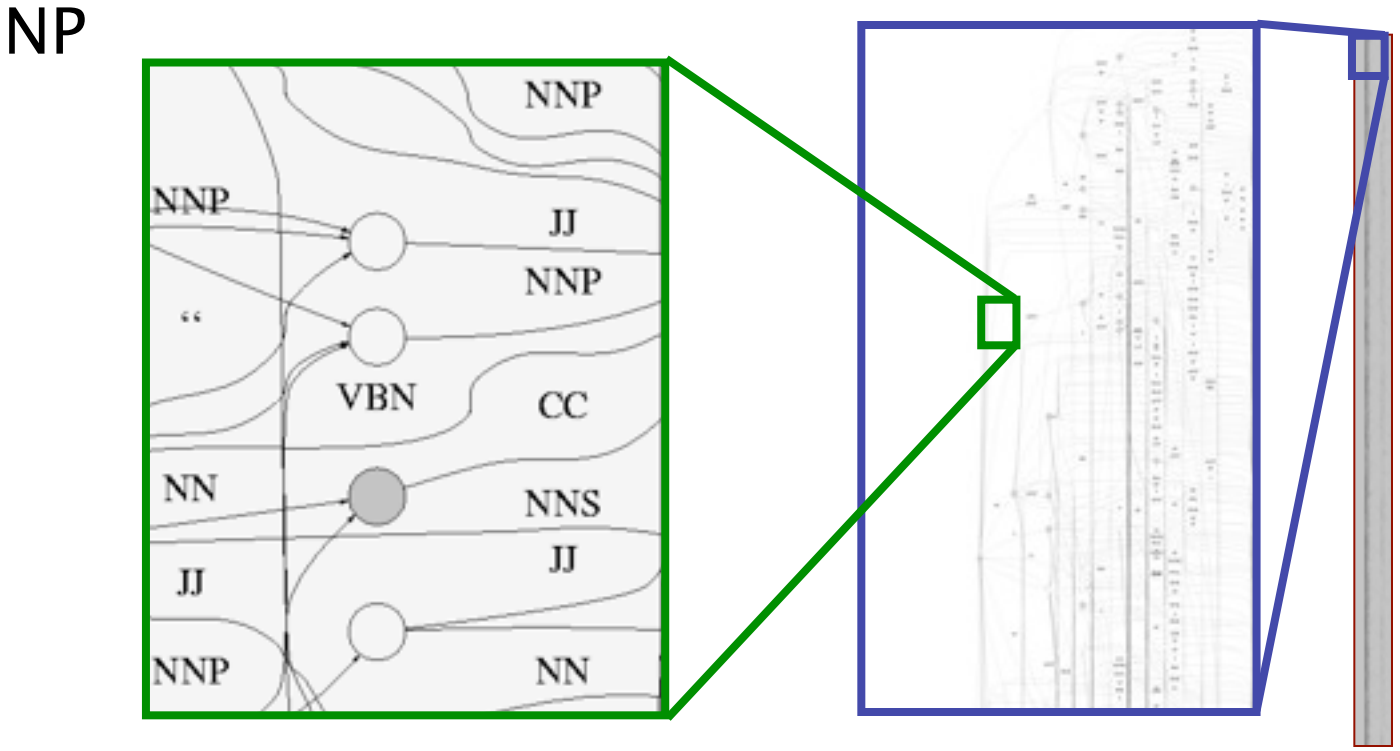
- Treebank grammars can be enormous
  - As FSAs, the raw grammar has ~10K states, excluding the lexicon
  - Better parsers usually make the grammars larger, not smaller





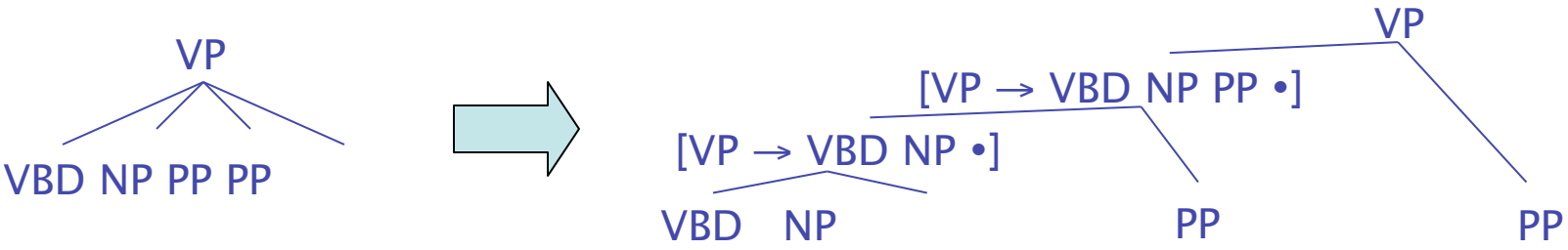
Treebank Grammar Scale

- 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



Chomsky Normal Form

- Chomsky normal form:
  - All rules of the form X → Y Z or X → w
  - In principle, this is no limitation on the space of (P)CFGs
    - N-ary rules introduce new non-terminals



    - Unaries / empties are "promoted"
  - 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!

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) *
                  bestScore(Z,k,j)
```

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

A Memoized Parser

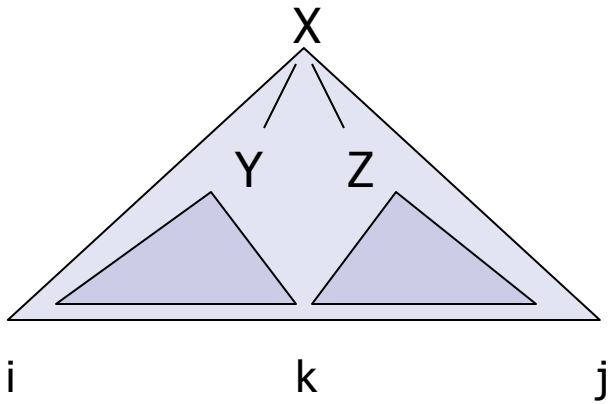
- One small change:

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

A Bottom-Up Parser (CKY)

- Can also organize things bottom-up

```
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]
```



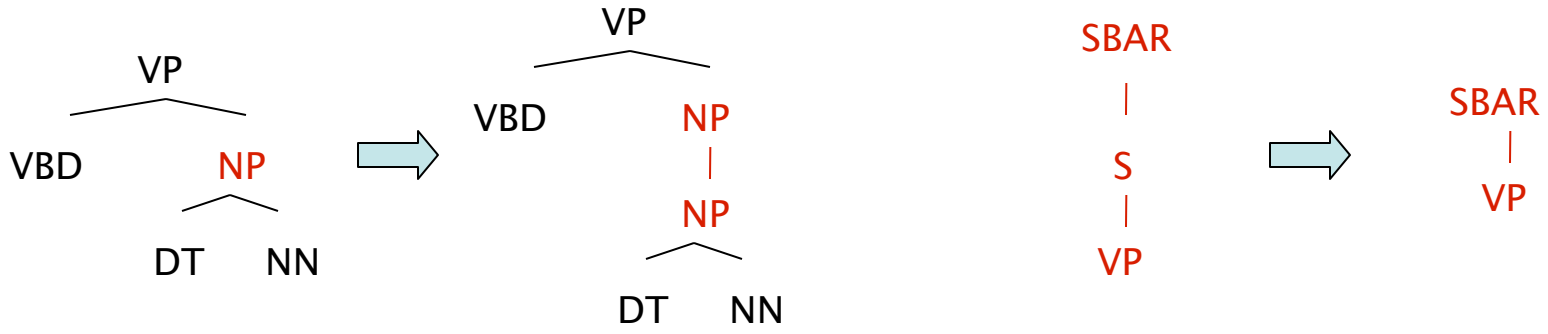
Unary Rules

- 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) *
       bestScore(Z,k,j)
              max score(X->Y) *
                   bestScore(Y,i,j)
```

CNF + Unary Closure

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

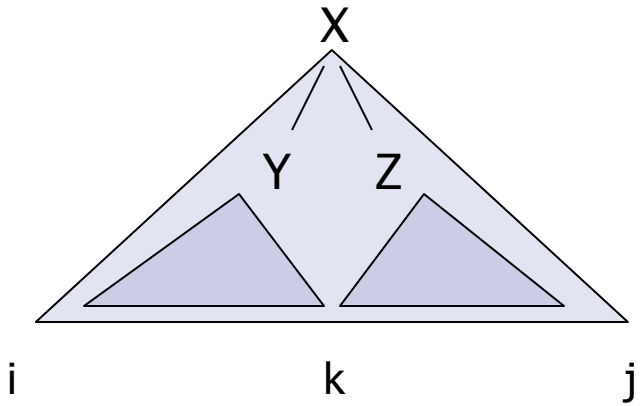


  - Alternate unary and binary layers
  - Reconstruct unary chains afterwards

Time: Theory

- How much time will it take to parse?

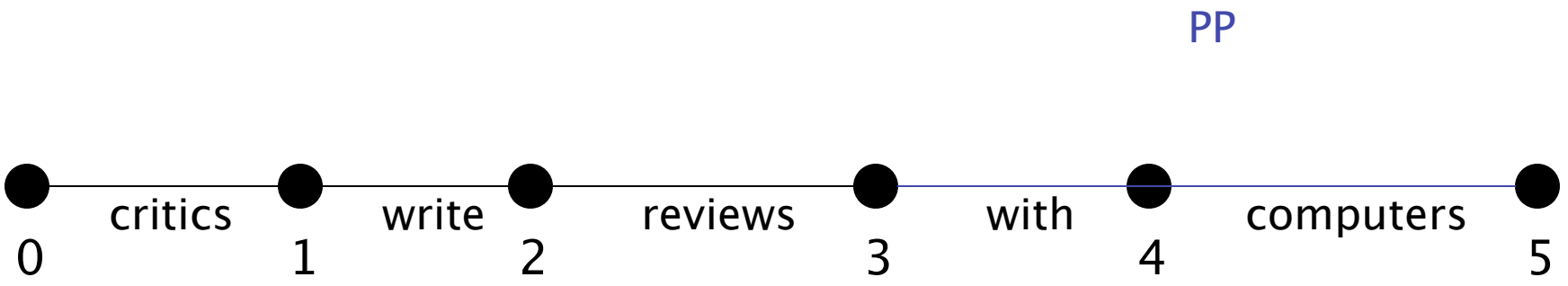
  - For each diff (<= n)
    - For each i (<= n)
      - For each rule X → Y Z
        - For each split point k
          Do constant work



  - Total time: |rules|*n3
  - Something like 5 sec for an unoptimized parse of a 20-word sentences, or 0.2sec for an optimized parser

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)



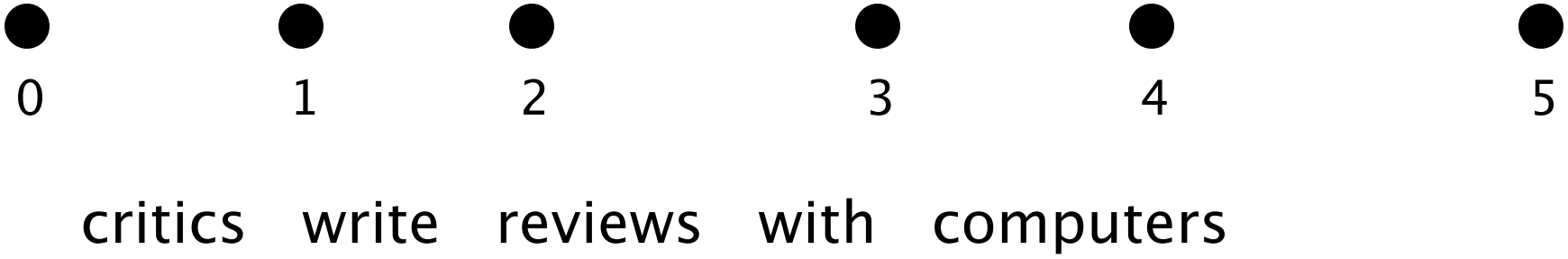
Word Items

- 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]



Item Successors

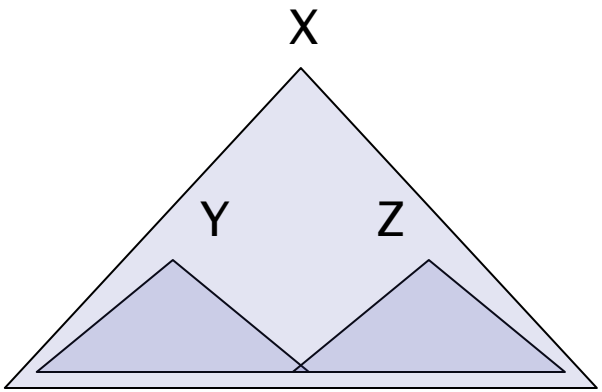
- When we pop items off of the agenda:
  - Graph successors: unary projections (NNS → critics, NP → NNS)

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

  - Hypergraph successors: combine with items already in our chart

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

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



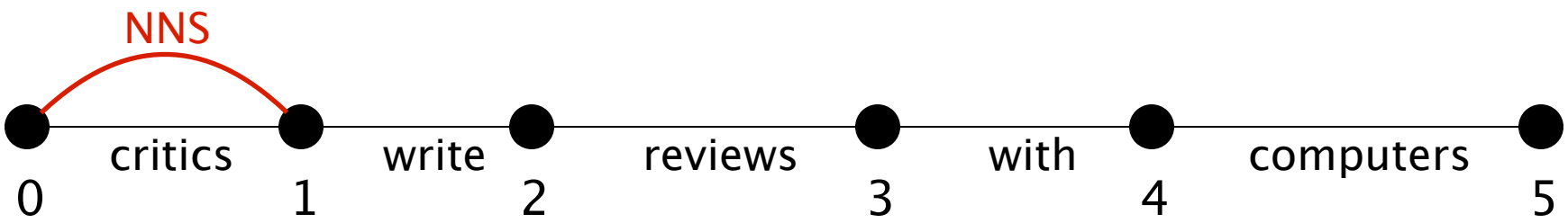
- Queries a chart must support:
  - Is edge X:[i,j] in the chart? (What score?)
  - What edges with label Y end at position j?
  - What edges with label Z start at position i?

An Example

NNS[0,1] VBP[1,2] NNS[2,3]IN[3,4]NNS[3,4]

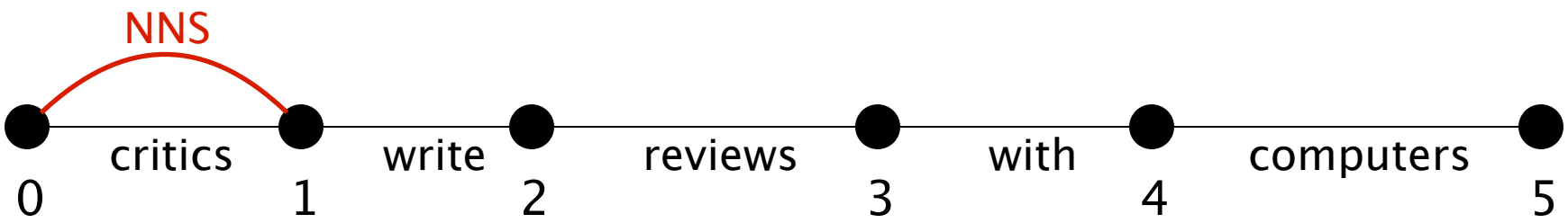
An Example

VBP[1,2] NNS[2,3]IN[3,4]NNS[3,4]



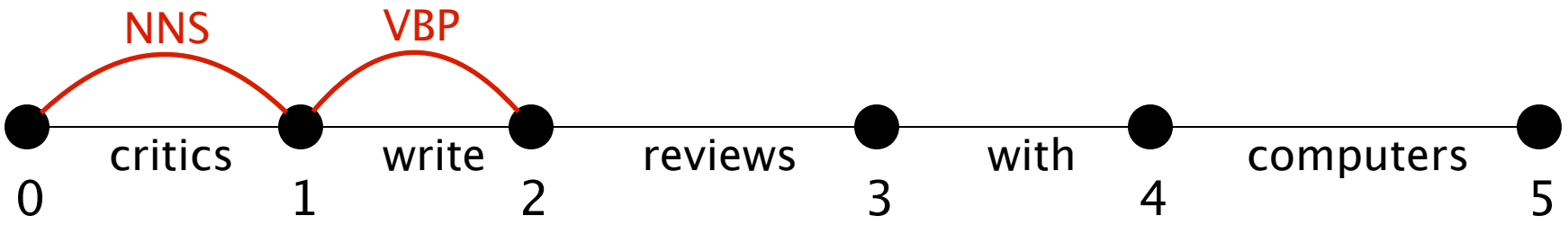
An Example

VBP[1,2] NNS[2,3]IN[3,4]NNS[3,4]NP[0,1]



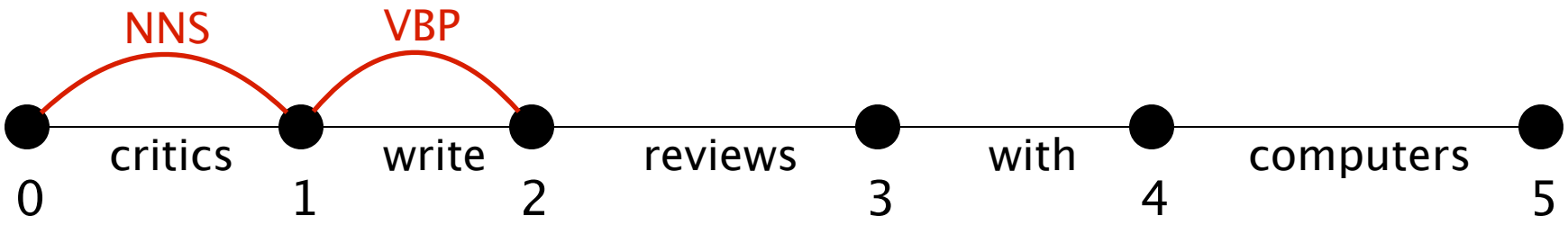
An Example

NNS[2,3] IN[3,4] NNS[3,4] NP[0,1]



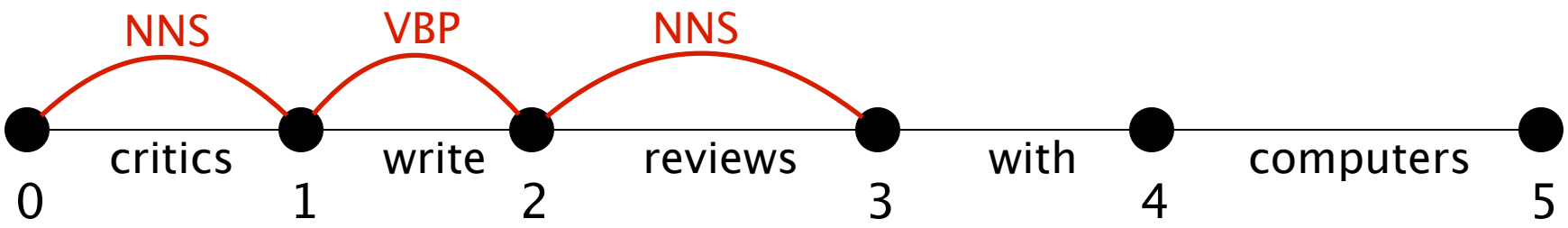
An Example

NNS[2,3] IN[3,4] NNS[3,4] NP[0,1] VP[1,2]



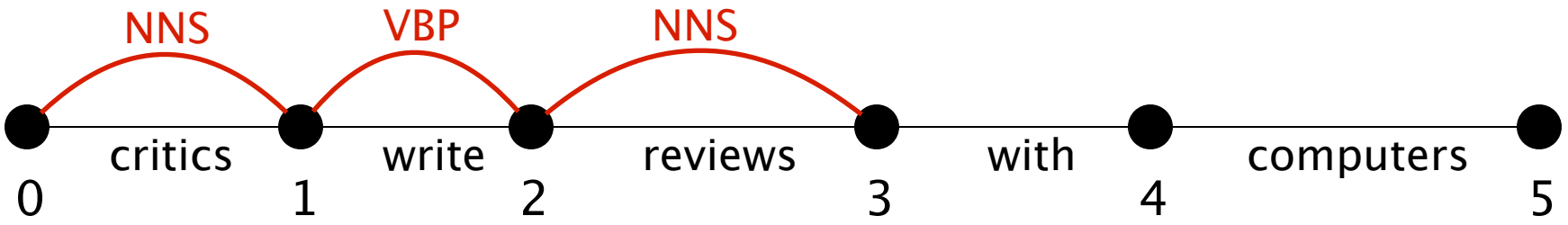
An Example

IN[3,4] NNS[3,4] NP[0,1] VP[1,2]



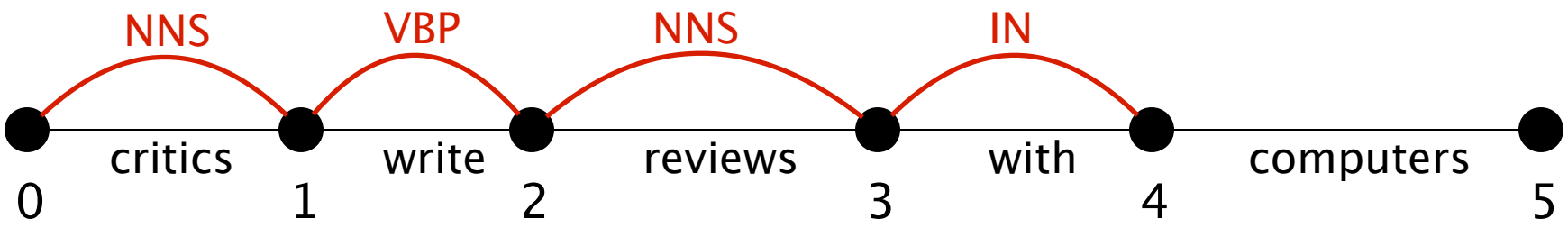
An Example

IN[3,4] NNS[3,4] NP[0,1] VP[1,2] NP[2,3]



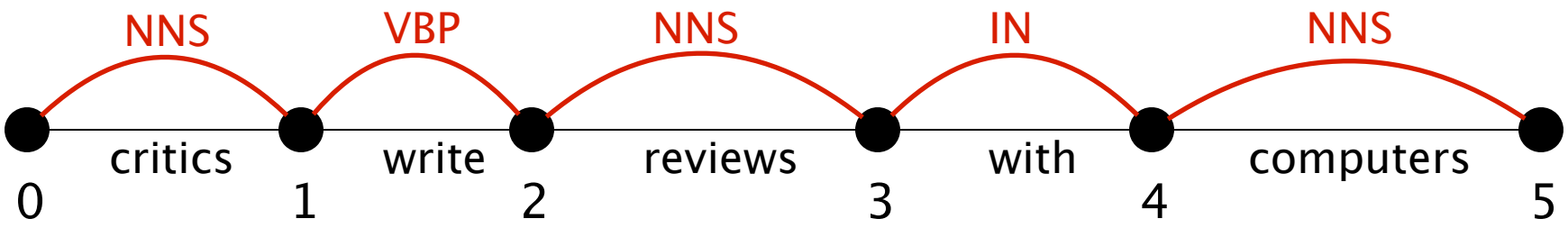
An Example

NNS[3,4] NP[0,1] VP[1,2] NP[2,3]



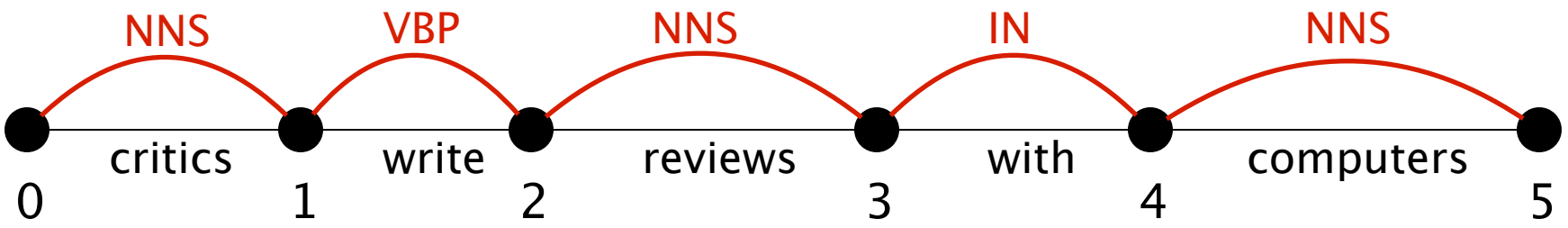
An Example

NP[0,1] VP[1,2] NP[2,3]



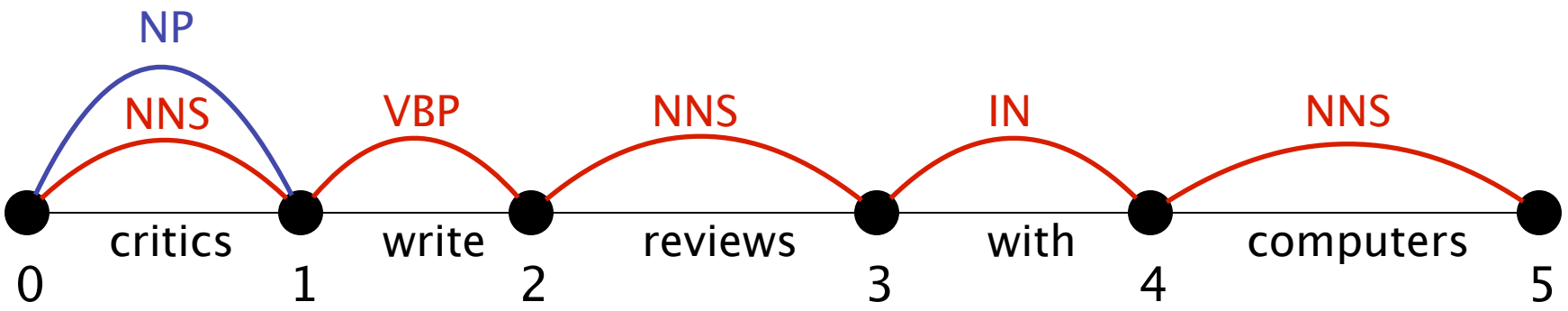
An Example

NP[0,1] VP[1,2] NP[2,3] NP[4,5]



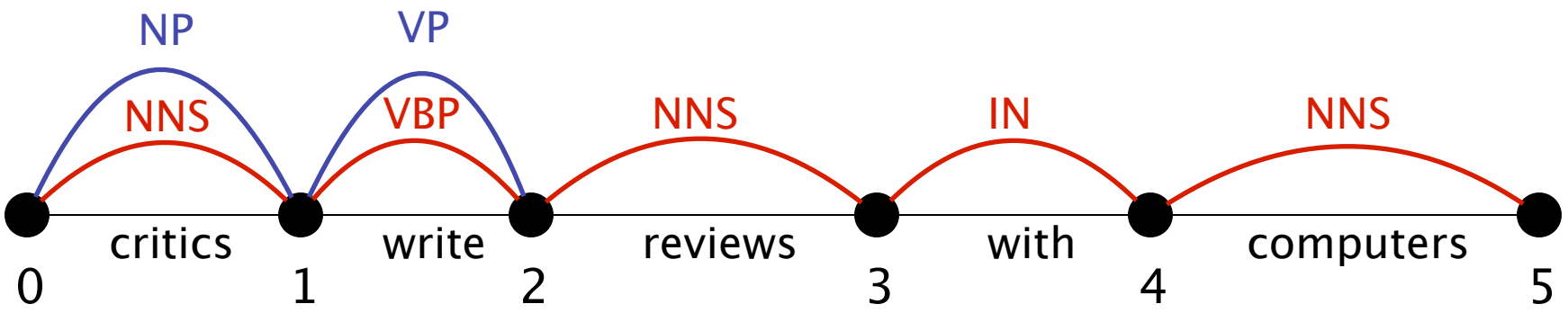
An Example

VP[1,2] NP[2,3] NP[4,5]



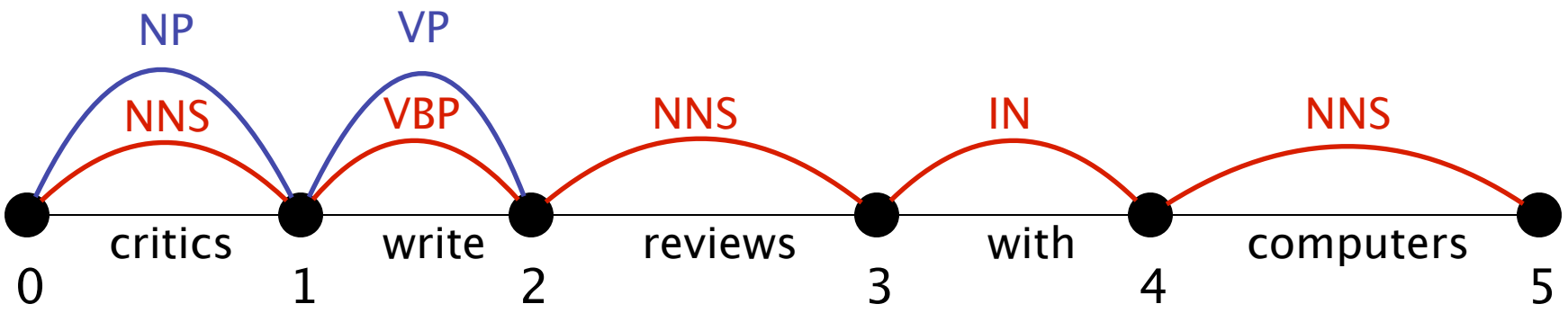
An Example

NP[2,3] NP[4,5]



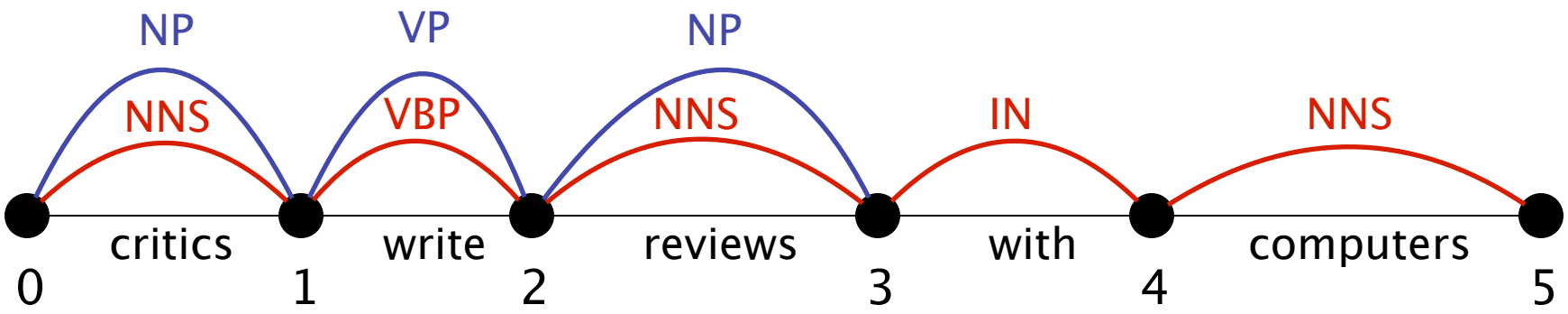
An Example

NP[2,3] NP[4,5] S[0,2]



An Example

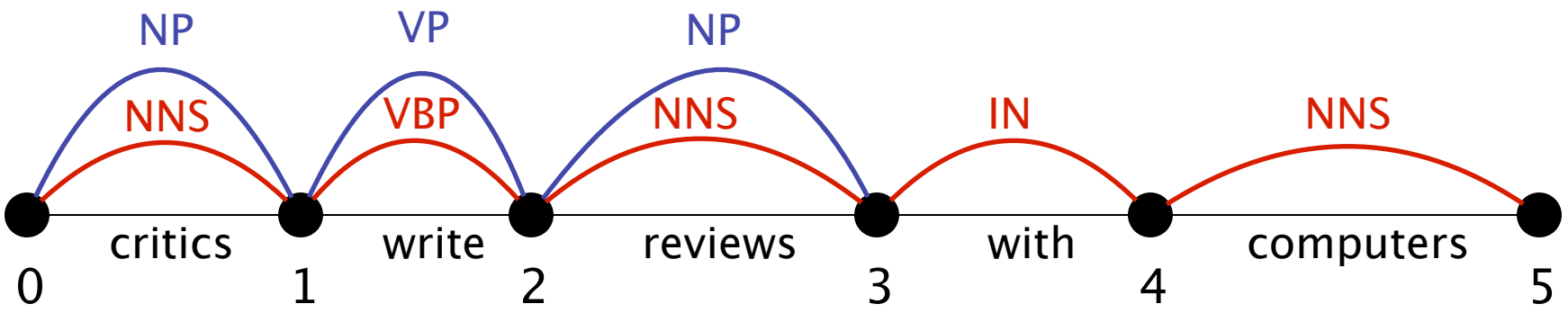
NP[4,5] S[0,2]



An Example

NP[4,5] S[0,2]

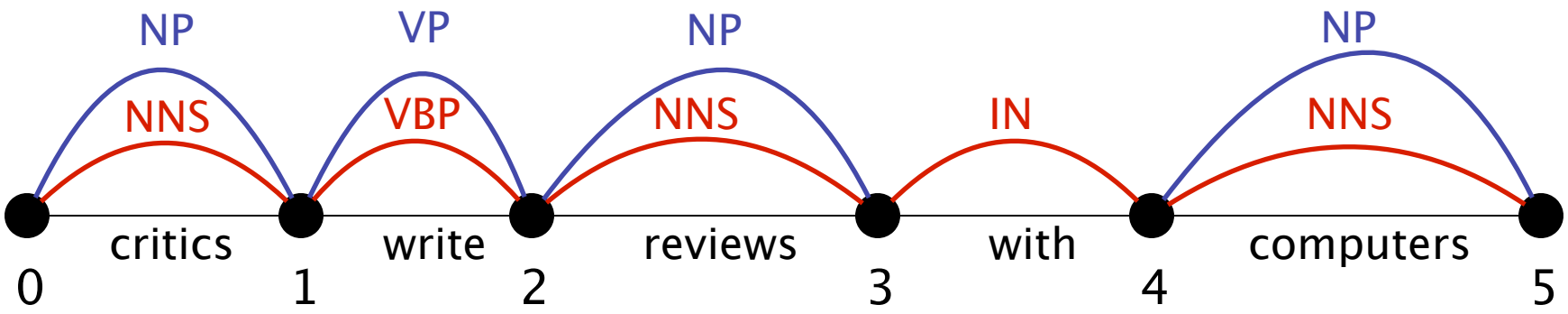
VP[1,3]



An Example

S[0,2]

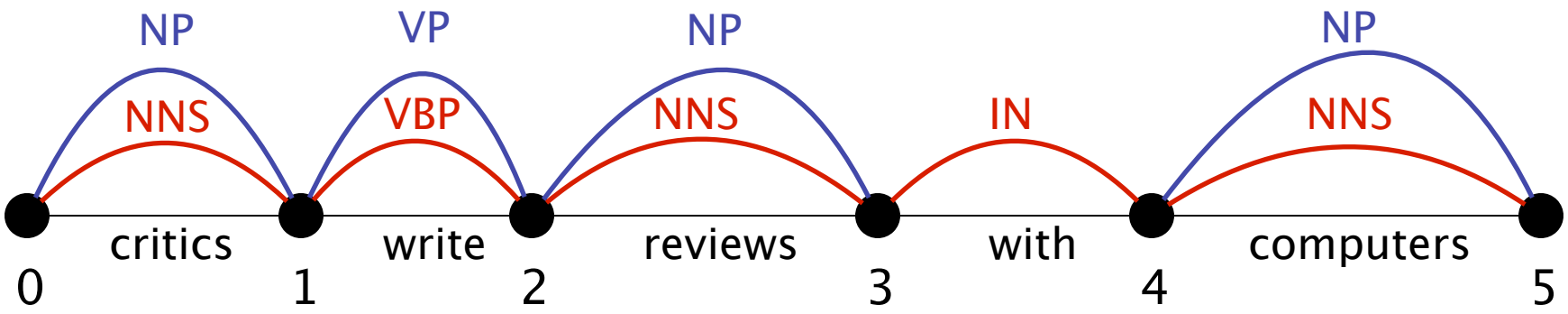
VP[1,3]



An Example

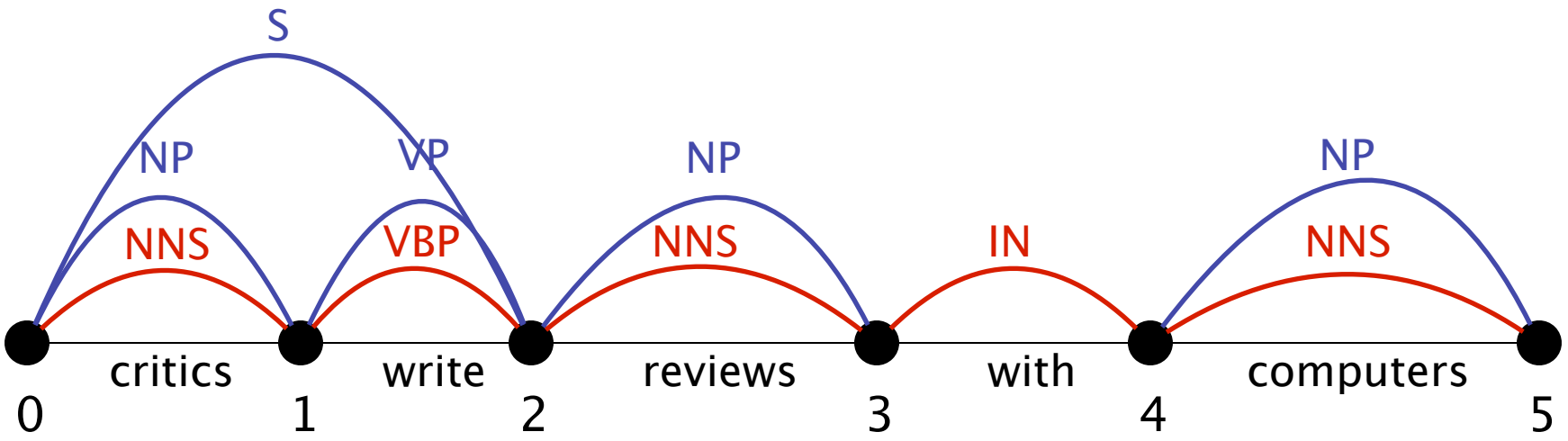
S[0,2]

VP[1,3] PP[3,5]



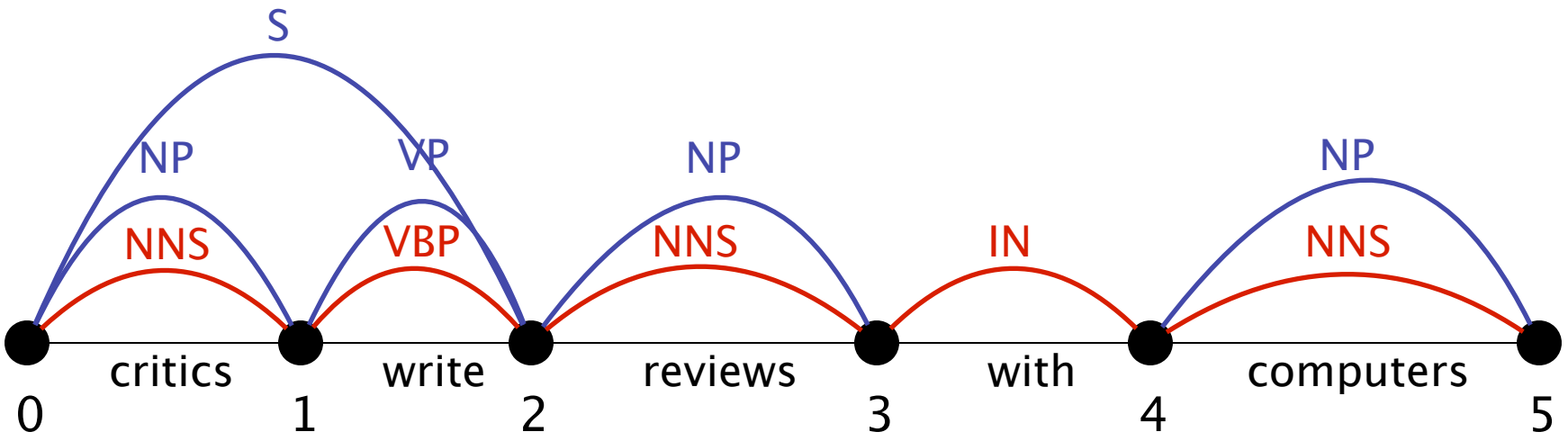
An Example

VP[1,3] PP[3,5]



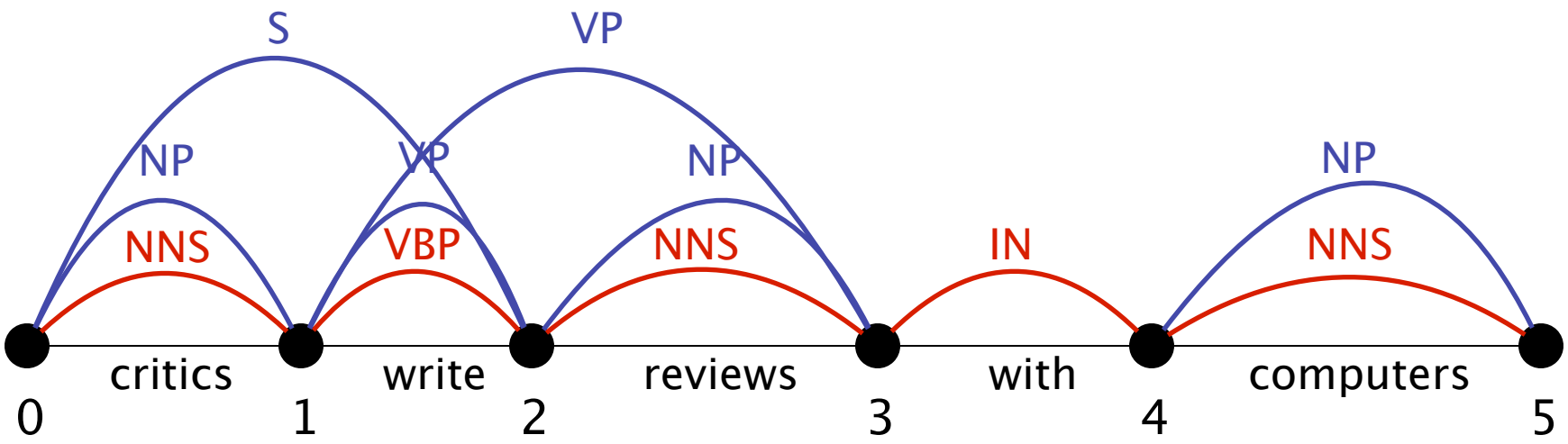
An Example

VP[1,3] PP[3,5] ROOT[0,2]



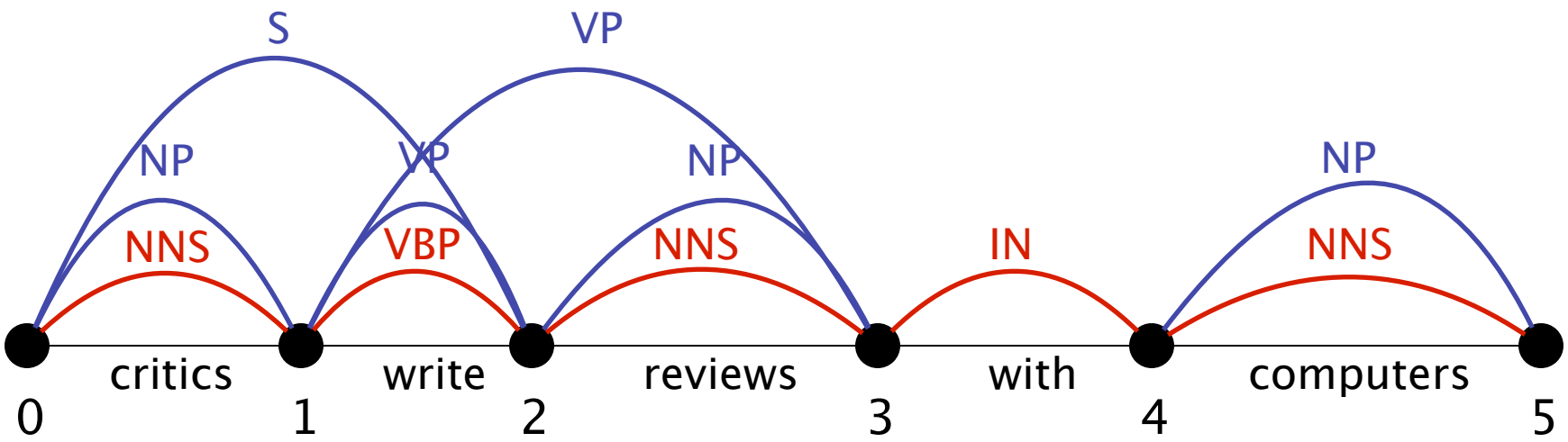
An Example

PP[3,5] ROOT[0,2]



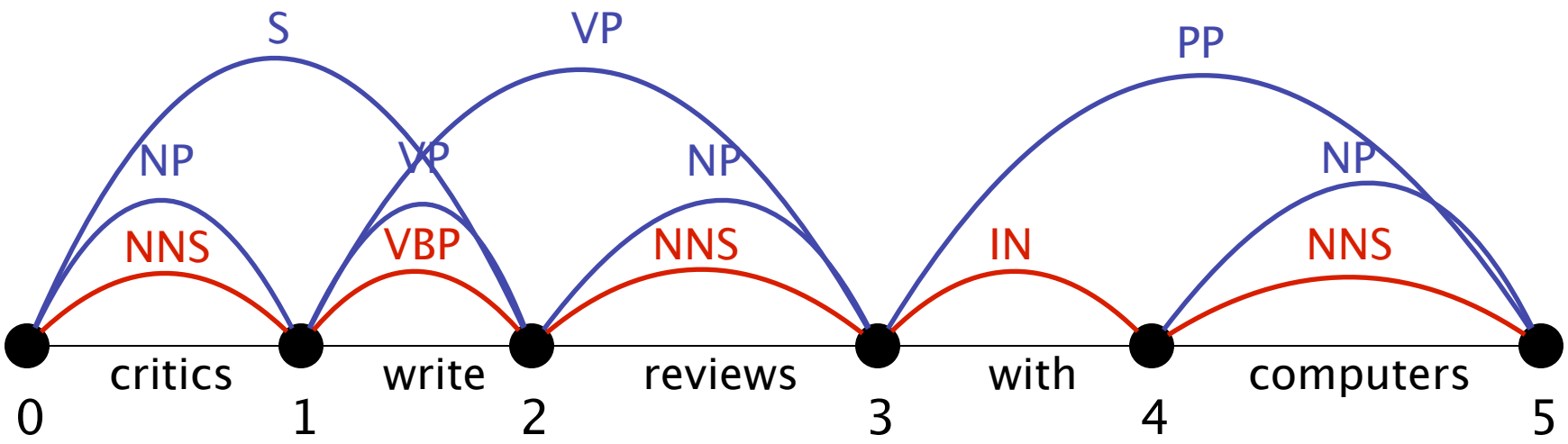
An Example

PP[3,5] ROOT[0,2] S[0,3]



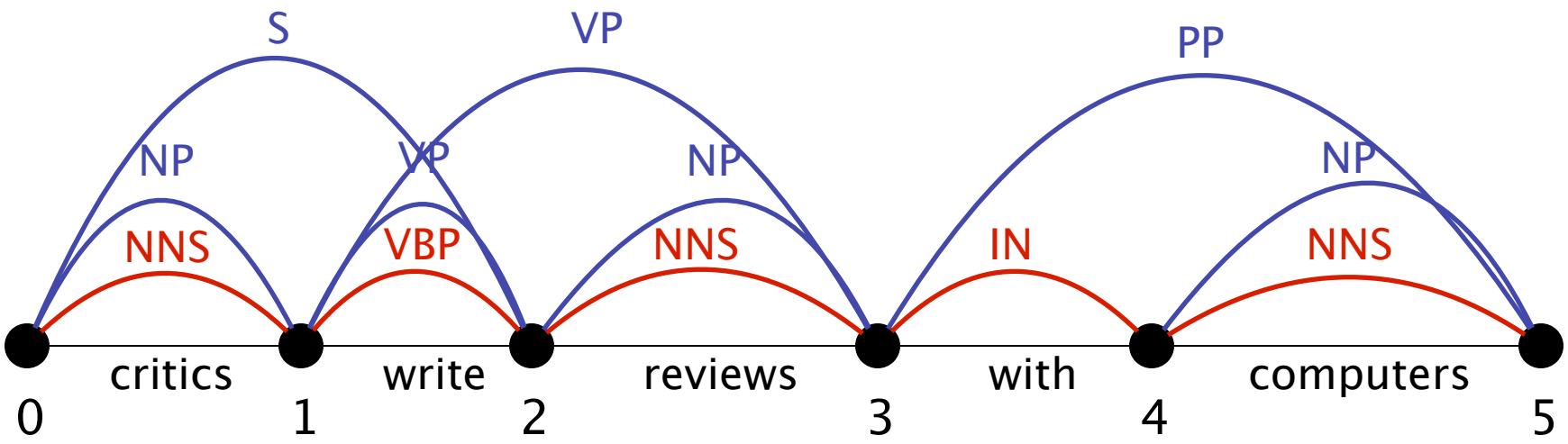
An Example

ROOT[0,2] S[0,3]



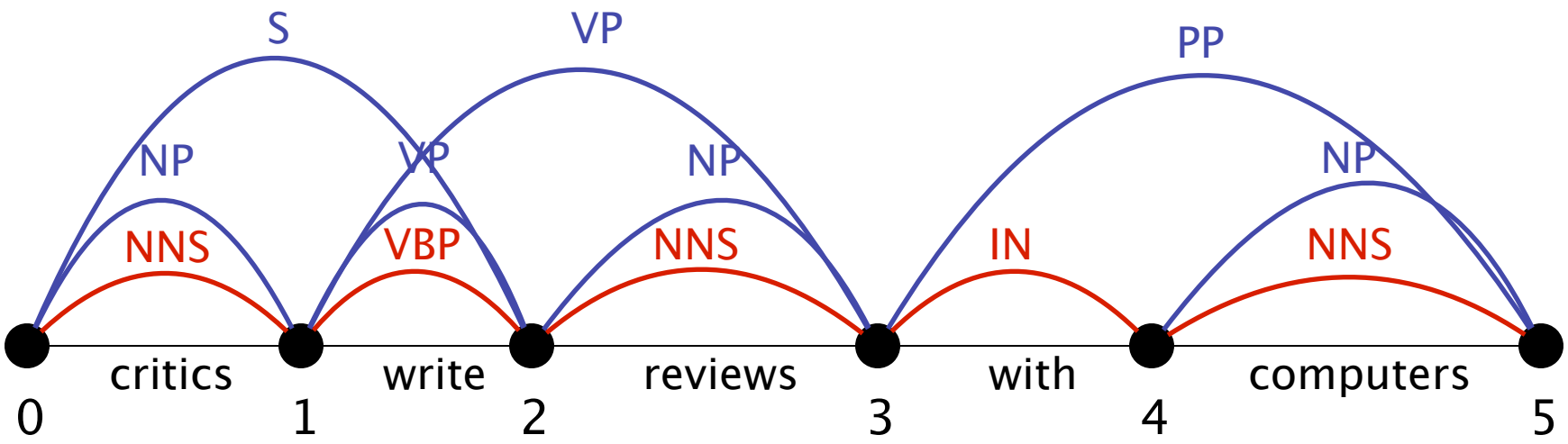
An Example

ROOT[0,2] S[0,3] VP[1,5]



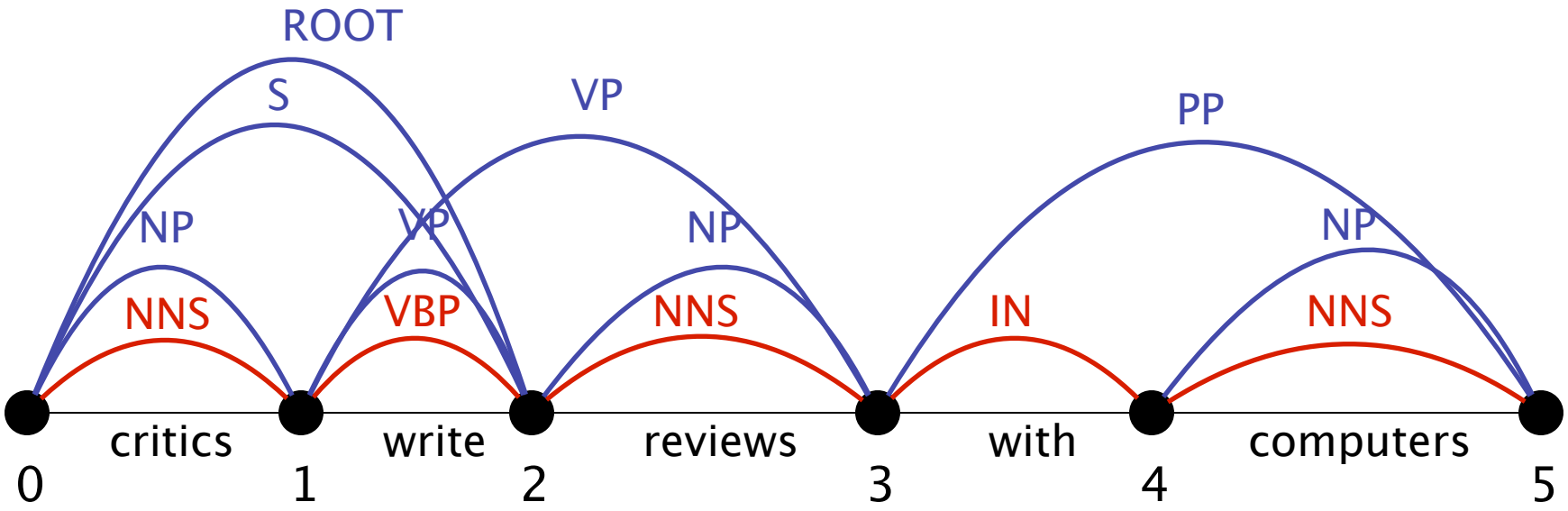
An Example

ROOT[0,2] S[0,3] VP[1,5] NP[2,5]

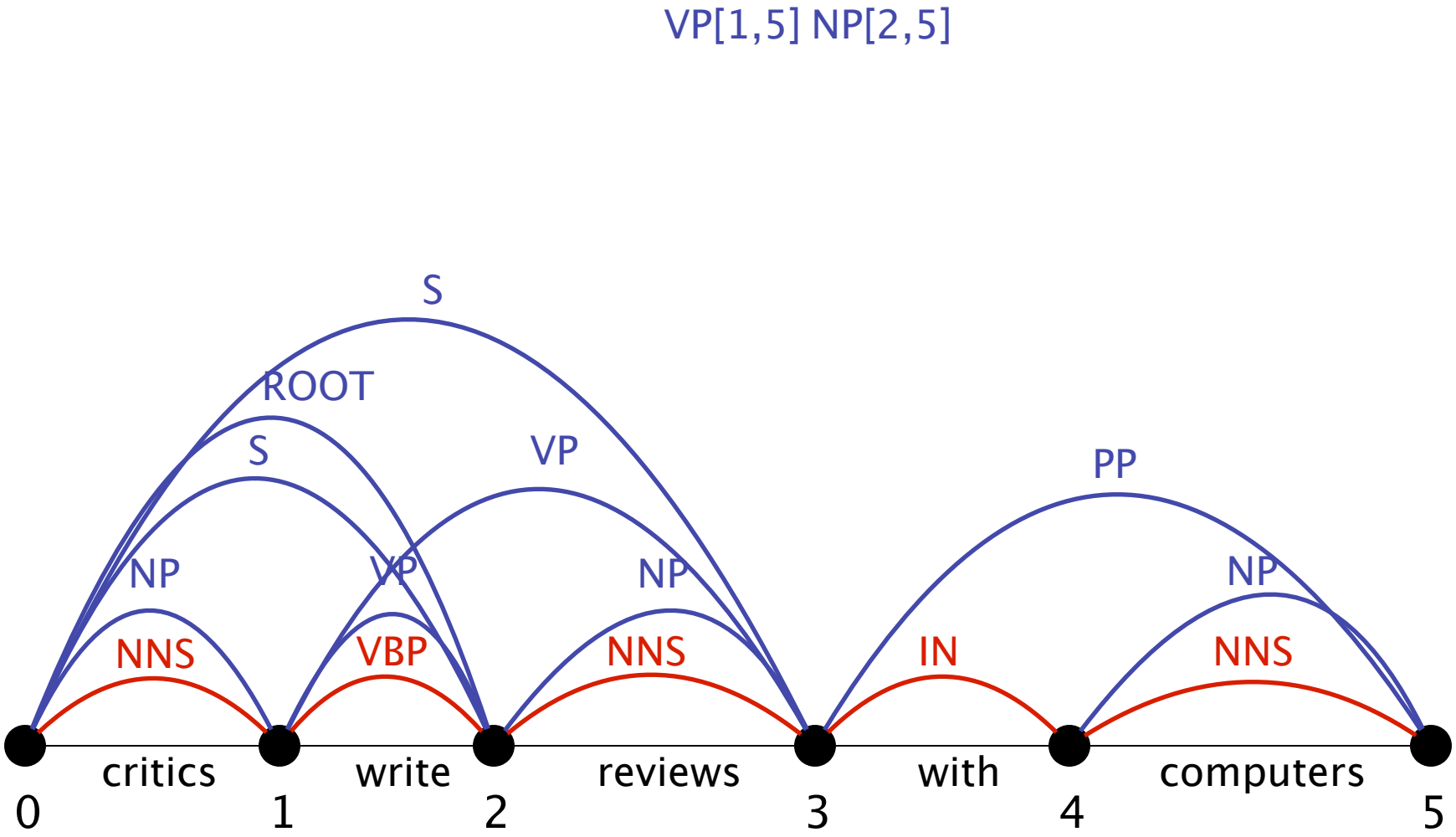


An Example

S[0,3] VP[1,5] NP[2,5]

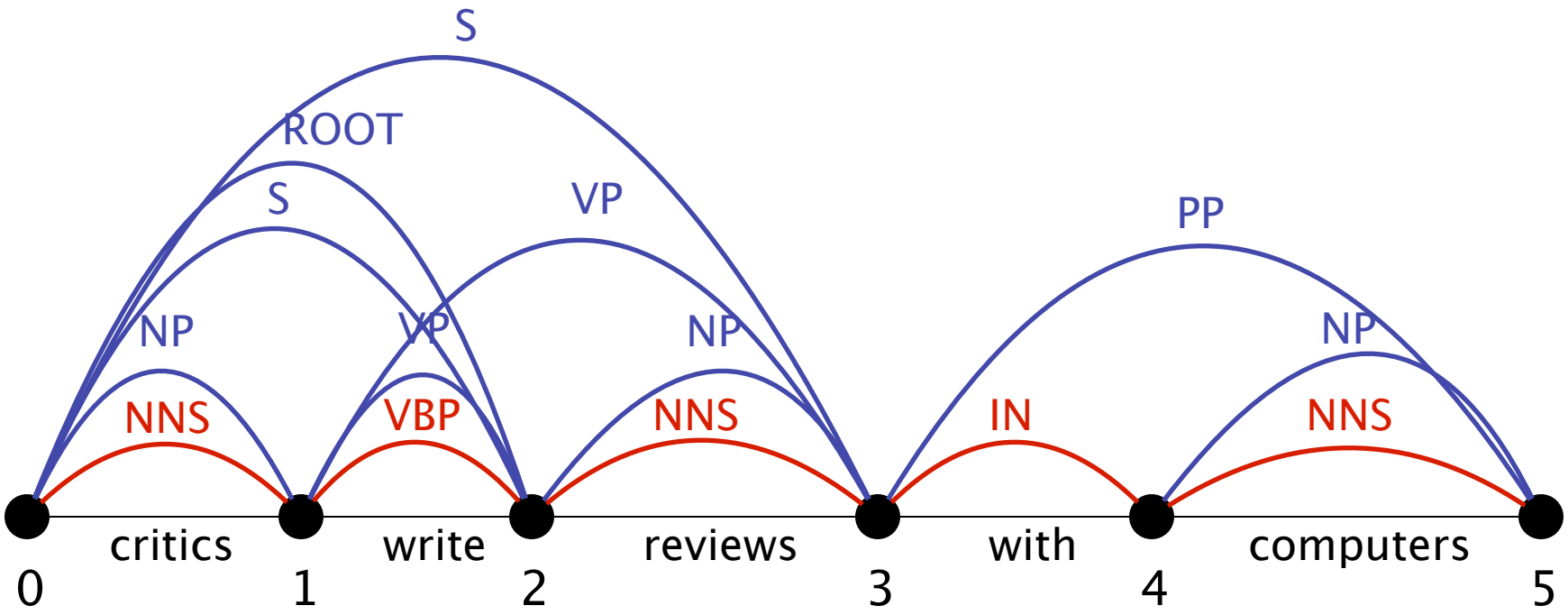


An Example


VP[1,5] NP[2,5]
S
ROOT
S
VP
PP
NP
VP
NP
NP
NNS
VBP
NNS
IN
NNS
critics
write
reviews
with
computers
0
1
2
3
4
5

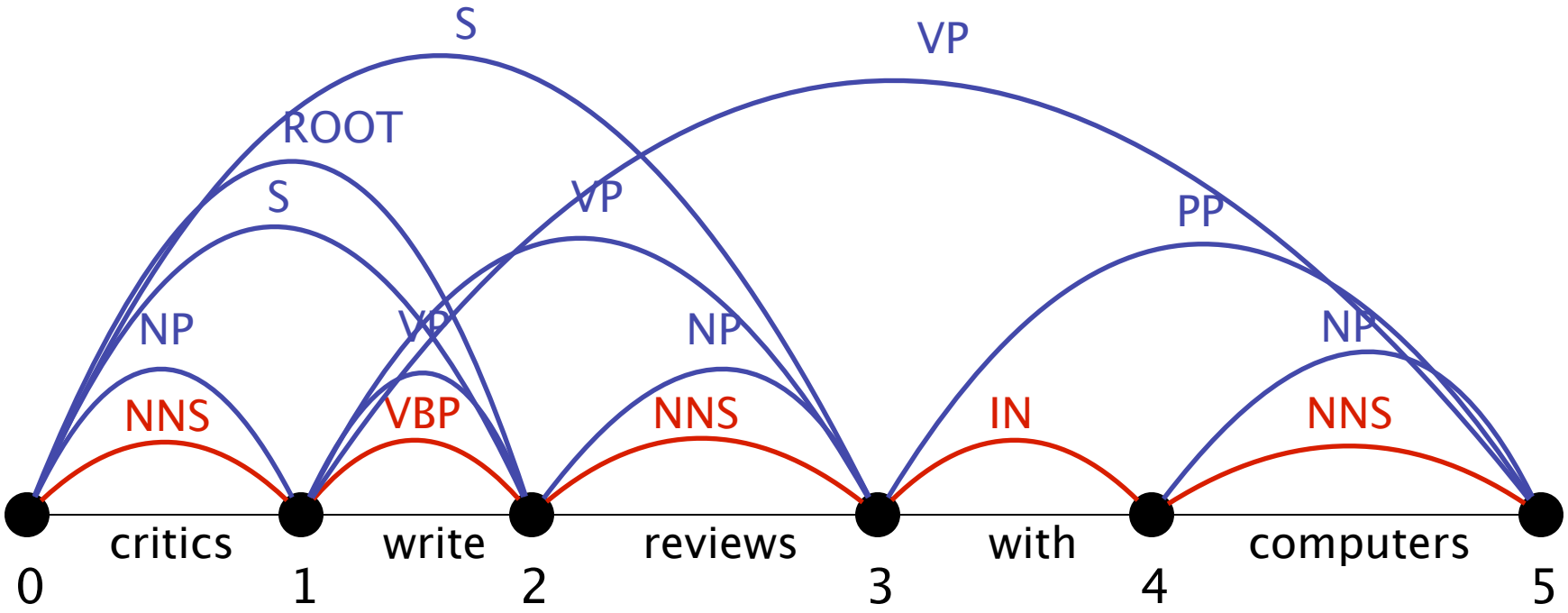
An Example

VP[1,5] NP[2,5] ROOT[0,3]



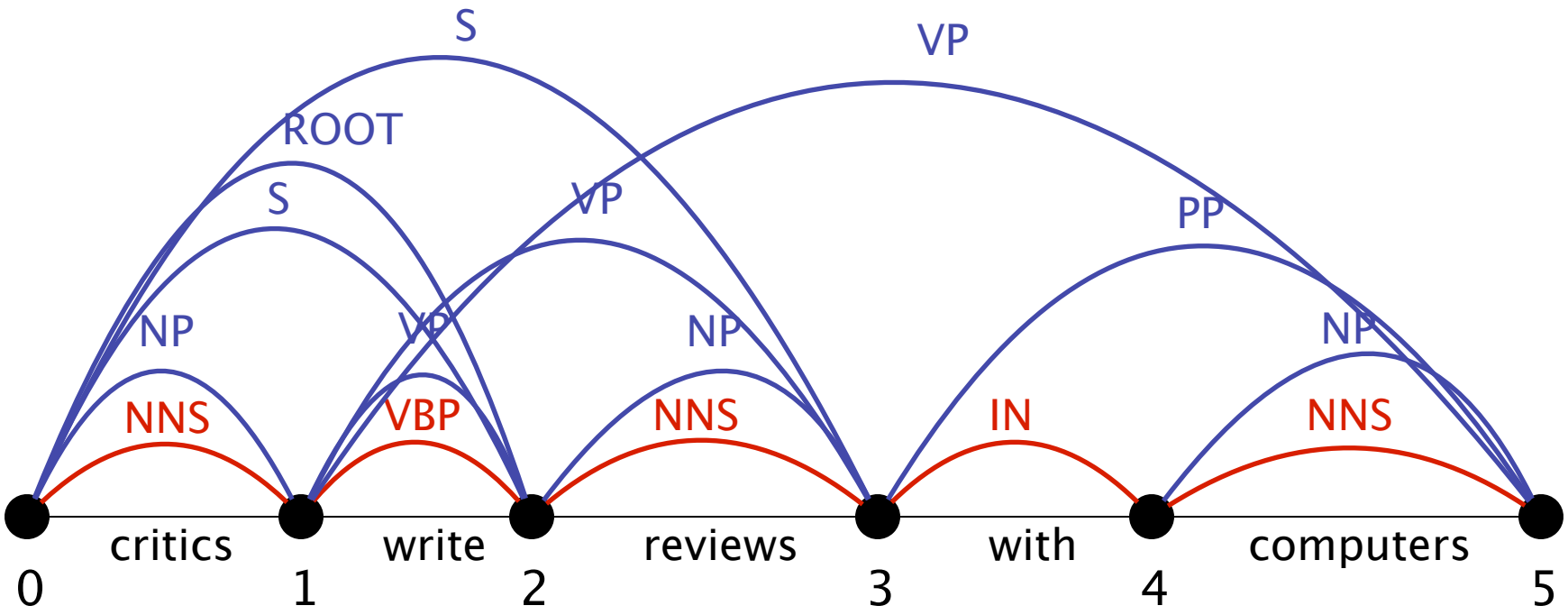
An Example

NP[2,5] ROOT[0,3]



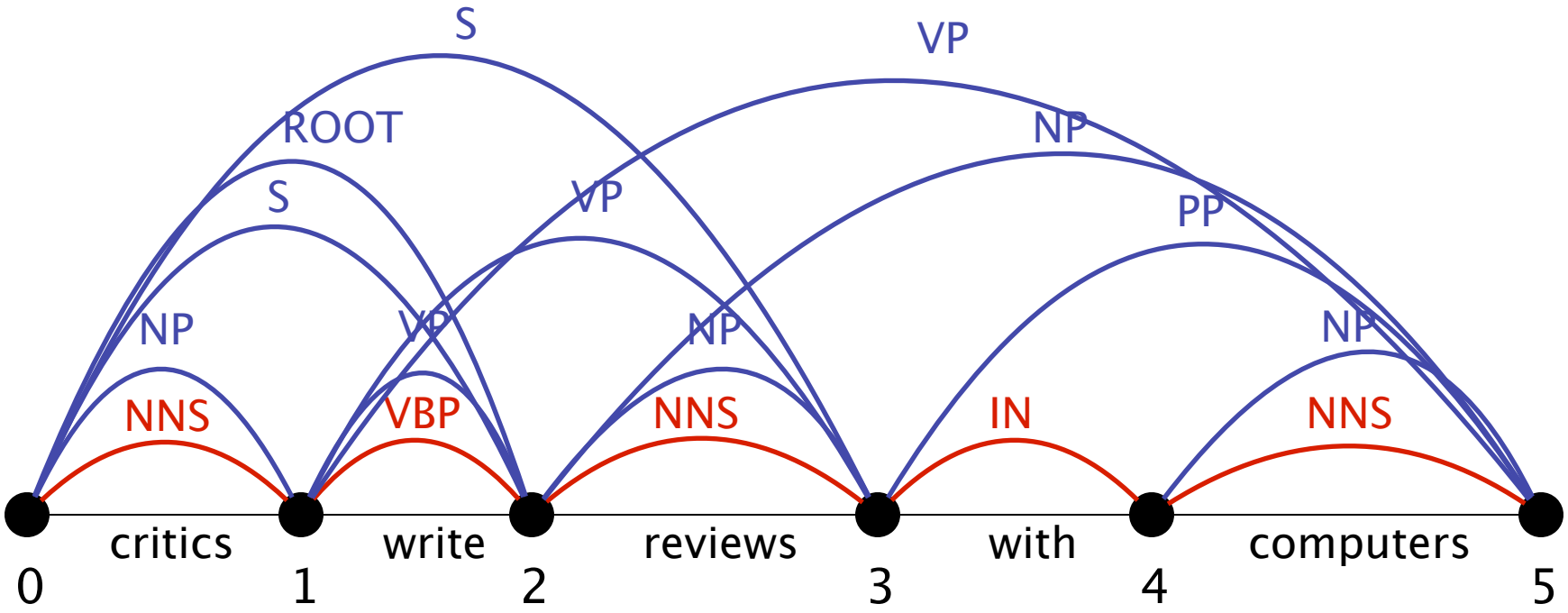
An Example

NP[2,5] ROOT[0,3] S[0,5]



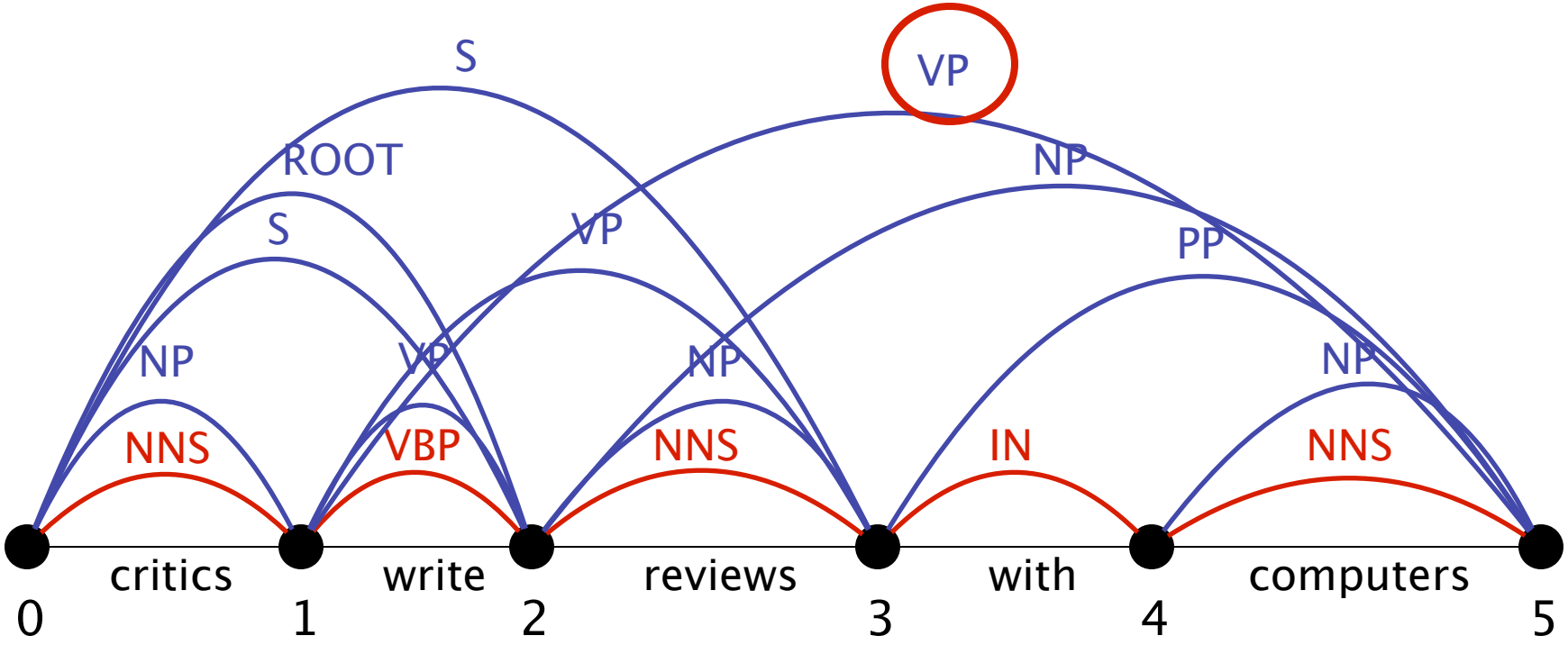
An Example

ROOT[0,3] S[0,5]



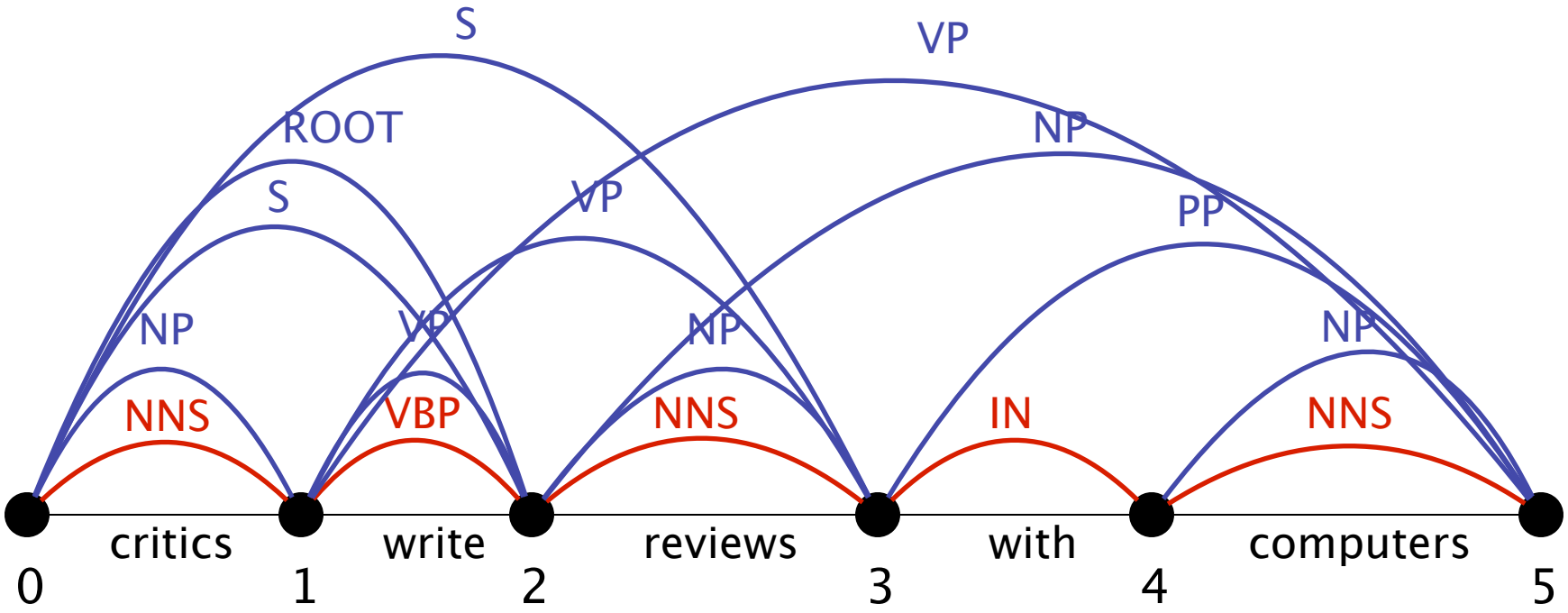
An Example

ROOT[0,3] S[0,5]

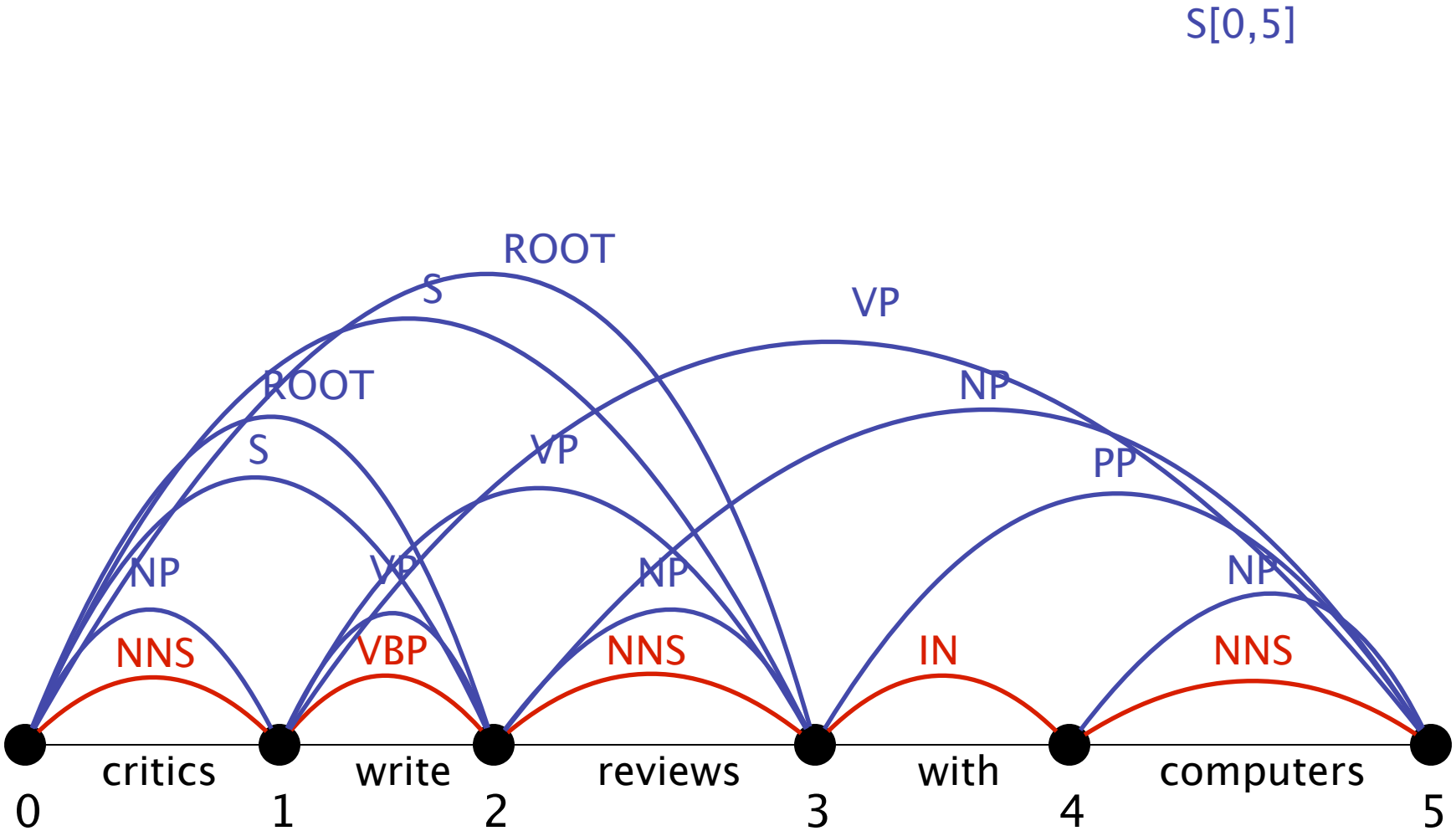


An Example

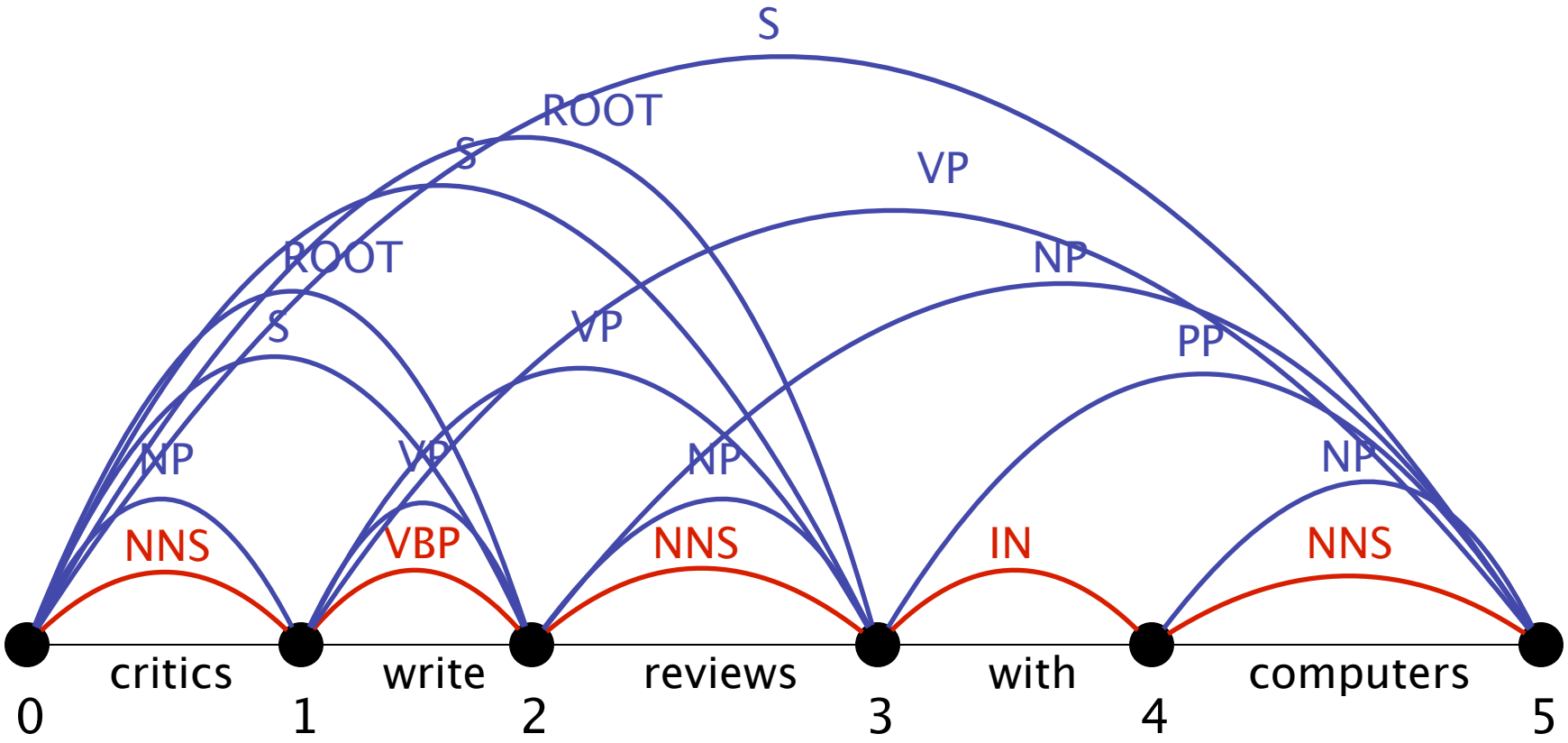
ROOT[0,3] S[0,5]



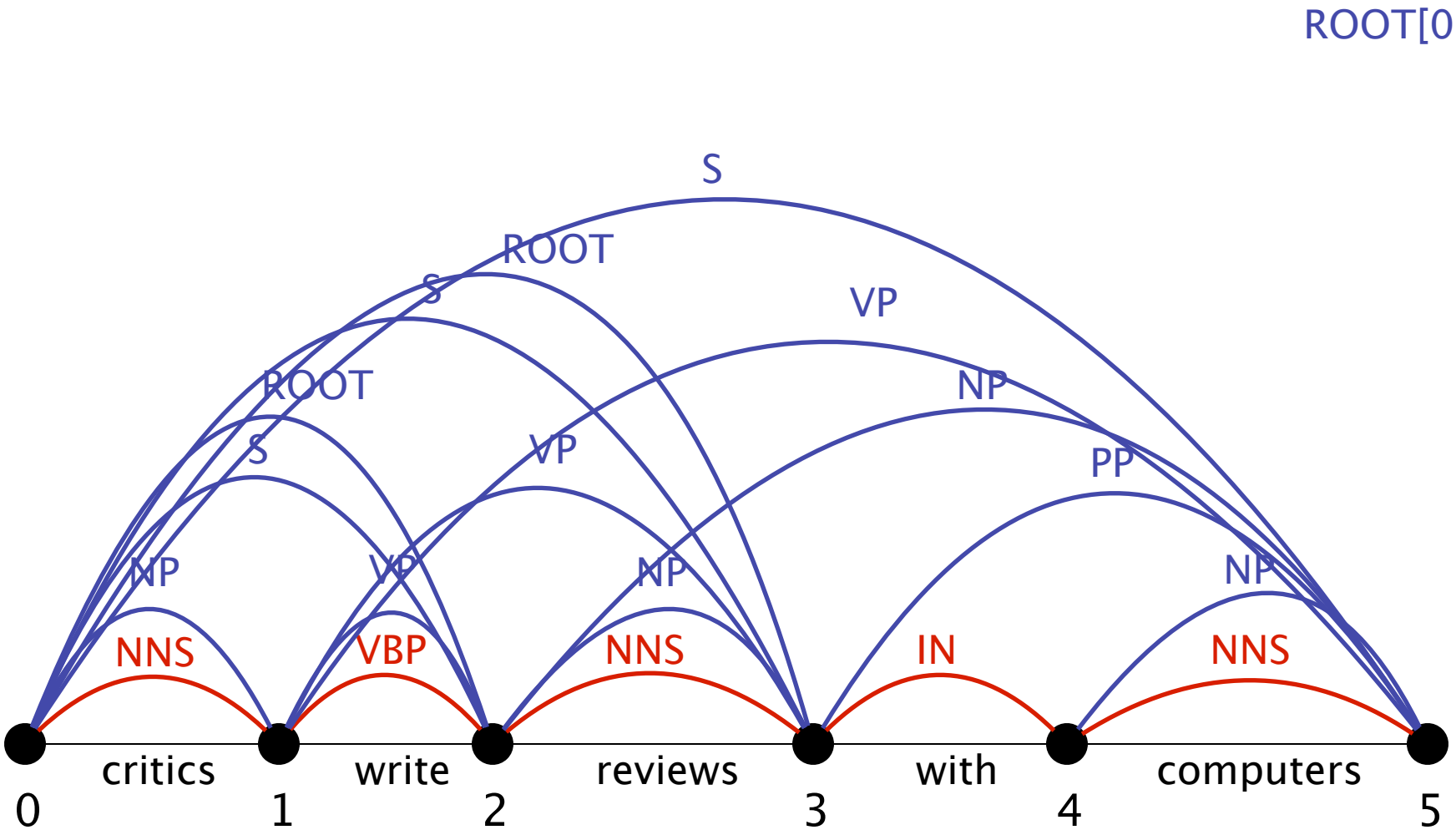
An Example


S[0,5]
ROOT
S
VP
ROOT
NP
S
VP
PP
NP
VP
NP
NP
NNS
VBP
NNS
IN
NNS
critics
write
reviews
with
computers
0
1
2
3
4
5

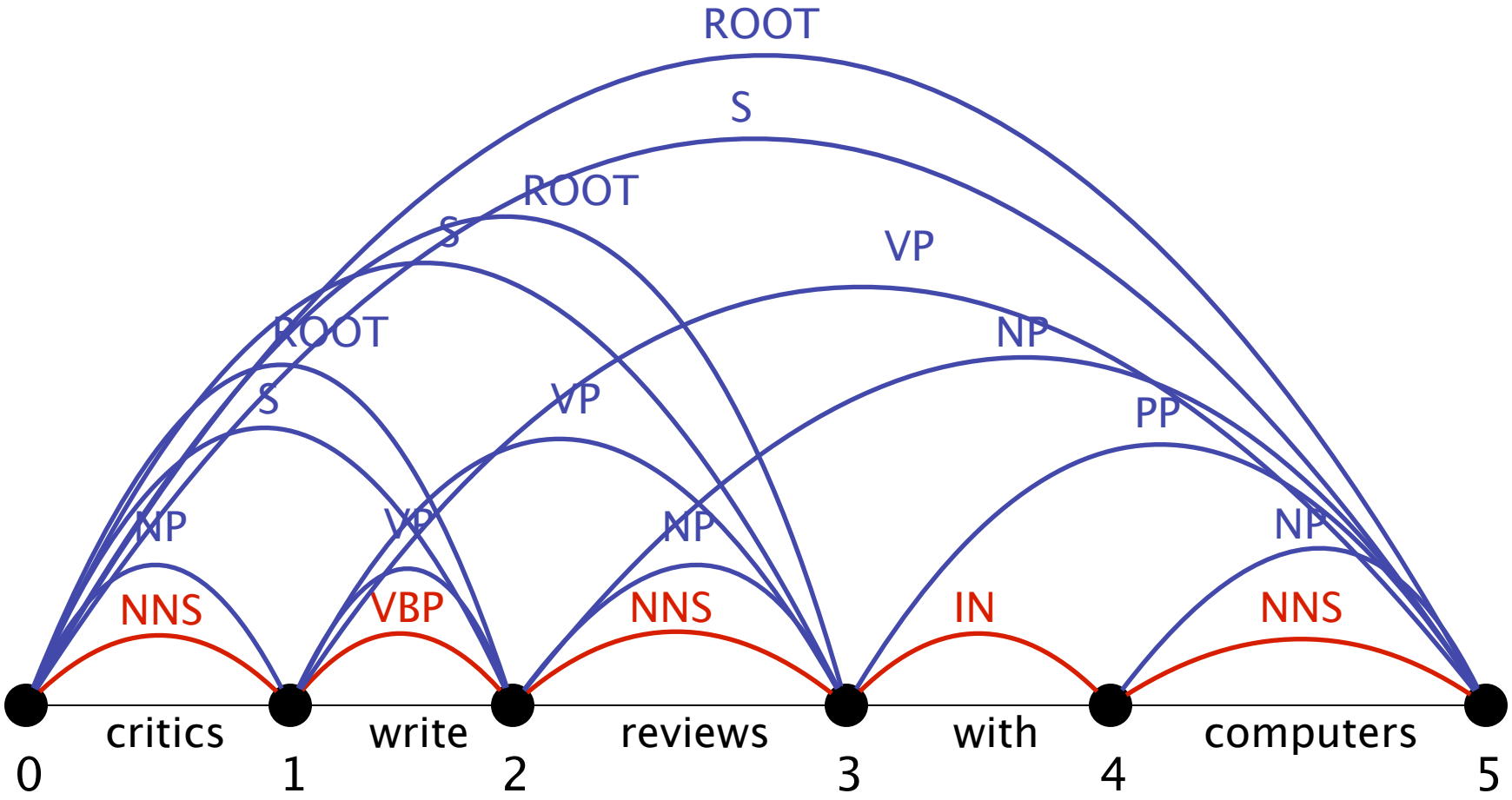
An Example


S
ROOT
S
VP
ROOT
NP
S
VP
PP
NP
VP
NP
NP
NNS
VBP
NNS
IN
NNS
critics
write
reviews
with
computers
0
1
2
3
4
5

An Example


ROOT[0,5]
S
ROOT
S
VP
ROOT
NP
S
VP
PP
NP
VP
NP
NP
NNS
VBP
NNS
IN
NNS
critics
write
reviews
with
computers
0
1
2
3
4
5

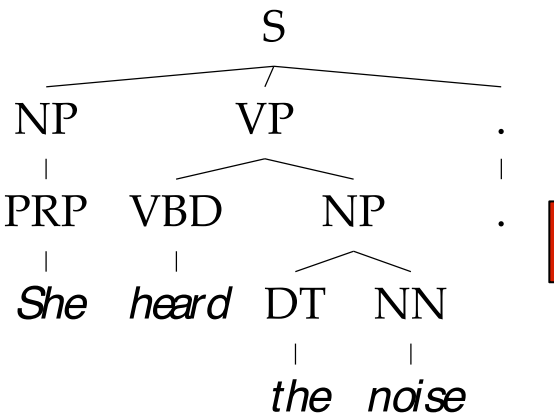
An Example

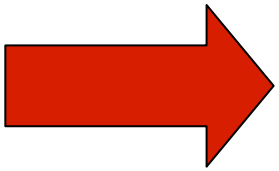

ROOT
S
ROOT
S
VP
ROOT
NP
S
VP
PP
NP
VP
NP
NP
NNS
VBP
NNS
IN
NNS
critics
write
reviews
with
computers
0
1
2
3
4
5

Treebank Grammars

[Charniak '96]

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





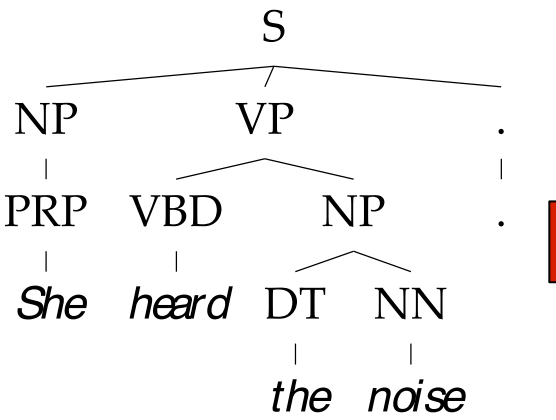
S → NP VP .	1
NP → PRP	0.5
NP → DT NN	0.5
VP → VBD NP	1
PRP → She	1
…	

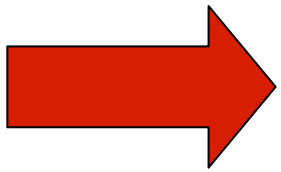
- Better results by enriching the grammar (e.g., lexicalization).
- Can also get reasonable parsers without lexicalization.

Treebank Grammars

[Charniak '96]

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





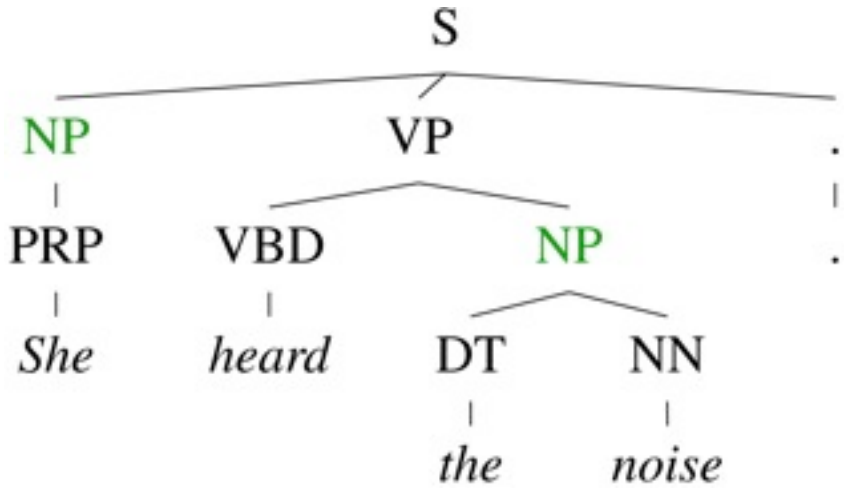
S → NP VP .	1
NP → PRP	0.5
NP → DT NN	0.5
VP → VBD NP	1
PRP → She	1
…	

- Better results by enriching the grammar (e.g., lexicalization).
- Can also get reasonable parsers

Model	F1
Charniak '96	72.0

Conditional Independence?

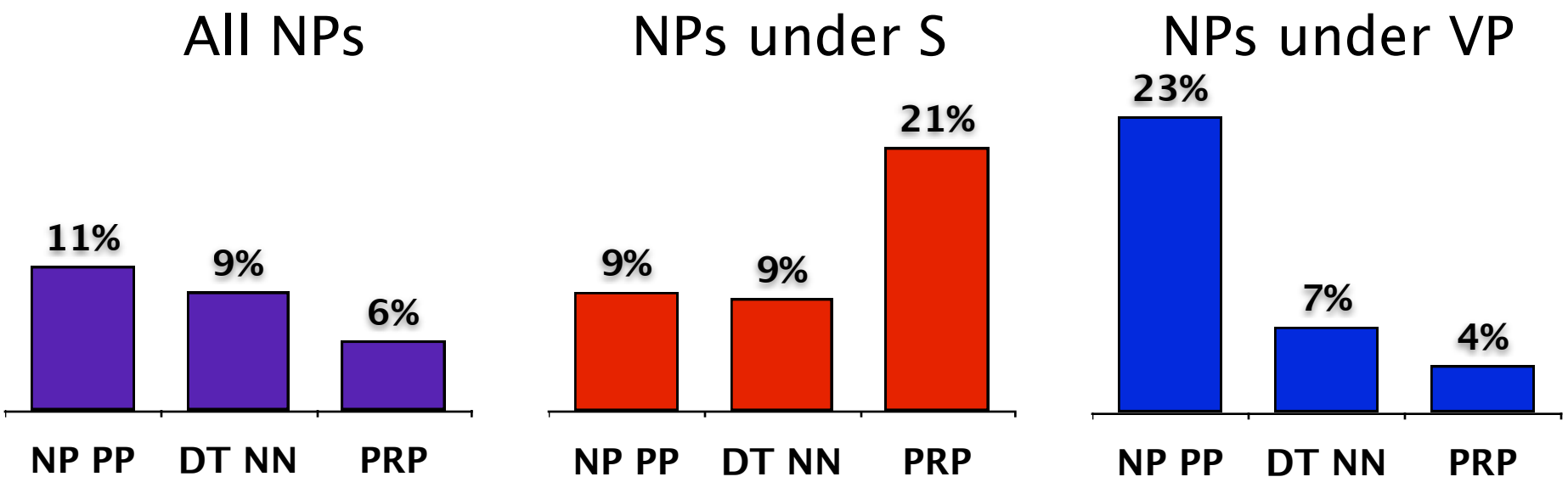
- Not every NP expansion can fill every NP slot



- A grammar with symbols like "NP" won't be context-free
- Statistically, conditional independence too strong

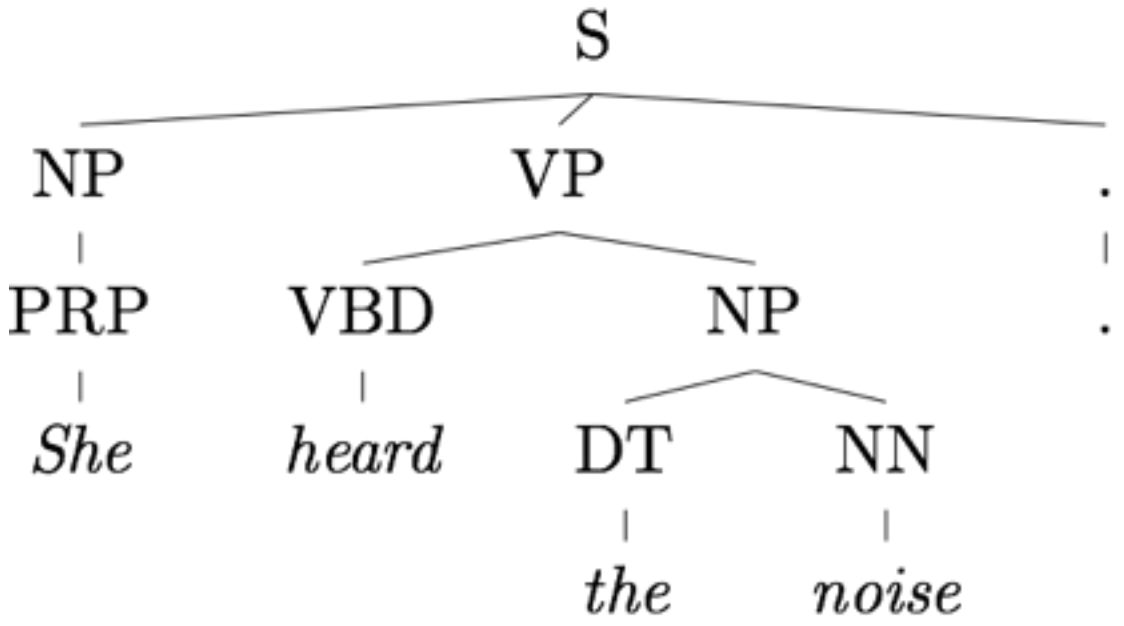
Non-Independence

- Independence assumptions are often too strong.

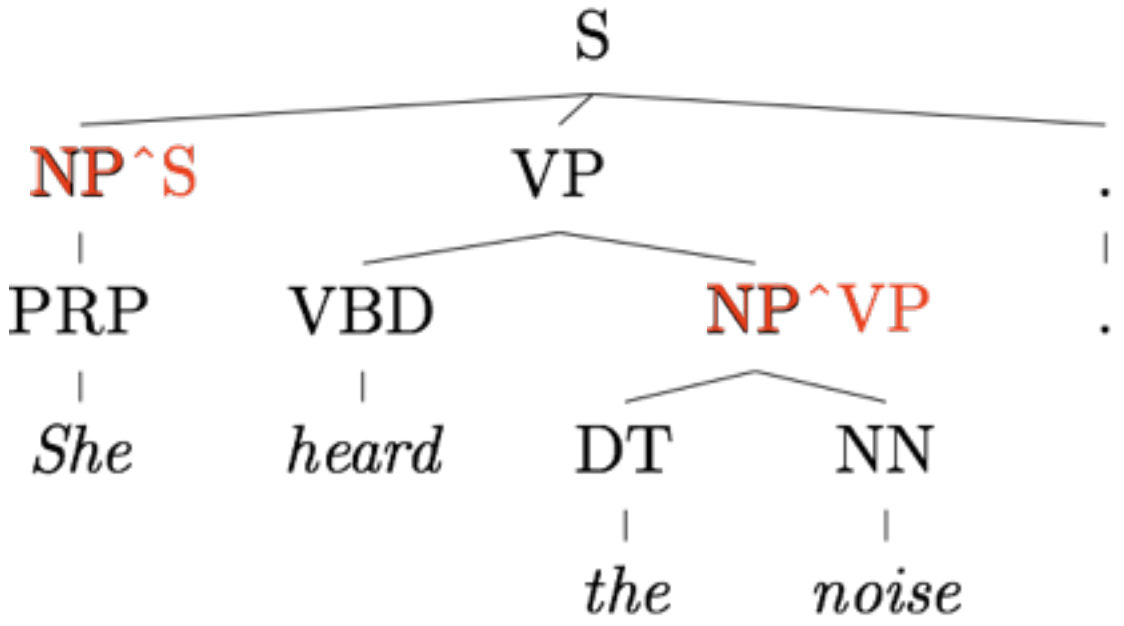


- Example: the expansion of an NP is highly dependent on the parent of the NP (i.e., subjects vs. objects).
- Also: the subject and object expansions are correlated!

The Game of Designing a Grammar

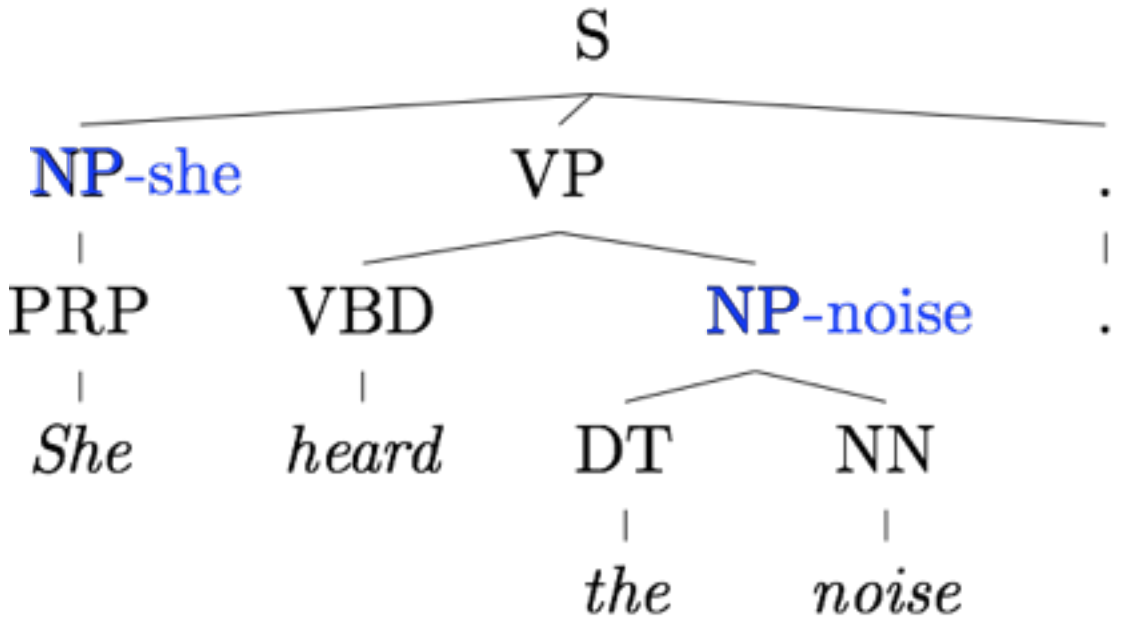


The Game of Designing a Grammar



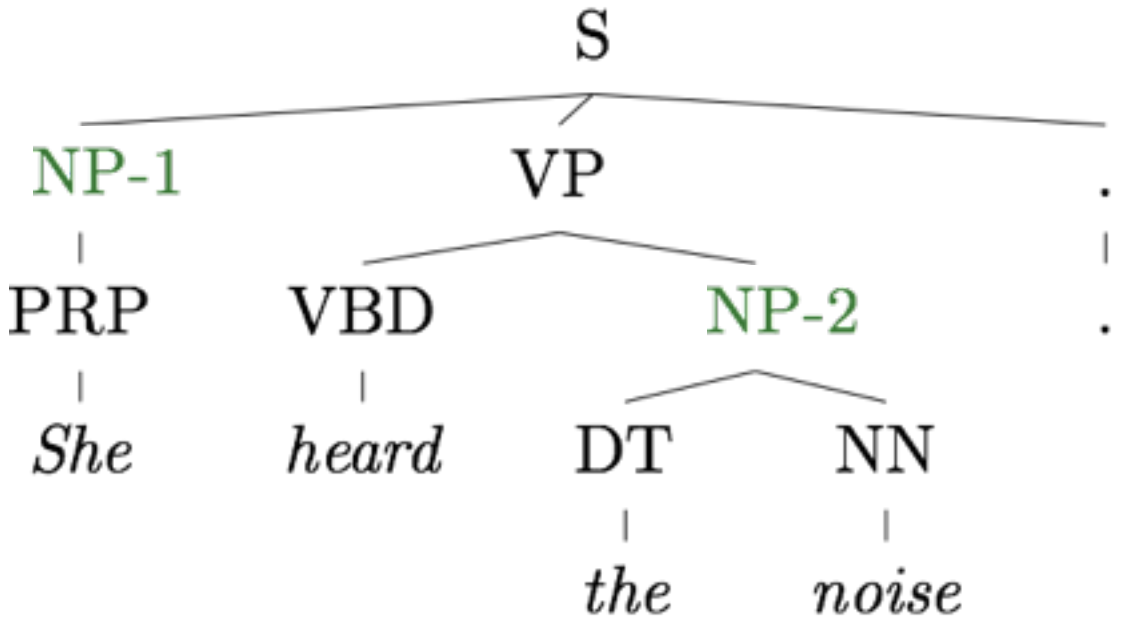
- Structure Annotation [Johnson ’98, Klein&Manning ’03]

The Game of Designing a Grammar



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

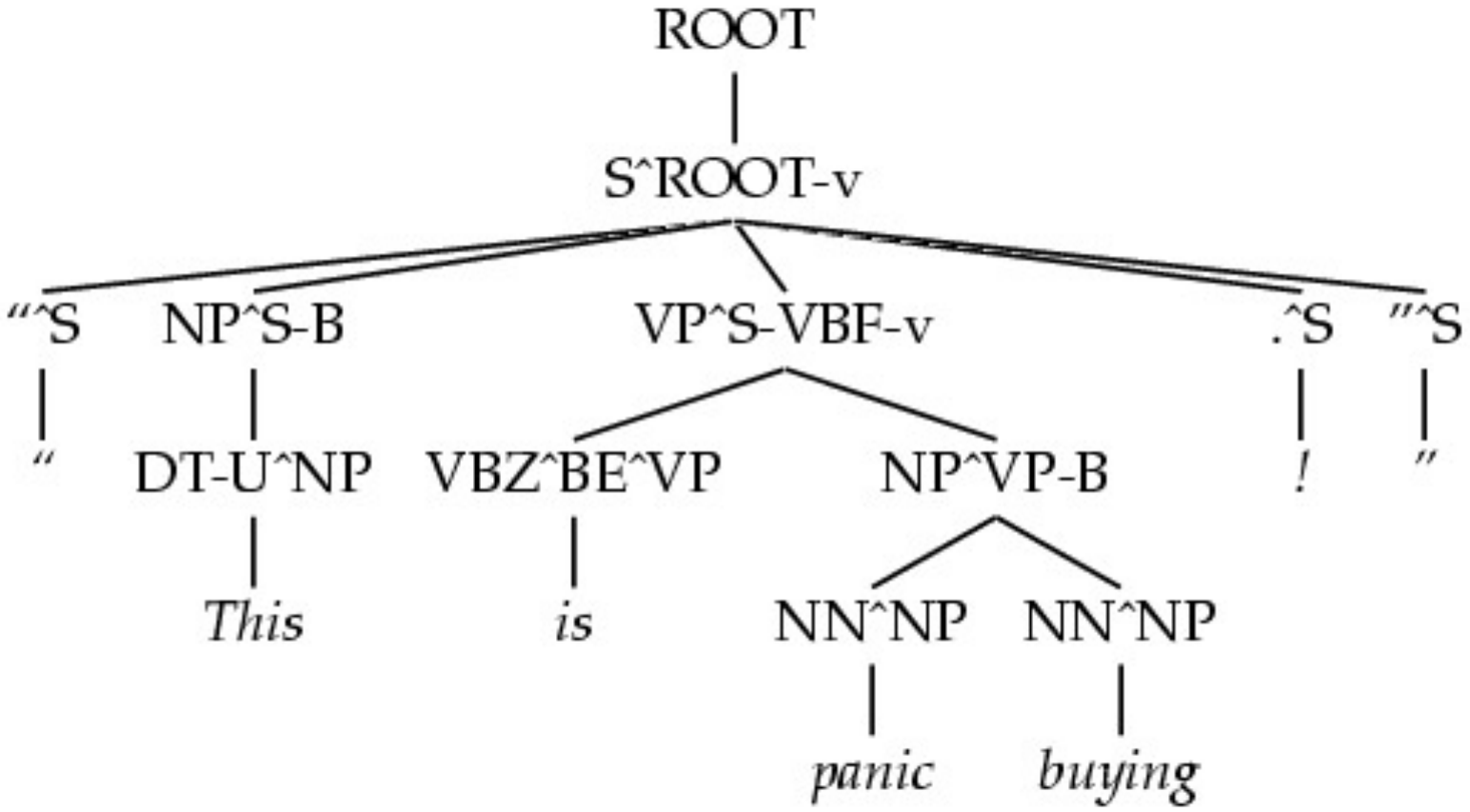
The Game of Designing a Grammar



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

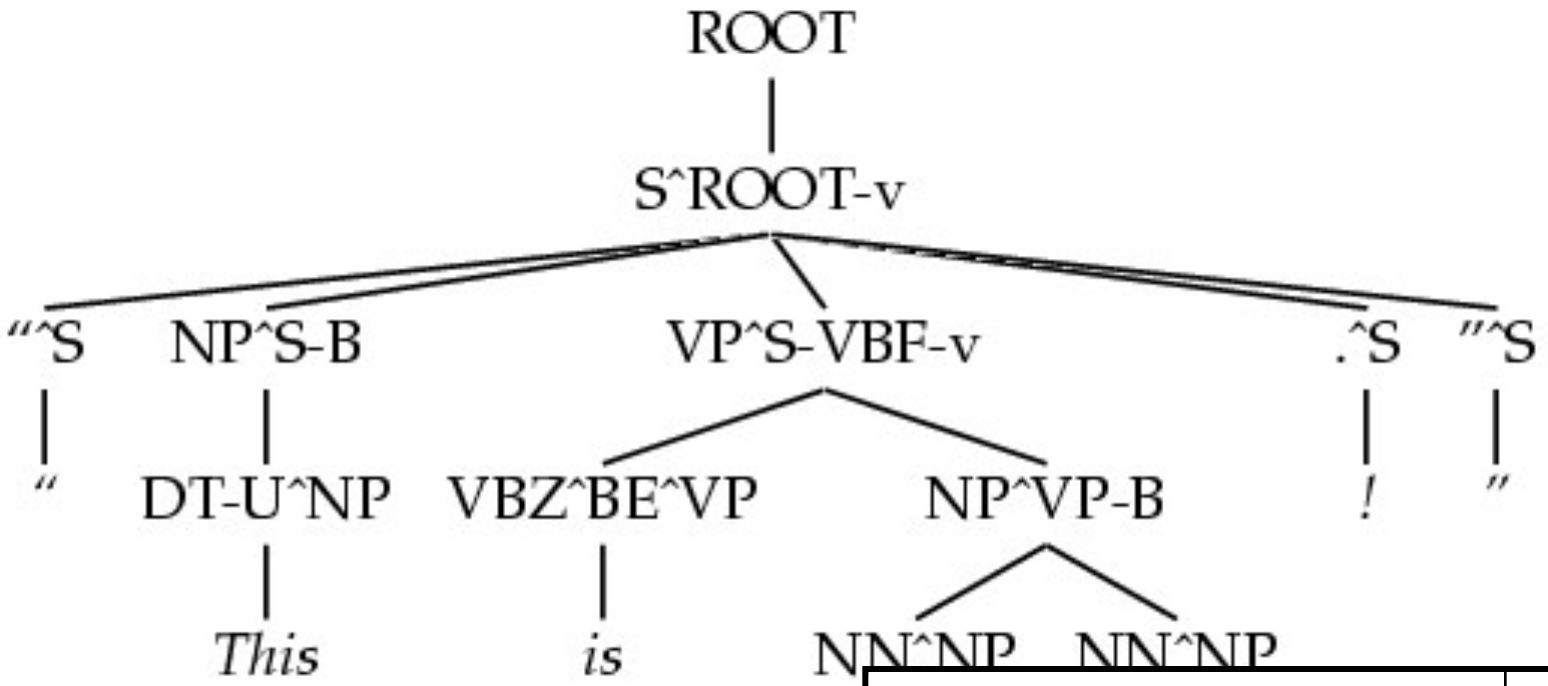
A Fully Annotated (Unlexicalized) Tree

[Klein & Manning '03]



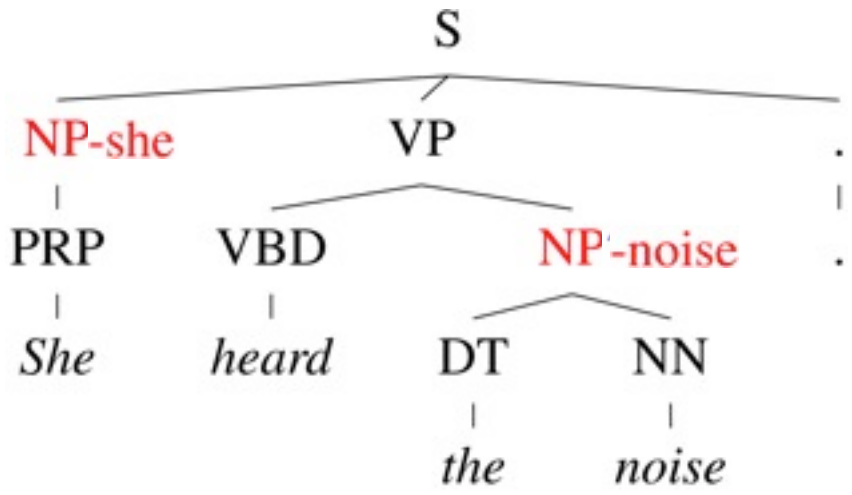
A Fully Annotated (Unlexicalized) Tree

[Klein & Manning '03]



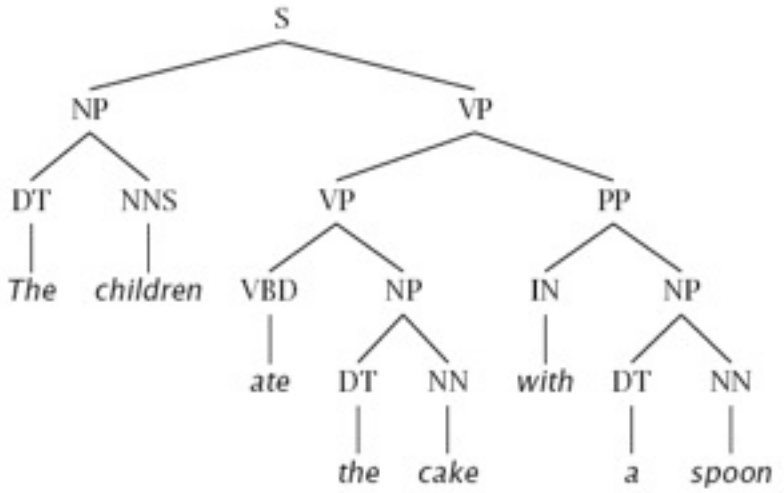
Model	F1
Charniak '96	72.0
Klein&Manning	86.3

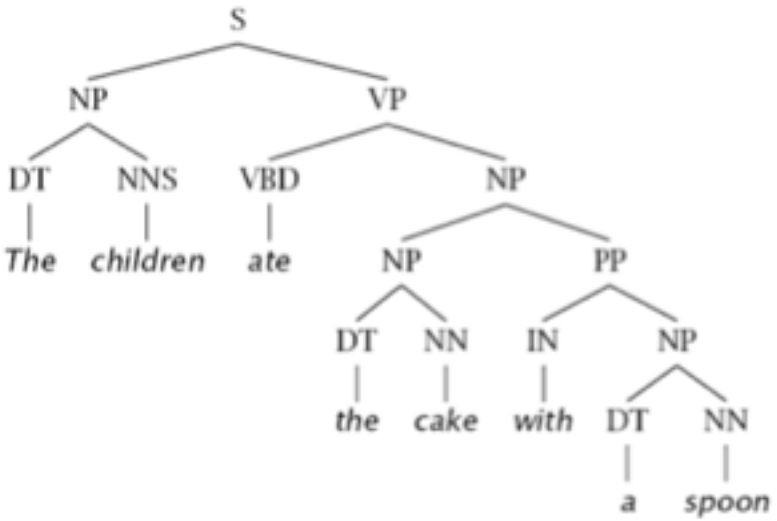
The Game of Designing a Grammar



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

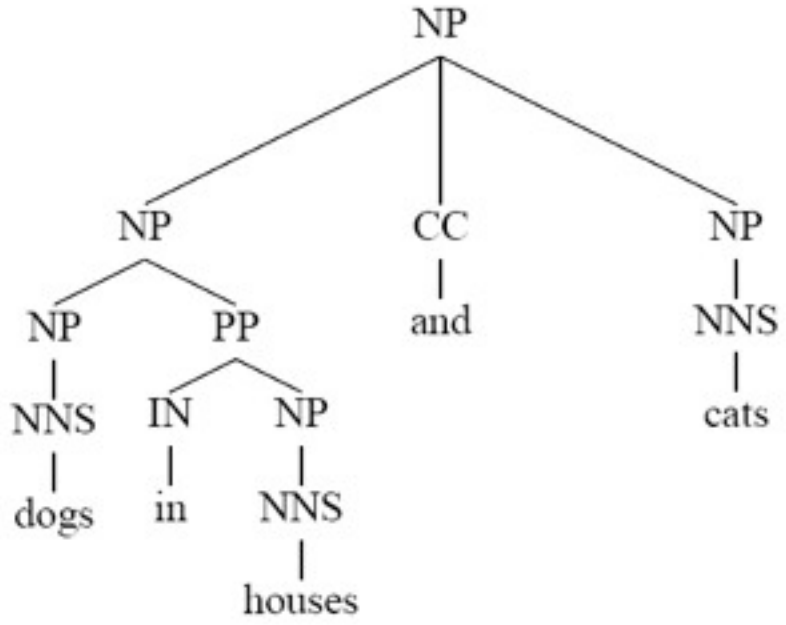
Problems with PCFGs

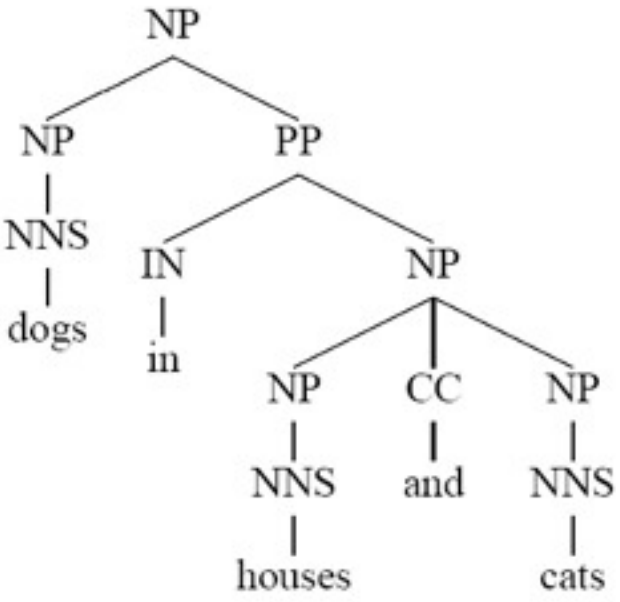




- If we do no annotation, these trees differ only in one rule:
  - VP → VP PP
  - NP → NP PP
- Parse will go one way or the other, regardless of words
- We addressed this in one way with unlexicalized grammars (how?)
- Lexicalization allows us to be sensitive to specific words

Problems with PCFGs



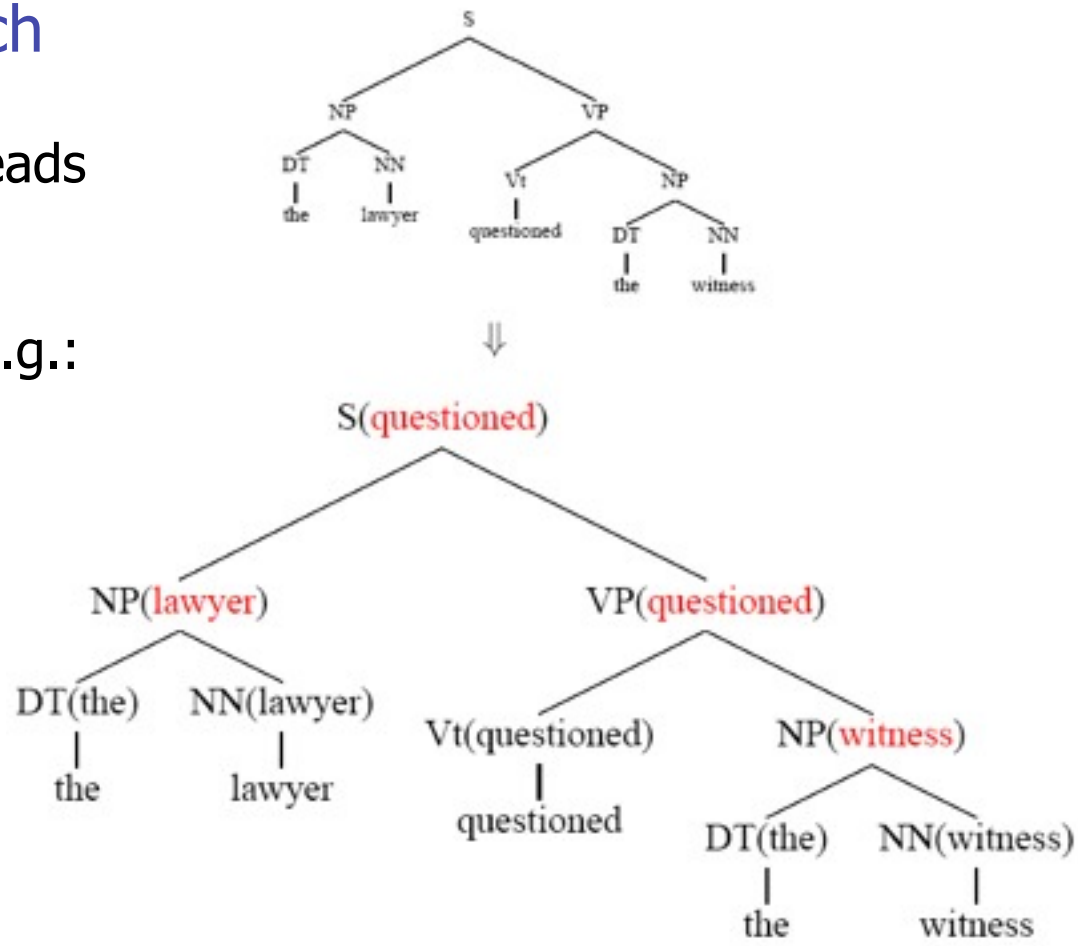


- What's different between basic PCFG scores here?
- What (lexical) correlations need to be scored?

Lexicalized Trees

[Charniak '97, Collins '97]

- Add "headwords" 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?

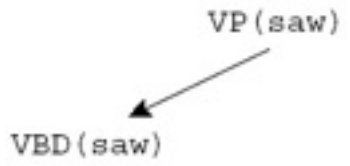
- Problem: we now have to estimate probabilities like

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

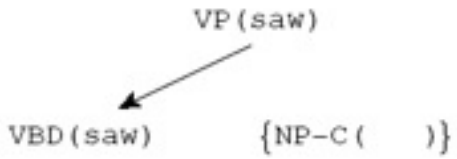
- Never going to get these atomically off of a treebank
- Solution: break up derivation into smaller steps

Lexical Derivation Steps

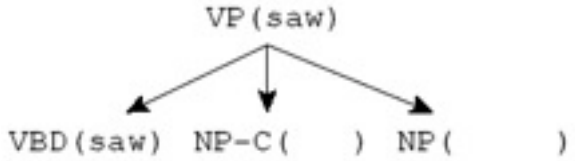
- A derivation of a local tree [Collins '99]



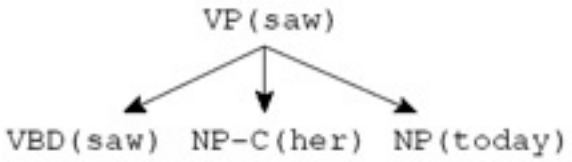
Choose a head tag and word



Choose a complement bag



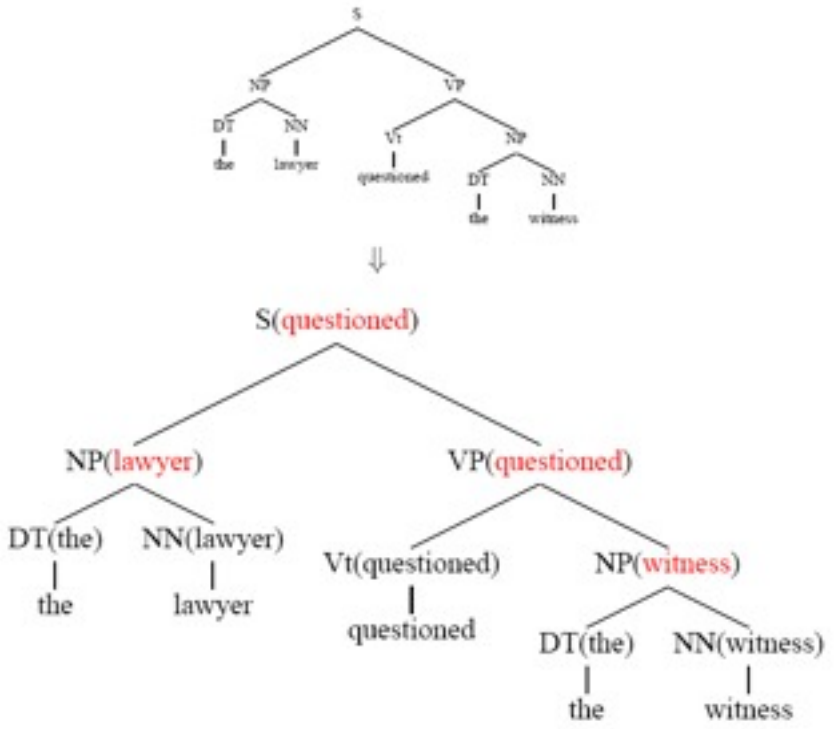
Generate children (incl. adjuncts)



Recursively derive children

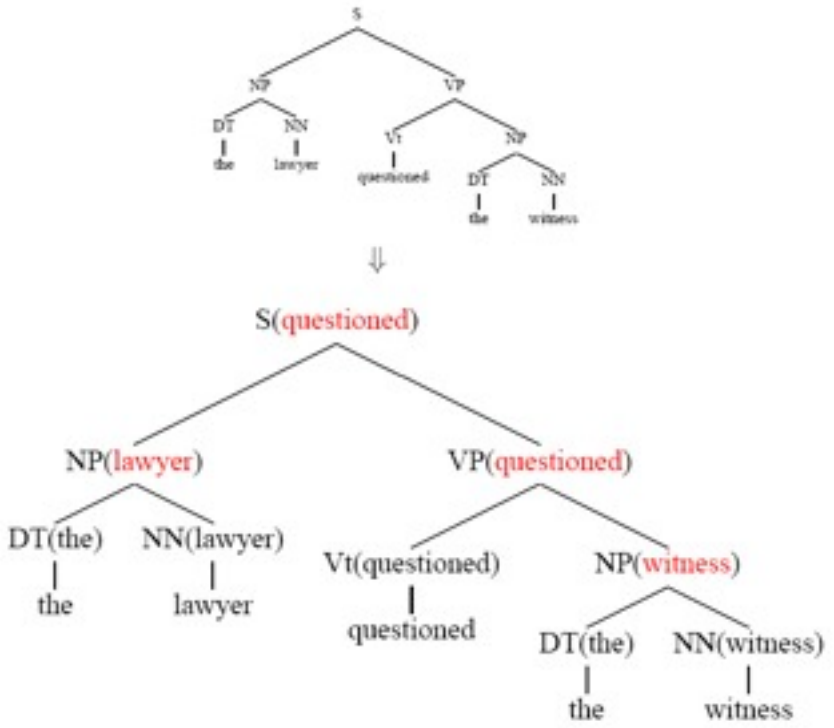
Lexicalized Grammars

- Challenges:
  - Many parameters to estimate: requires sophisticated smoothing techniques
  - Exact inference is too slow: requires pruning heuristics
  - Difficult to adapt to new languages: At least head rules need to be specified, typically more changes needed



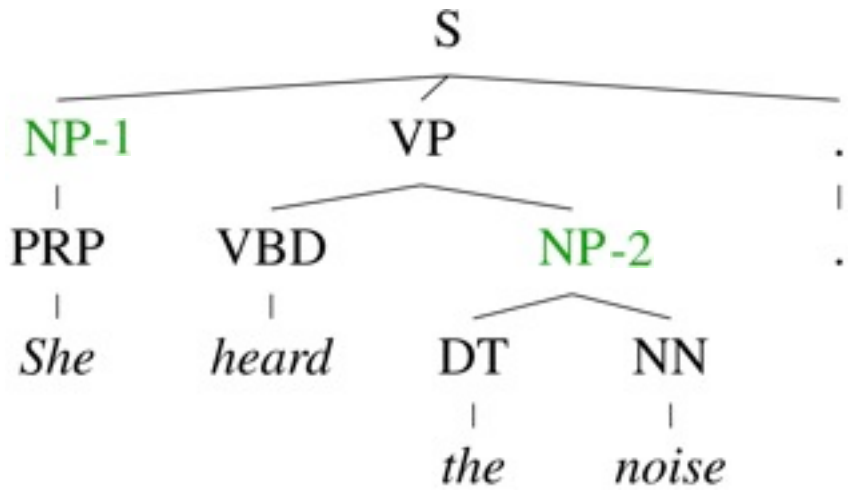
Lexicalized Grammars

- Challenges:
  - Many parameters to estimate: requires sophisticated smoothing techniques
  - Exact inference is too slow: requires pruning heuristics
  - Difficult to adapt to new languages: At least head rules need to be specified, typically more changes needed



Model	F1
Klein&Manning	86.3
Charniak '00	90.1

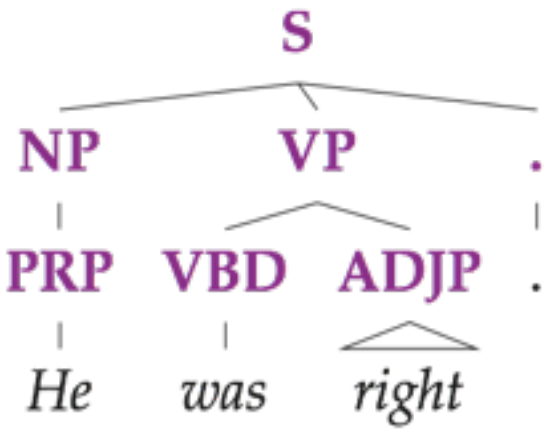
The Game of Designing a Grammar



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

Latent Variable Grammars

[Matsuzaki et al. ’05, Petrov et al. ‘06]

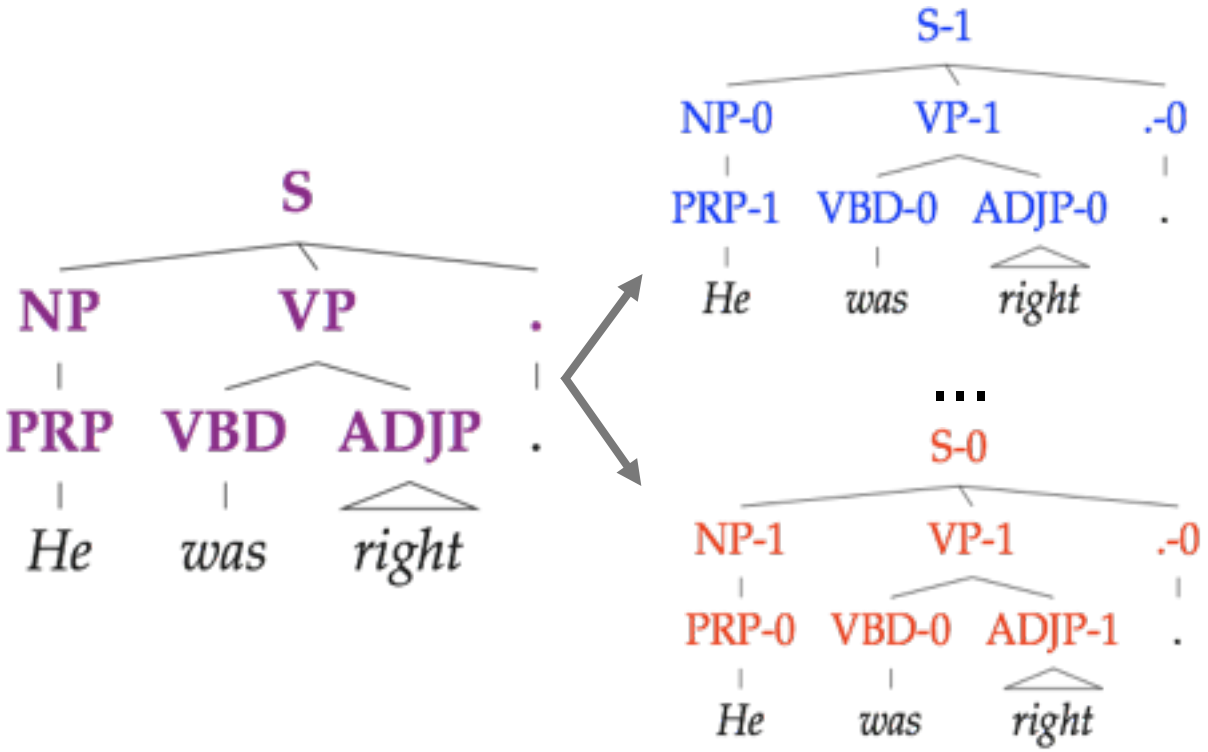


Parse Tree T

Sentence w

Latent Variable Grammars

[Matsuzaki et al. '05, Petrov et al. '06]



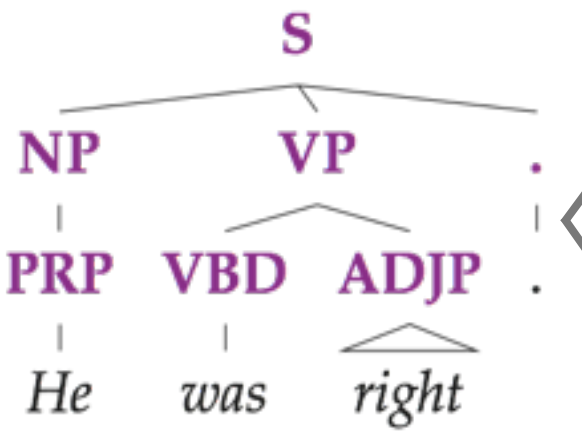
Parse Tree T

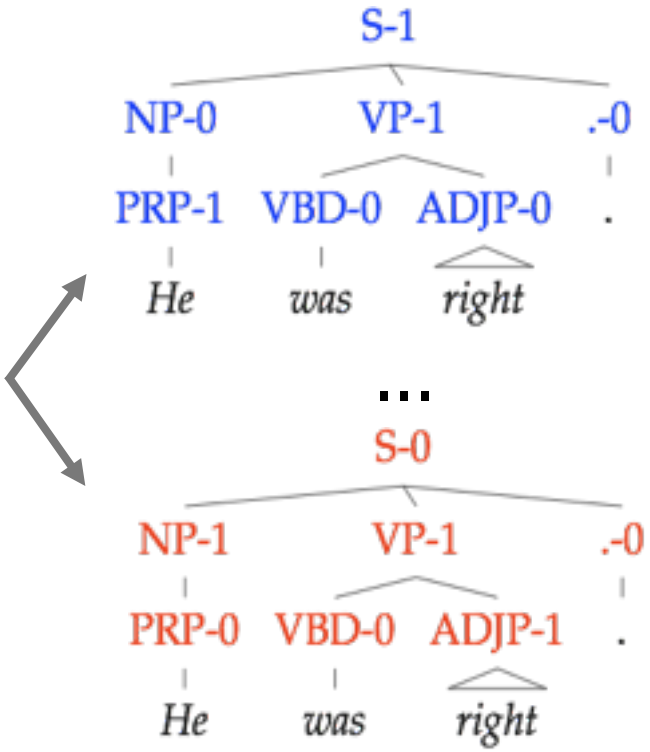
Sentence w

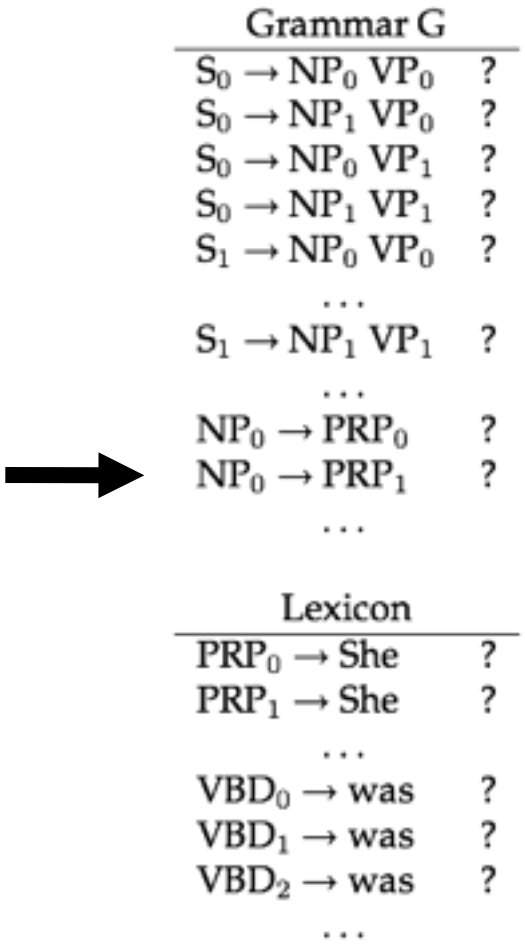
Derivations $t : T$

Latent Variable Grammars

[Matsuzaki et al. '05, Petrov et al. '06]







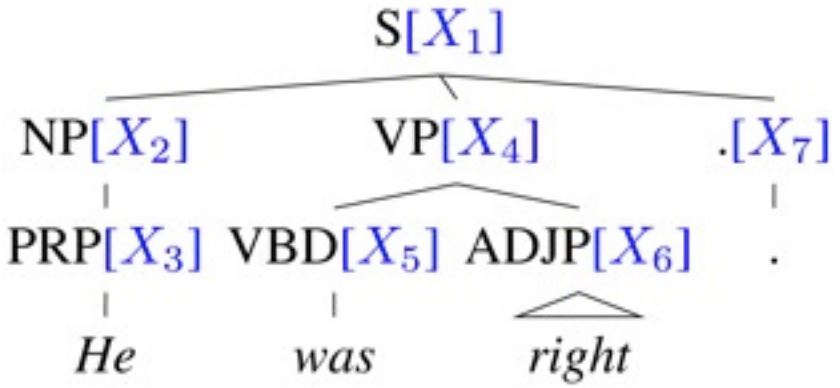
Parse Tree T
Sentence w

Derivations $t : T$

Parameters θ

Learning Latent Annotations

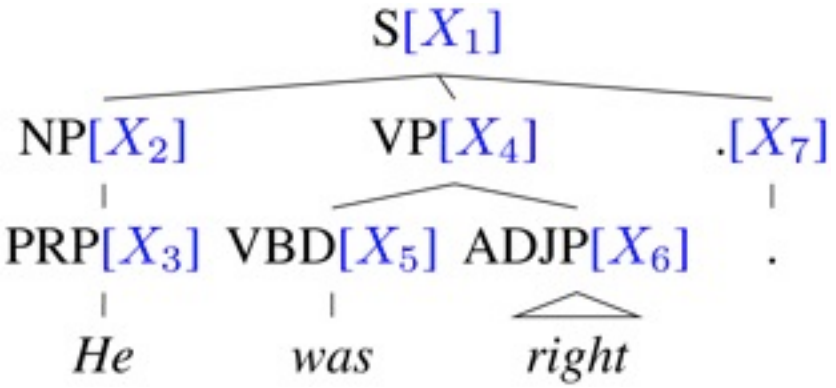
EM algorithm:



Learning Latent Annotations

EM algorithm:

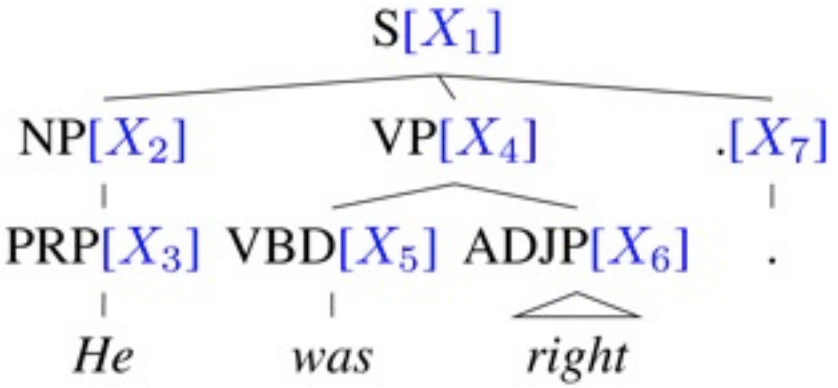
- Brackets are known
- Base categories are known
- Only induce subcategories

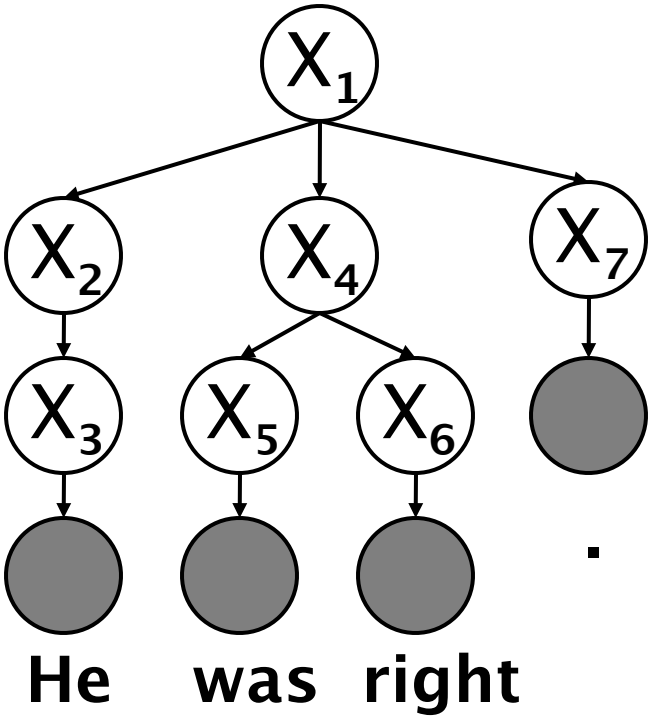


Learning Latent Annotations

EM algorithm:

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

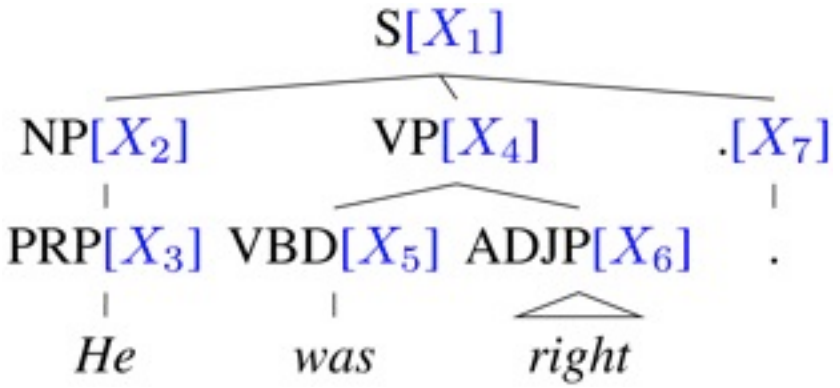


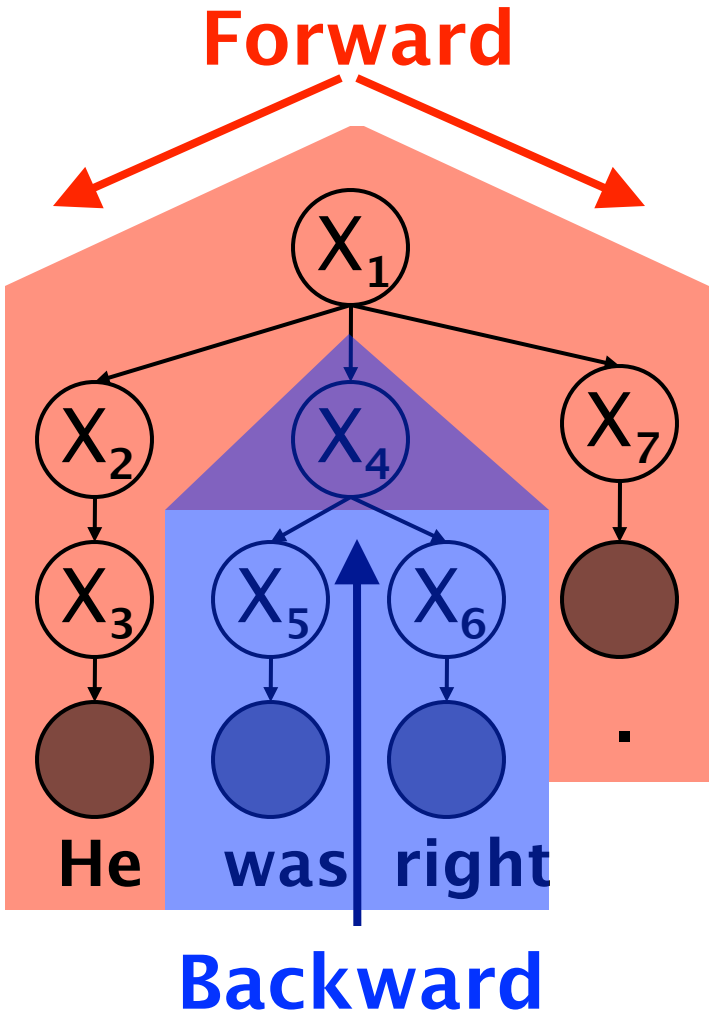


Learning Latent Annotations

EM algorithm:

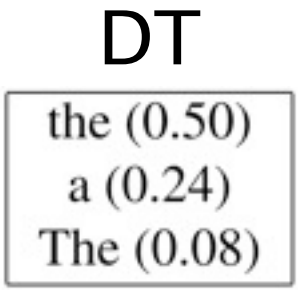
- Brackets are known
- Base categories are known
- Only induce subcategories



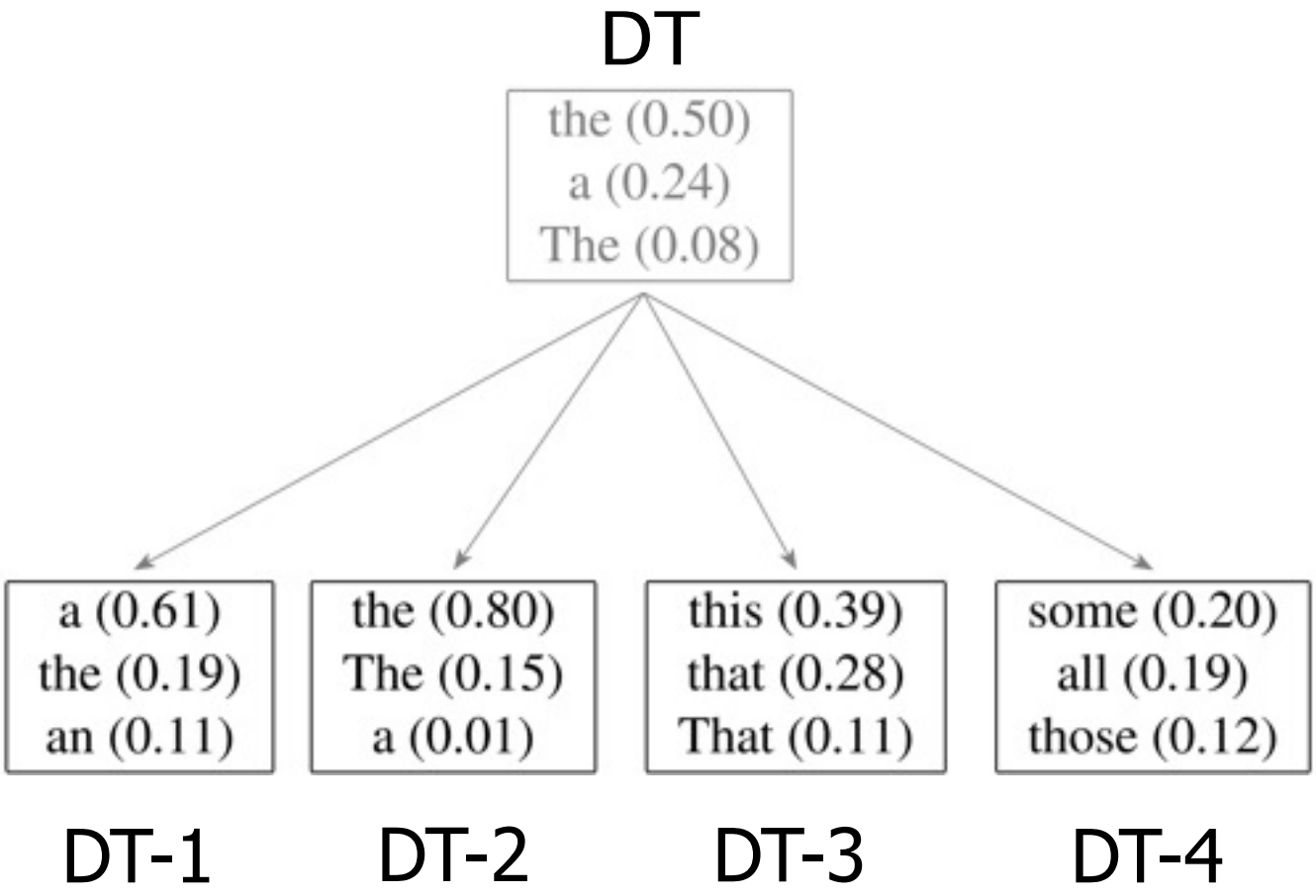


Just like Forward-Backward for HMMs.

Refinement of the DT tag

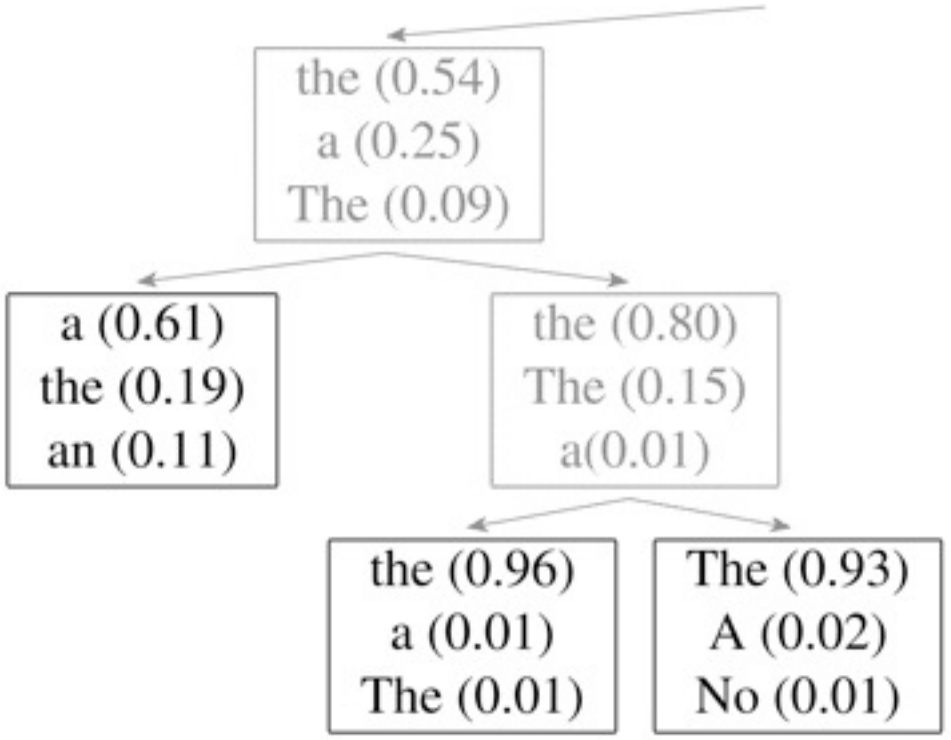


Refinement of the DT tag



Adaptive Splitting

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



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

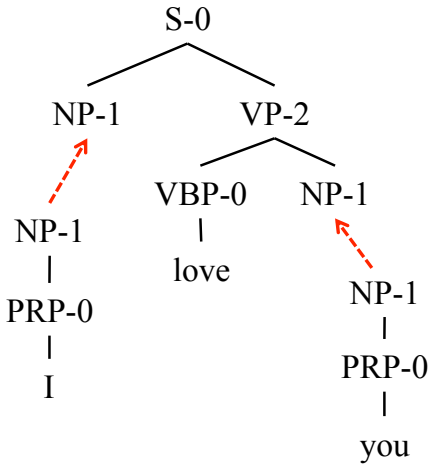
- Cardinal Numbers (CD):

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

Bayesian Symbol Refined TSG

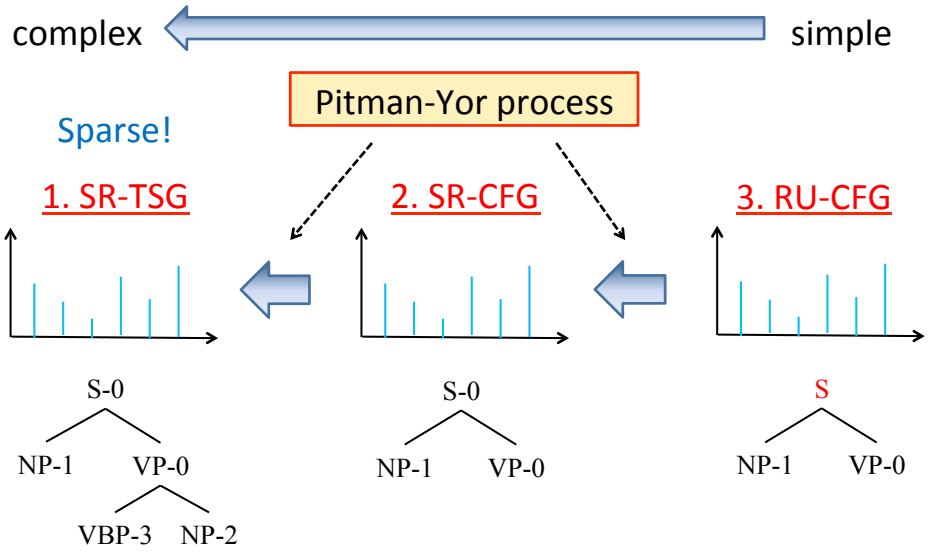
[Shindo et al. '12]

- Latent Variable Tree-Substitution Grammar



Hierarchical Generation Process:

- Joint model of everything
- Complex sampling scheme needed



Spectral Learning for PCFGs

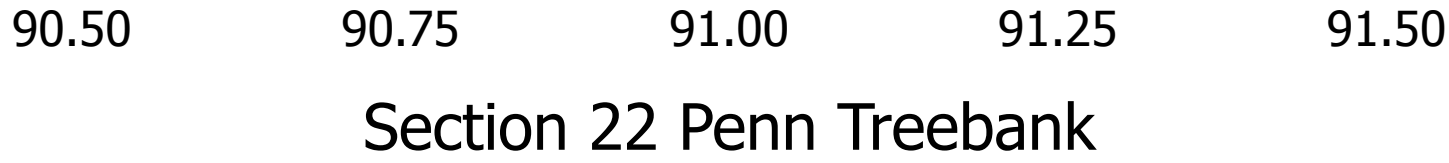
[Luque et al. '12, Cohen et al. '12]

- EM is a local method
  - Can never be sure to have the global optimum
  - Significant variance between different runs

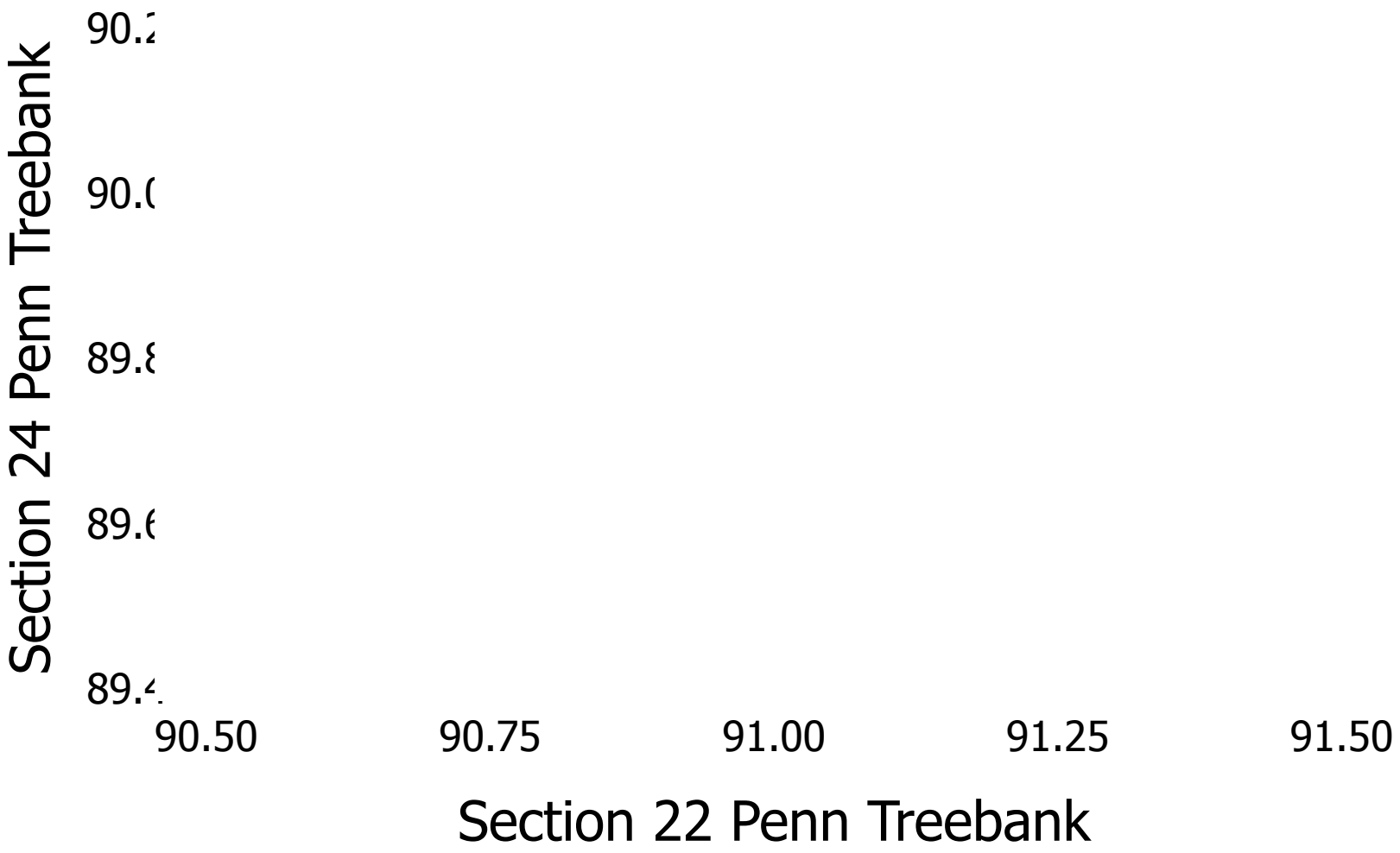
- Spectral methods
  - Provably find the global optimum
  - Compute SVD of training data
  - Efficient to run
  - But currently not competitive in practice

Parser Evaluation

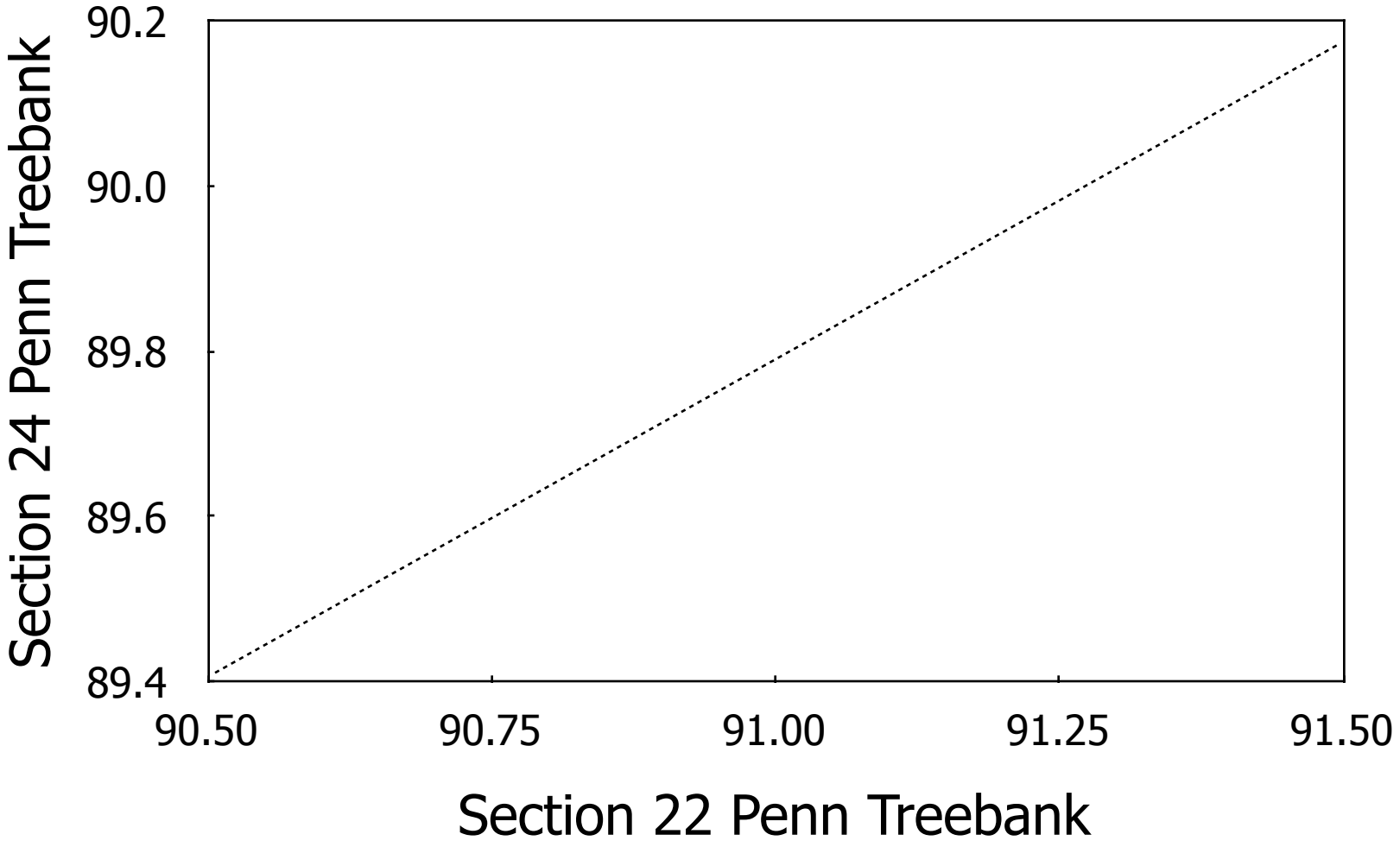
Parser Evaluation



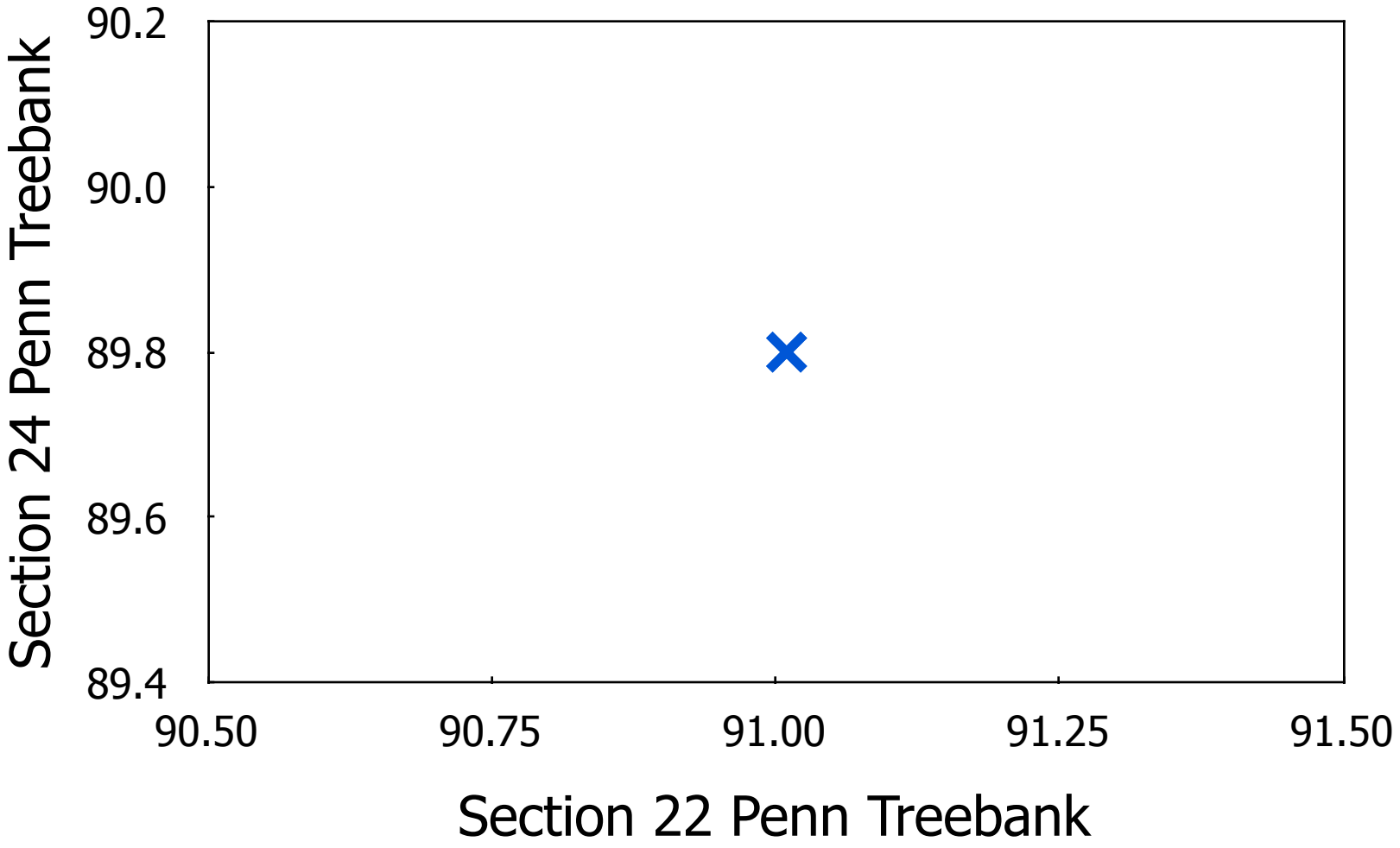
Parser Evaluation



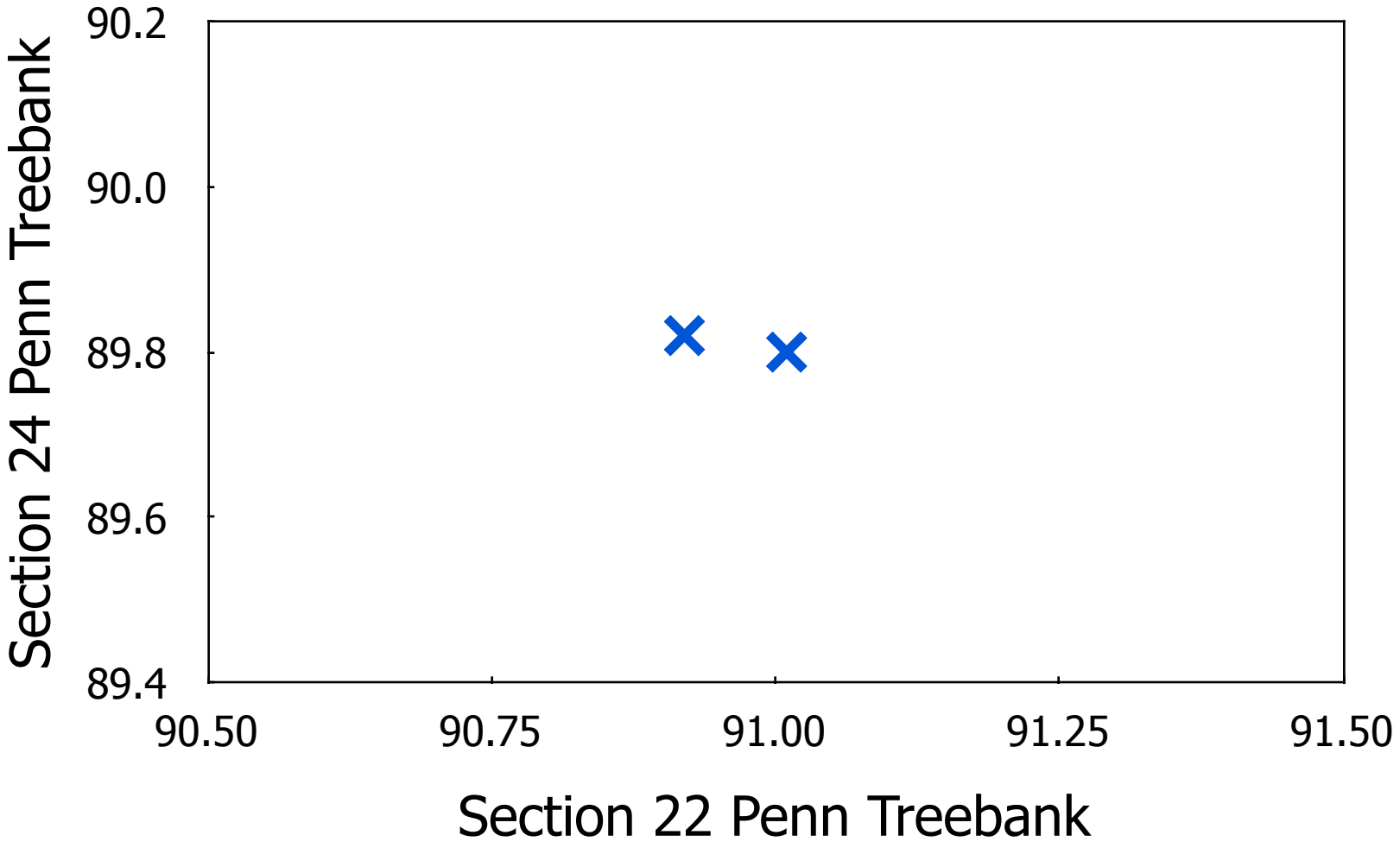
Parser Evaluation



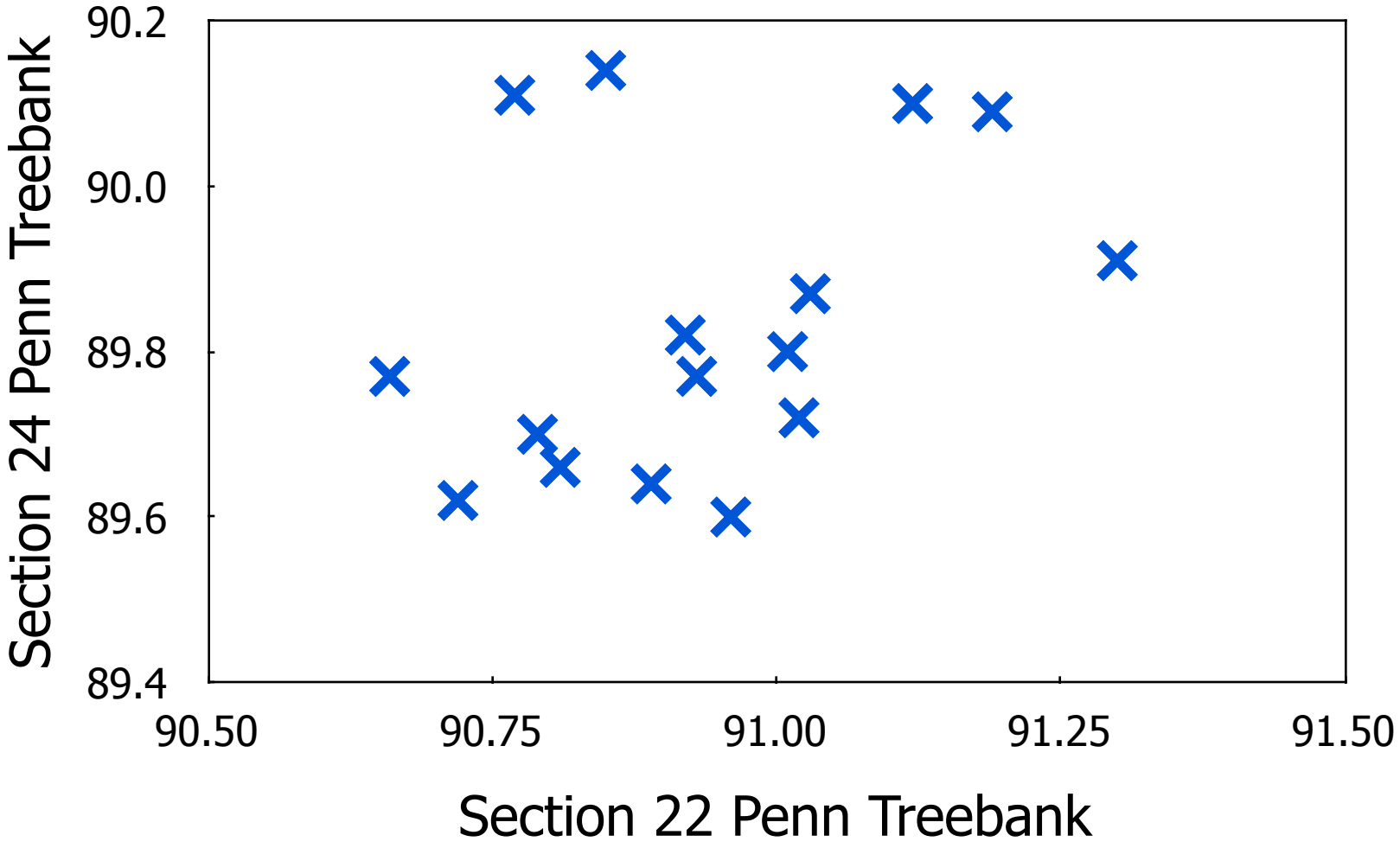
Parser Evaluation



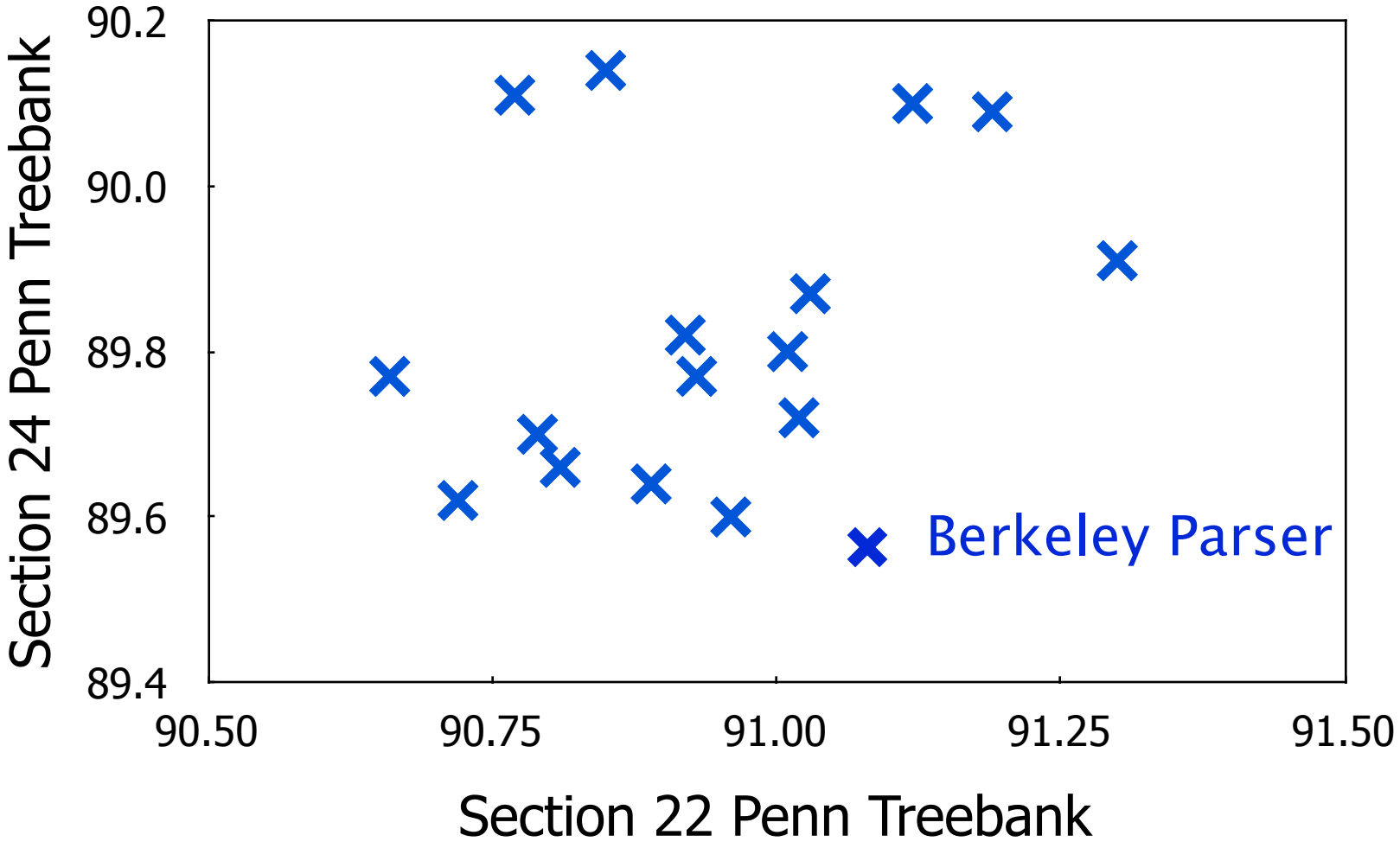
Parser Evaluation



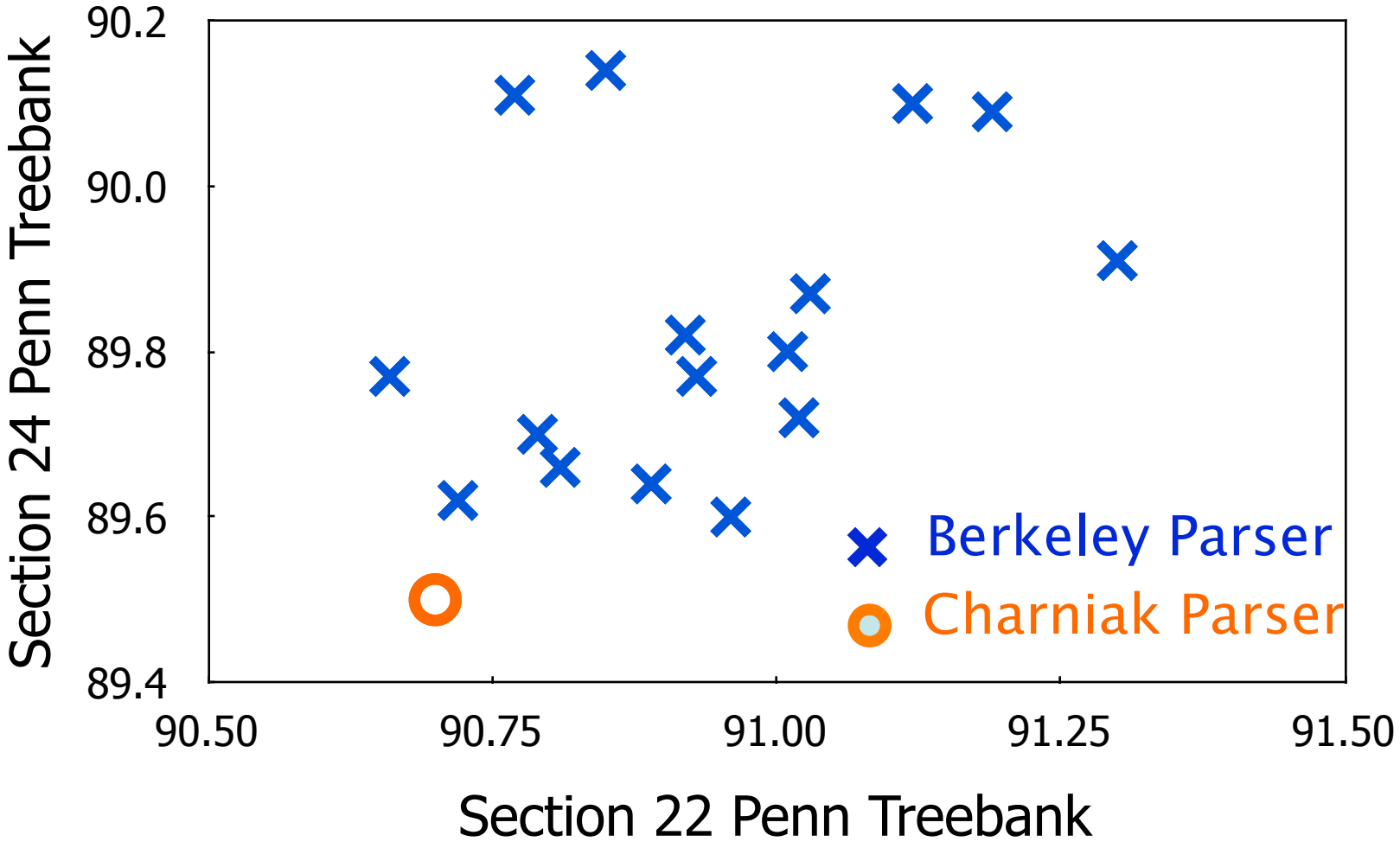
Parser Evaluation



Parser Evaluation

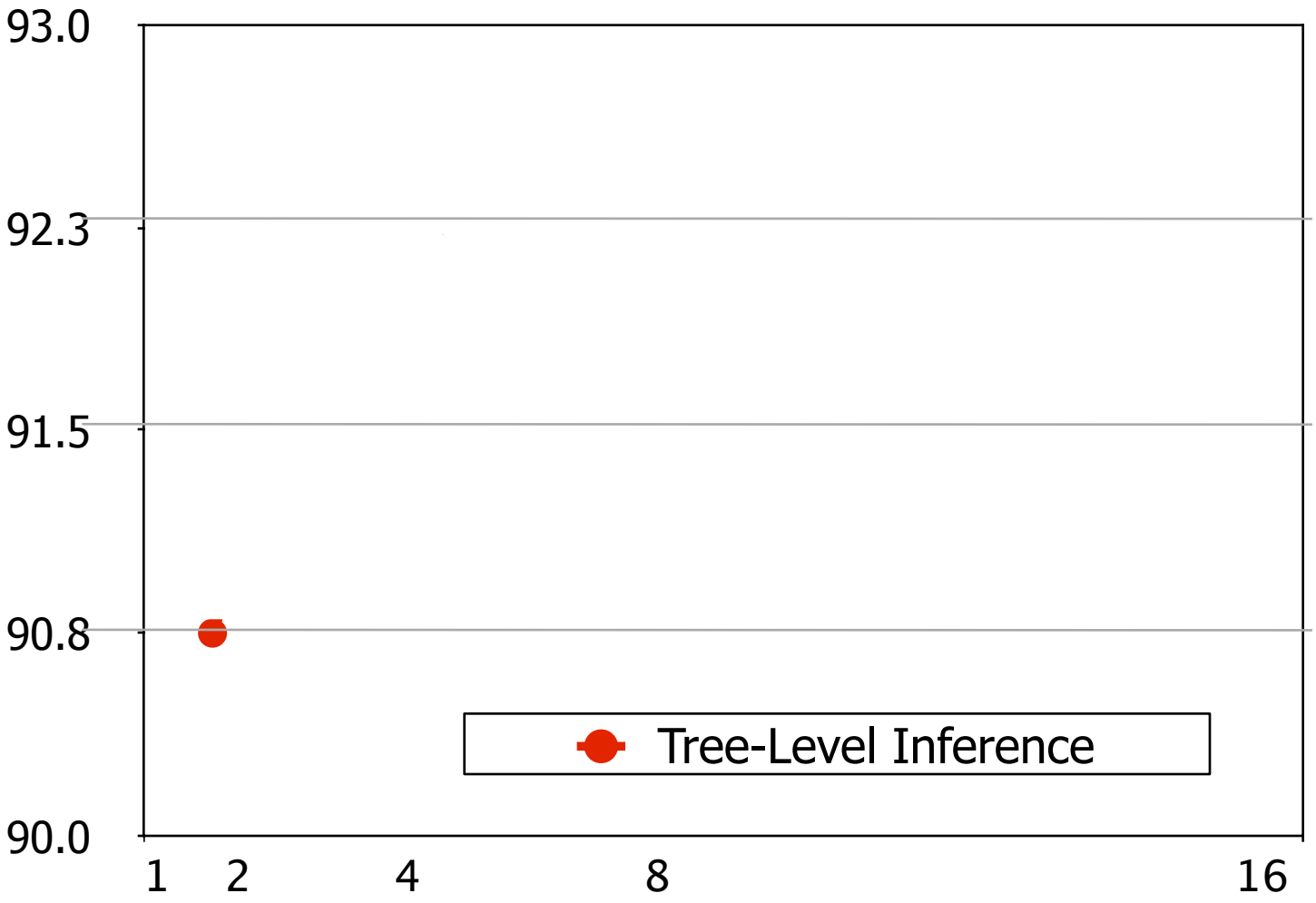


Parser Evaluation



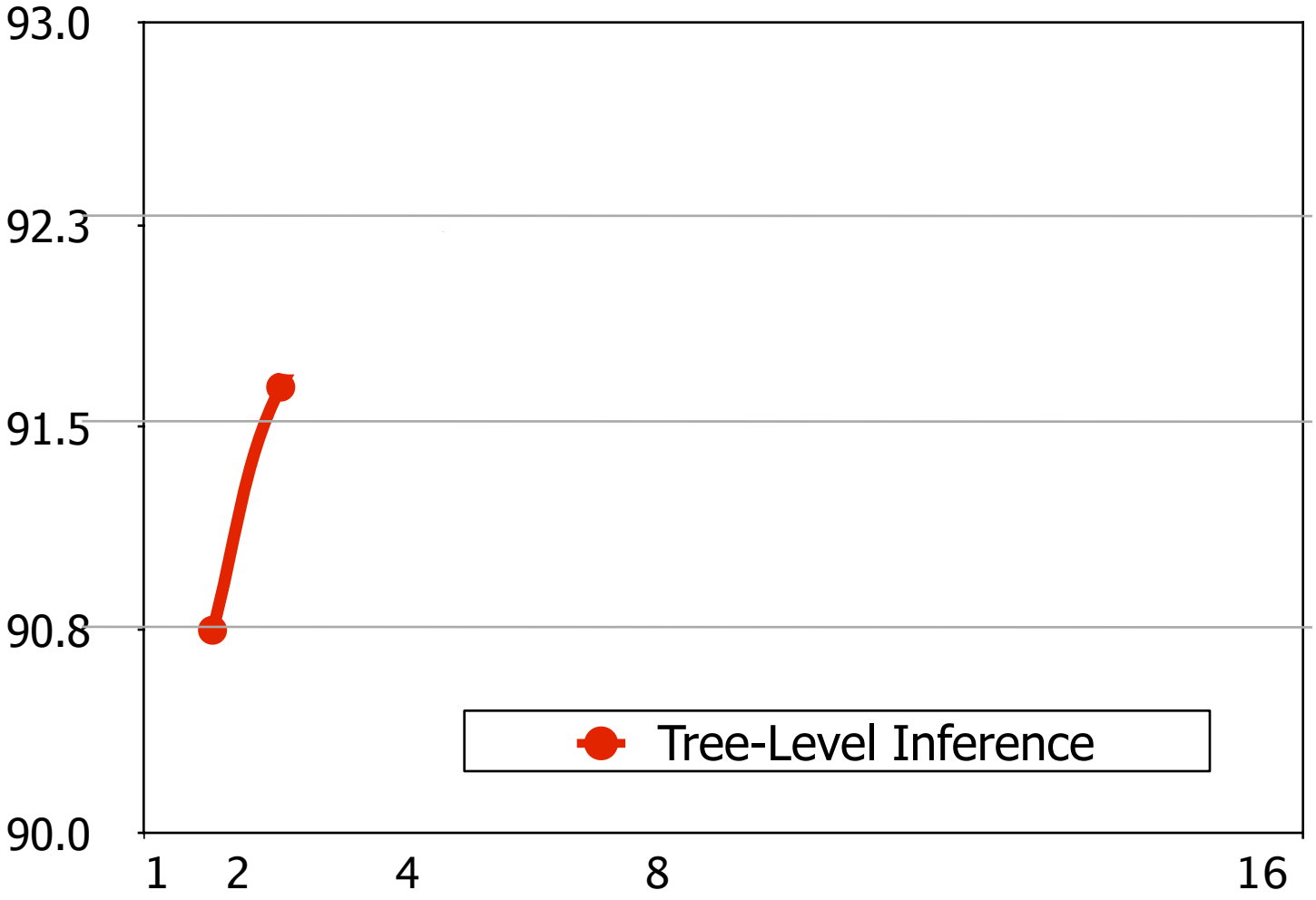
Product of Latent Variable Grammars

[Petrov ’10]



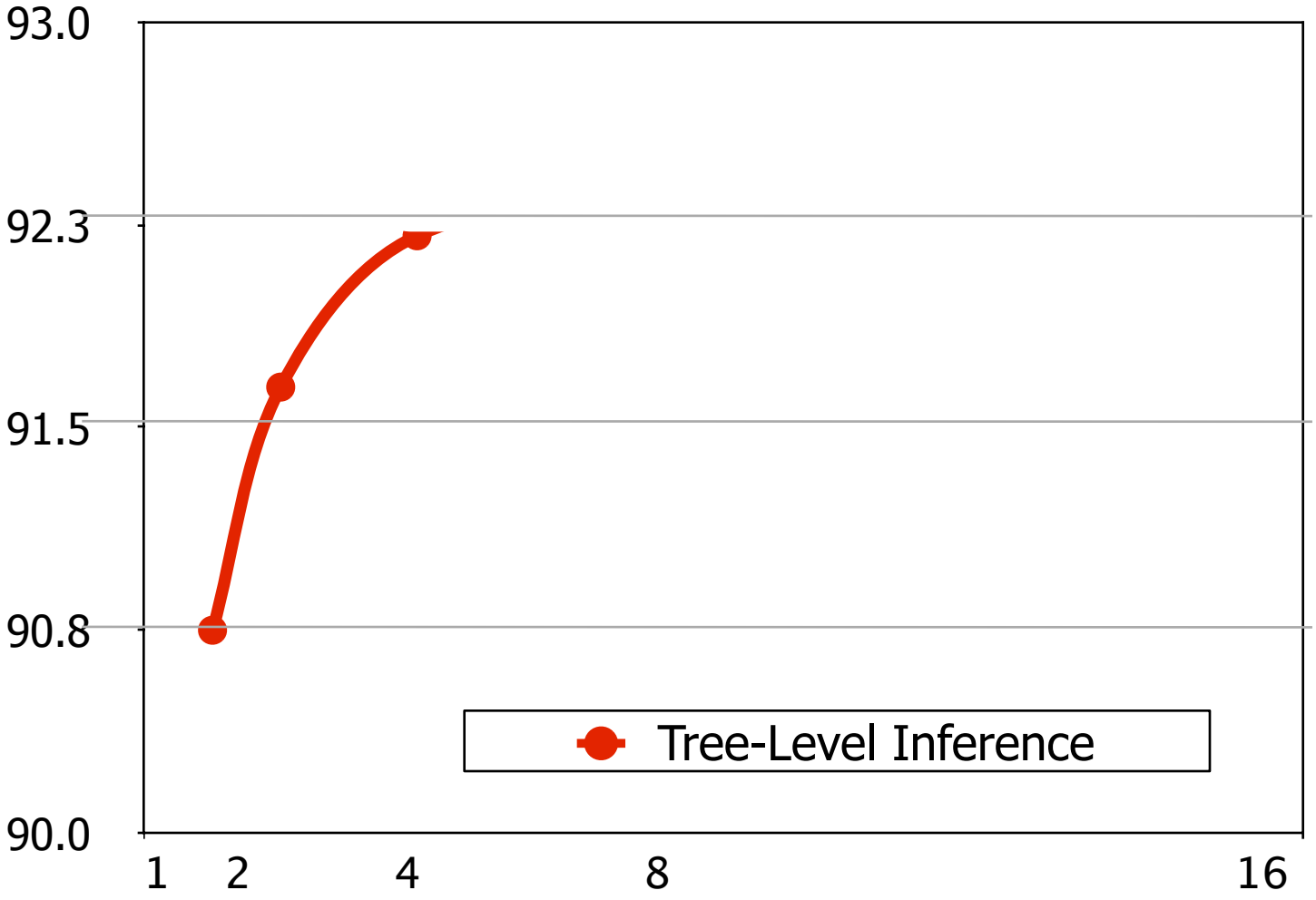
Product of Latent Variable Grammars

[Petrov '10]



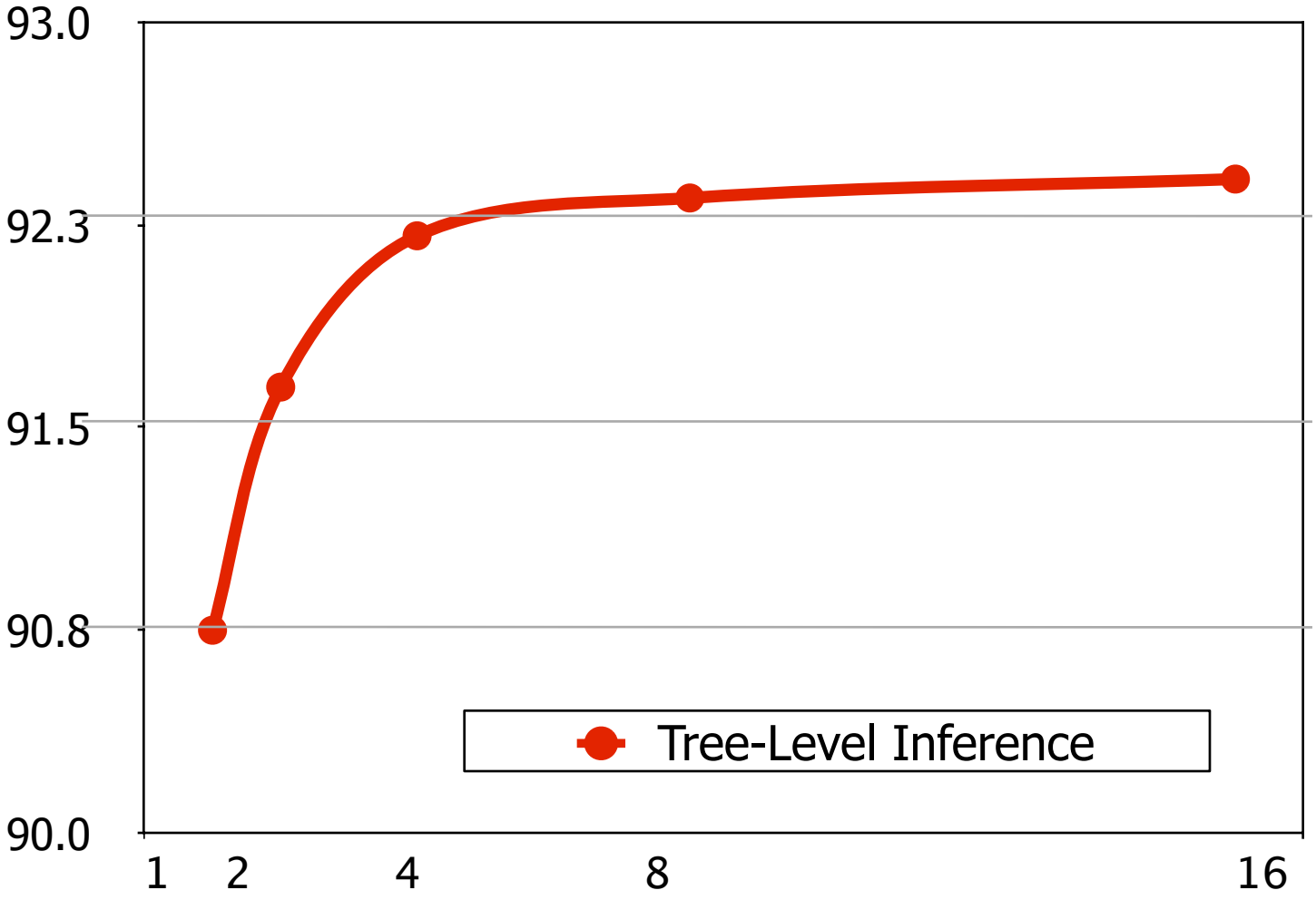
Product of Latent Variable Grammars

[Petrov '10]



Product of Latent Variable Grammars

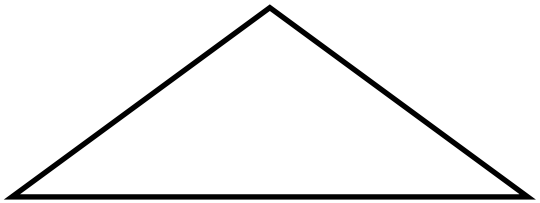
[Petrov ’10]



Coarse-to-Fine Inference

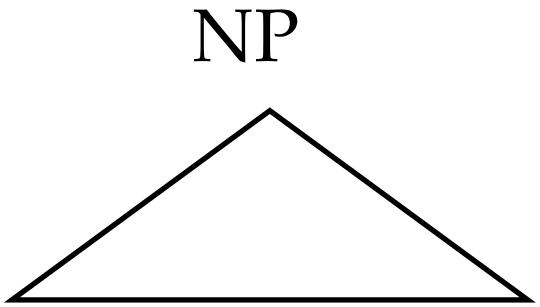
They solved the problem with statistics

Coarse-to-Fine Inference



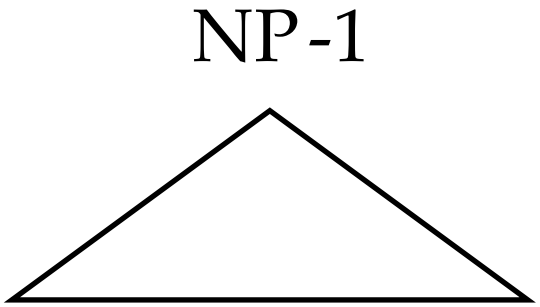
They solved the problem with statistics

Coarse-to-Fine Inference



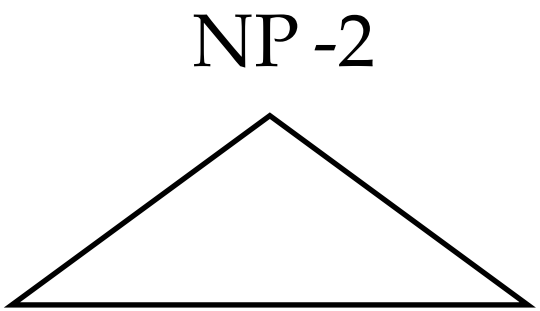
They solved the problem with statistics

Coarse-to-Fine Inference



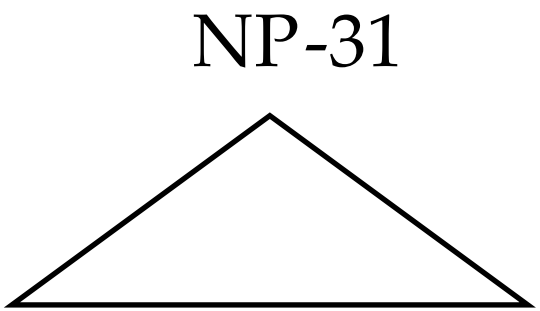
They solved the problem with statistics

Coarse-to-Fine Inference



They solved the problem with statistics

Coarse-to-Fine Inference

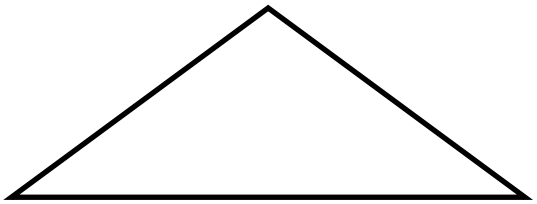


They solved the problem with statistics

Coarse-to-Fine Inference

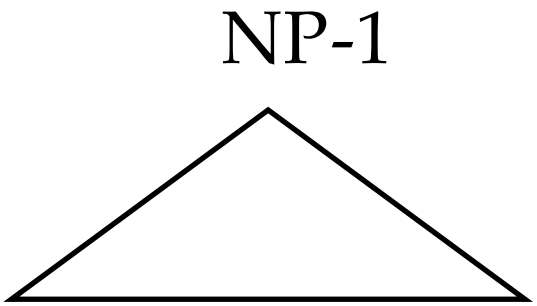
They solved the problem with statistics

Coarse-to-Fine Inference



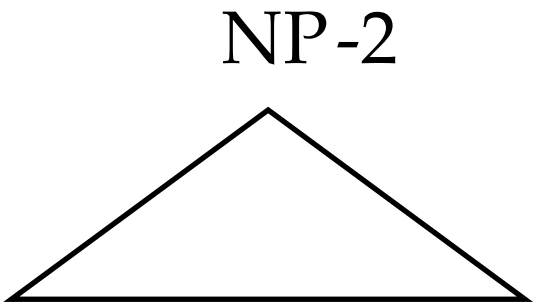
They solved the problem with statistics

Coarse-to-Fine Inference



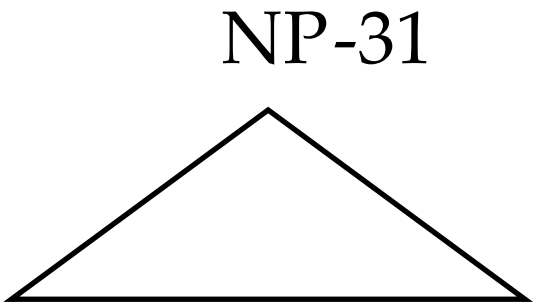
They solved the problem with statistics

Coarse-to-Fine Inference



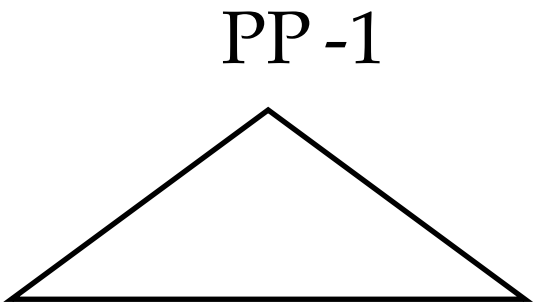
They solved the problem with statistics

Coarse-to-Fine Inference



They solved the problem with statistics

Coarse-to-Fine Inference

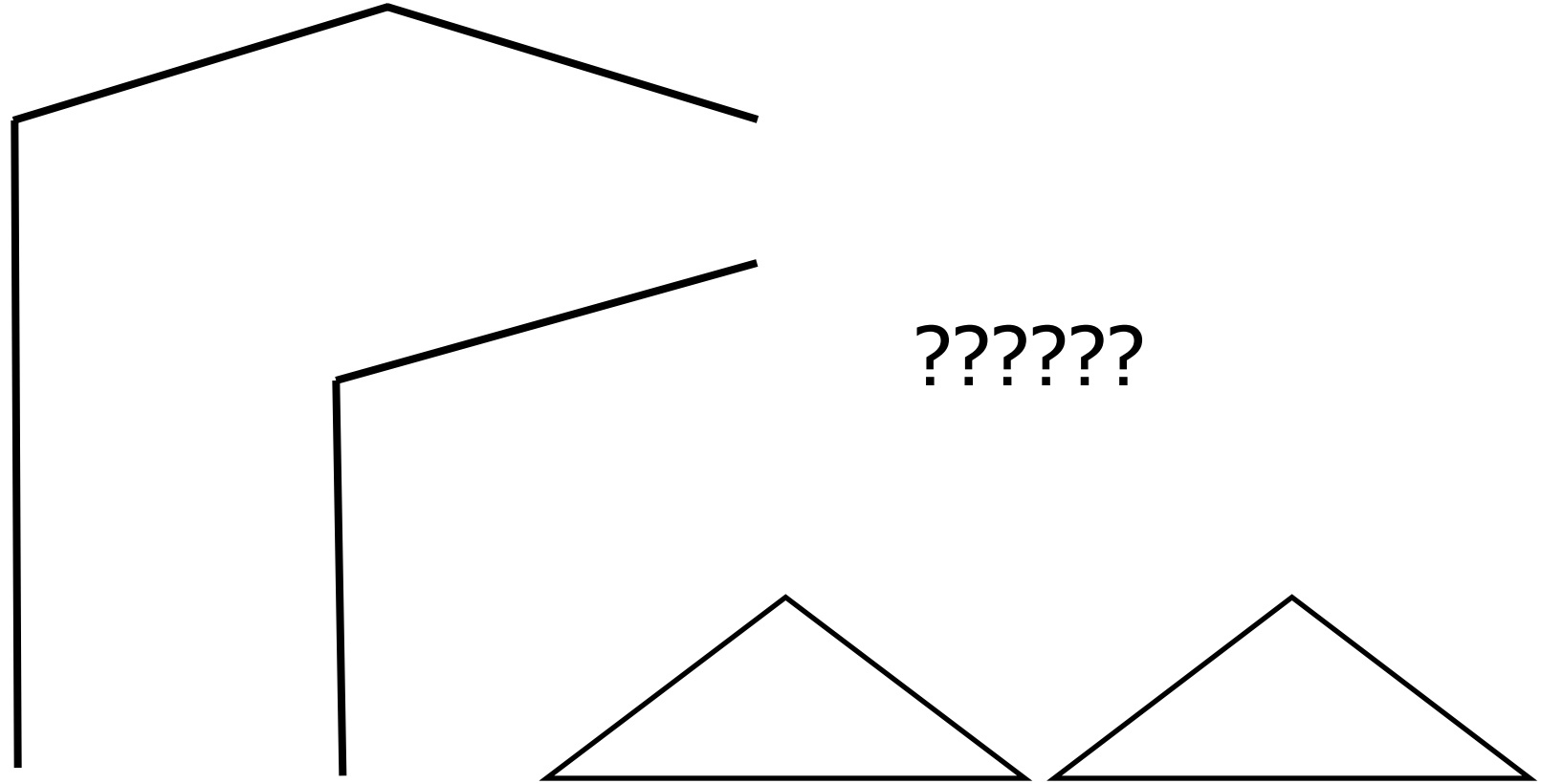


They solved the problem with statistics

Coarse-to-Fine Inference

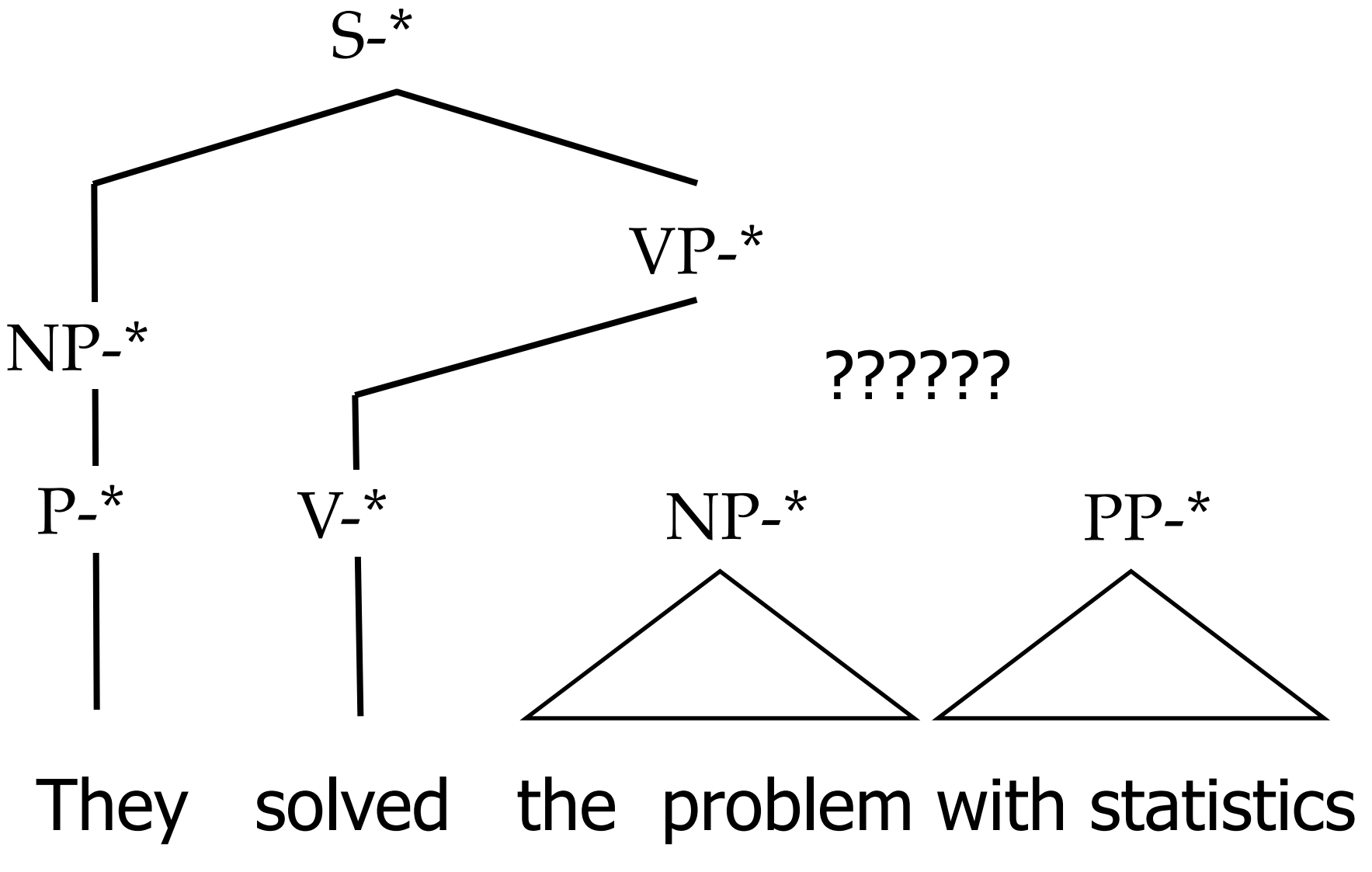
They solved the problem with statistics

Coarse-to-Fine Inference



They solved the problem with statistics

Coarse-to-Fine Inference



Prune?

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

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

E.g. consider the span 5 to 12:

coarse: … | QP | NP | VP | …

Prune?

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

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

E.g. consider the span 5 to 12:

coarse: … QP NP VP …

Prune?

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

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

E.g. consider the span 5 to 12:

coarse: … QP NP VP …

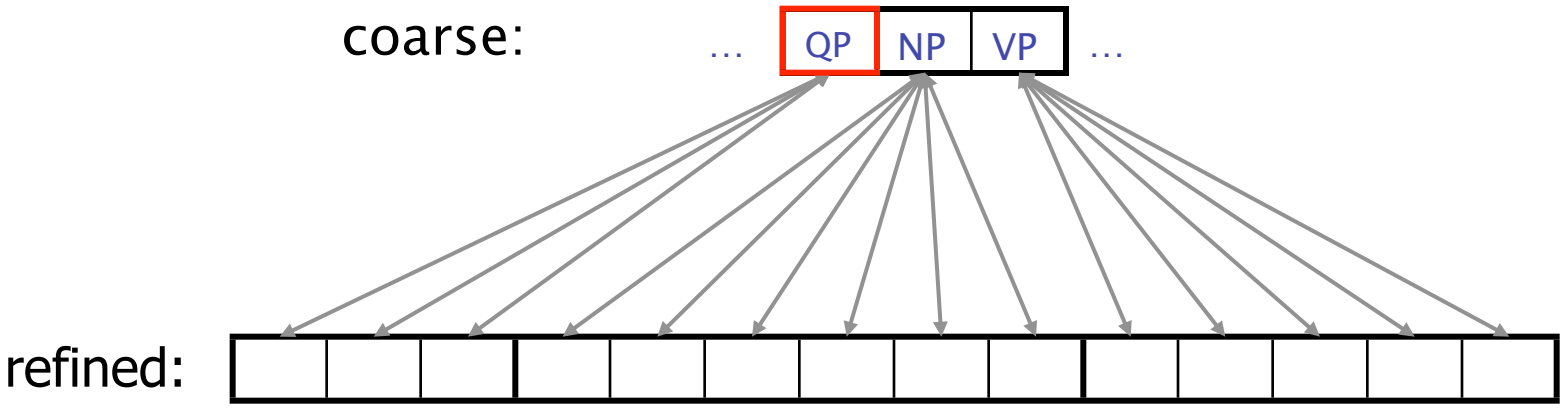
refined:

Prune?

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

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

E.g. consider the span 5 to 12:

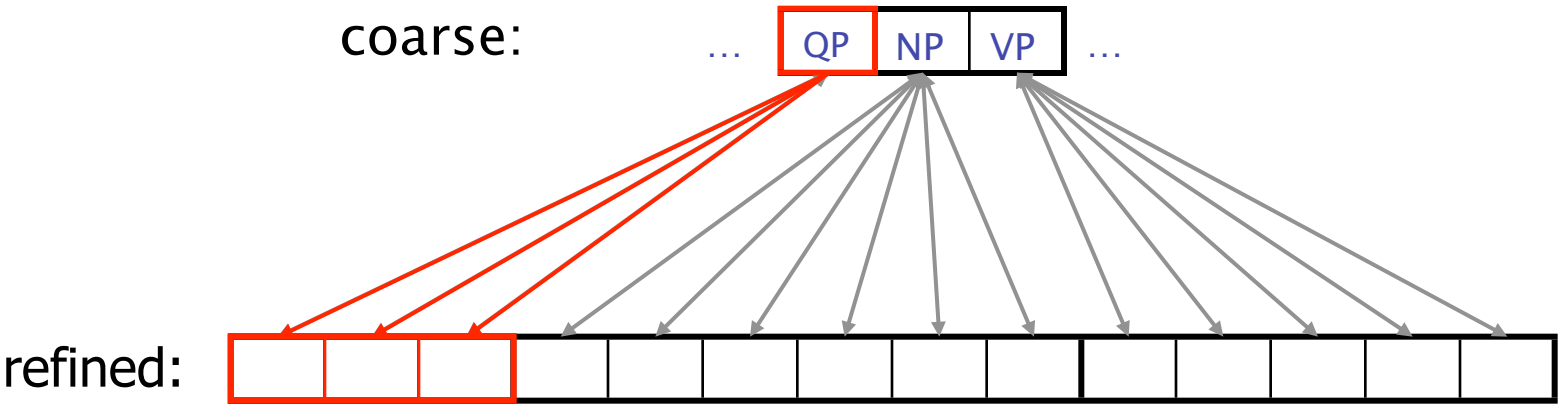


Prune?

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

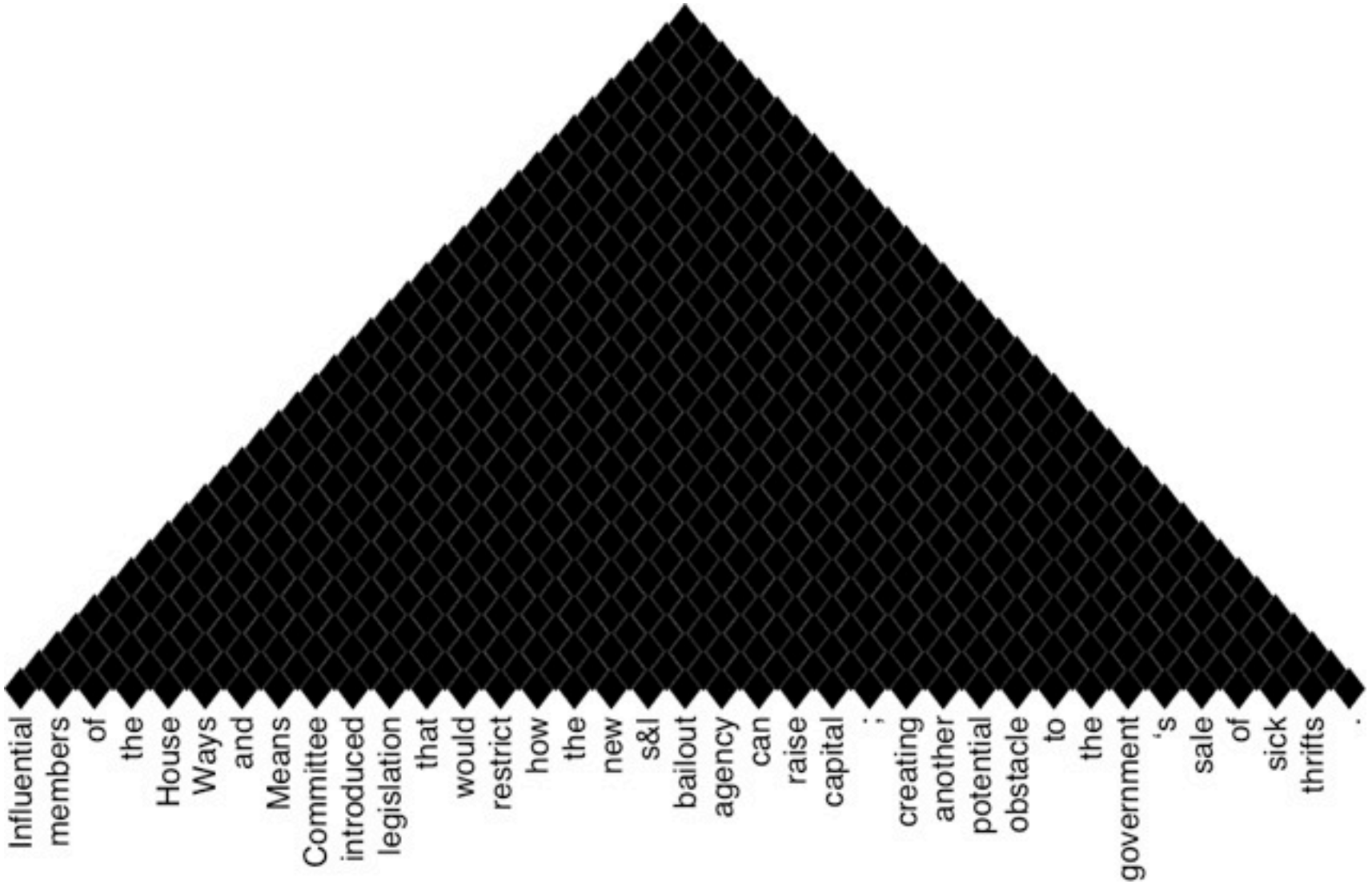
$$\frac{\mathrm{P_{IN}}(X,i,j)\cdot\mathrm{P_{OUT}}(X,i,j)}{\mathrm{P_{IN}}(root,0,n)} \quad \textbf{< threshold}$$

E.g. consider the span 5 to 12:

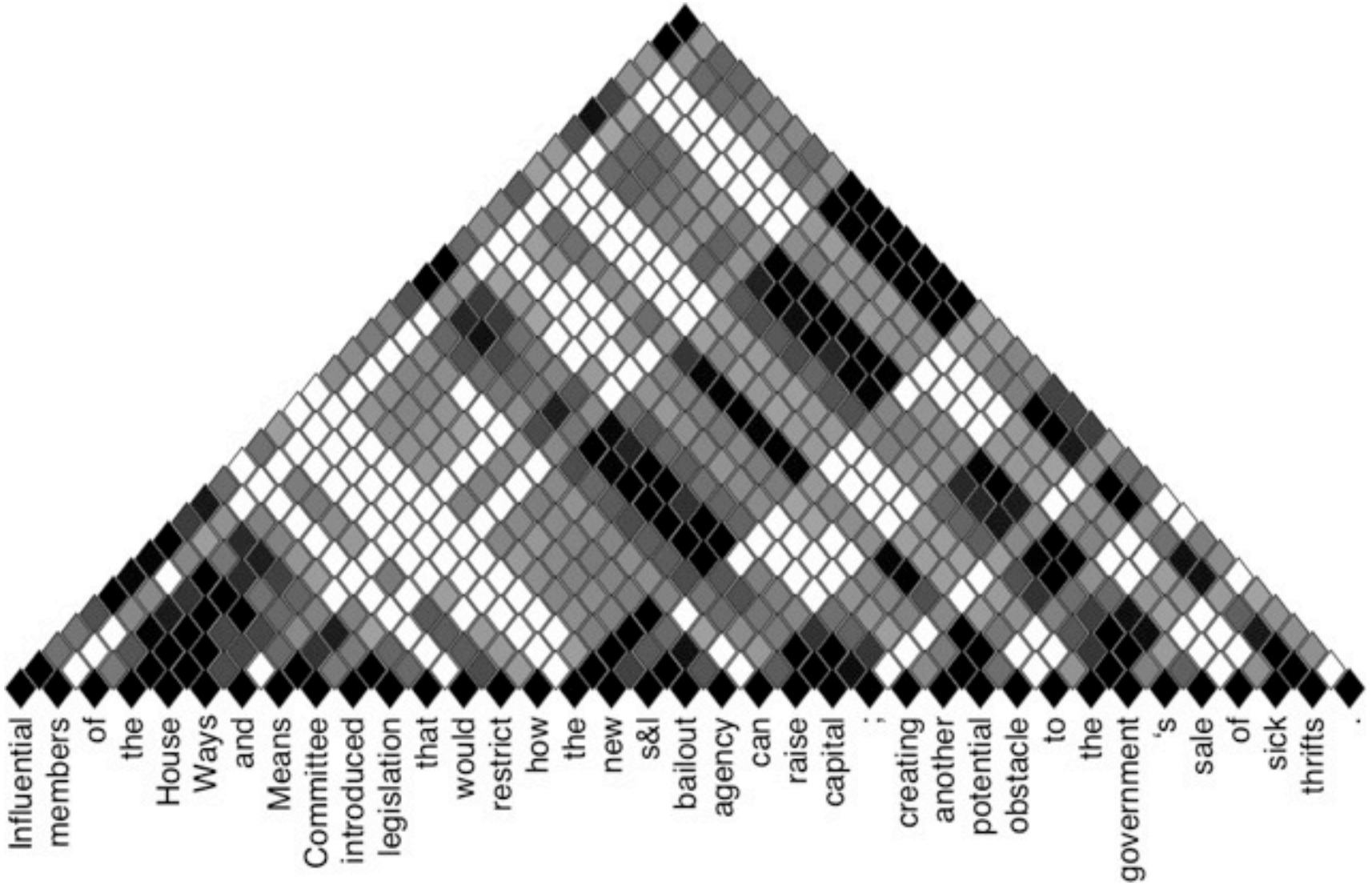


Bracket Posteriors

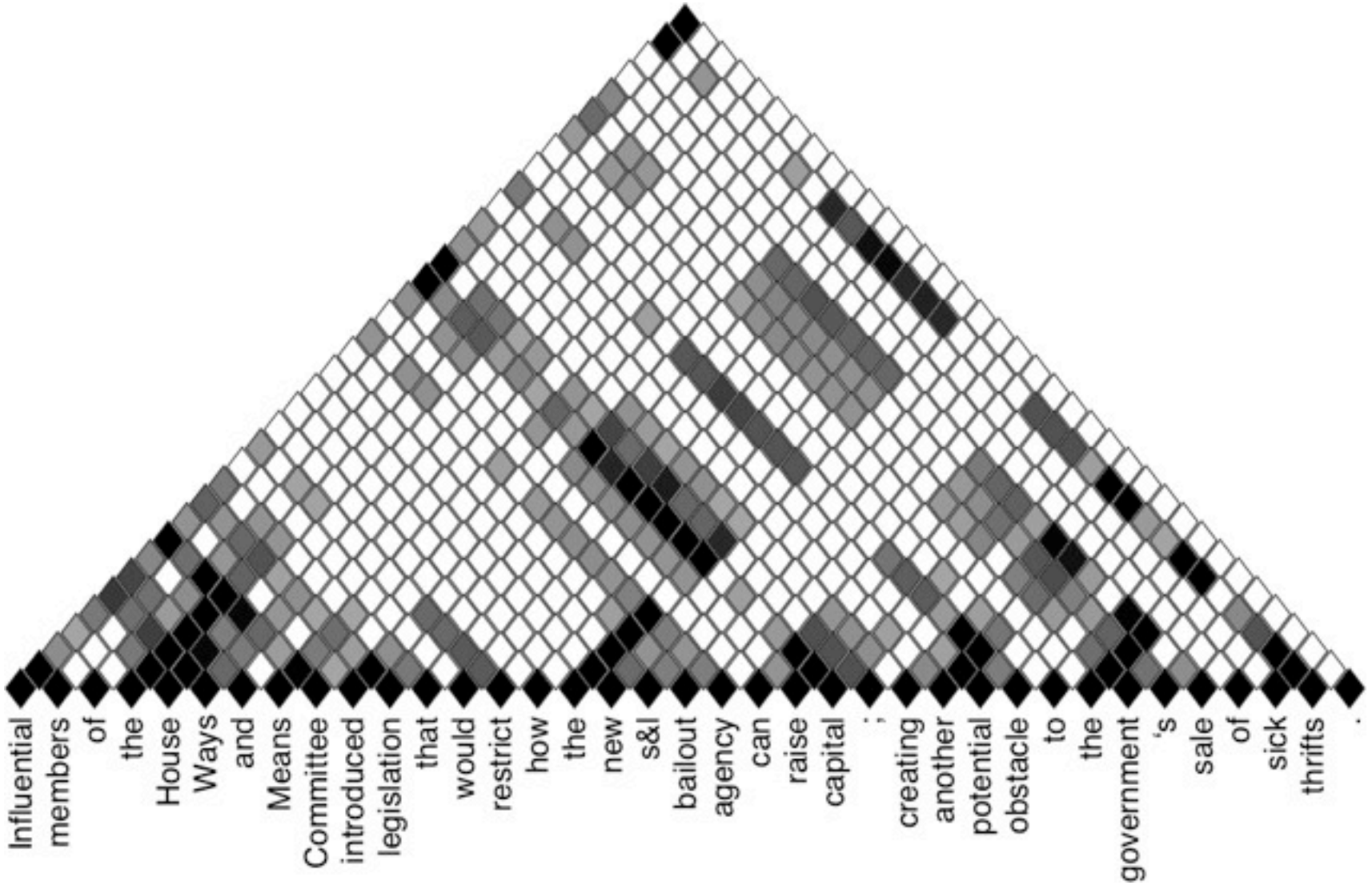
Bracket Posteriors



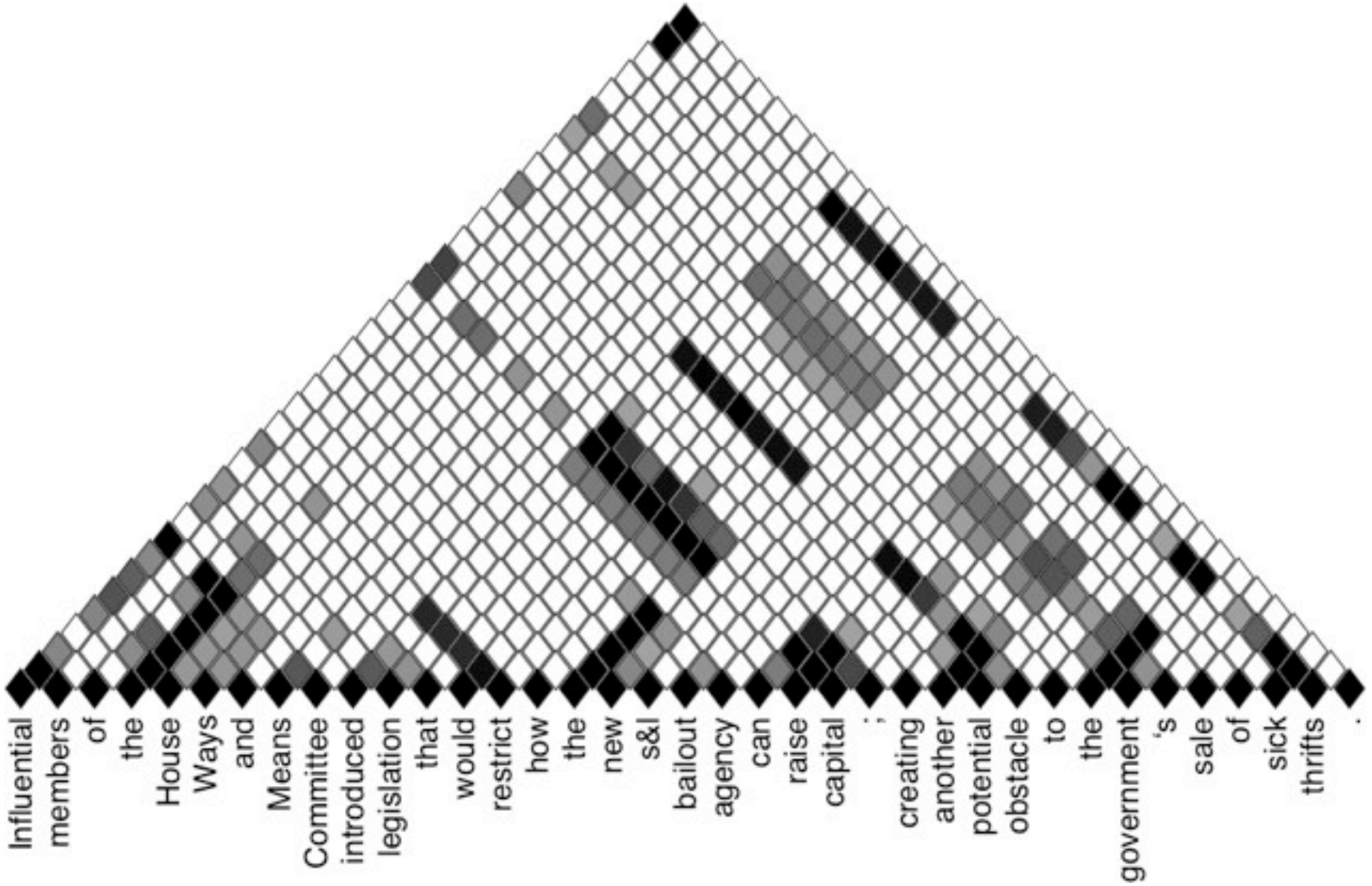
Bracket Posteriors



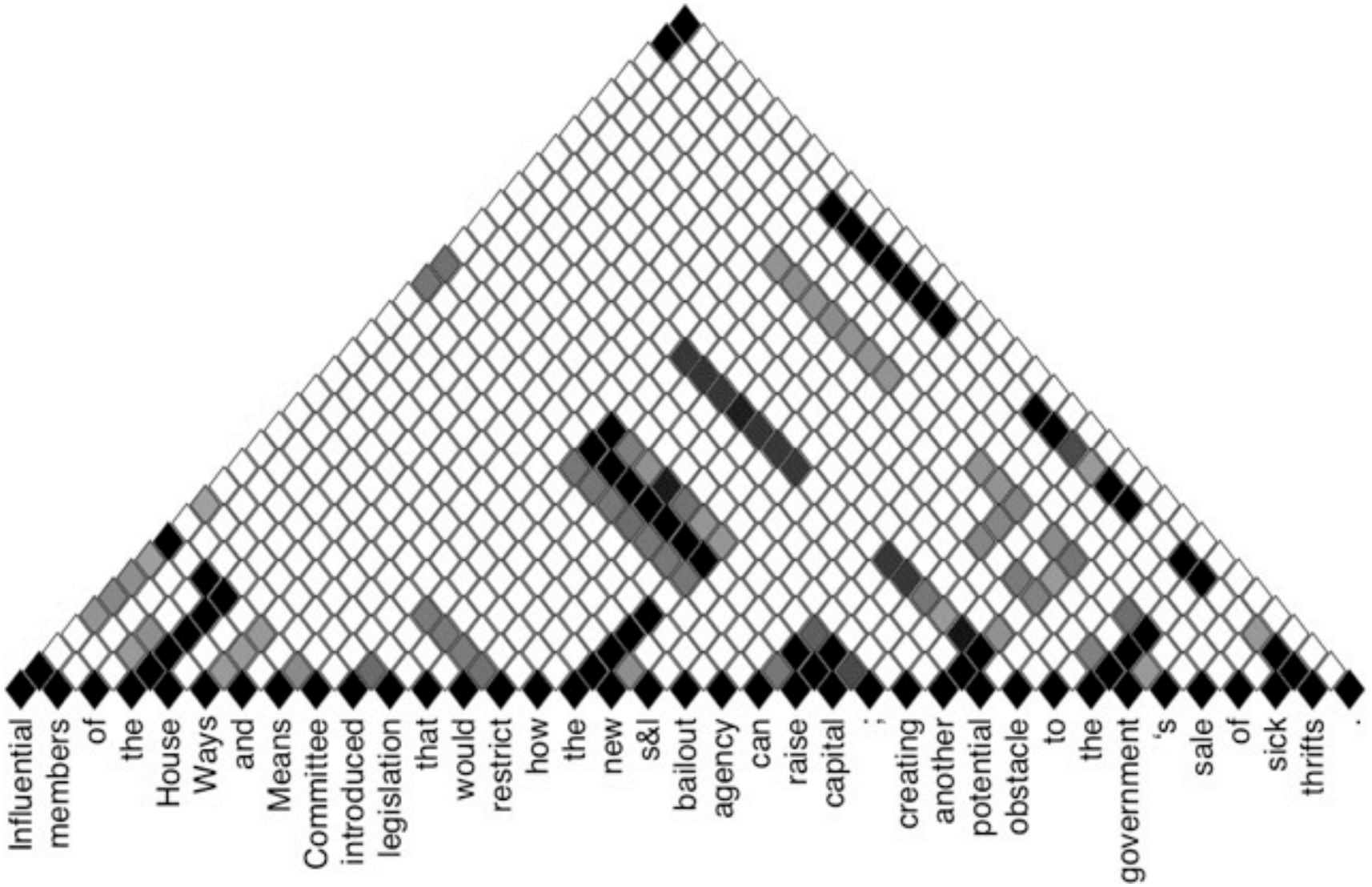
Bracket Posteriors



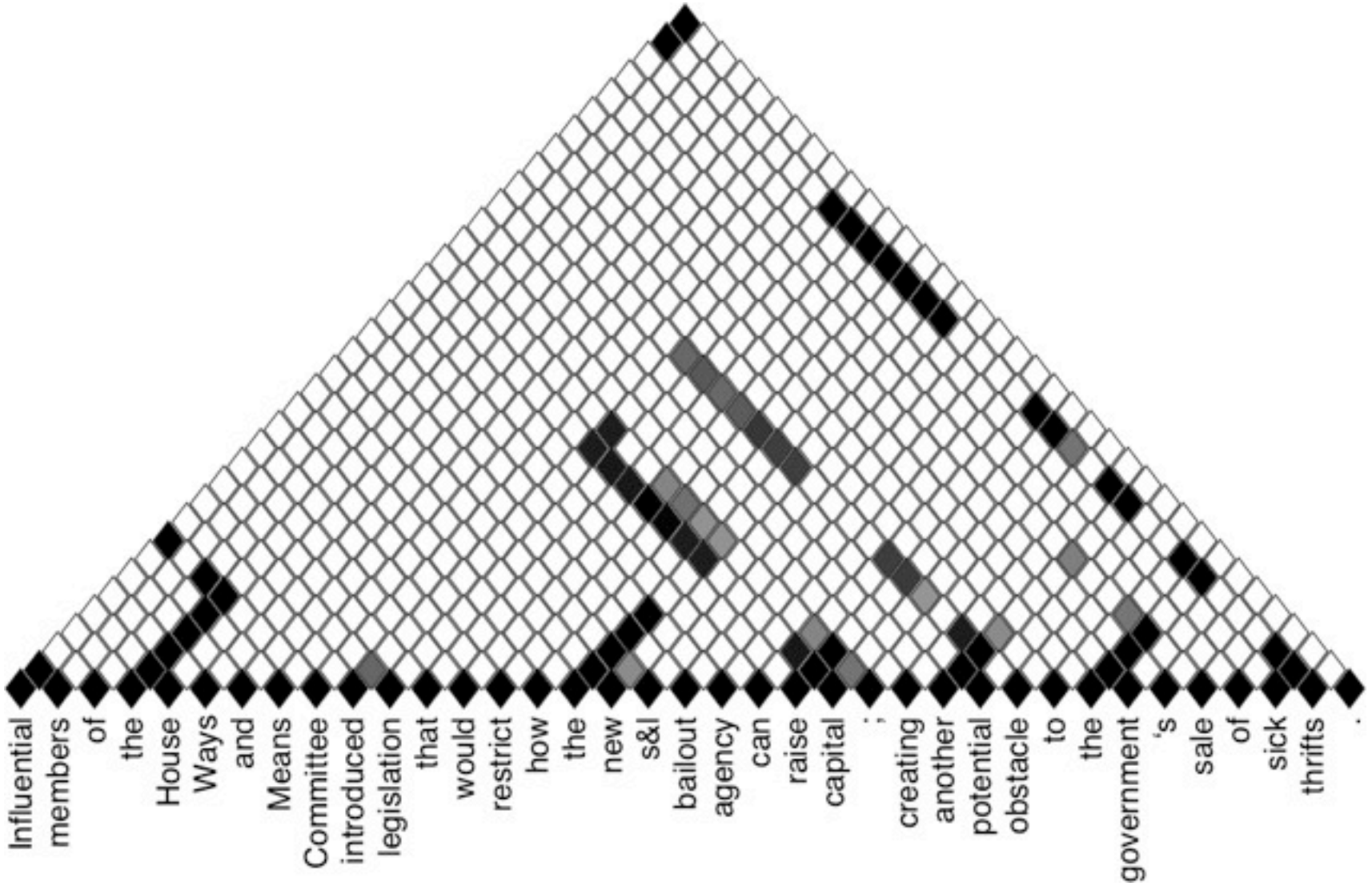
Bracket Posteriors



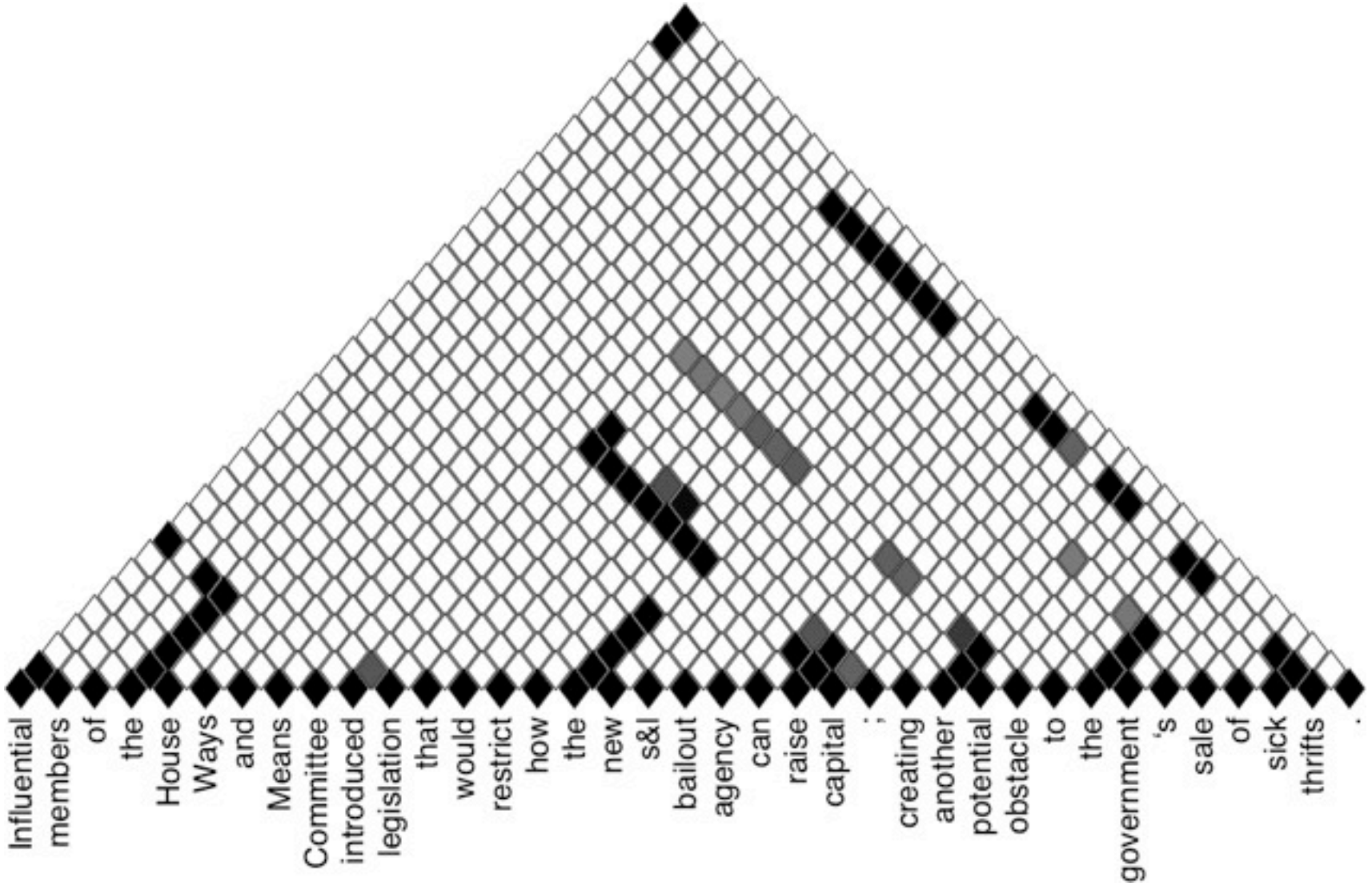
Bracket Posteriors



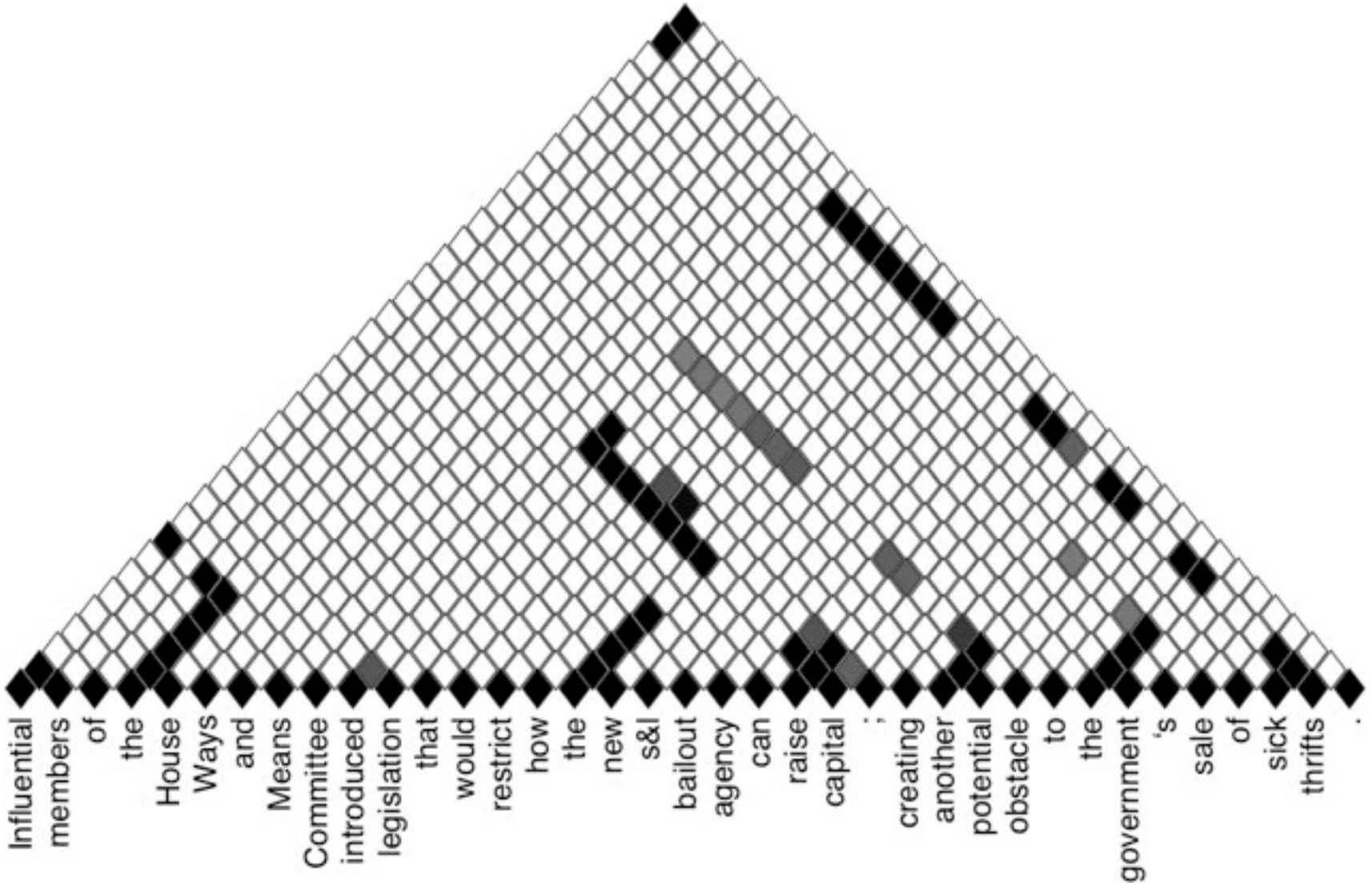
Bracket Posteriors



Bracket Posteriors



Bracket Posteriors



Hierarchical Pruning

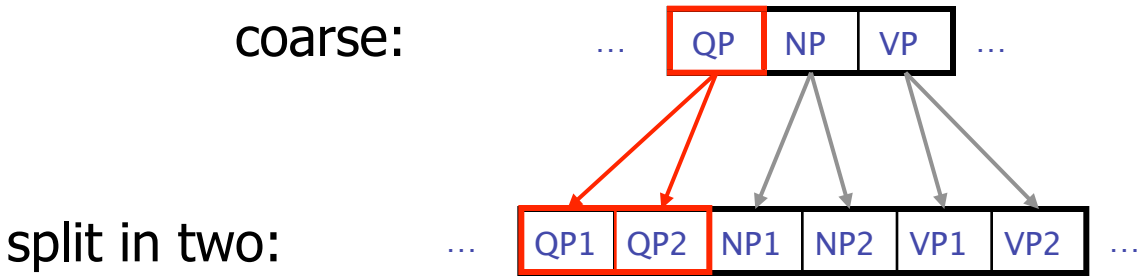
Hierarchical Pruning

coarse:

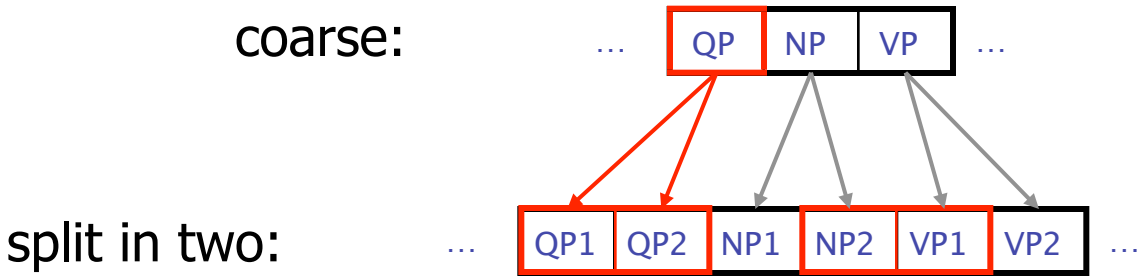
Hierarchical Pruning

coarse:

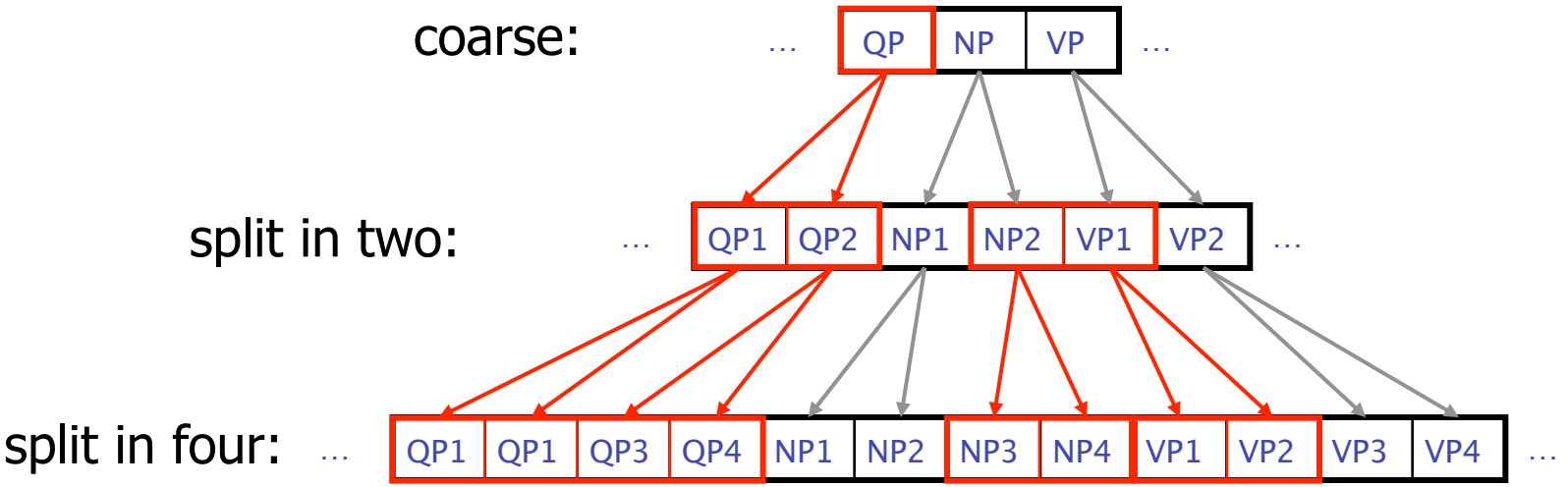
Hierarchical Pruning



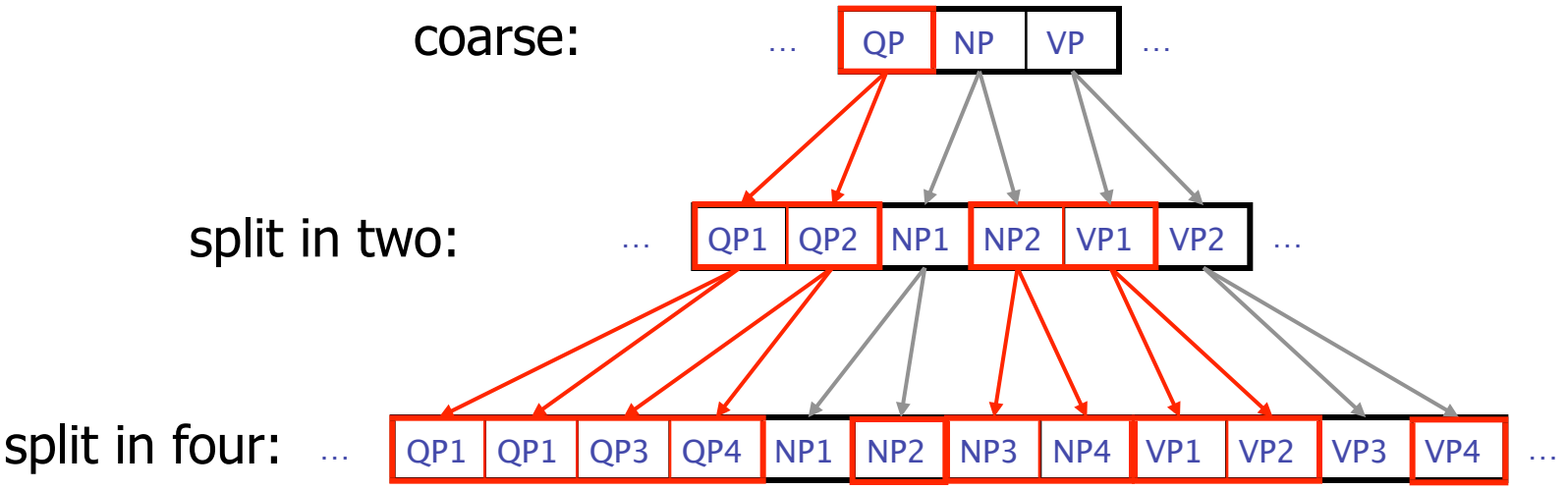
Hierarchical Pruning



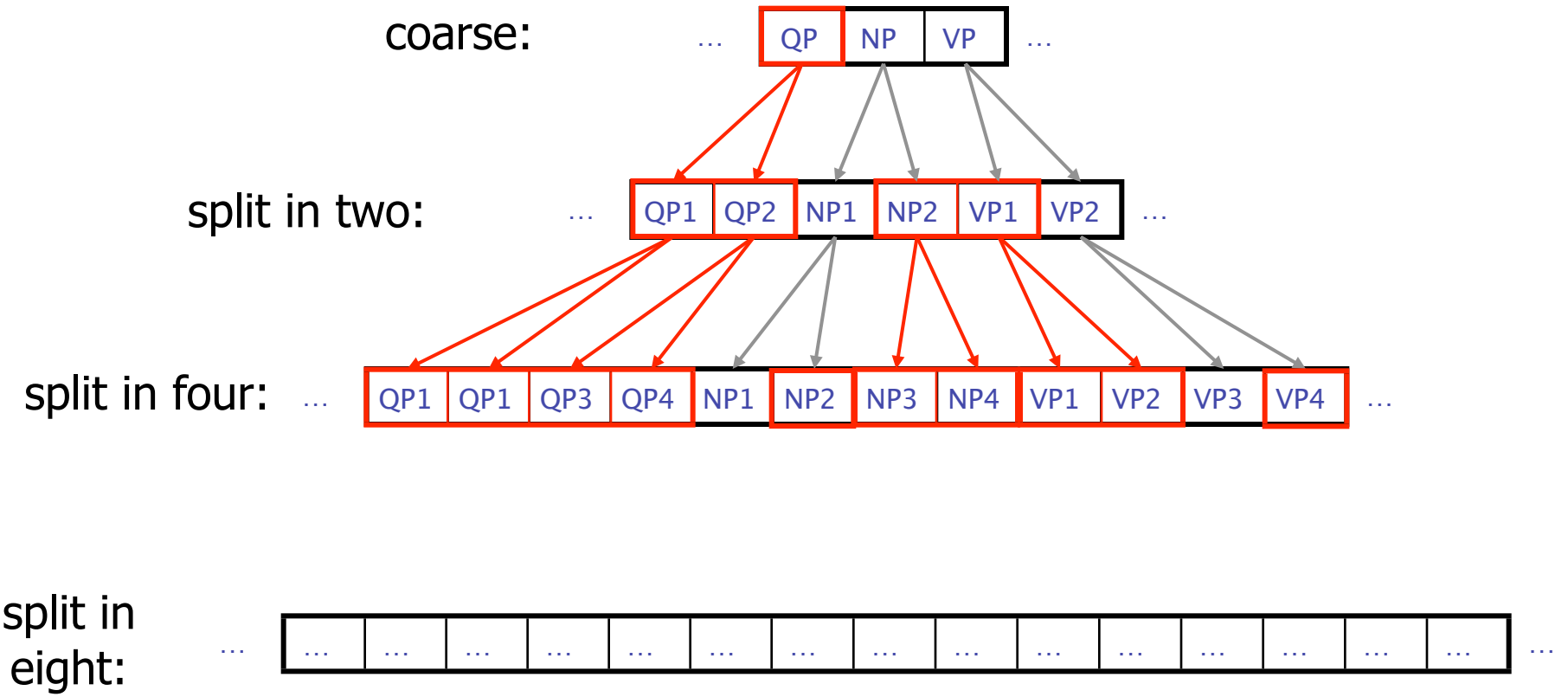
Hierarchical Pruning



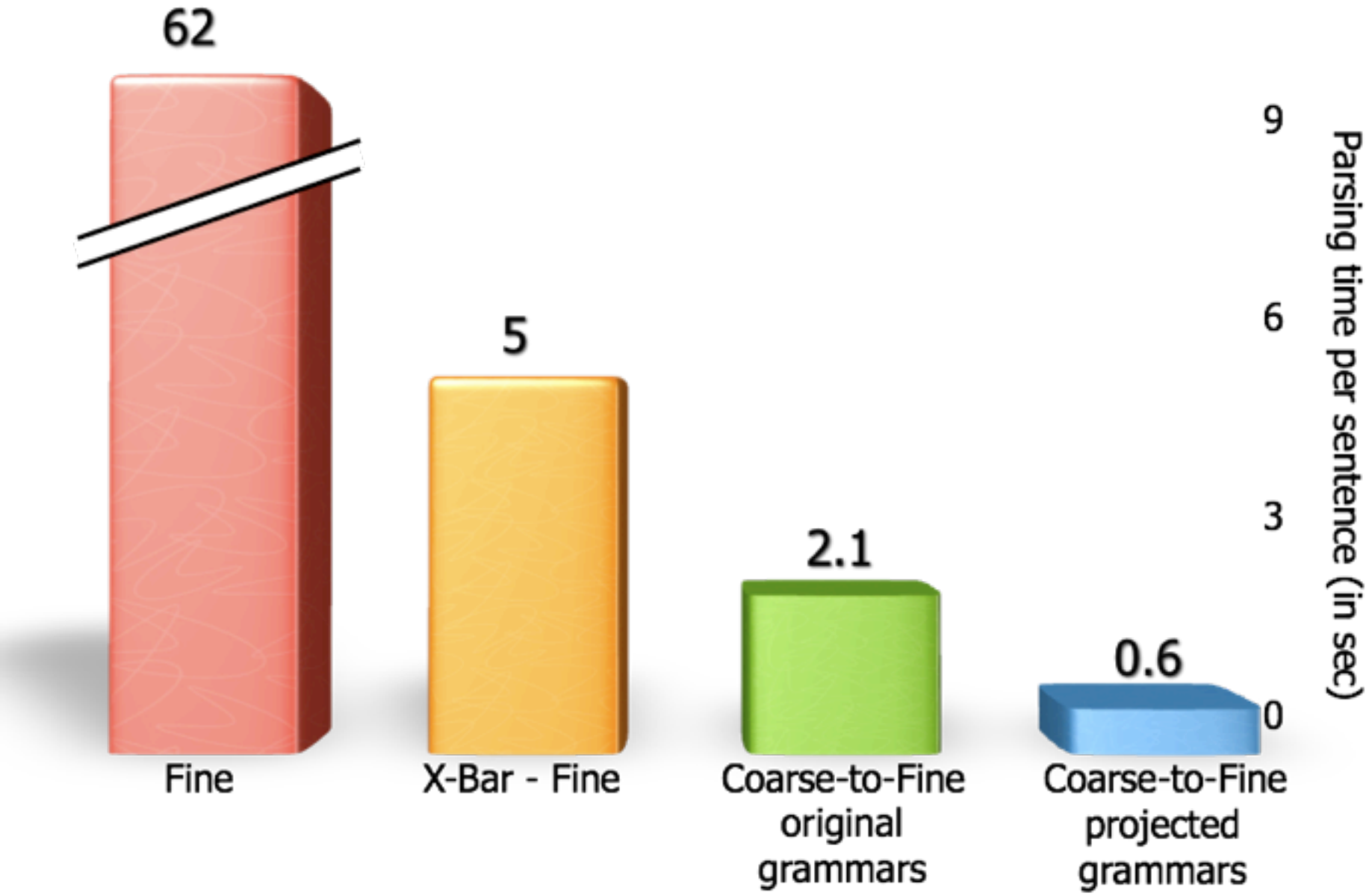
Hierarchical Pruning



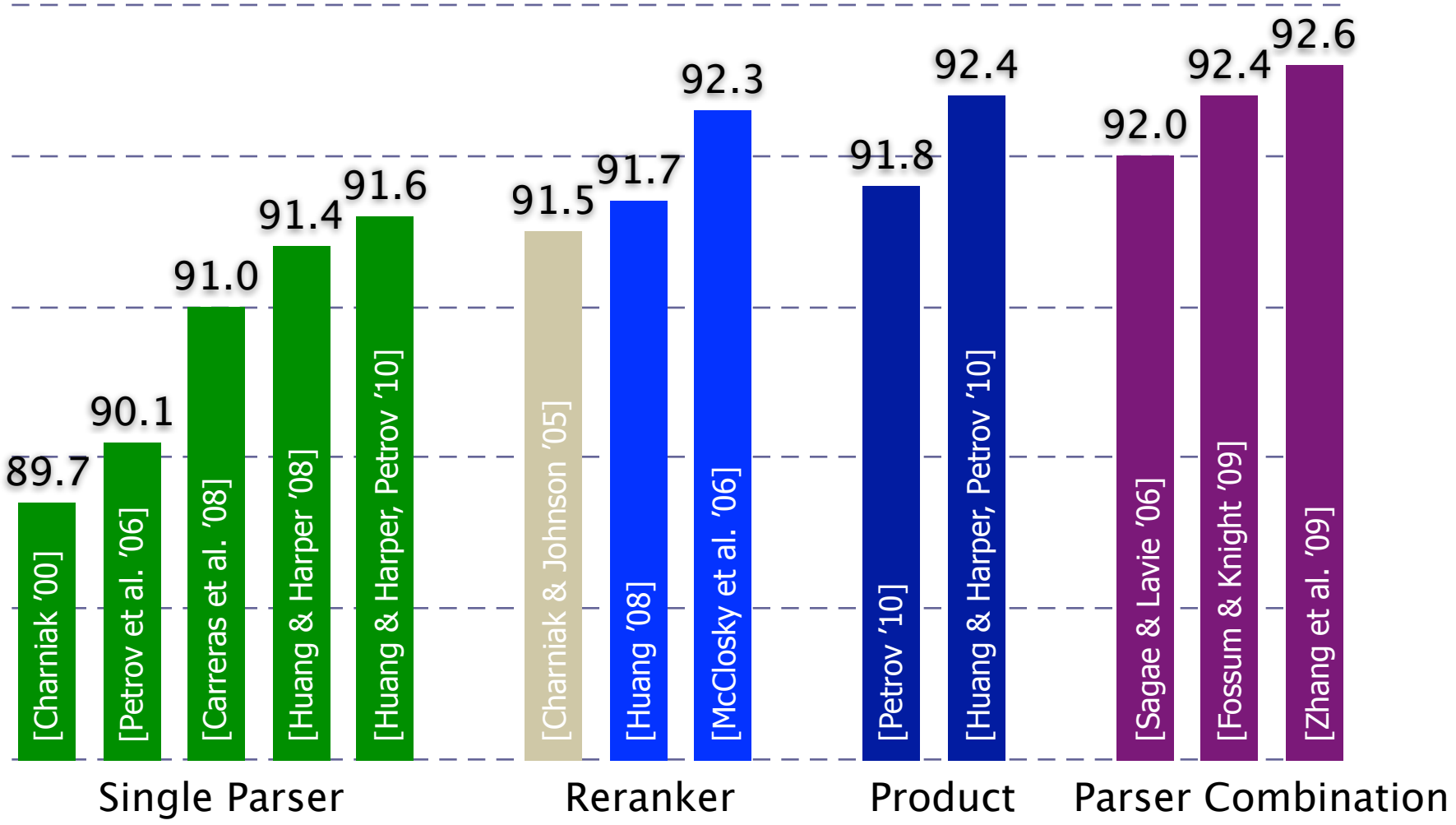
Hierarchical Pruning



Parsing Times (per sentence)

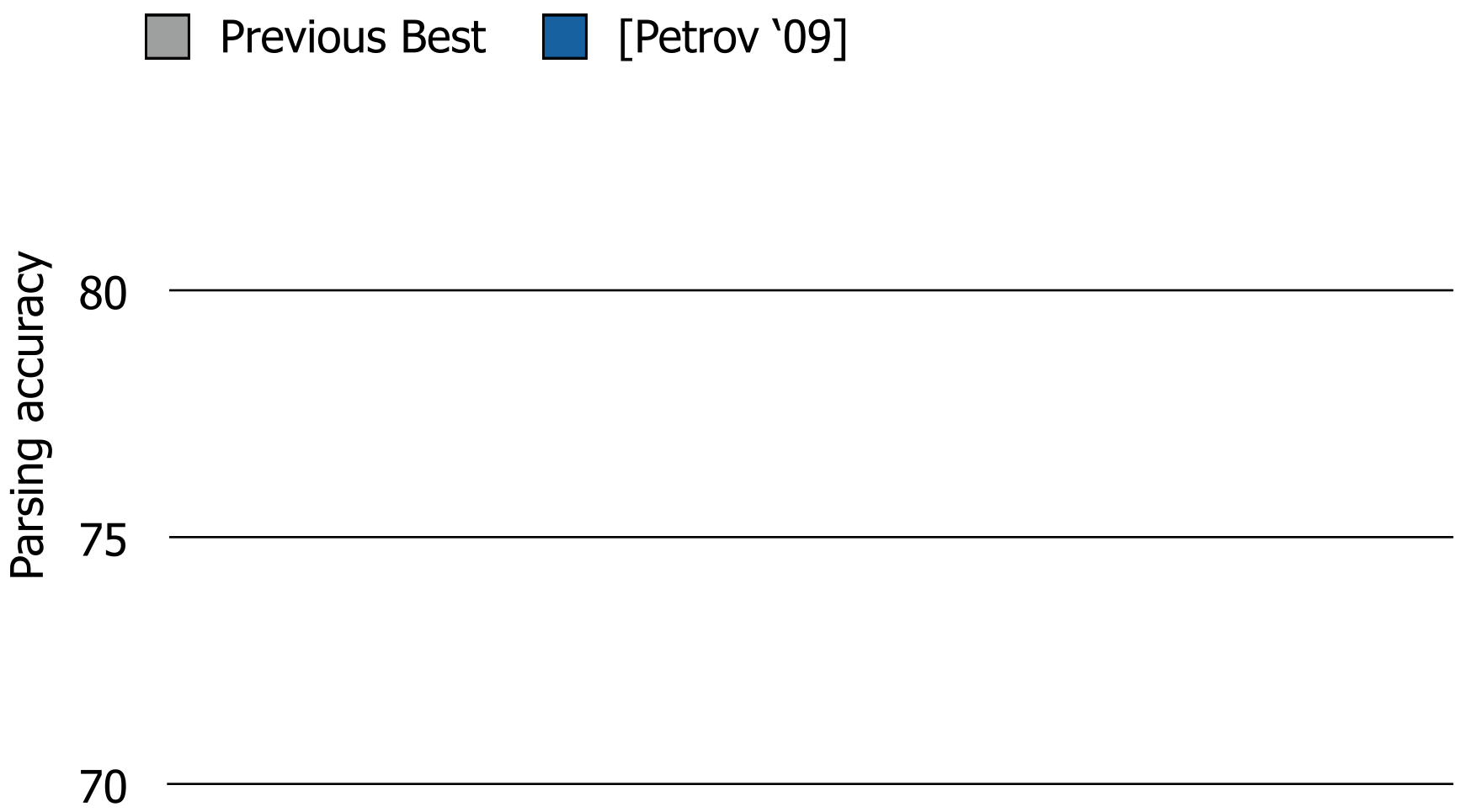


Detailed English Results

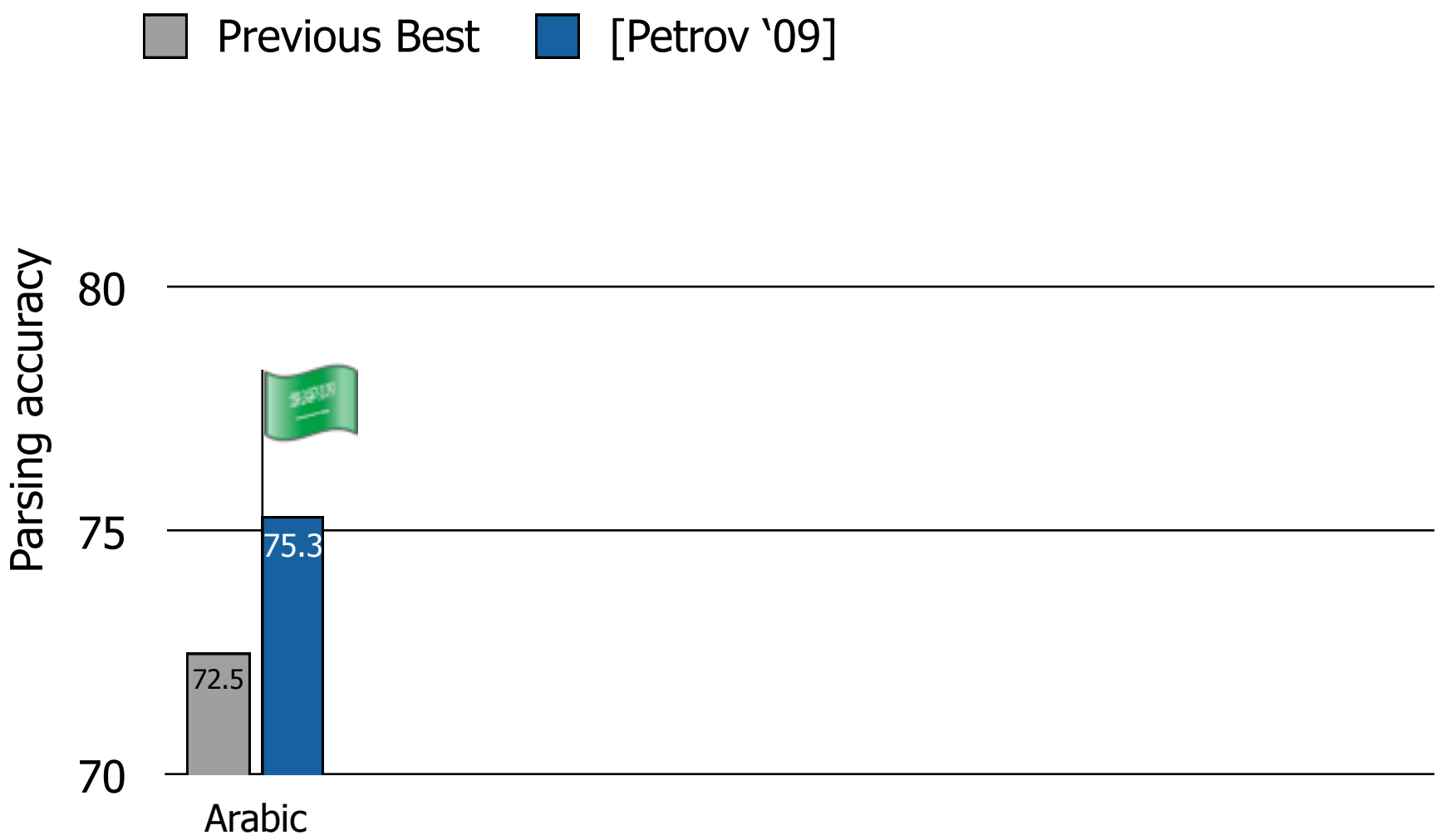


Multi-Lingual Results

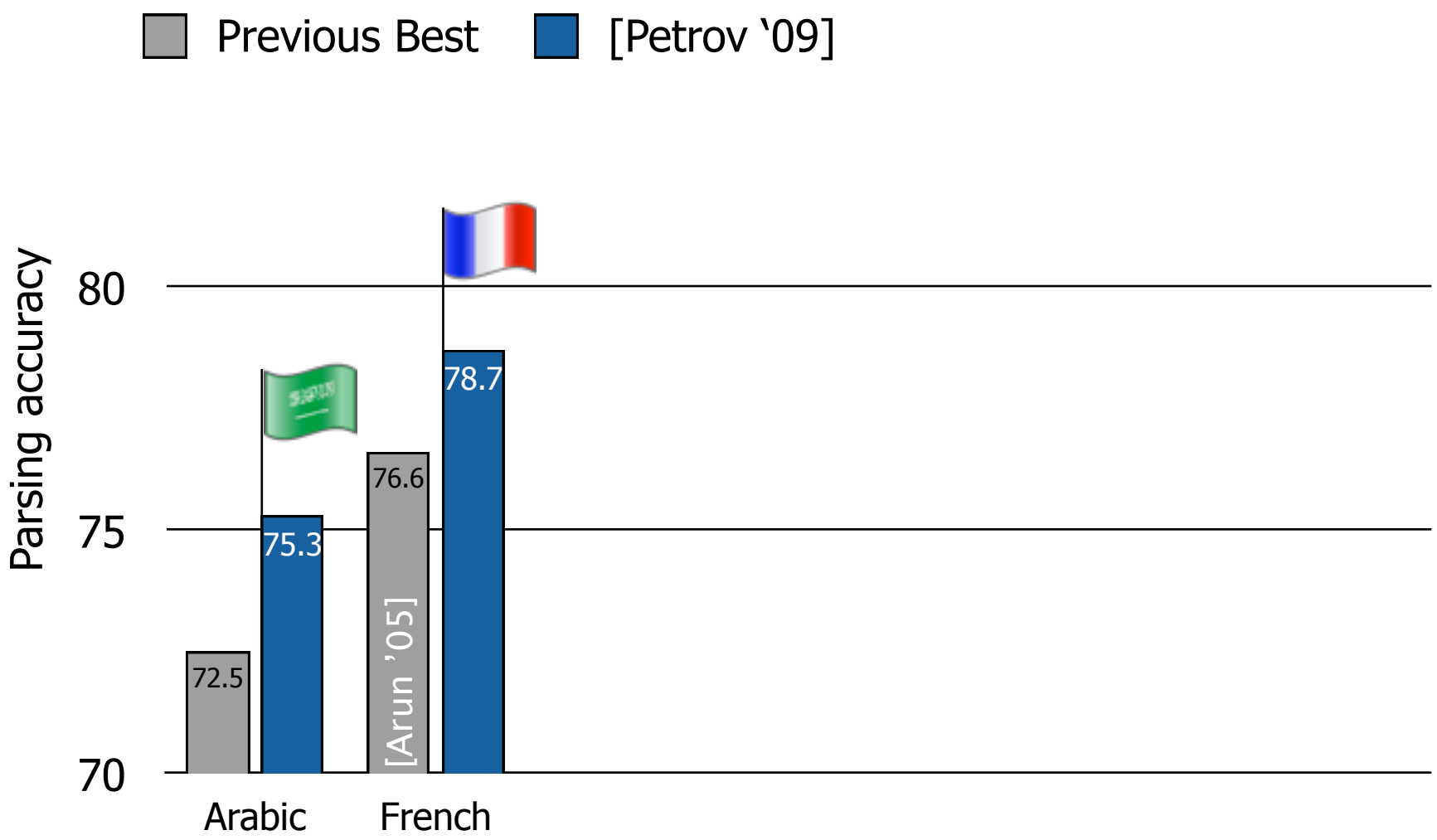
Multi-Lingual Results



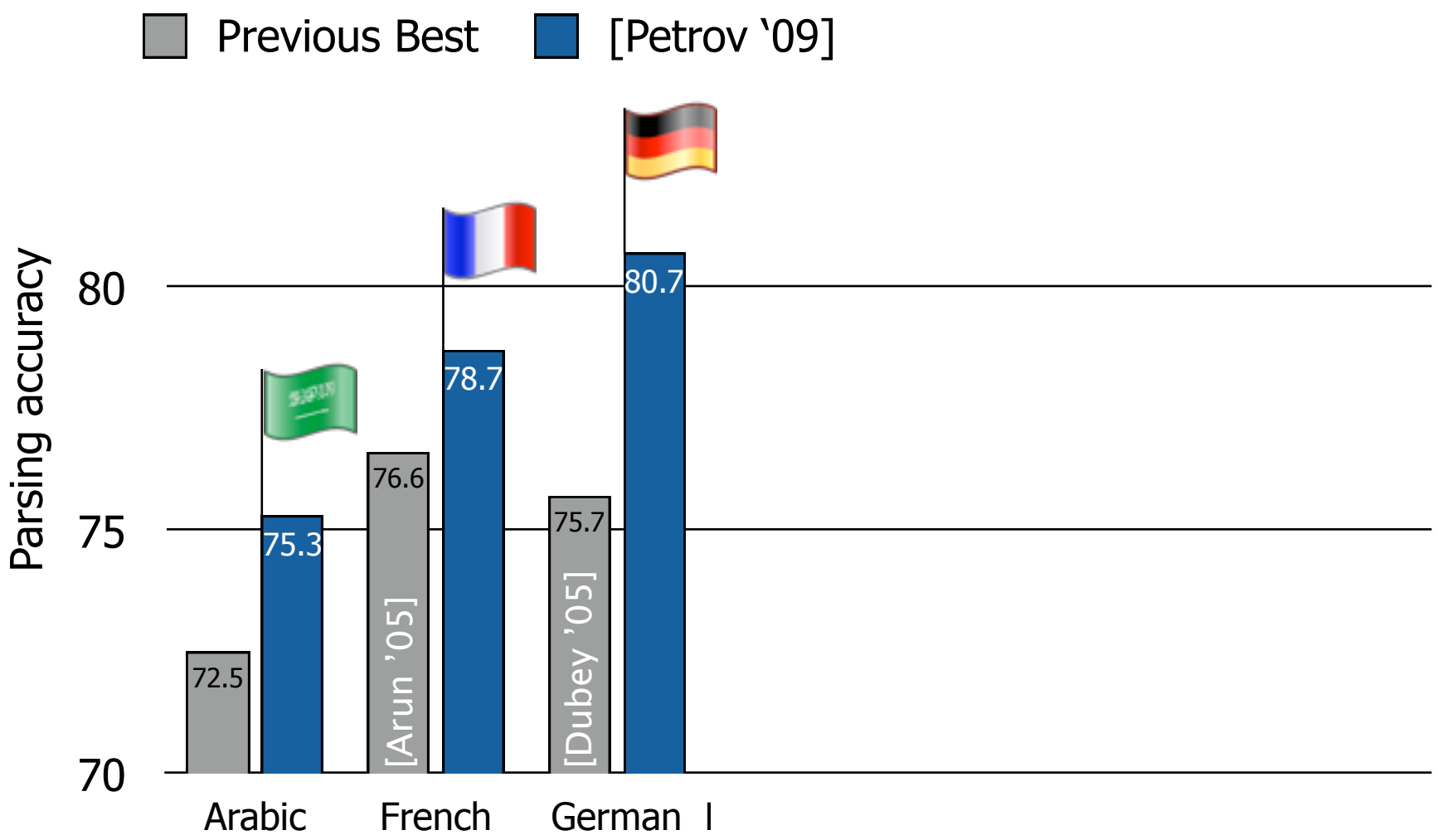
Multi-Lingual Results



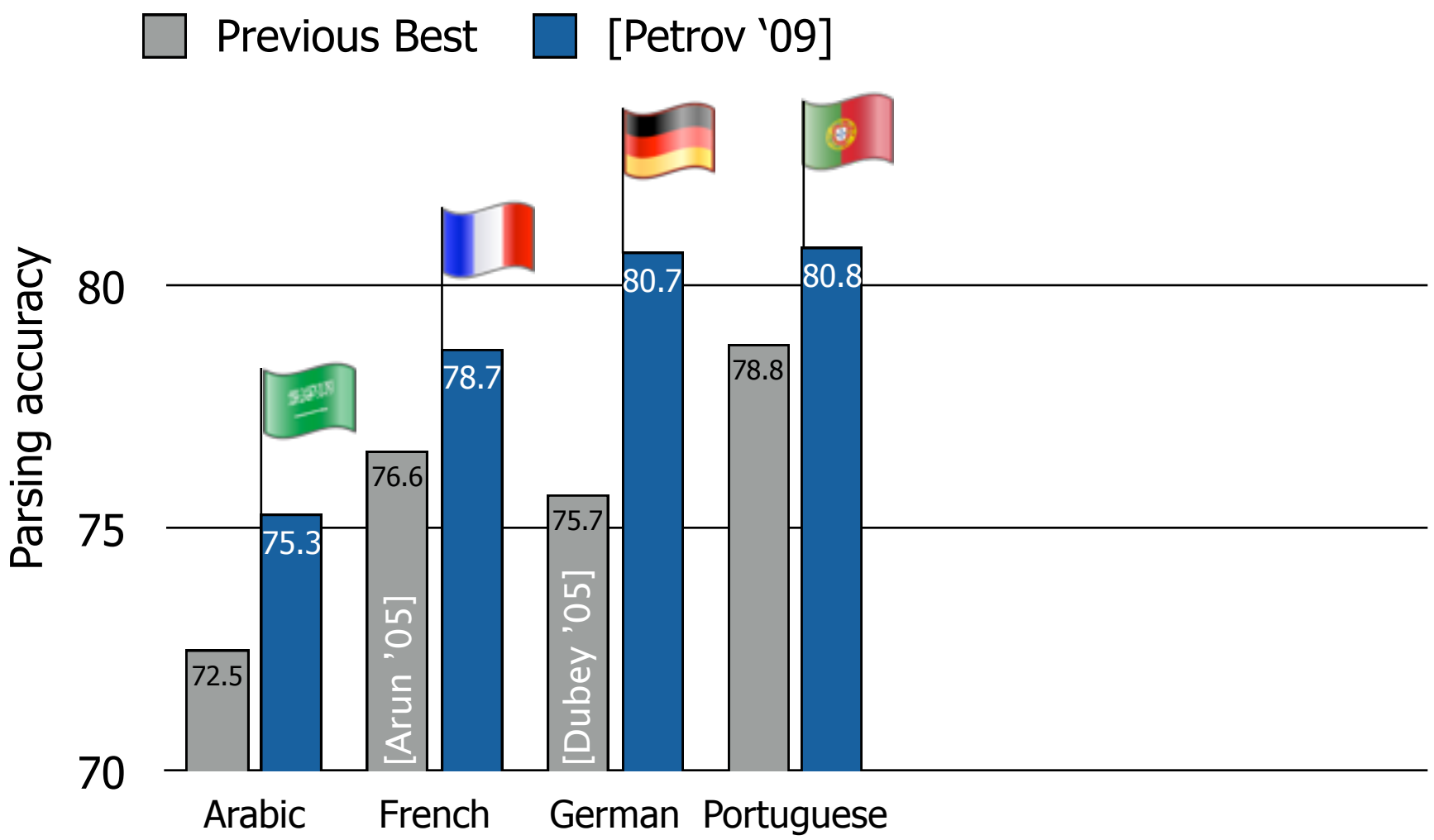
Multi-Lingual Results



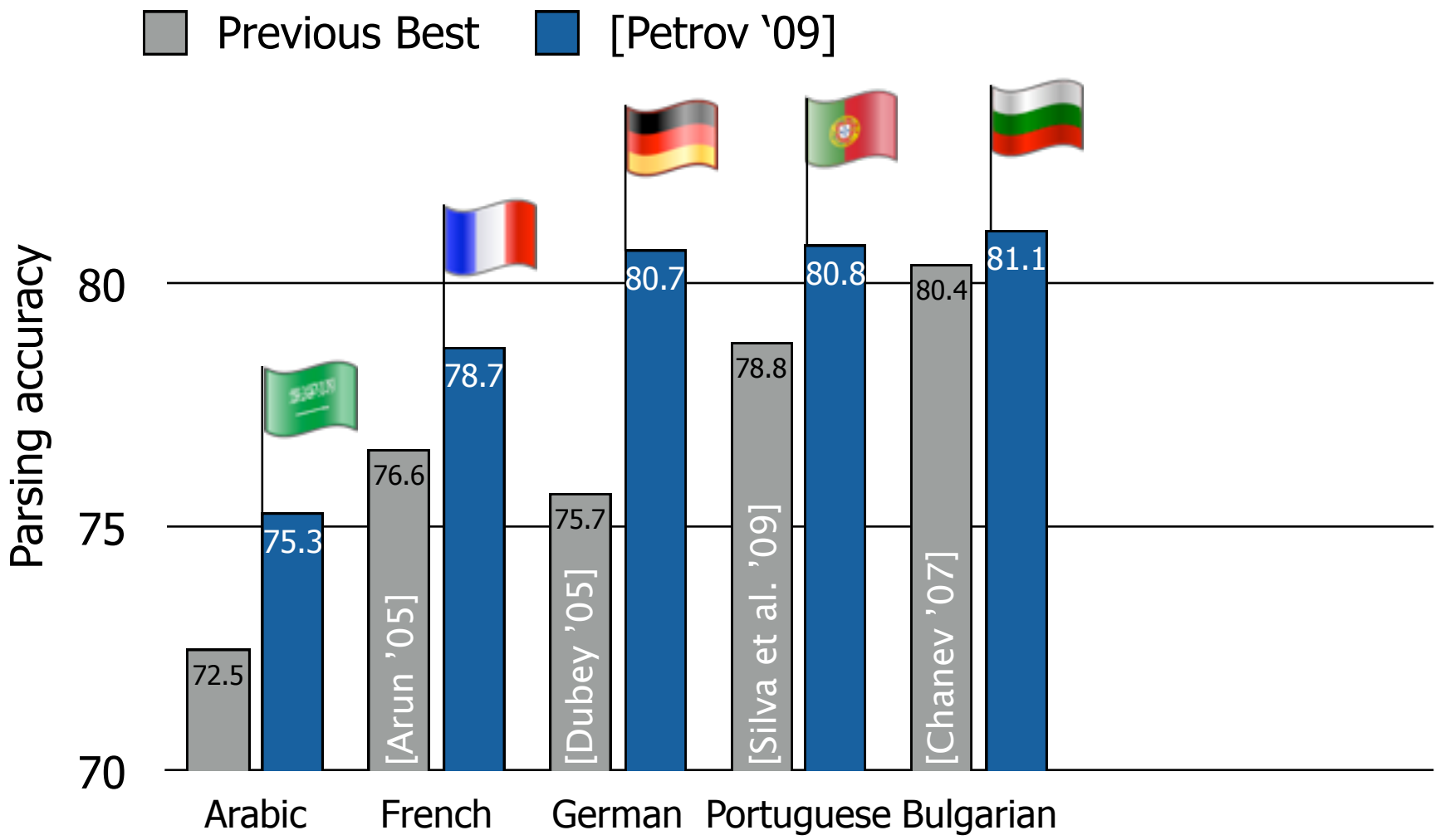
Multi-Lingual Results



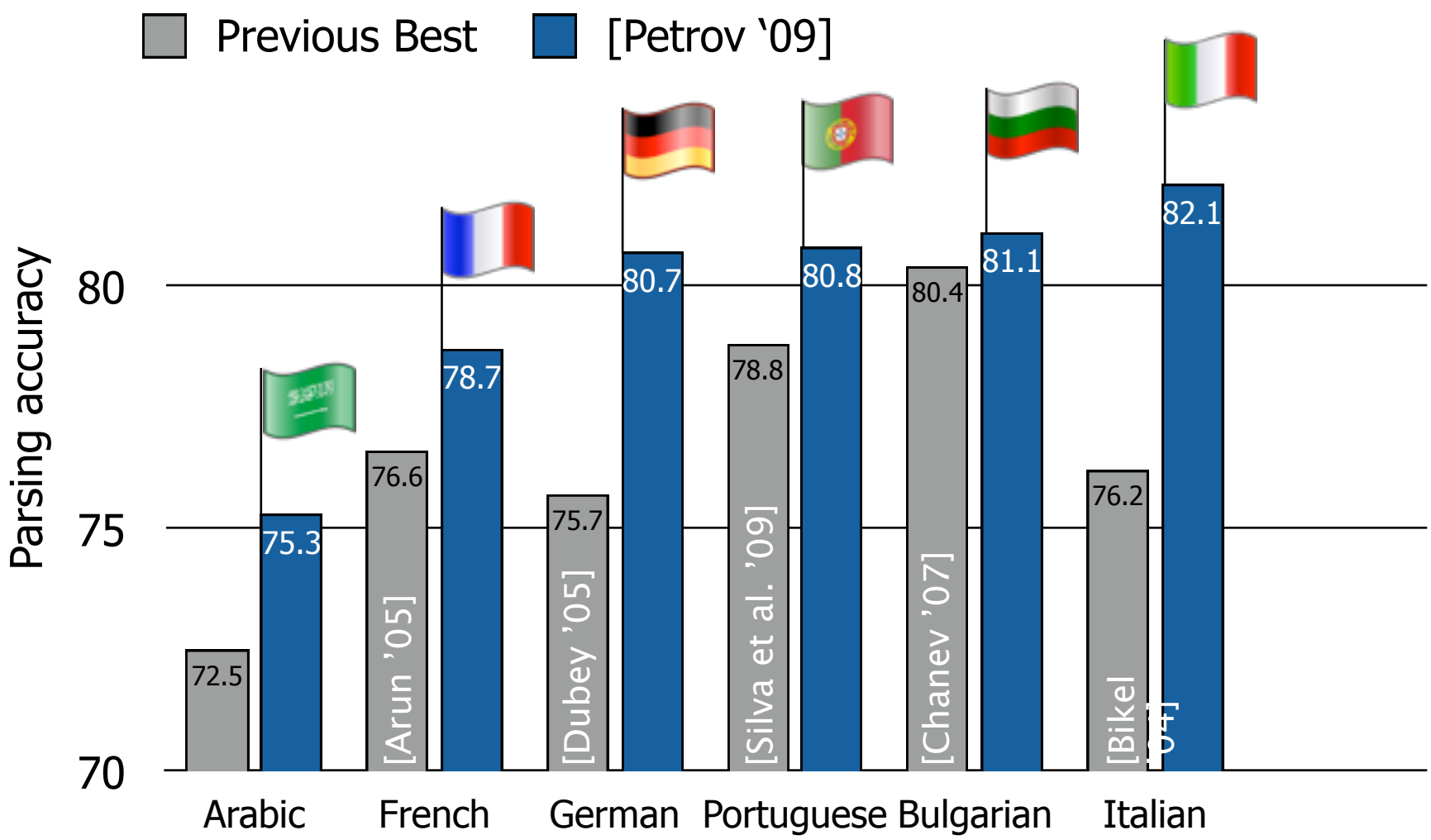
Multi-Lingual Results



Multi-Lingual Results



Multi-Lingual Results



Multi-Lingual Results