Refining Generative Grammars

Parser Evaluation

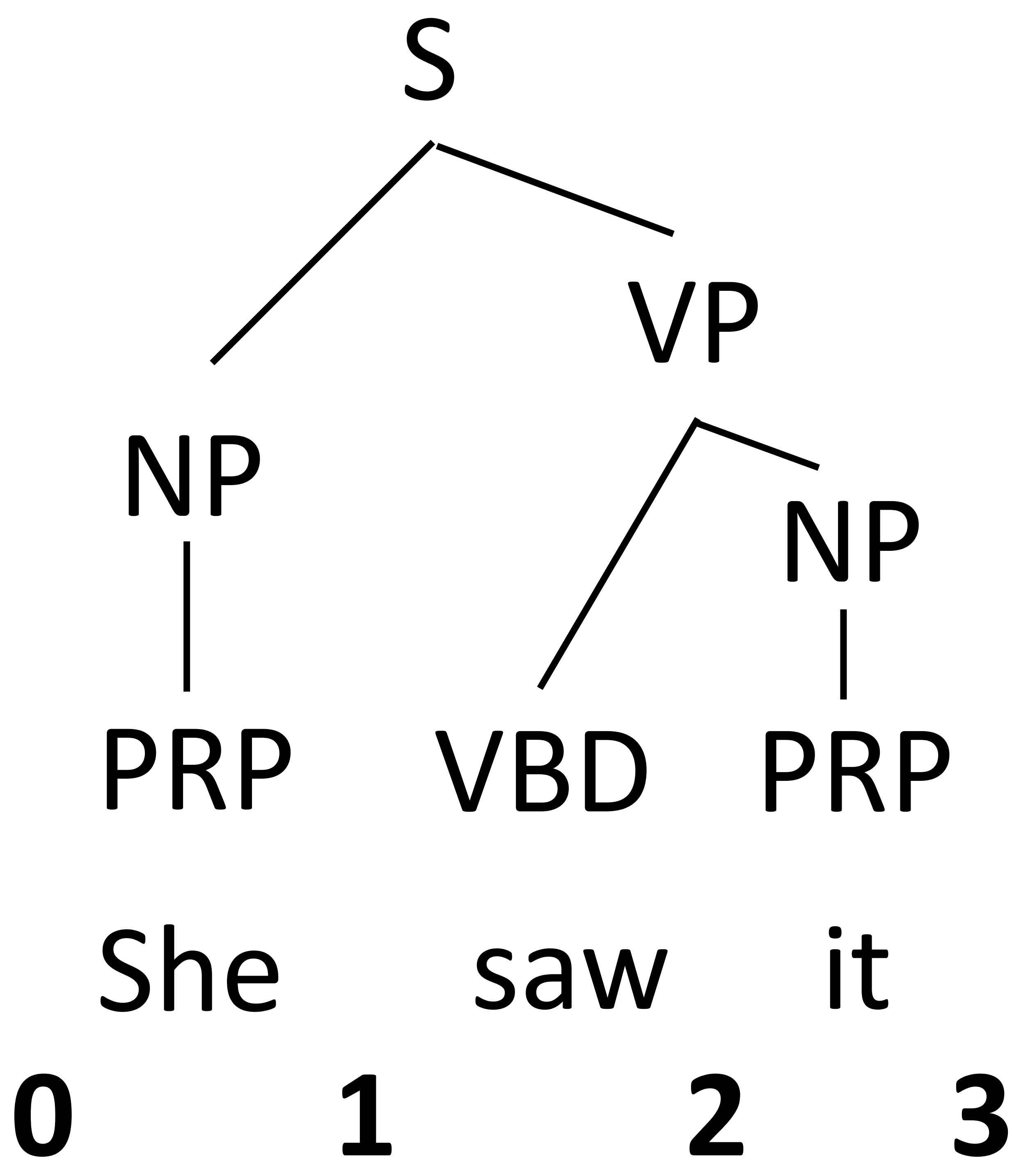
- View a parse as a set of labeled *brackets* / constituents

S(0,3)

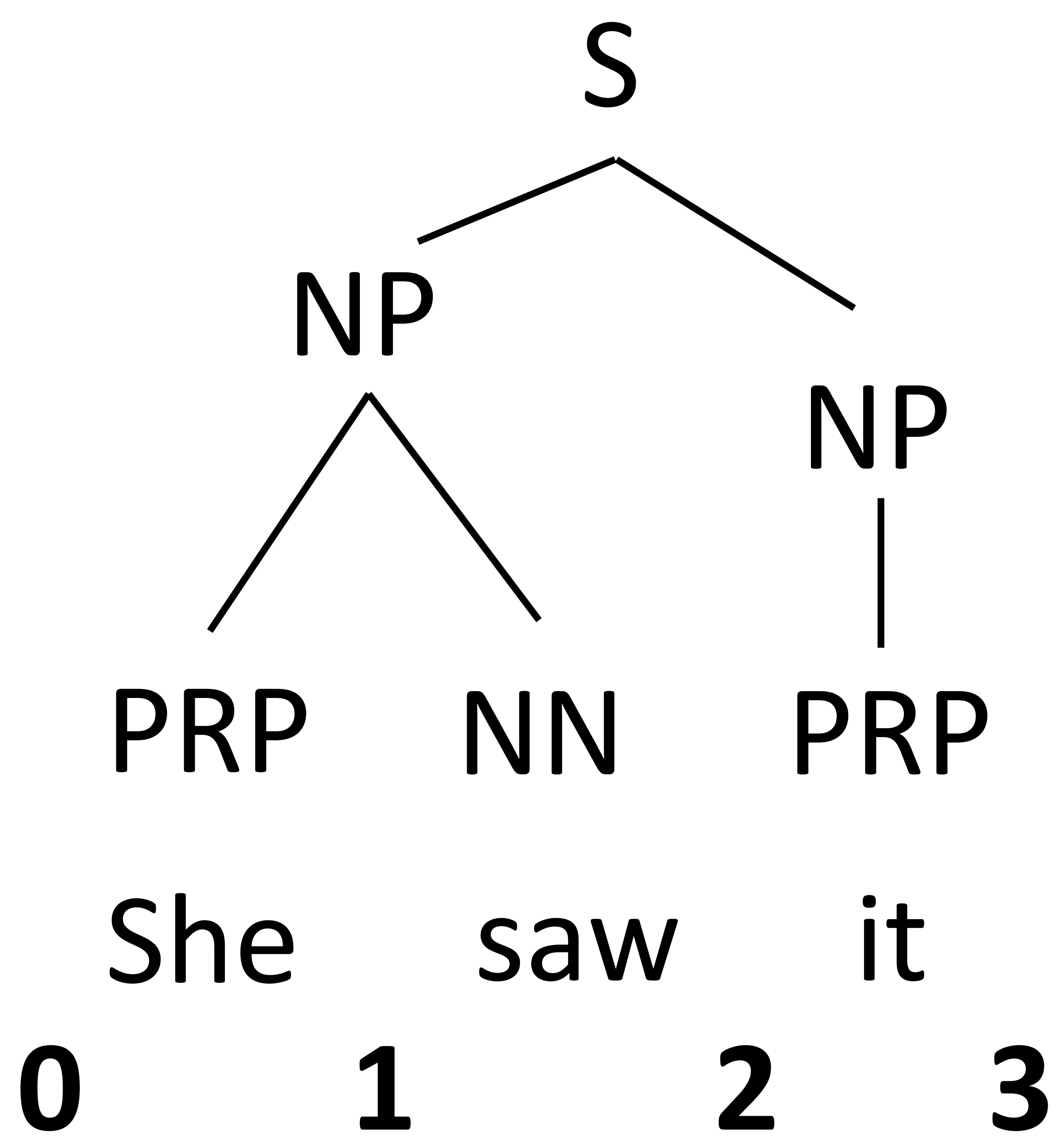
NP(0,1)

PRP(0,1) (but standard evaluation *does not count POS tags*)

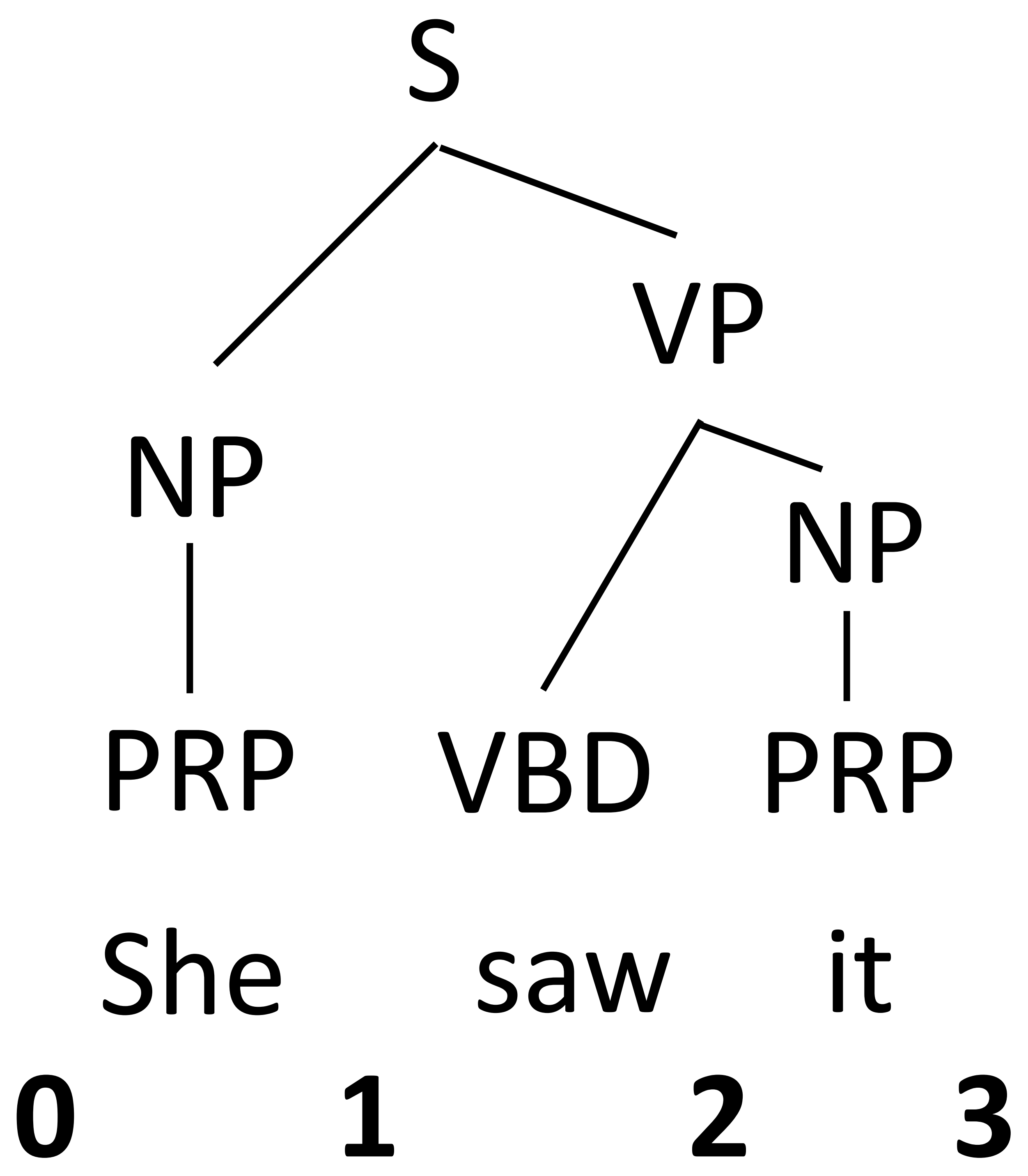
VP(1,3), VBD(1,2), NP(2,3), PRP(2,3)



Parser Evaluation



S(0,3),
NP(0,2),
NP(2,3),
~~PRP(0,1),~~
~~NN(1,2),~~
~~PRP(2,3)~~



S(0,3),
NP(0,1),
VP(1,3),
NP(2,3),
~~PRP(0,1),~~
~~VBD(1,2),~~
~~PRP(2,3)~~

- Precision: number of correct predictions / number of predictions = 2/3
- Recall: number of correct predictions / number of golds = 2/4
- F1: harmonic mean of precision and recall = $(1/2 * ((2/4)^{-1} + (2/3)^{-1}))^{-1}$

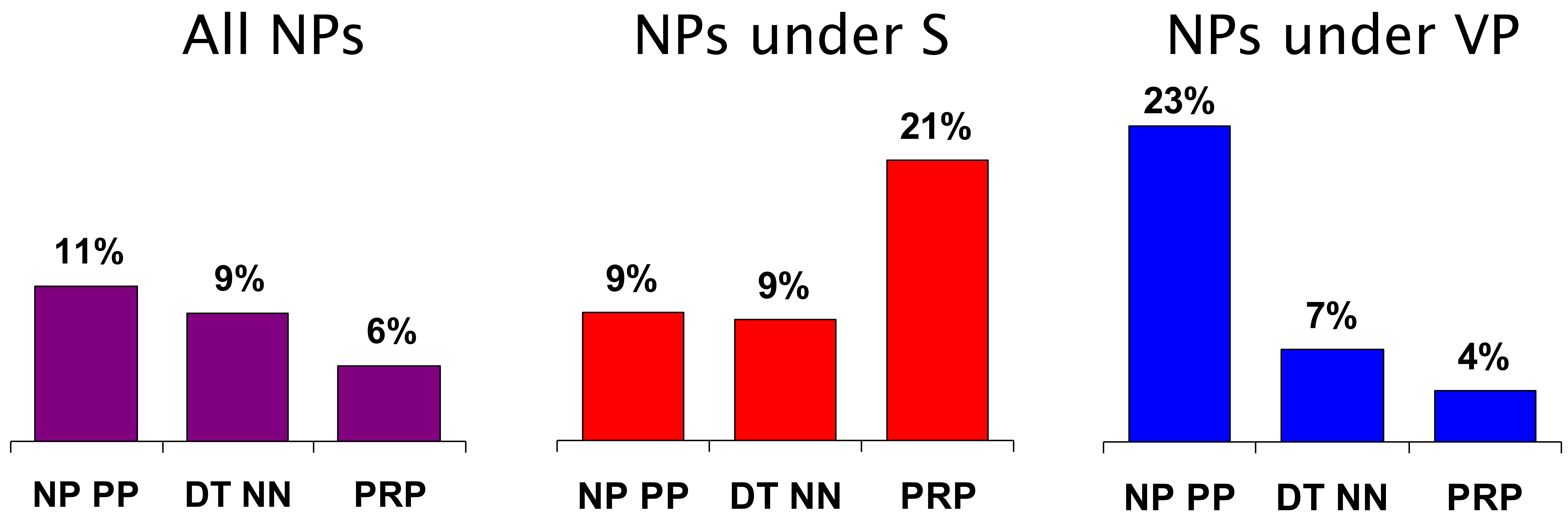
  = 0.57 (closer to min)

Results

- Standard dataset for English: Penn Treebank (Marcus et al., 1993)
- "Vanilla" PCFG: ~75 F1
- Best PCFGs for English: ~90 F1
- State-of-the-art discriminative models (using unlabeled data): 95 F1
- Other languages: results vary widely depending on annotation + complexity of the grammar

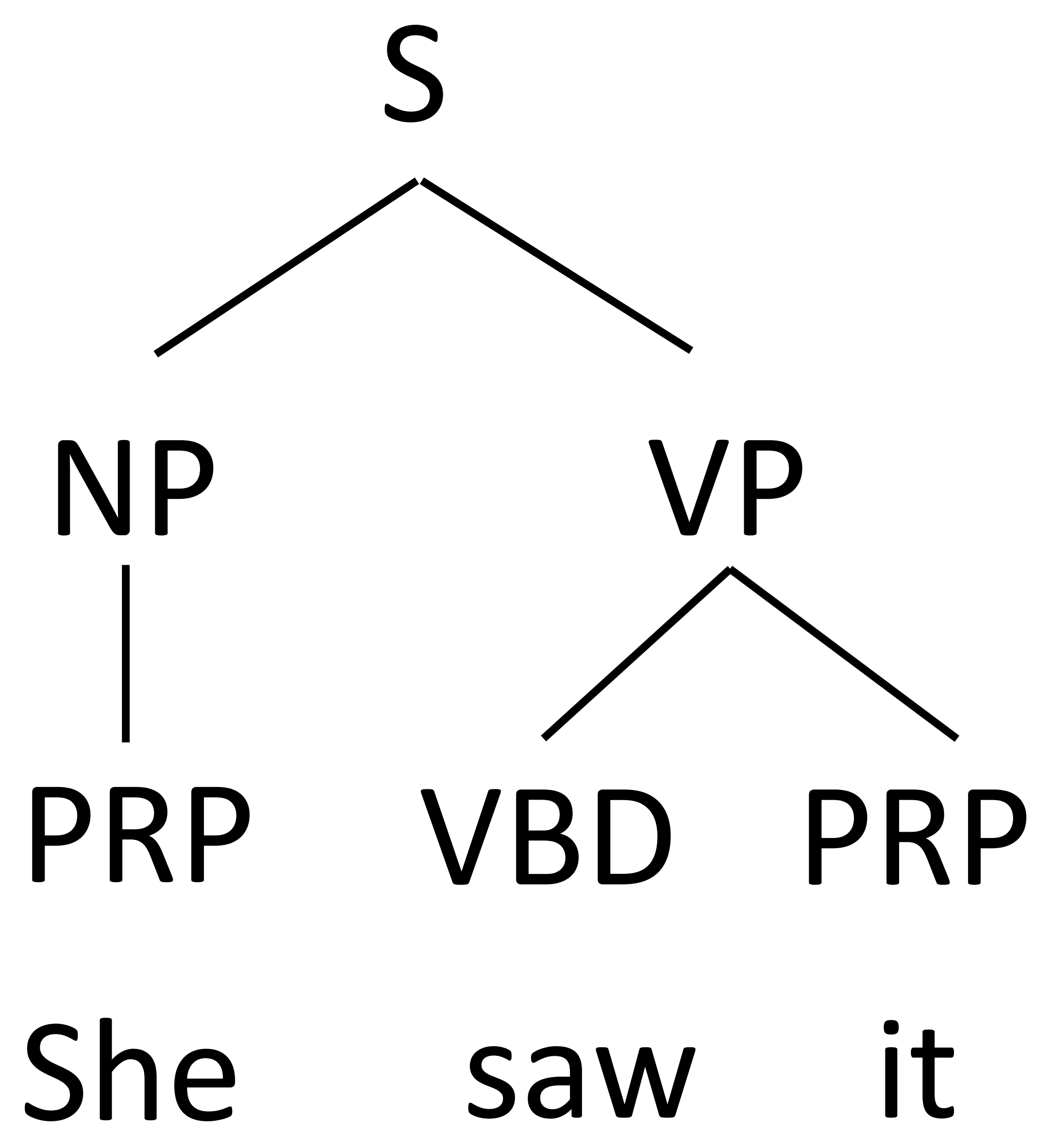
Klein and Manning (2003)

PCFG Independence Assumptions

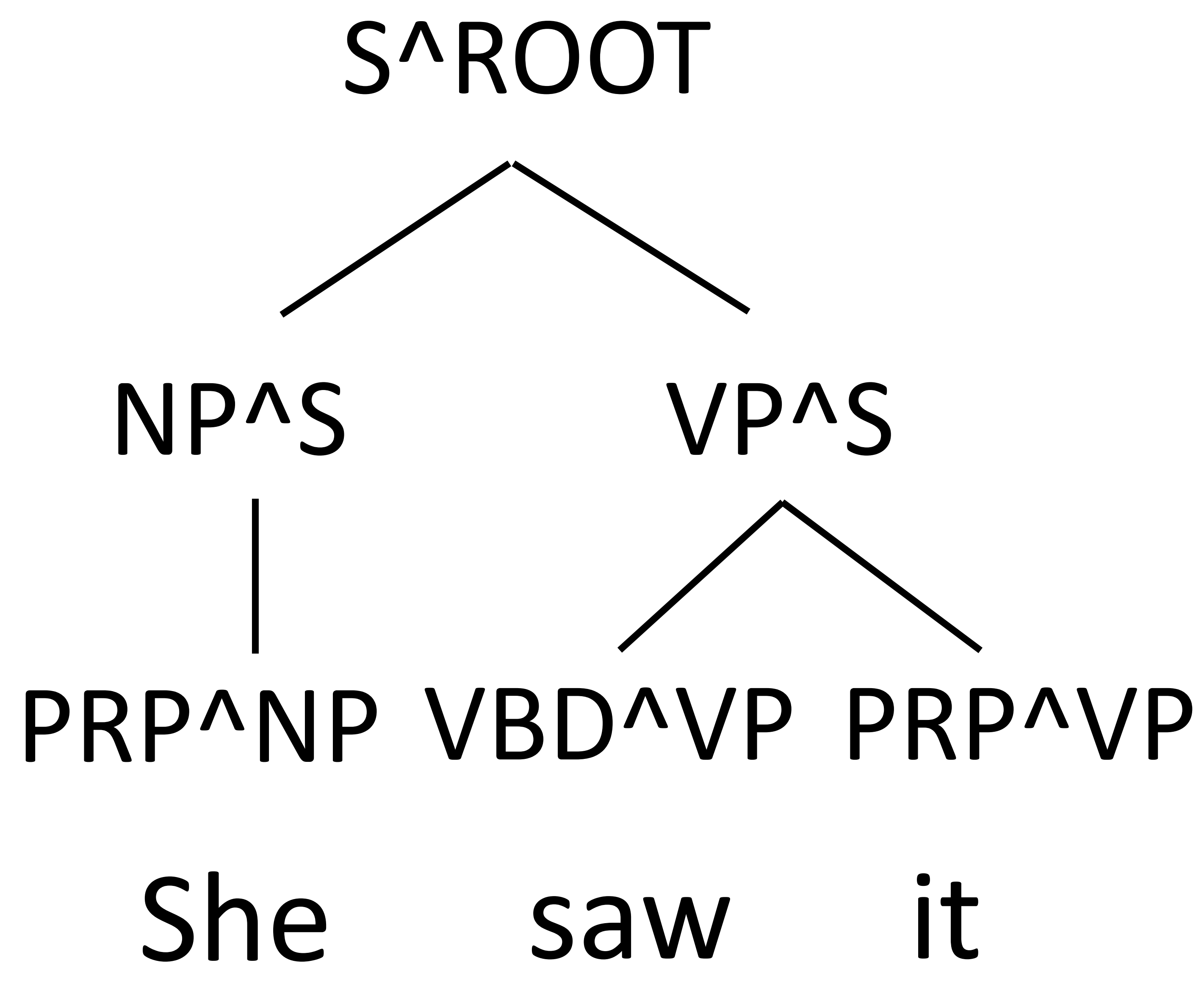


- Language is not context-free: NPs in different contexts rewrite differently
- [They]$_{NP}$ received [the package of books]$_{NP}$

Vertical Markovization



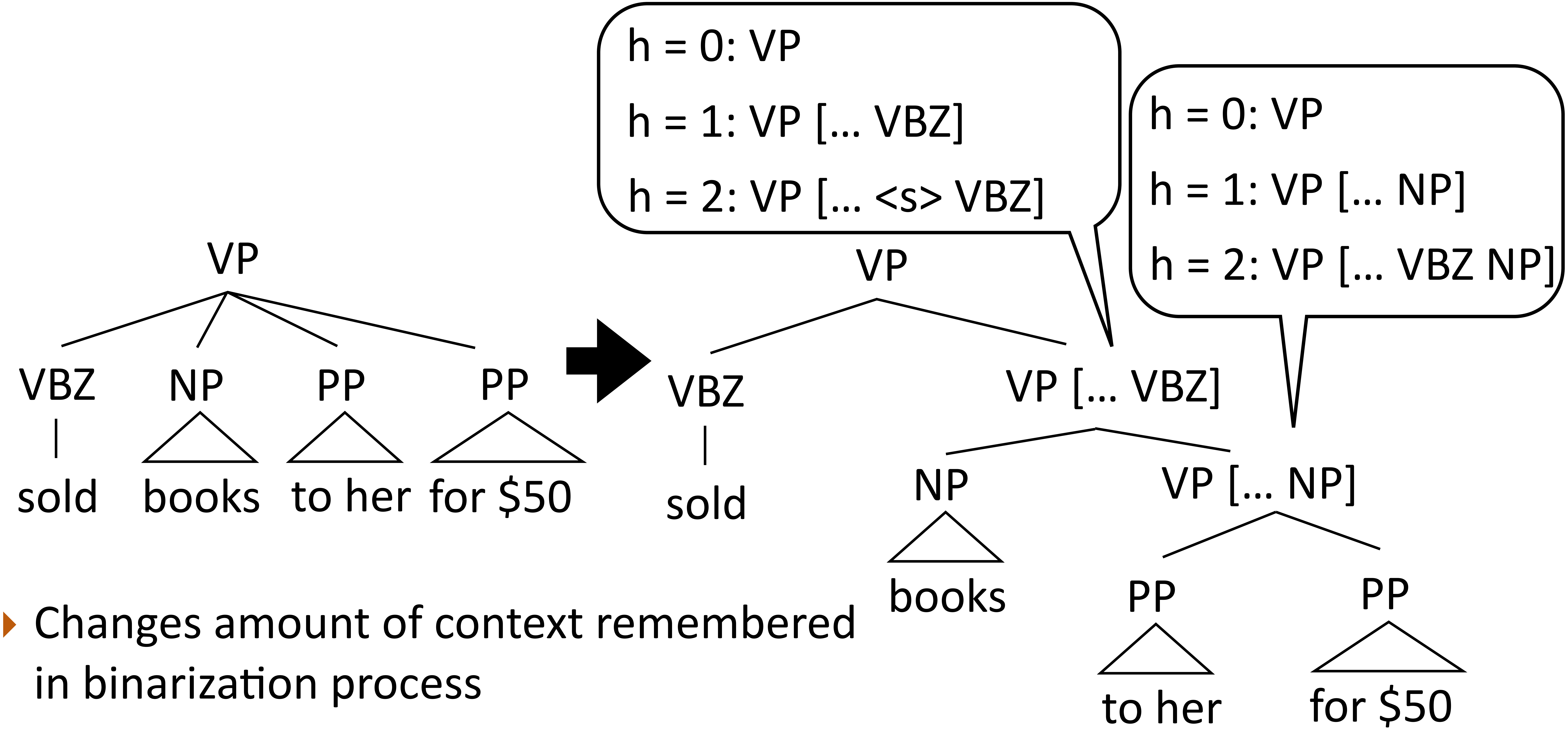
Basic tree (v = 0)



v = 1 Markovization

- Why is this a good idea?

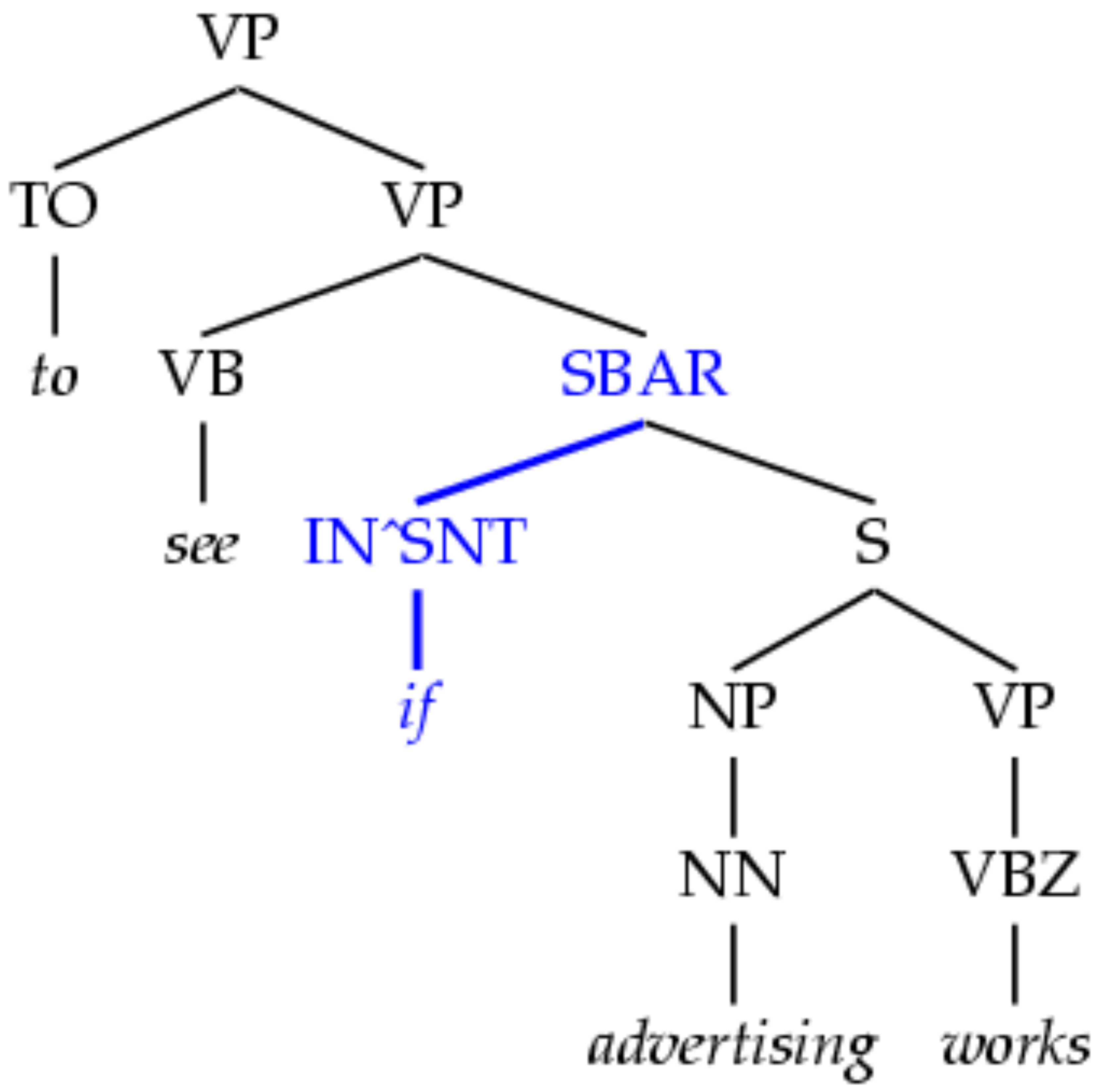
Horizontal Markovization



- Changes amount of context remembered in binarization process

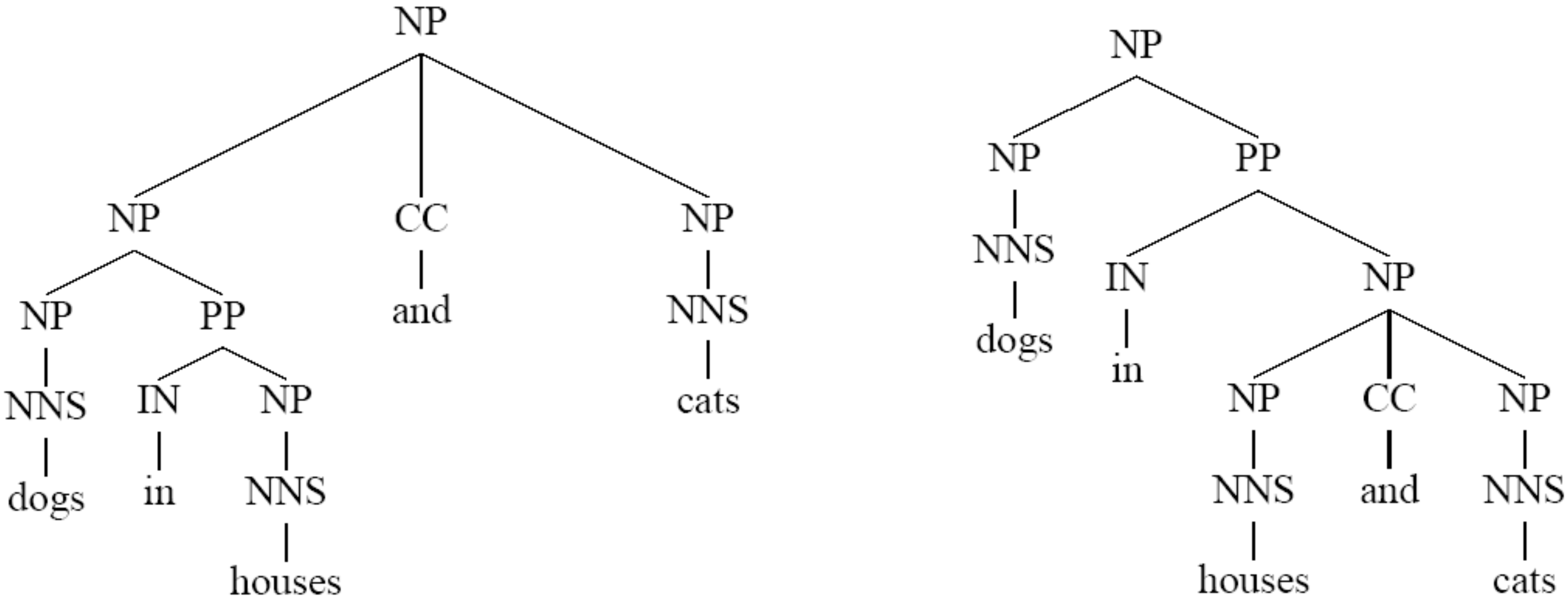
Tag Splits

- Can do some other ad hoc tag splits
- Sentential prepositions behave differently from other prepositions
- 75 F1 with basic PCFG => 86.3 F1 with a highly customized PCFG (v = 2, h = 2, other hacks like this)



Klein and Manning (2003)

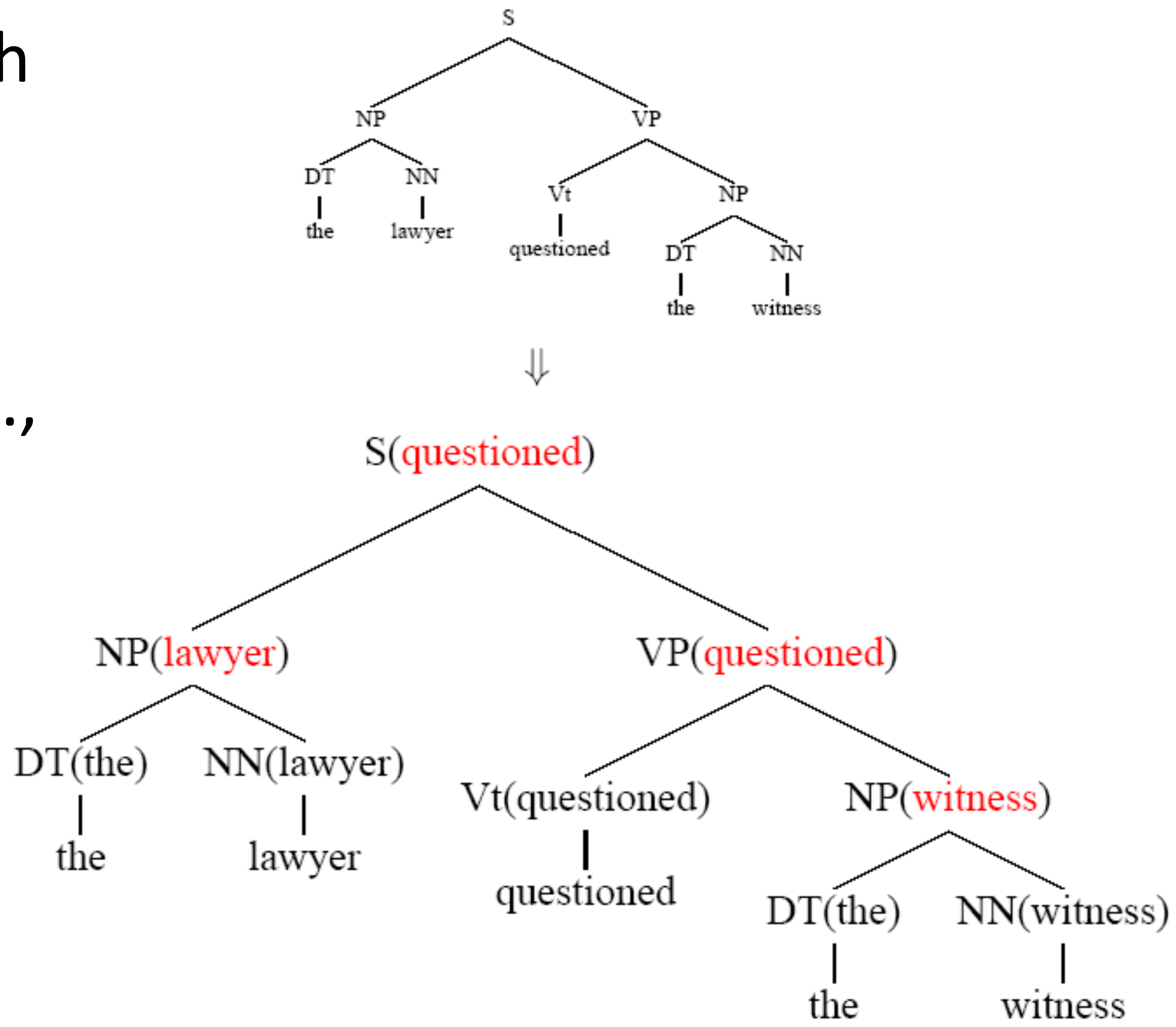
Lexicalized Parsers



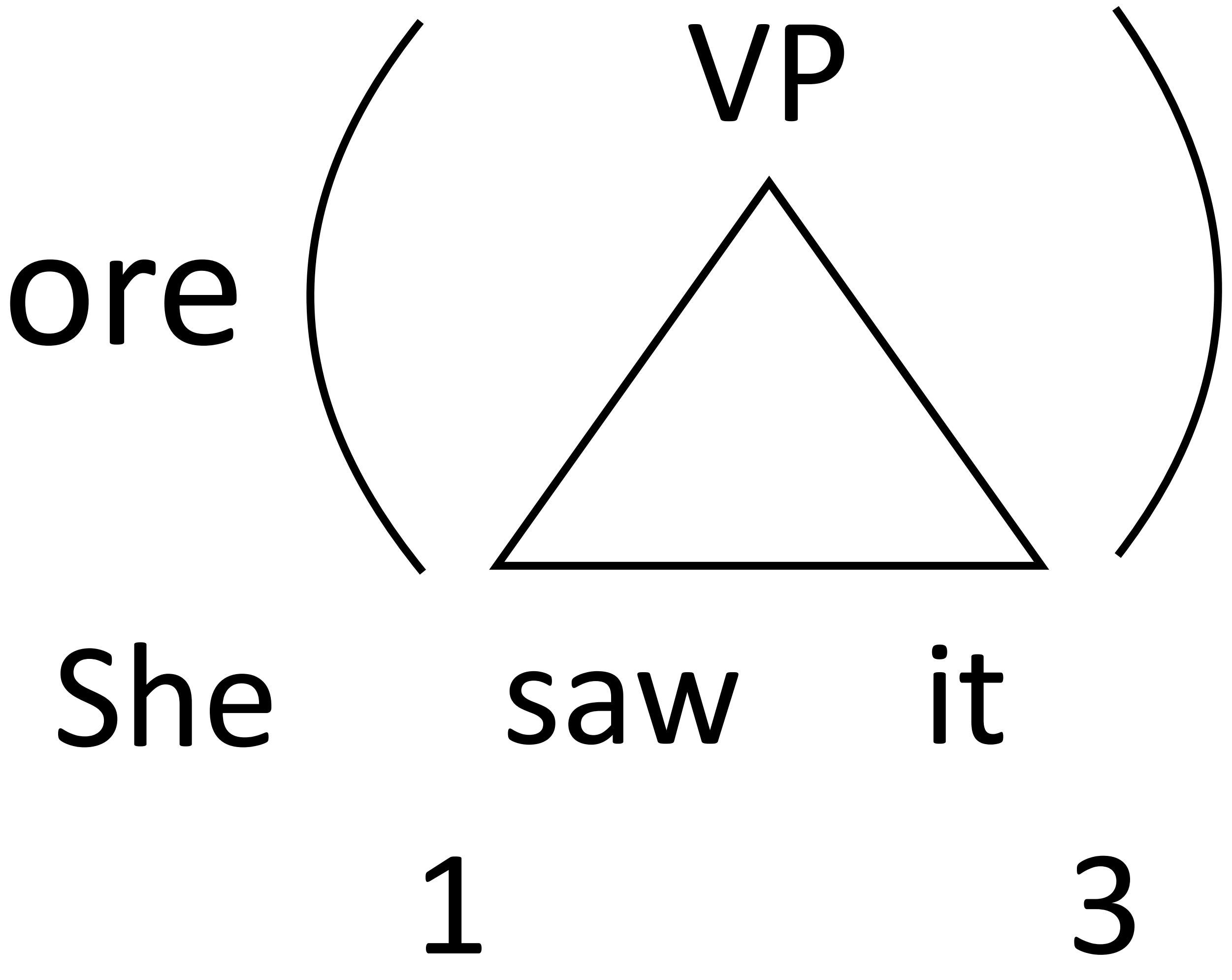
- Even with parent annotation, these trees have the same rules. Need to use the words

Lexicalized Parsers

- Annotate each grammar symbol with its “head word”: most important word of that constituent
- Rules for identifying headwords (e.g., the last word of an NP before a preposition is typically the head)
- Collins and Charniak (late 90s): ~89 F1 with these



Discriminative Parsers

score 

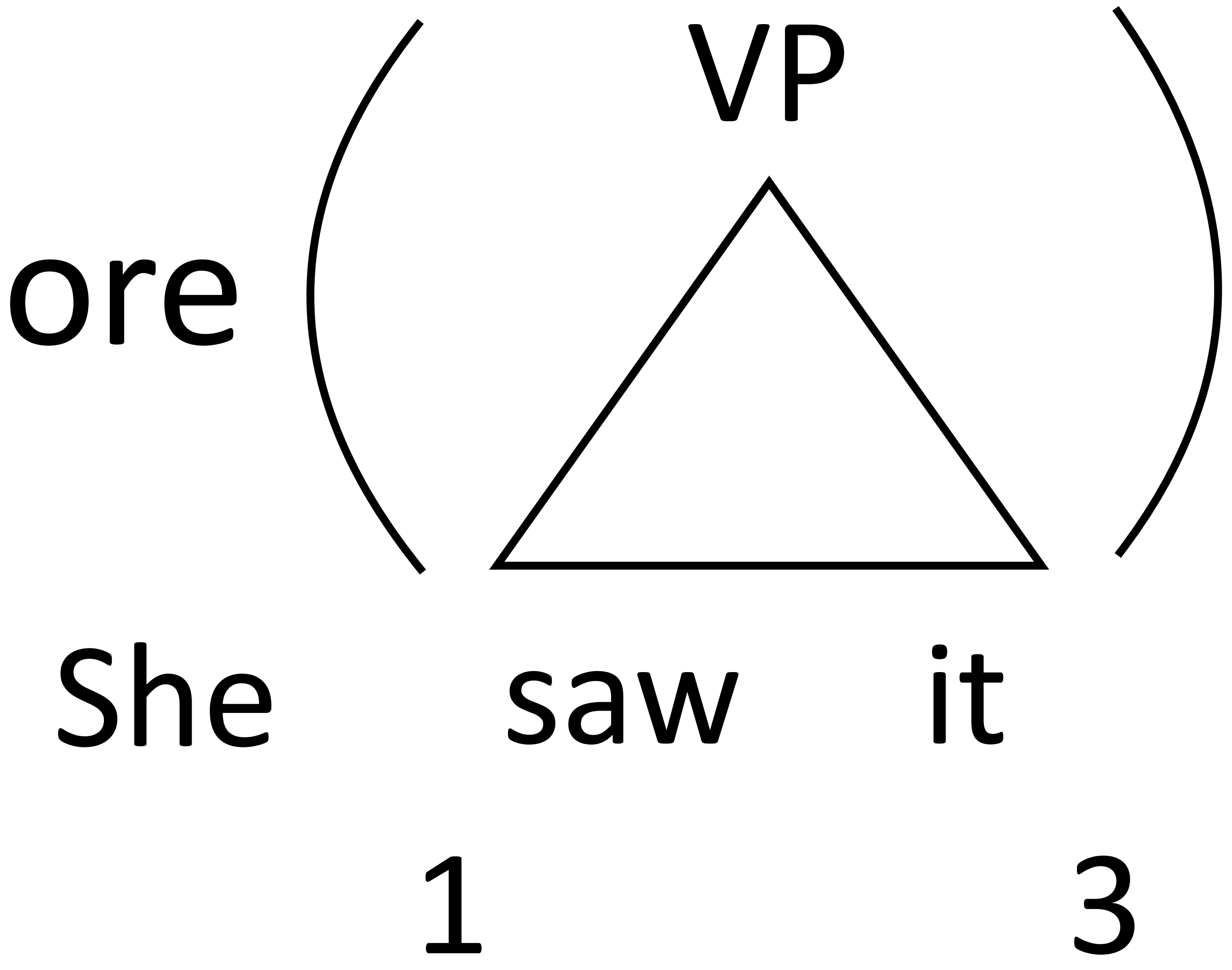
$= w^\top f\left(\begin{array}{c}\text{She saw it} \\ \;\;1 \qquad\quad 3\end{array}\right)$

Taskar et al. (2004), Hall et al. (2014), Stern et al. (2017), Kitaev et al. (2018)

- Features: I[first word = saw & VP]
  I[last word = it & VP]
  I[word before span = She & VP]
- ...or use neural networks

- Score *constituents* with a feature-based model
- Simple version of this model: Train a span classifier to predict type of span or NONE if it's not in the tree

Discriminative Parsers



$$\text{score}(\cdot) = w^\top f\left(\begin{array}{c}\text{She saw it} \\ 1 \quad\quad 3\end{array}\right)$$

Taskar et al. (2004), Hall et al. (2014), Stern et al. (2017), Kitaev et al. (2018)

- CKY: instead of rule probabilities, maximize sum of scores of the spans included in a tree
    - Why is CKY still necessary? Why can't we just independently label spans with our classifier?
- Neural net models get 91-93 F1, 95 F1 with other tricks we'll see later. Works well for other languages too!