Syntax and Parsing I

Constituency Parsing

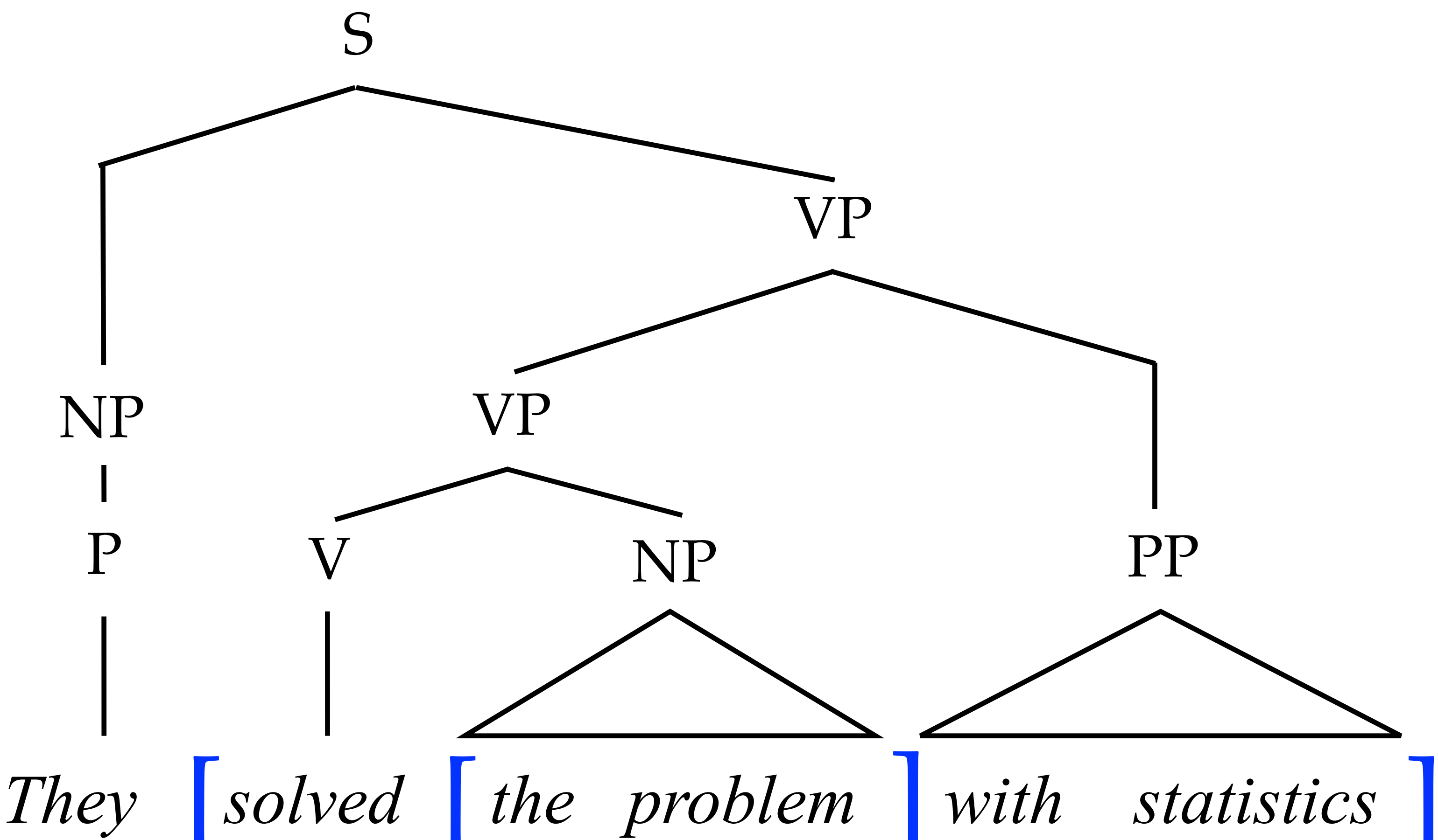
Slav Petrov – Google

Thanks to:

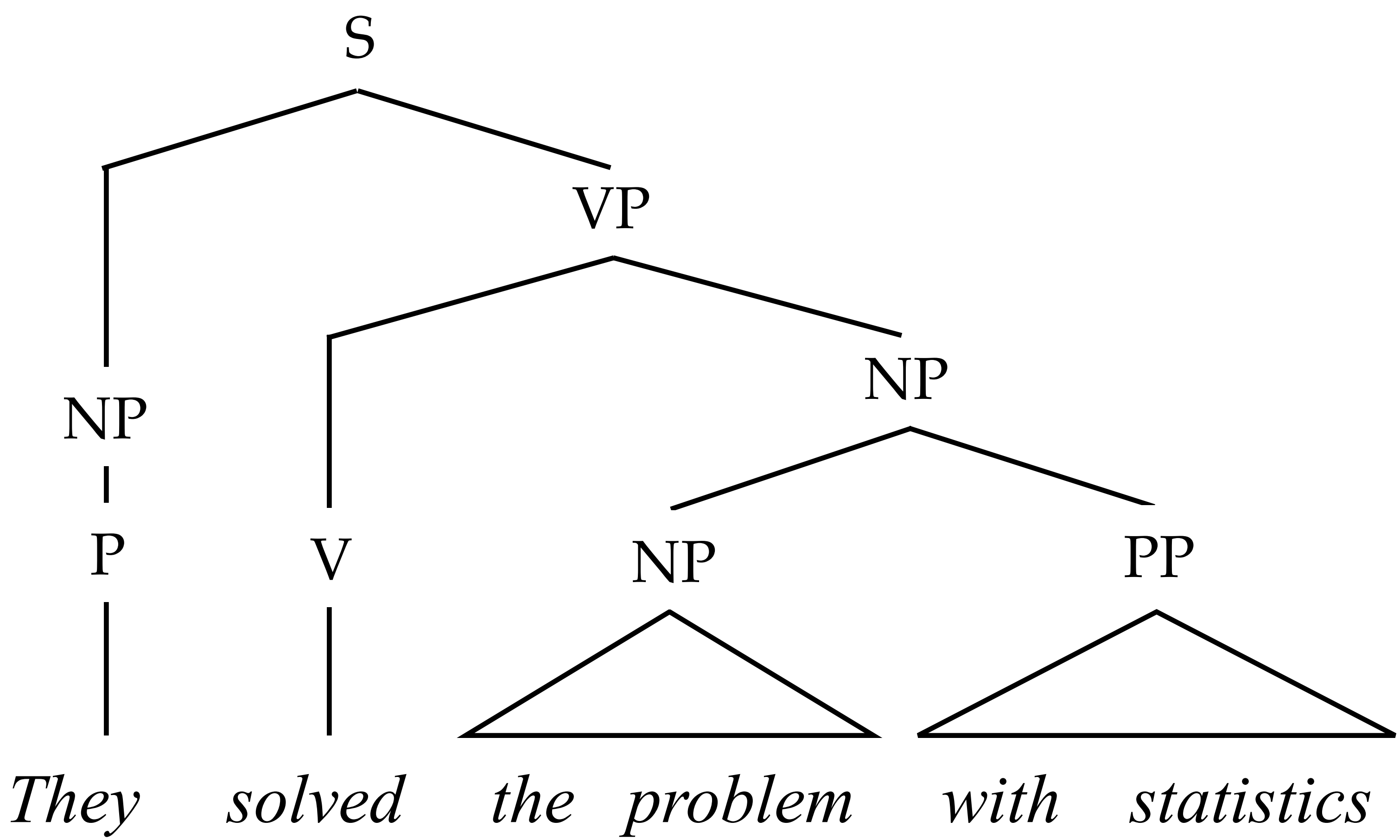
Dan Klein, Ryan McDonald, Alexander Rush, Joakim Nivre,
Greg Durrett, David Weiss

Lisbon Machine Learning School 2016

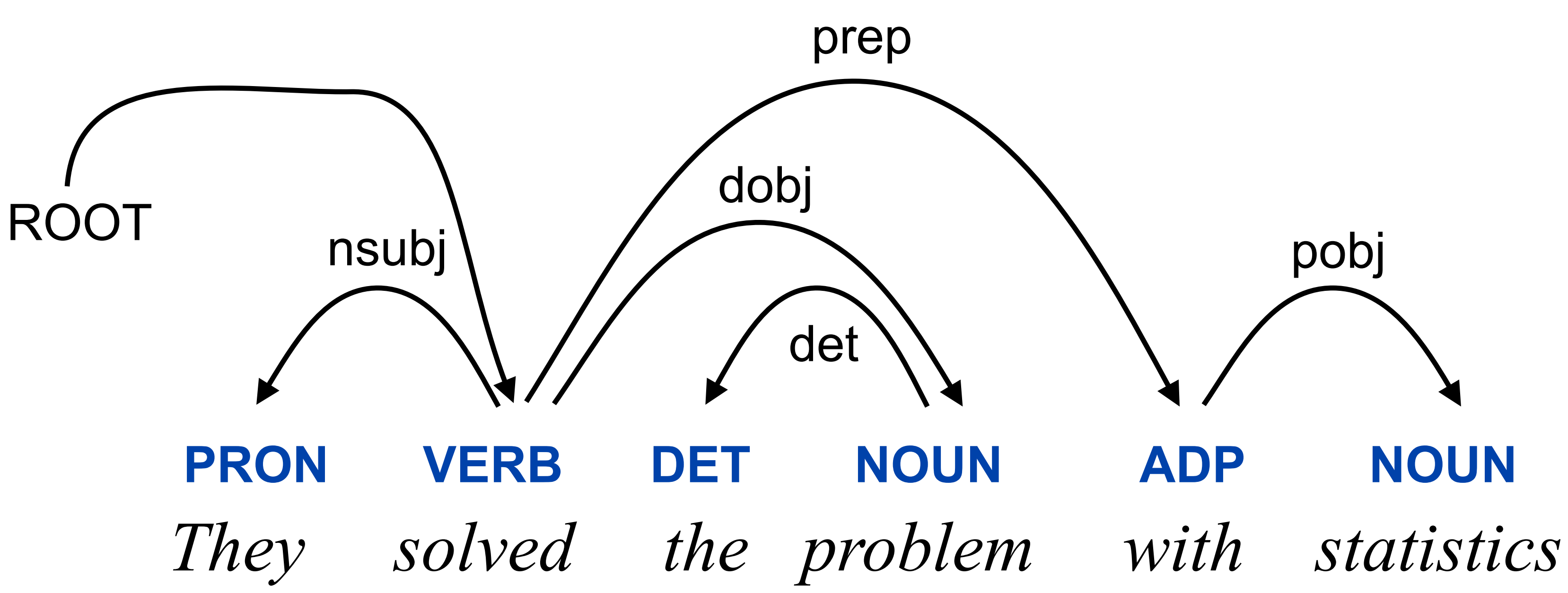
Analyzing Natural Language



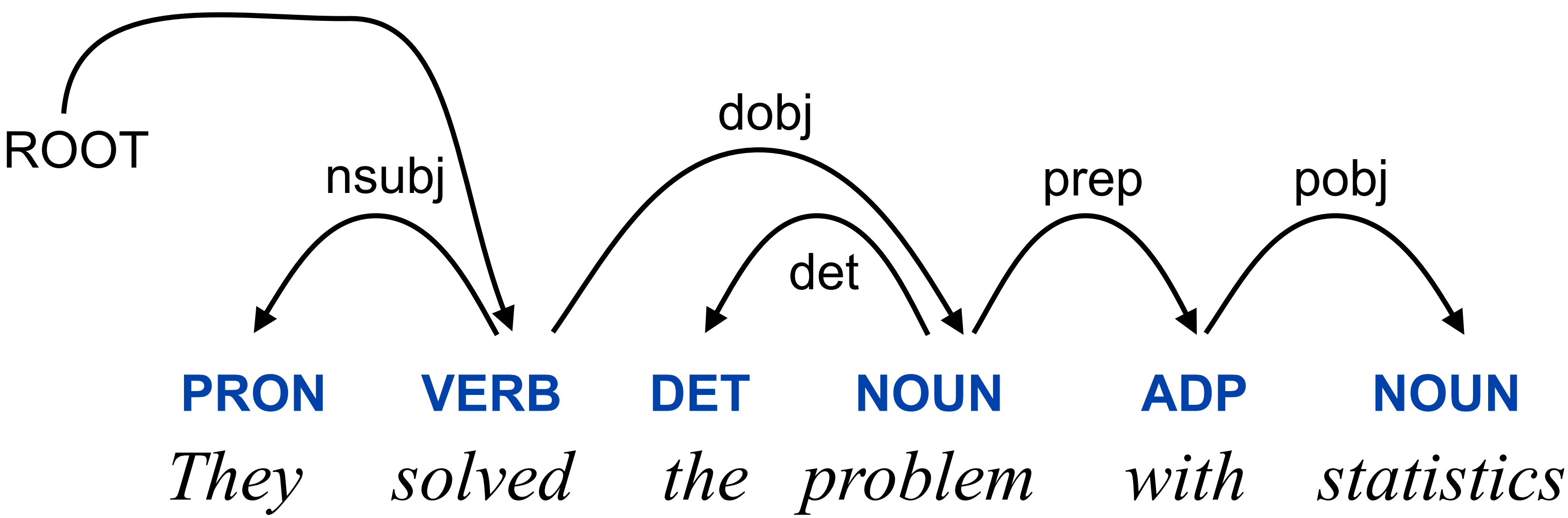
Syntax and Semantics



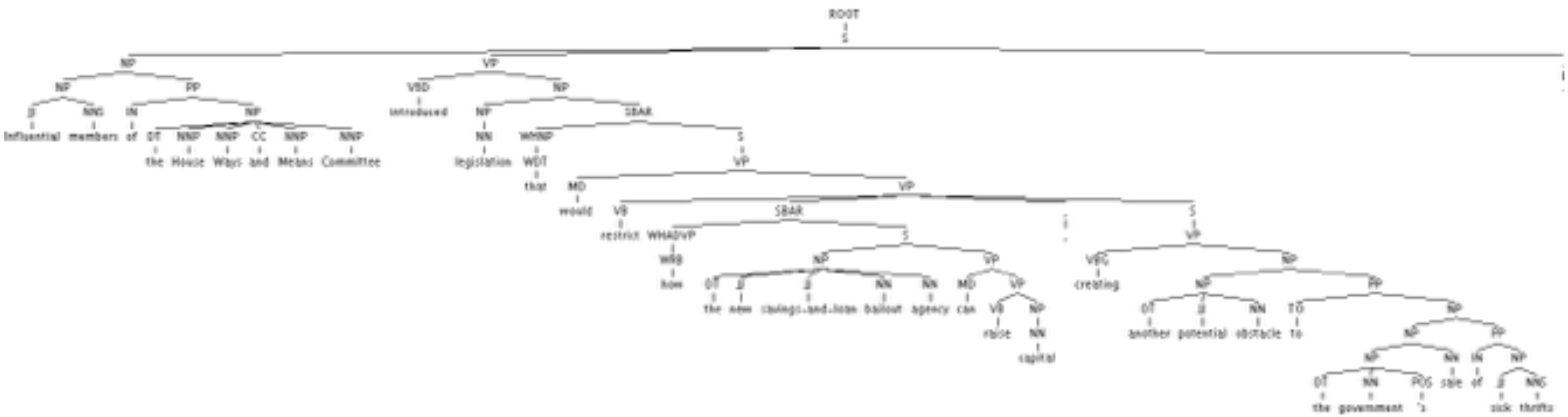
Constituency and Dependency



Constituency and Dependency



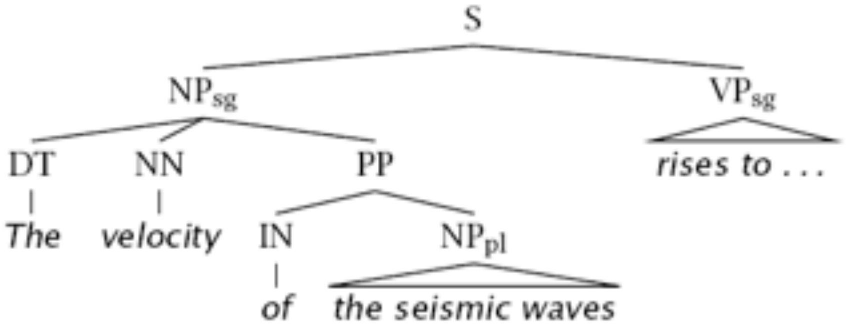
A “real” Sentence



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.

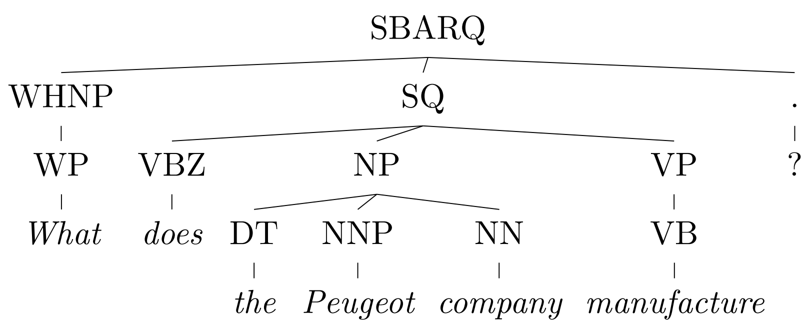
Phrase Structure Parsing

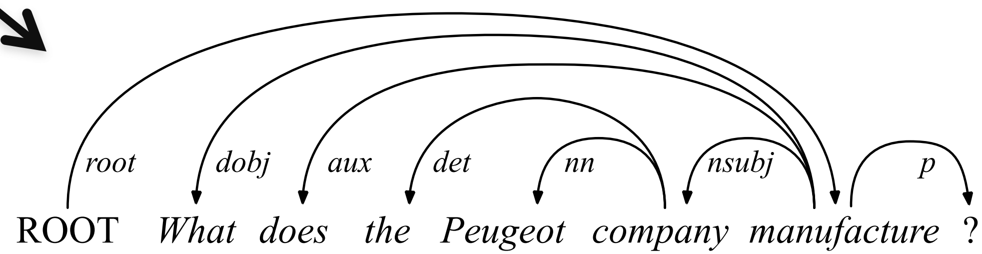
- Phrase structure parsing organizes syntax into *constituents* or *brackets*
- In general, this involves nested trees
- Linguists can, and do, argue about details
- Lots of ambiguity
- Not the only kind of syntax…
- First part of today's lecture



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:

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

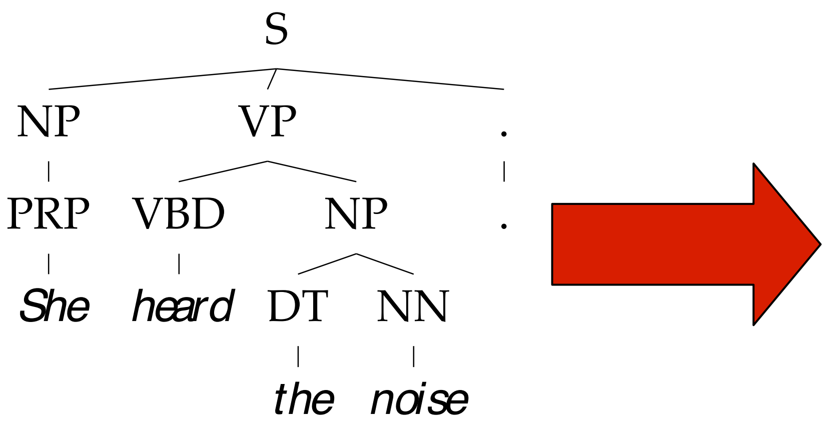
- I cleaned the dishes from dinner
- I cleaned the dishes with detergent
- I cleaned the dishes in my pajamas
- I cleaned the dishes in the sink

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 $P(Y1\ Y2 \ldots Yk \mid X)$

Treebank Grammars

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



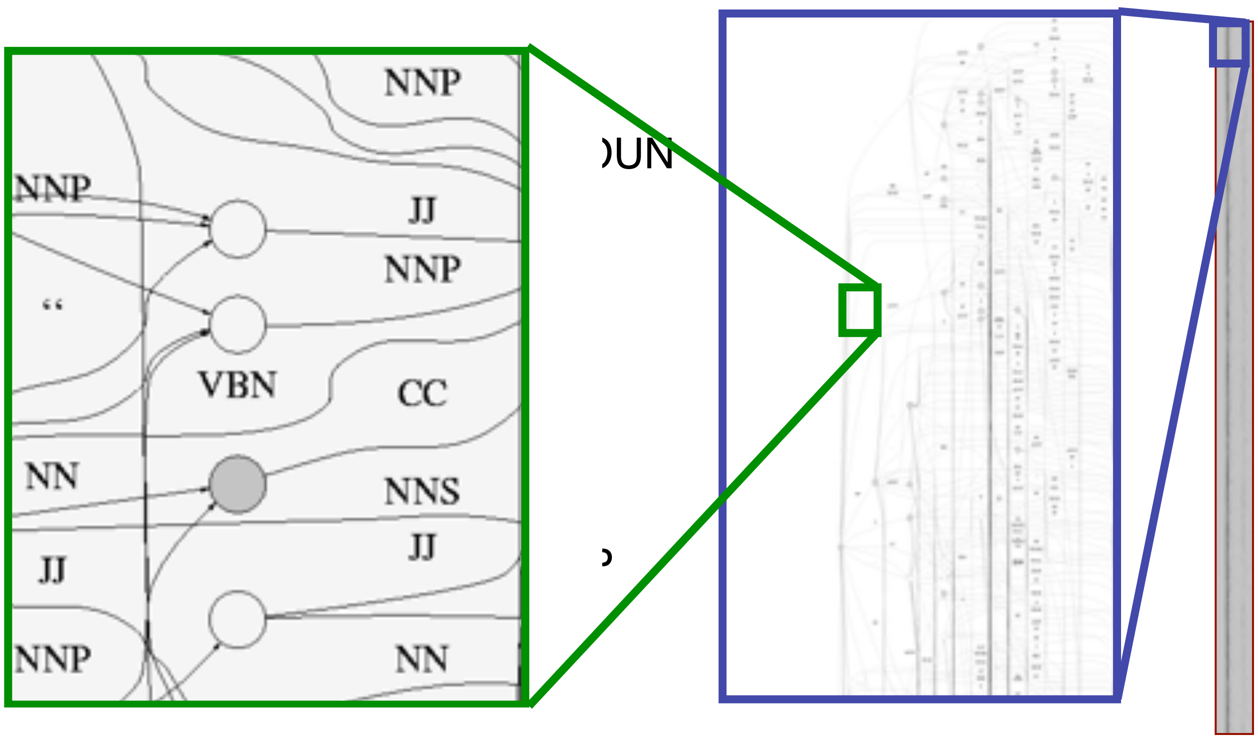
S → NP VP .	1.0
NP → PRP	0.5
NP → DT NN	0.5
VP → VBD NP	1.0
PRP → She	1.0
…	

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

Treebank Grammar Scale

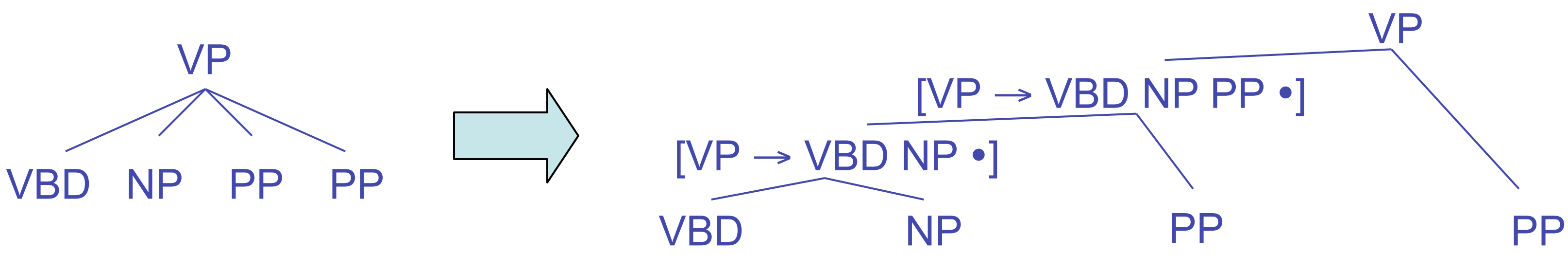
- Treebank grammars can be enormous
  - As FSAs, the raw grammar has ~10K states, excluding the lexicon
  - Better parsers usually make the grammars larger, not smaller

NP



Chomsky Normal Form

- Chomsky normal form:
  - All rules of the form X → 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

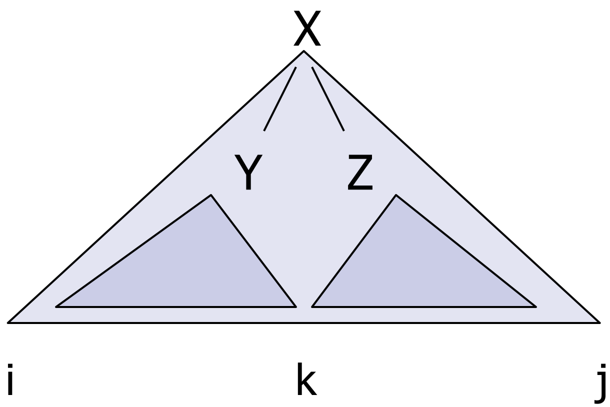
- One small change:

```
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

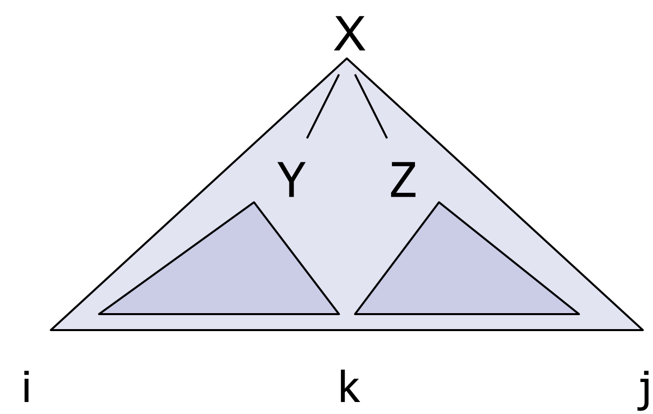
```
bestScore(s)
  for (i : [0,n-1])
      for (X : tags[s[i]])
      score[X][i][i+1] =
          tagScore(X,s[i])
  for (diff : [2,n])
    for (i : [0,n-diff])
      j = i + diff
      for (X->YZ : rule)
        for (k : [i+1, j-1])
       score[X][i][j] = max score[X][i][j],
                            score(X->YZ) *
                            score[Y][i][k] *
                            score[Z][k][j]
```



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

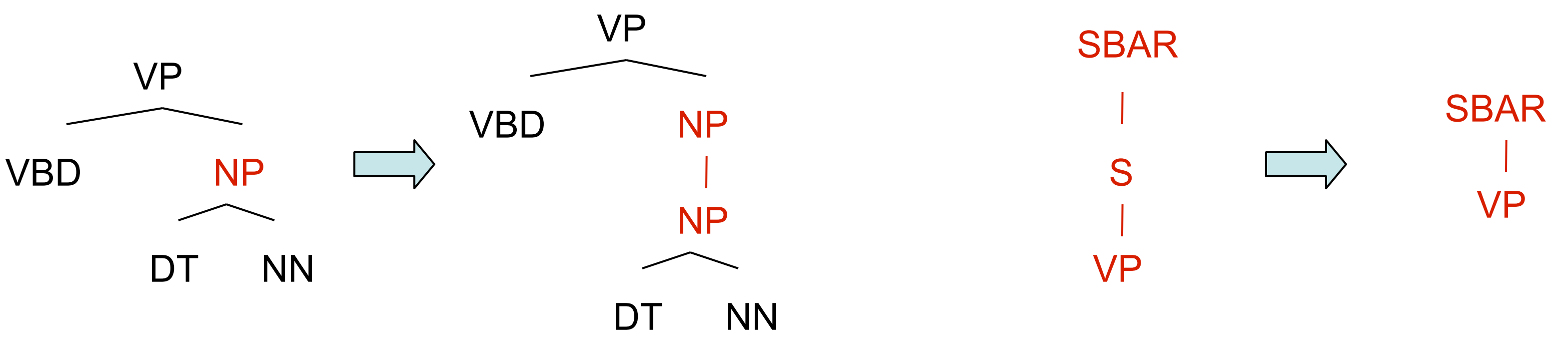
Unary Rules

- 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

Alternating Layers

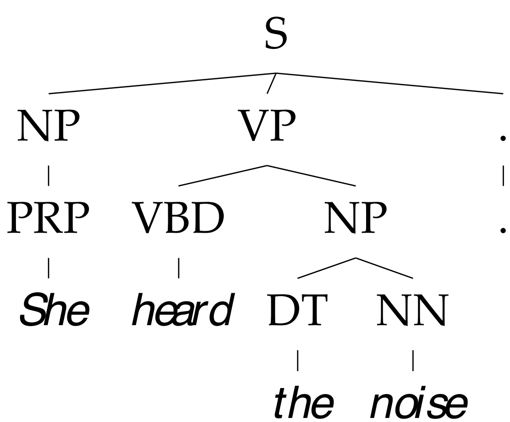
```
bestScoreB(X,i,j,s)
      return max max score(X->YZ) *
     bestScoreU(Y,i,k) *
     bestScoreU(Z,k,j)


bestScoreU(X,i,j,s)
     if (j = i+1)
      return tagScore(X,s[i])
     else
      return max max score(X->Y) *
                  bestScoreB(Y,i,j)
```

Treebank Grammars

[Charniak '96]

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



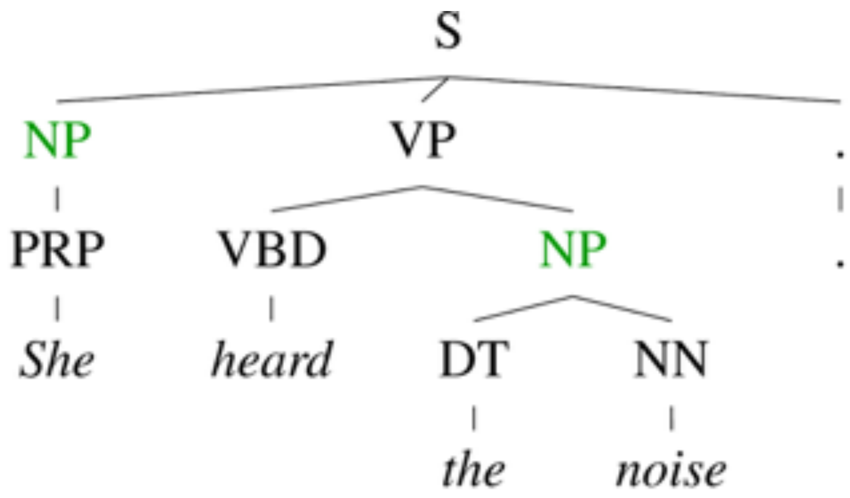
S → NP VP .	1.0
NP → PRP	0.5
NP → DT NN	0.5
VP → VBD NP	1.0
PRP → She	1.0
…	

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

Model	*F1*
Charniak '96	72.0

Conditional Independence?

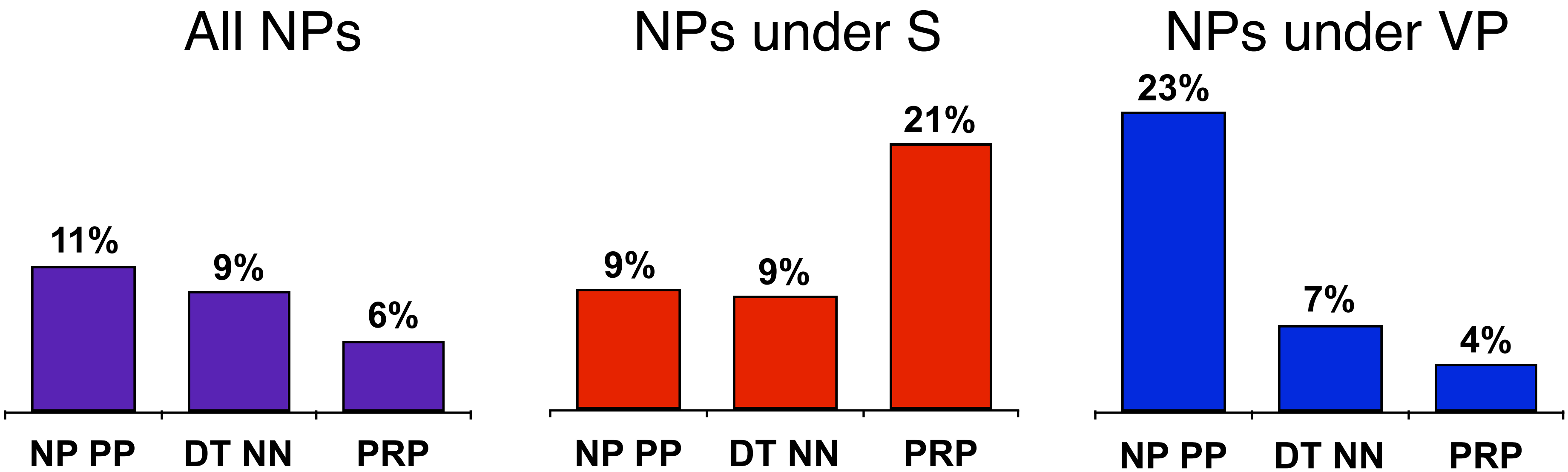
- Not every NP expansion can fill every NP slot



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

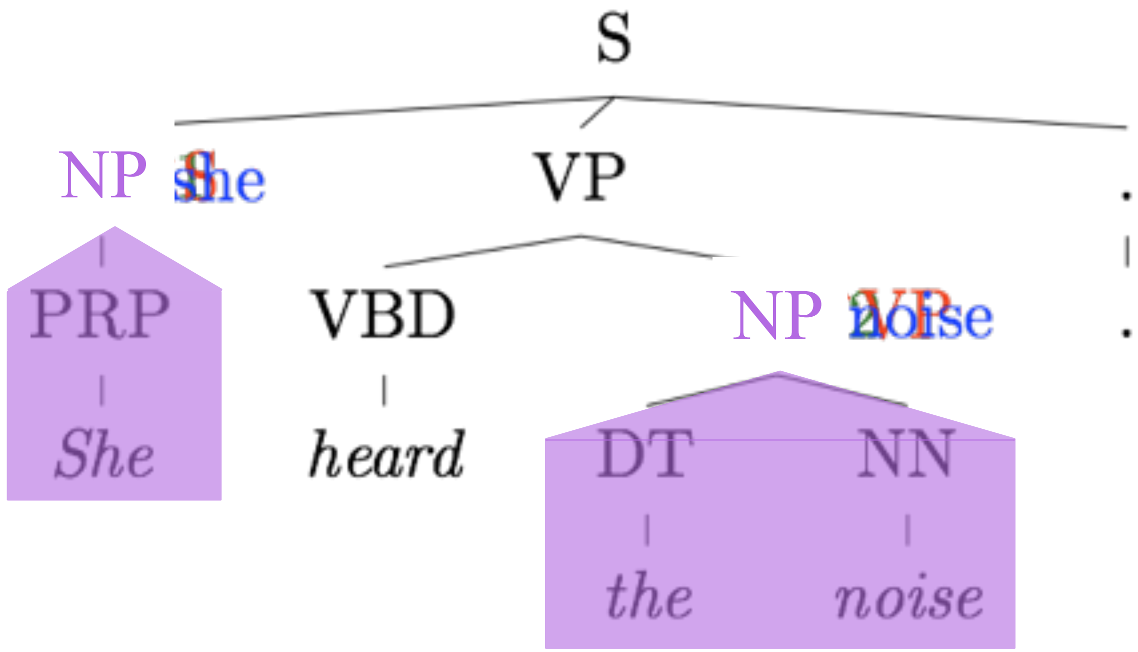
Non-Independence

- 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!

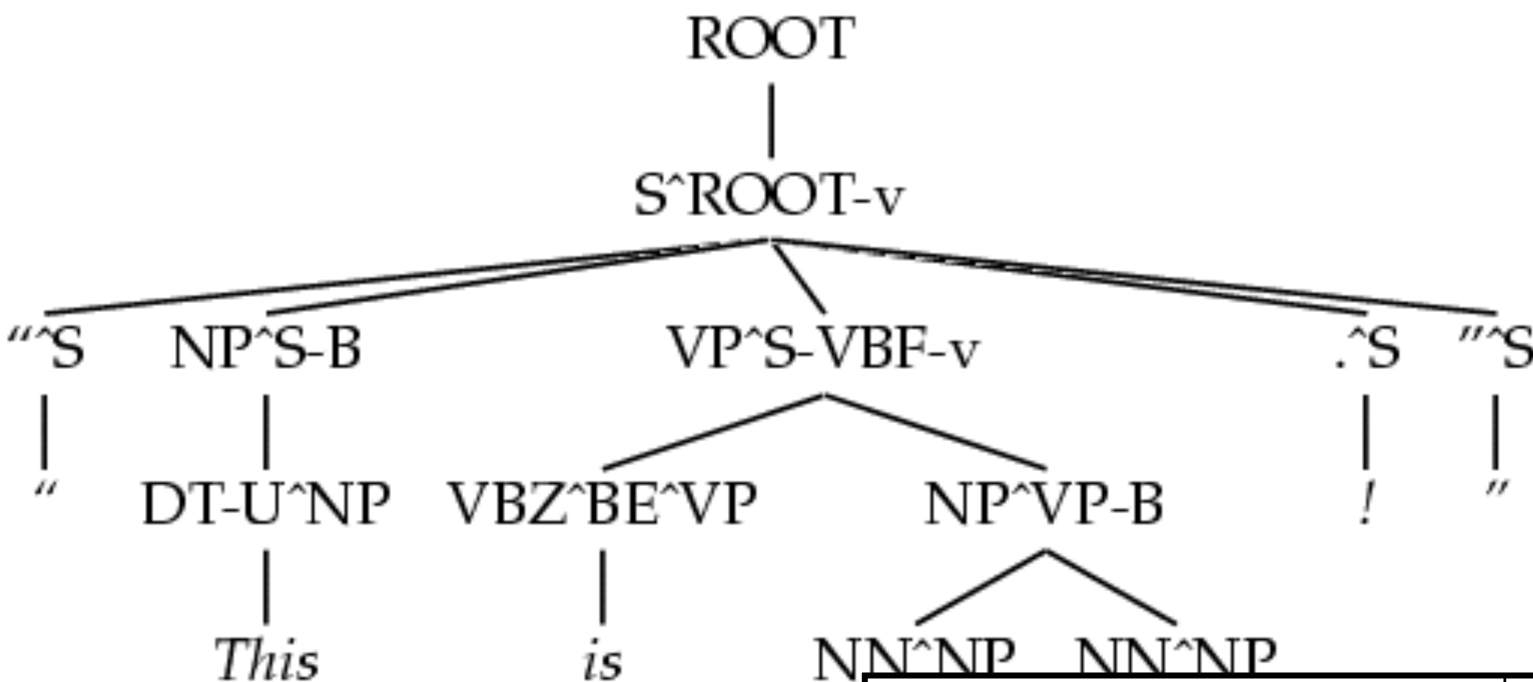
The Game of Designing a Grammar



- Structure Annotation [Johnson ’98, Klein & Manning ’03]
- Lexicalization [Collins ’99, Charniak ’00]
- Latent Variables [Matsuzaki et al. 05, Petrov et al. ’06]
- (Neural) CRF Parsing [Hall et al. ’14, Durrett & Klein ’15]

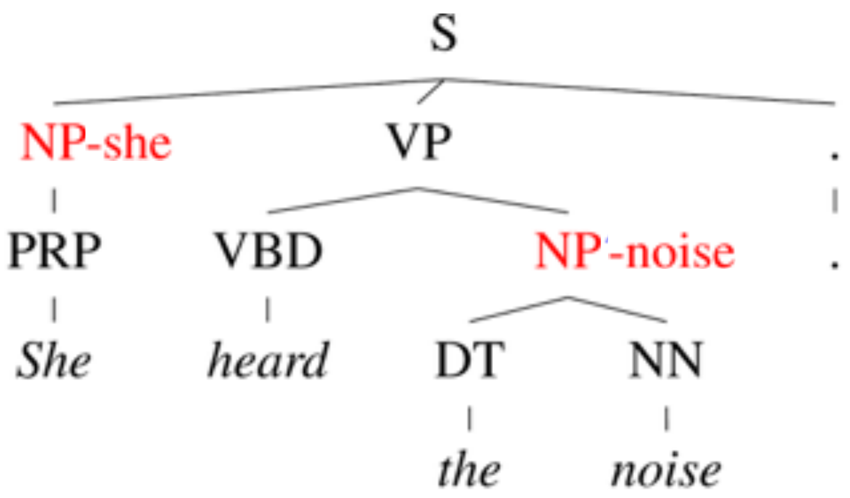
A Fully Annotated (Unlexicalized) Tree

[Klein & Manning '03]



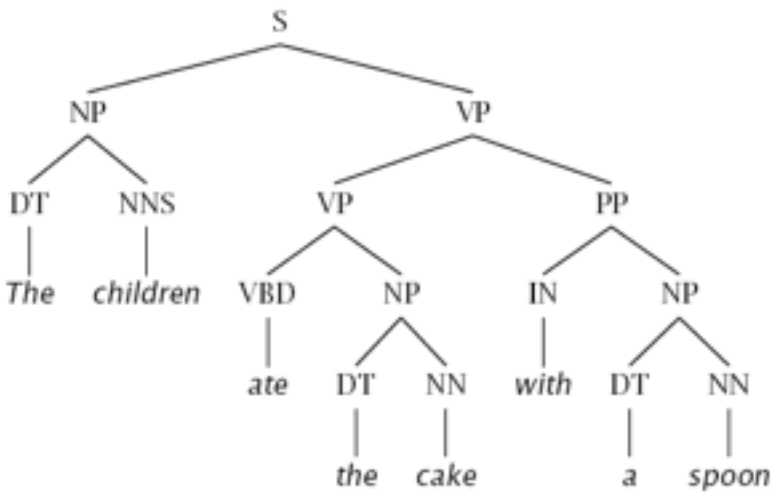
Model	*F1*
Charniak '96	72.0
Klein&Manning '03	86.3

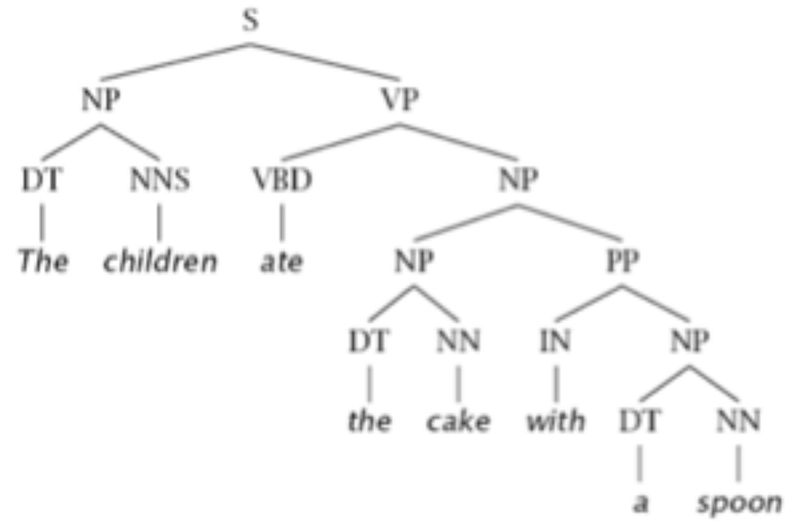
The Game of Designing a Grammar



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

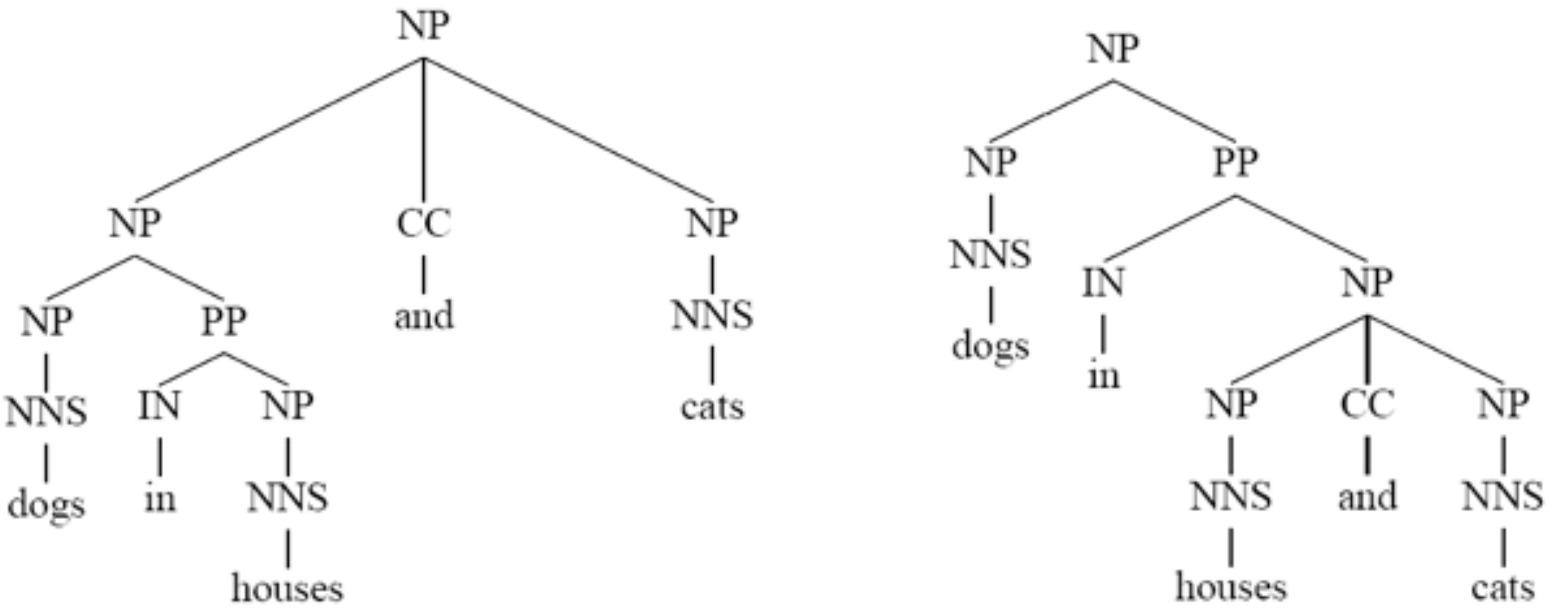
Problems with PCFGs





- 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

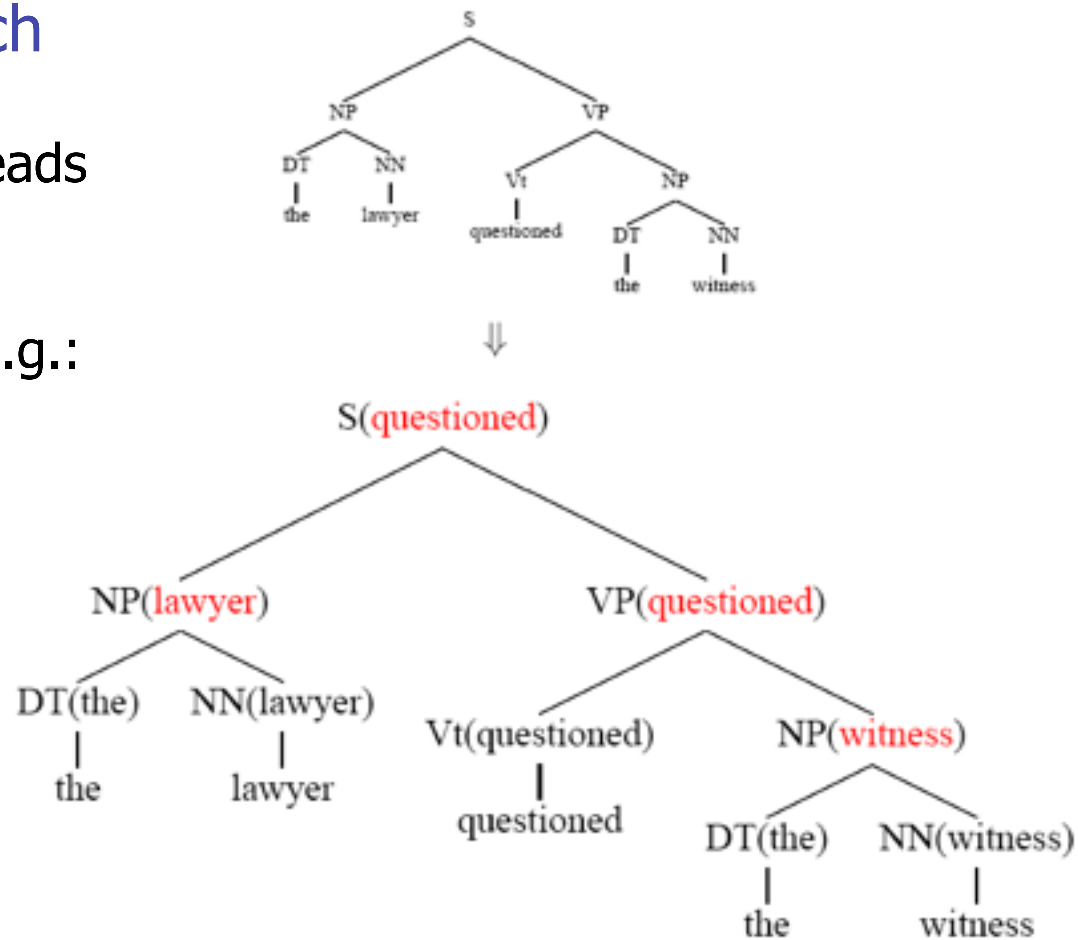


- What's different between basic PCFG scores here?
- What (lexical) correlations need to be scored?

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)
```

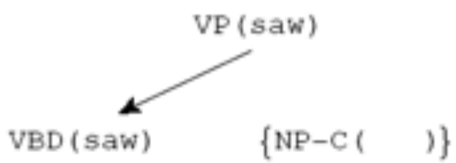
- Never going to get these atomically off of a treebank

- Solution: break up derivation into smaller steps

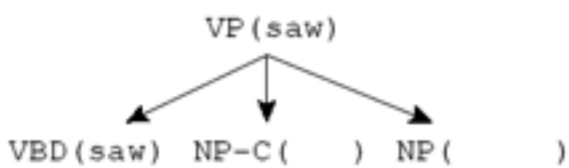
Lexical Derivation Steps

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

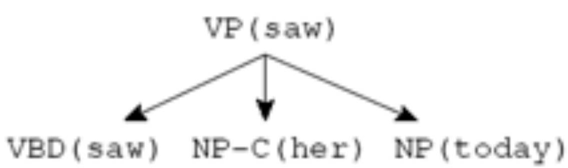
Choose a head tag and word



Choose a complement bag



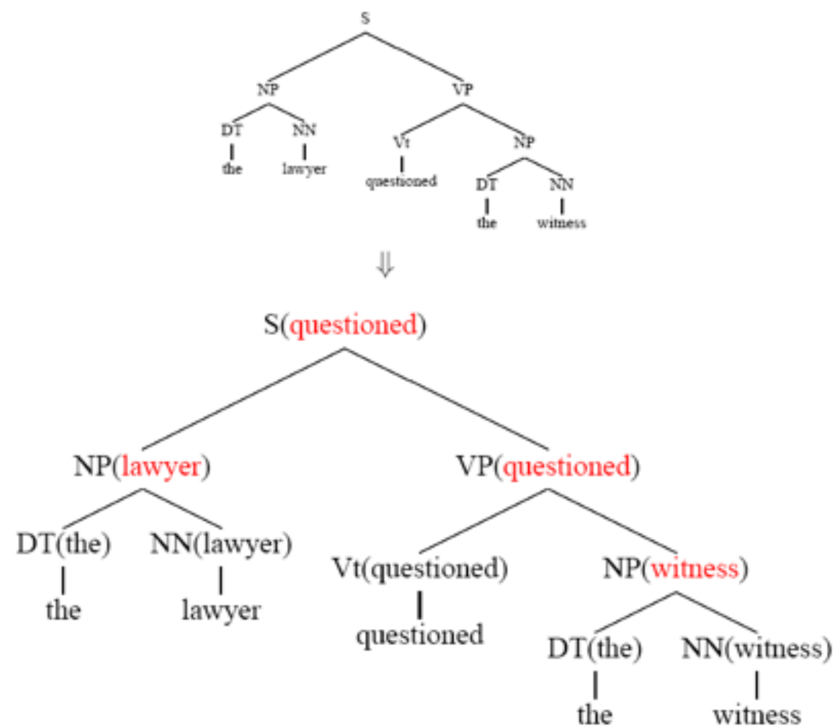
Generate children (incl. adjuncts)



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

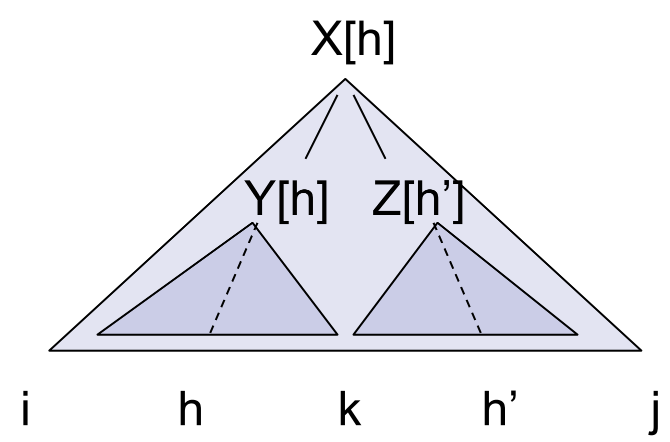


Model	*F1*
Klein&Manning ’03	86.3
Charniak ’00	90.1

Lexicalized CKY

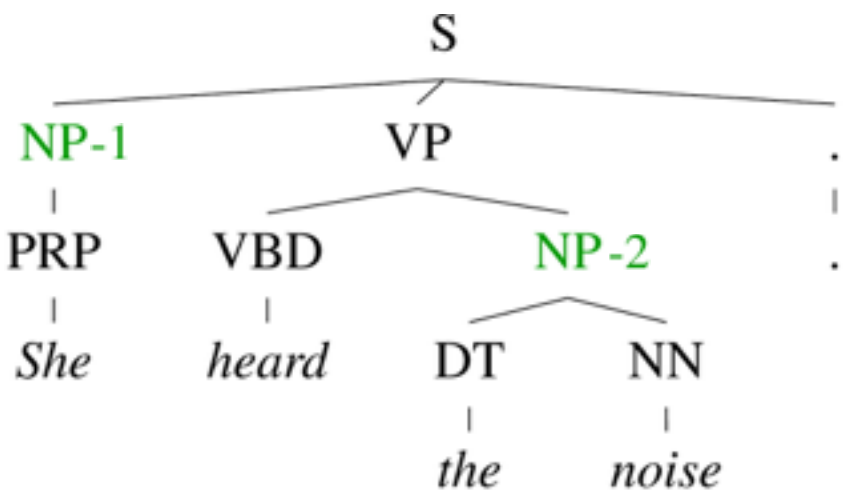
(VP->VBD...NP •)[saw]

(VP->VBD •)[saw] NP[her]




```
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)
```

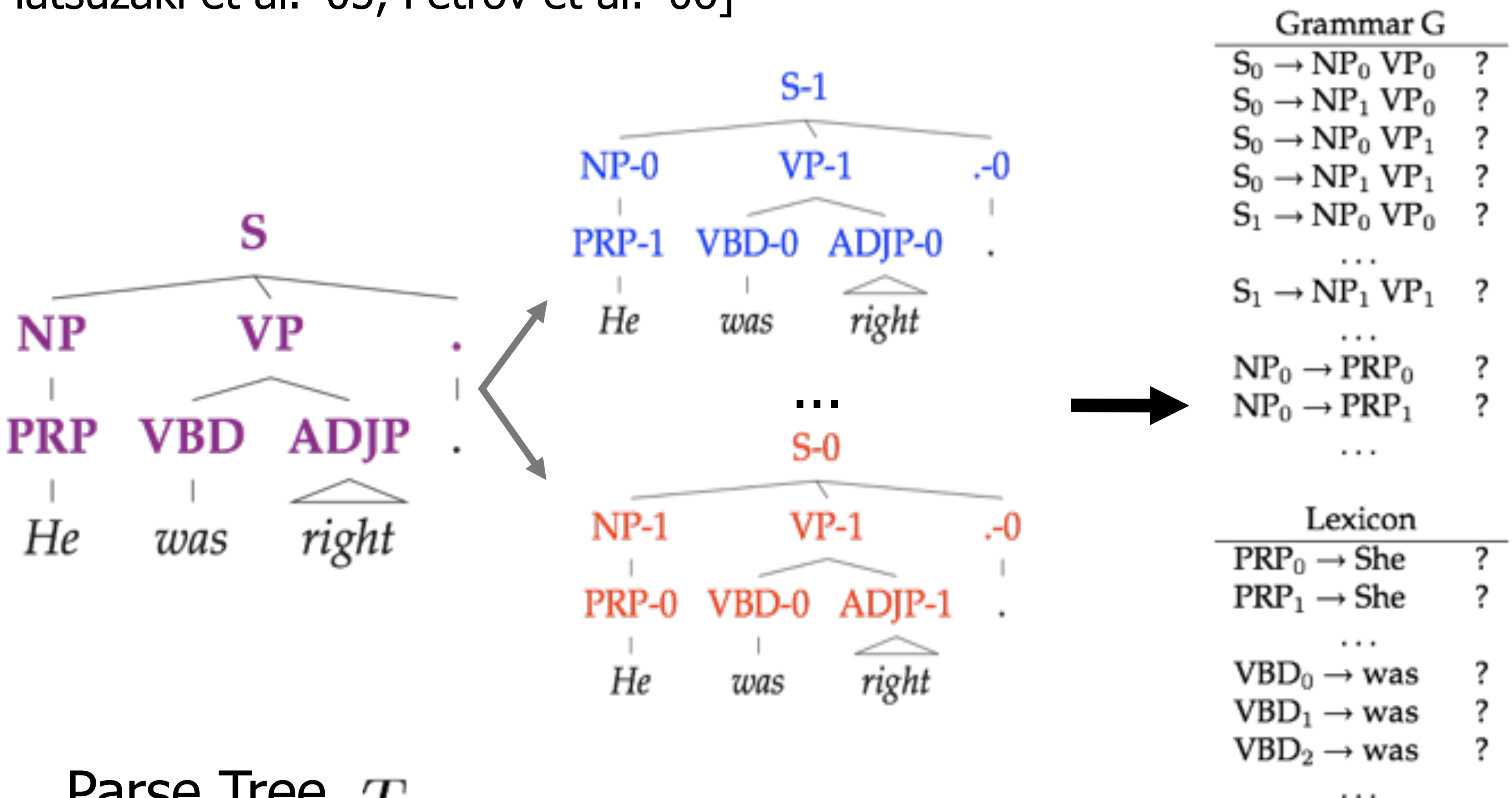
The Game of Designing a Grammar



- 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

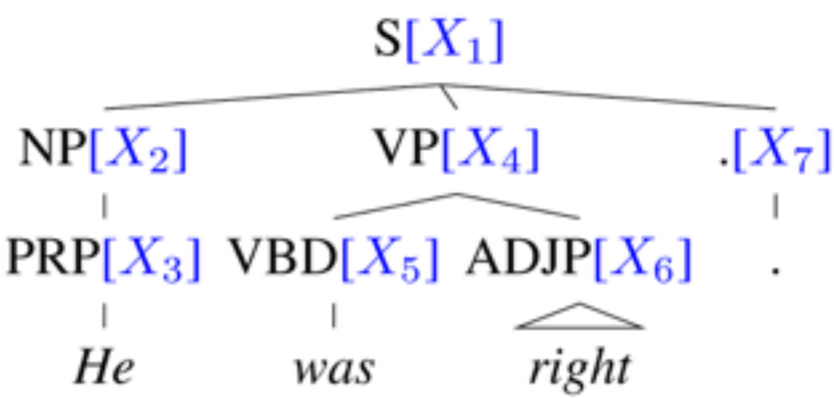
Derivations $t : T$

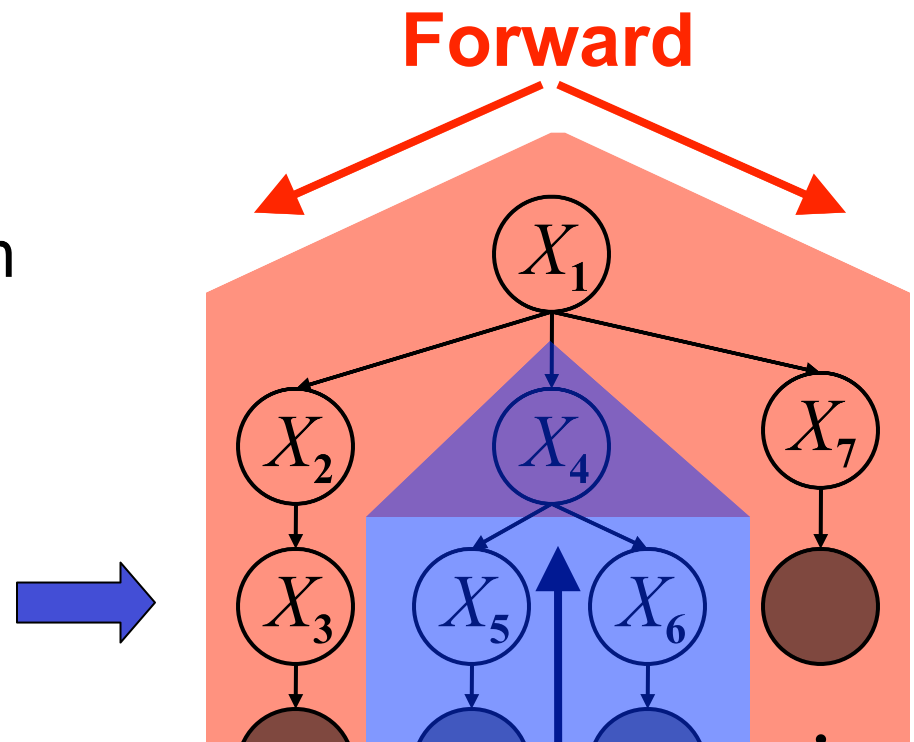
Parameters θ

Learning Latent Annotations

EM algorithm:

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

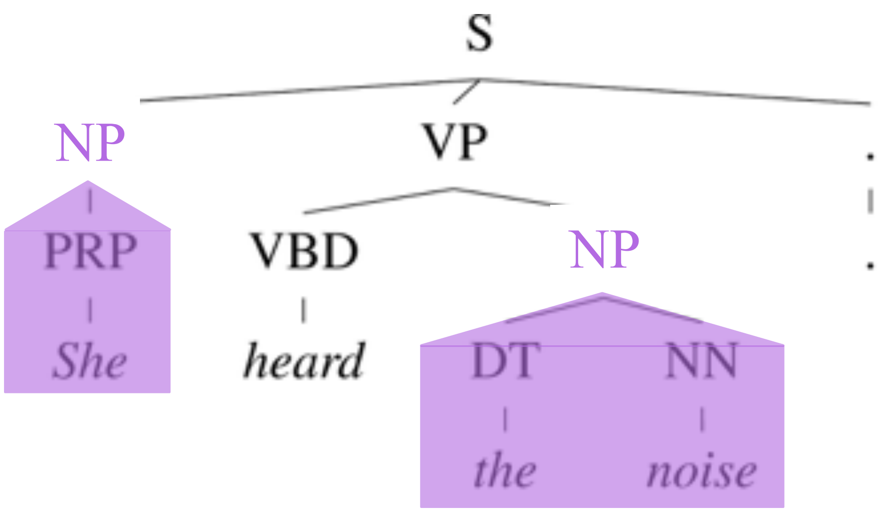




Just like Forward-Backwa
for HMMs.

Model	*F1*
Charniak '00	90.1
Petrov et al. '06	90.6

The Game of Designing a Grammar

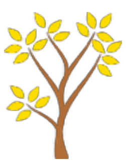


- Annotation refines base treebank symbols to improve statistical fit of the grammar
  - CRF Parsing (+Neural Network Representations)

Generative vs. Discriminative

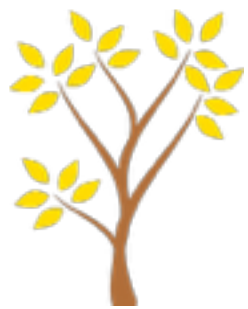
Generative

Maximize joint likelihood
of gold tree **and** sentence



$w_1\ w_2 \ldots w_n$

EM-algorithm



$w_1\ w_2 \ldots w_n$

EASY: expectations over observed trees

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

Discriminative

Maximize conditional likelihood
of gold tree **given** sentence

$w_1\ w_2 \ldots w_n$

Gradient-based algorithm

HARD: expectations over all trees

[Petrov & Klein '07, '08]

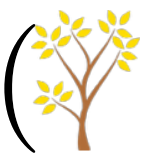
Objective Functions

Generative Objective Function:

$\max_{\theta} \; \mathcal{L}_{\theta}\big($  $, w_1 \ldots w_n\big)$

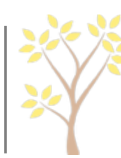
[Petrov, Barrett, Thibaux & Klein ’06]

Discriminative Objective Function:

$\max_{\theta} \; \mathcal{L}_{\theta}\big($  $\big| \; w_1 \ldots w_n\big)$

[Petrov & Klein ’08, Finkel et. al ’08]

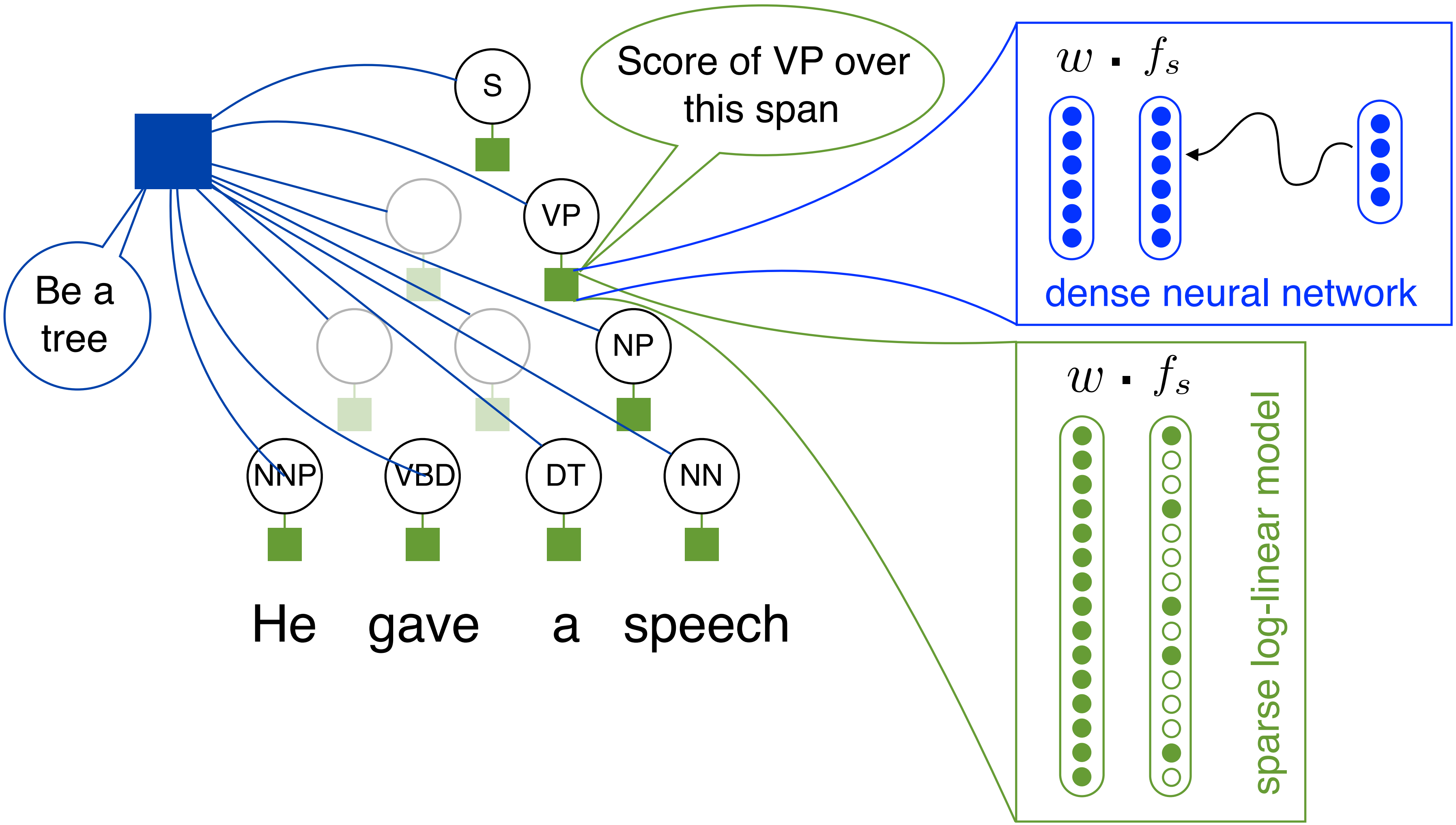
Bayesian Objective Function:

$\max_{\theta} \; \mathcal{P}\big(\theta \,\big|$  $\big)\mathcal{L}_{\theta}\big($  $, w_1 \ldots w_n\big)$

[Liang, Petrov, Jordan & Klein ’07]

(Neural) CRF Parsing

[Taskar et al. ’04, Petrov & Klein ’07, Hall et al. ’14, Durrett et al. ’15]



CRF Parsing Sparse Features

$$P(T|x) \propto \prod_{r \in T} \exp\left(\text{score}(r)\right)$$

$$\text{score}({}_2\text{NP}_7 \rightarrow {}_2\text{NP}_4\ {}_4\text{PP}_7) = w^\top f({}_2\text{NP}_7 \rightarrow {}_2\text{NP}_4\ {}_4\text{PP}_7)$$

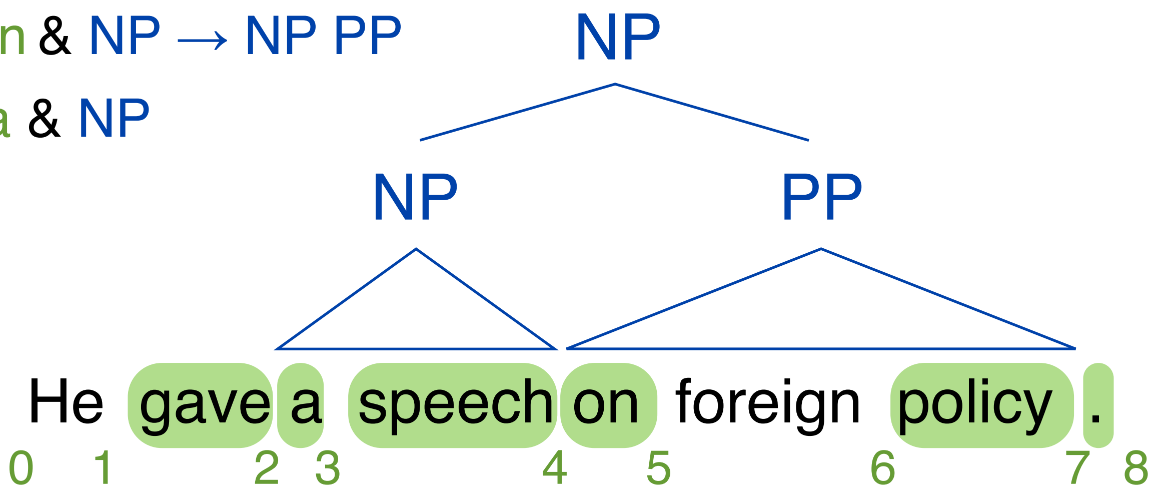
FirstWord = a & NP $\rightarrow$ NP PP

PrevWord = gave & NP $\rightarrow$ NP PP

AfterSplit = on & NP $\rightarrow$ NP PP

FirstWord = a & NP

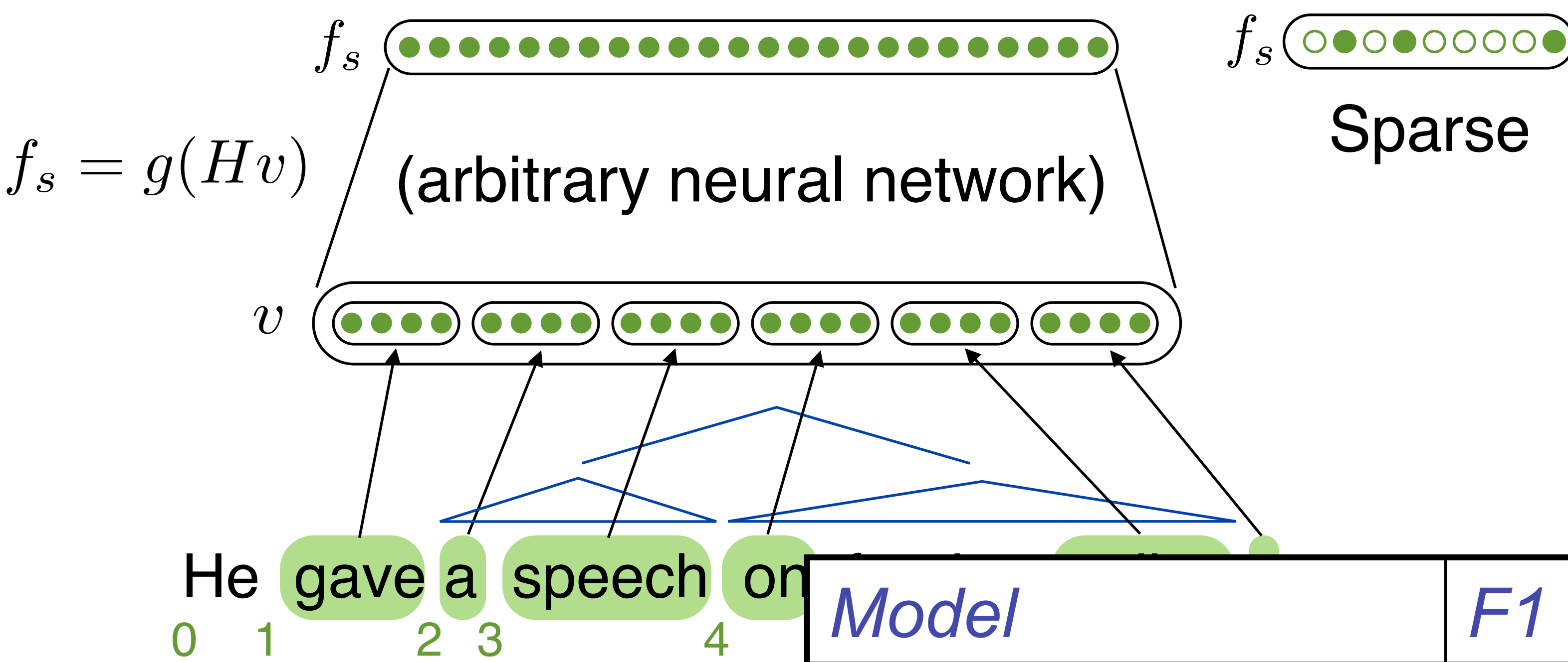
…



Neural CRF Model

$$\text{score}({}_2\text{NP}_7 \rightarrow {}_2\text{NP}_4\ {}_4\text{PP}_7) =$$

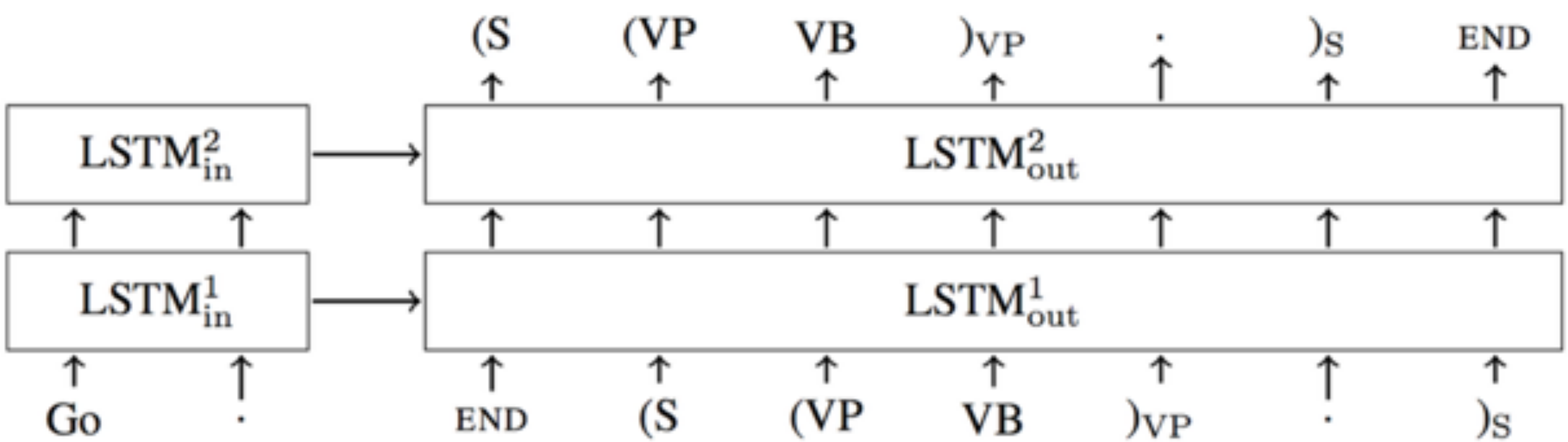
$$W \odot \left(f_s({}_2\text{X}_7 \rightarrow {}_2\text{X}_4\ {}_4\text{X}_7)\, f_o^\top(\text{NP} \rightarrow \text{NP PP}) \right)$$



Model	*F1*
Petrov et al. '06	90.6
Durrett et al. '16	91.3

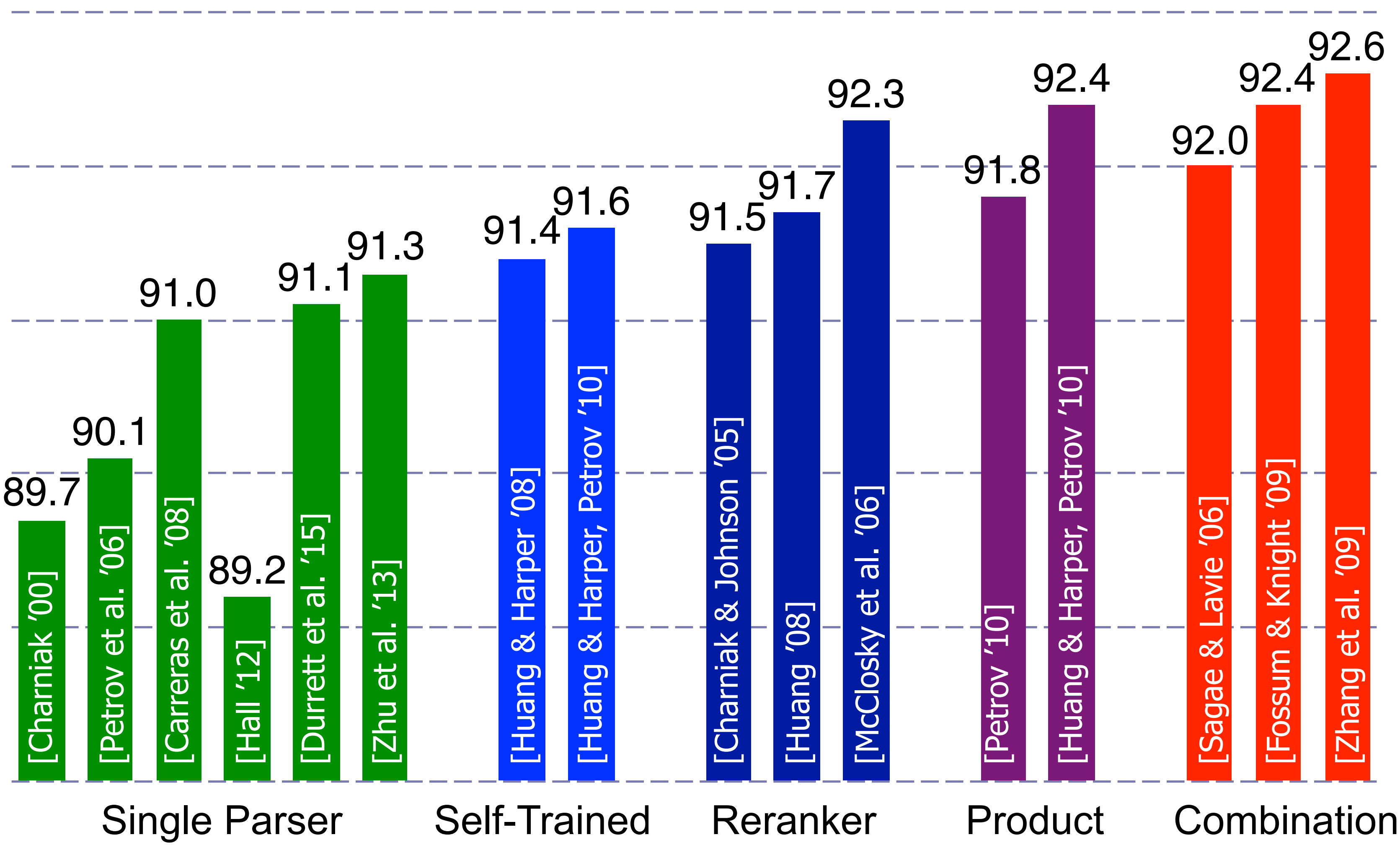
LSTM Parsing [Vinyals et al. '15]

- Treat parsing as a sequence-to-sequence prediction problem
- Completely ignores tree structure, uses LSTMs as black boxes



$$P(y_1, \ldots, y_{T'} | x_1, \ldots, x_T) = \prod_{t=1}^{T'} p(y_t | v, y_1, \ldots, y_{t-1})$$

Detailed English Results



Multi-Lingual Results