# Natural Language Processing: Statistical Parsing

Reading: Chap 13, Jurafsky & Martin

## Statistical Parsing

- Statistical parsing uses a probabilistic model of syntax in order to assign probabilities to each parse tree
- Provides principled approach to resolving syntactic ambiguity
- Allows supervised learning of parsers from tree-banks of parse trees provided by human linguists
- Also allows unsupervised learning of parsers from unannotated text, but the accuracy of such parsers has been limited

## Probabilistic Context Free Grammar (PCFG)

- PCFG: probabilistic version of a CFG where each production has a probability
- Probabilities of all productions rewriting a given non-terminal must add to 1, defining a distribution for each non-terminal
- String generation is now probabilistic where production probabilities are used to non-deterministically select a production for rewriting a given non-terminal

# Simple PCFG for ATIS English

| Grammar | Prob | |
|---|---|---|
| S → NP VP | 0.8 | |
| S → Aux NP VP | 0.1 | + 1.0 |
| S → VP | 0.1 | |
| NP → Pronoun | 0.2 | |
| NP → Proper-Noun | 0.2 | + 1.0 |
| NP → Det Nominal | 0.6 | |
| Nominal → Noun | 0.3 | |
| Nominal → Nominal Noun | 0.2 | + 1.0 |
| Nominal → Nominal PP | 0.5 | |
| VP → Verb | 0.2 | |
| VP → Verb NP | 0.5 | + 1.0 |
| VP → VP PP | 0.3 | |
| PP → Prep NP | 1.0 | |

## Lexicon

Det → the | a | that | this
0.6 0.2 0.1 0.1

Noun → book | flight | meal | money
0.1 0.5 0.2 0.2

Verb → book | include | prefer
0.5 0.2 0.3

Pronoun → I | he | she | me
0.5 0.1 0.1 0.3

Proper-Noun → Houston | NWA
0.8 0.2

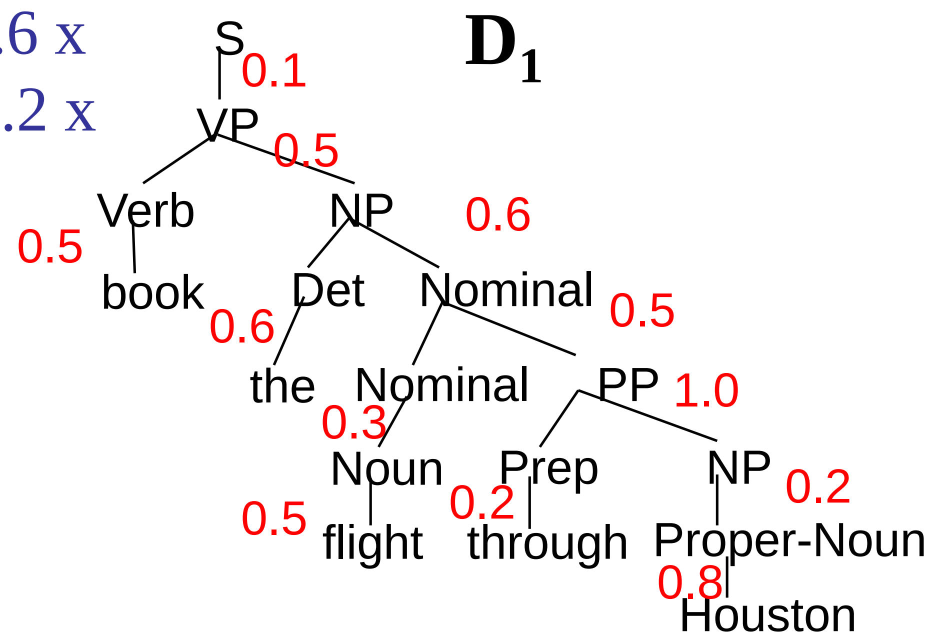
Aux → does
1.0

Prep → from | to | on | near | through
0.25 0.25 0.1 0.2 0.2

# Sentence Probability

- Assume productions for each node are *chosen independently*
- Probability of derivation is the product of the probabilities of its productions

$P(D_1) = 0.1 \times 0.5 \times 0.5 \times 0.6 \times 0.6 \times$
$0.5 \times 0.3 \times 1.0 \times 0.2 \times 0.2 \times$
$0.5 \times 0.8$
$= 0.0000216$

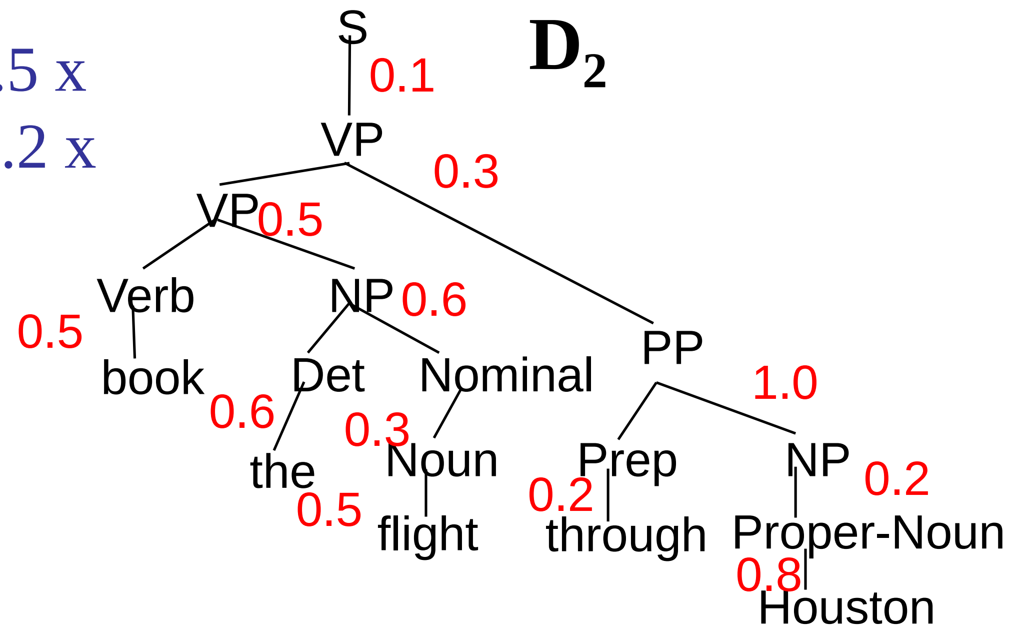


## Syntactic Disambiguation

- Resolve ambiguity by picking most probable parse tree

$P(D_2) = 0.1 \times 0.3 \times 0.5 \times 0.6 \times 0.5 \times 0.6 \times 0.3 \times 1.0 \times 0.5 \times 0.2 \times 0.2 \times 0.8$

$= 0.00001296$



## Sentence Probability

- Probability of a sentence is the sum of the probabilities of all of its derivations

P("book the flight through Houston") =
$P(D_1) + P(D_2) = 0.0000216 + 0.00001296$
$= 0.00003456$

## Three Useful PCFG Tasks

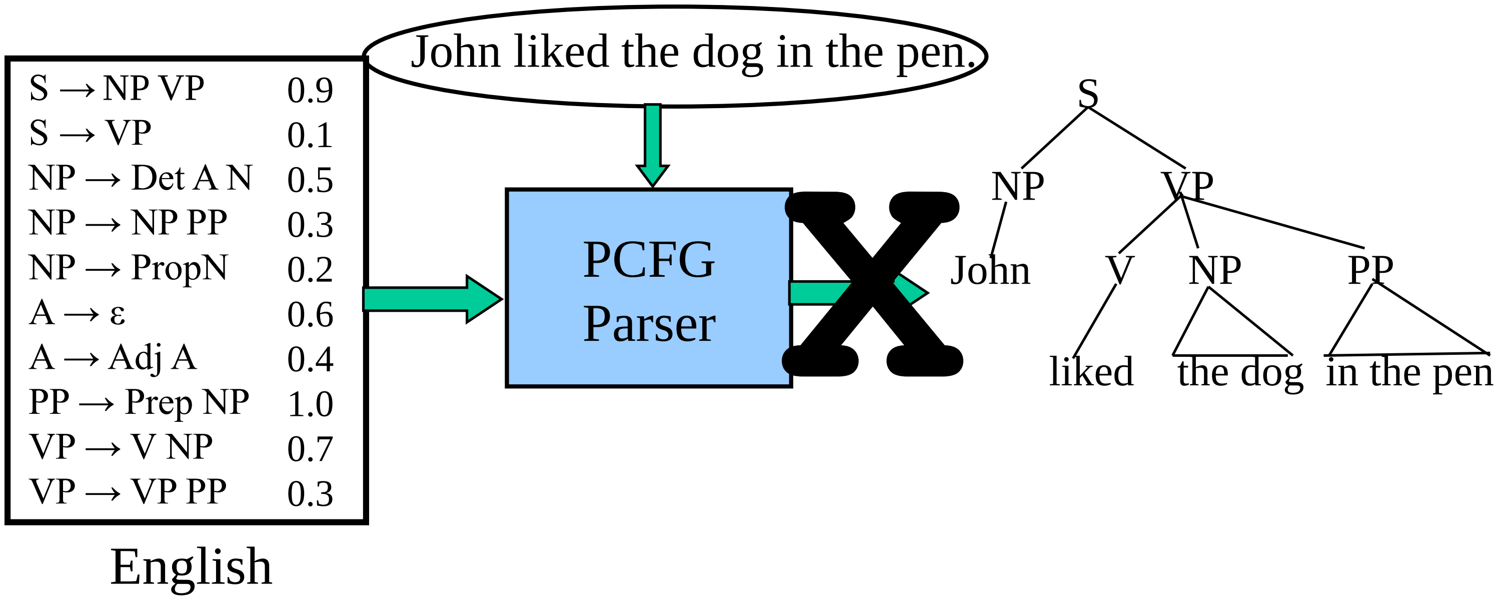
- Observation likelihood: To classify and order sentences

- Most likely derivation: To determine the most likely parse tree for a sentence

- Maximum likelihood training: To train a PCFG to fit empirical training data

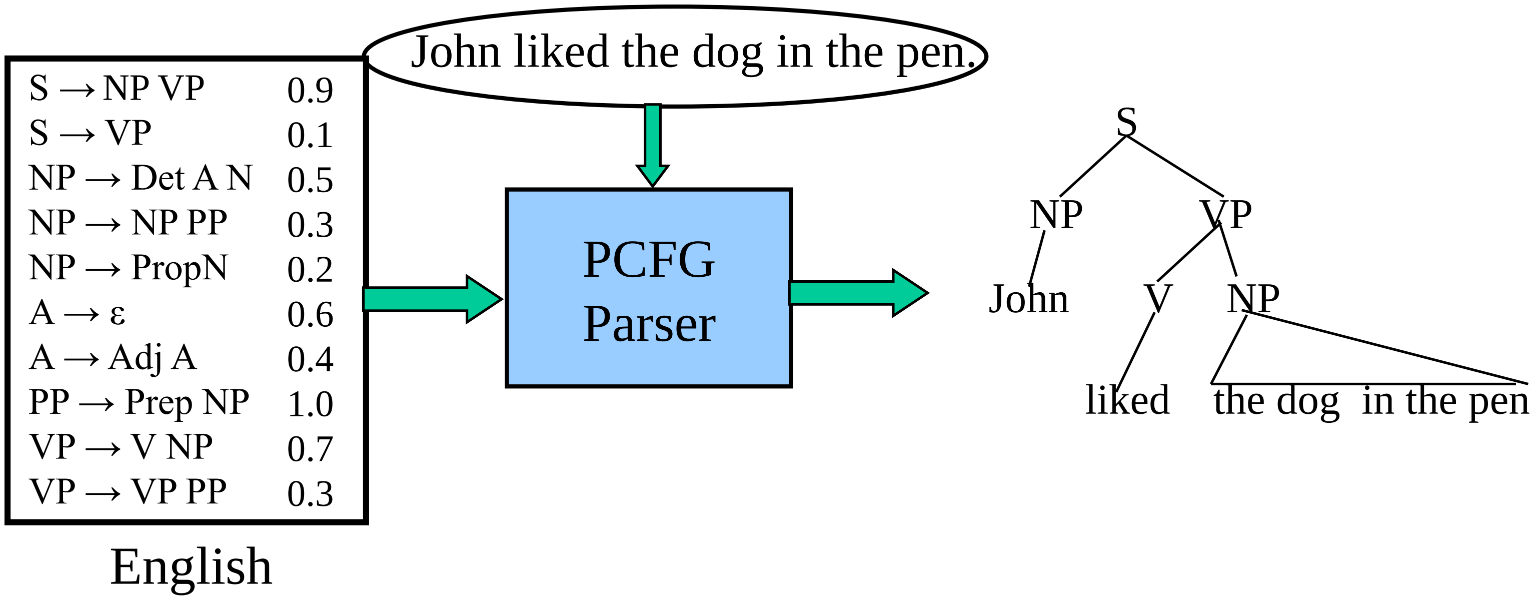
# PCFG: Most Likely Derivation

- There is an analog to the Viterbi algorithm to efficiently determine the most probable derivation (parse tree) for a sentence



# PCFG: Most Likely Derivation

- There is an analog to the Viterbi algorithm to efficiently determine the most probable derivation (parse tree) for a sentence



| | |
|---|---|
| S → NP VP | 0.9 |
| S → VP | 0.1 |
| NP → Det A N | 0.5 |
| NP → NP PP | 0.3 |
| NP → PropN | 0.2 |
| A → ε | 0.6 |
| A → Adj A | 0.4 |
| PP → Prep NP | 1.0 |
| VP → V NP | 0.7 |
| VP → VP PP | 0.3 |

English

## Probabilistic CKY: Cocke-Kasami-Younger

- CKY (**bottom up parsing method**) can be modified for PCFG parsing by including in each cell a probability for each non-terminal
- Cell[$i$,$j$] must retain the ***most probable*** derivation of each constituent (**non-terminal**) covering words $i$ +1 through $j$ together with its associated probability
- When transforming the grammar to CNF, must set production probabilities to preserve the probability of derivations

## Why normal forms?

- If all productions of the grammar could be expressed in the same form(s), then:

a. It becomes easy to design algorithms that use the grammar

b. It becomes easy to show proofs and properties

## Chomsky Normal Form

- A CFG is said to be in *Chomsky Normal Form* if every production is one of these two forms:
  1. A -> BC (right side is two variables)
  2. A -> a (right side is a single terminal)
- Theorem: If L is a CFL, then L – {ε} has a CFG in CNF

# Proof of CNF Theorem

- Step 1: "Clean" the grammar, so every production right side is either a single terminal or of length at least 2
- Step 2: For each right side $\neq$ a single terminal, make the right side all variables.
  - For each terminal *a* create new variable $A_a$ and production $A_a \to a$.
  - Replace *a* by $A_a$ in right sides of length > 2.

## Example: Step 2

- Consider production A -> BcDe
- We need variables $A_c$ and $A_e$. with productions $A_c$ -> c and $A_e$ -> e
  - Note: you create at most one variable for each terminal, and use it everywhere it is needed.
- Replace A -> BcDe by A -> $BA_cDA_e$

## CNF Proof – Continued

- Step 3: Break right sides longer than 2 into a chain of productions with right sides of two variables
- Example: A -> BCDE is replaced by A -> BF, F -> CG, and G -> DE.
  - F and G must be used nowhere else

## Example of Step 3 – Continued

- Recall A -> BCDE is replaced by

  A -> BF, F -> CG, and G -> DE
- In the new grammar, A => BF => BCG => BCDE
- More importantly: Once we choose to replace A by BF, we must continue to BCG and BCDE
  - Because F and G have only one production

# Probabilistic Grammar Conversion

| Original Grammar | | Chomsky Normal Form | |
|---|---|---|---|
| S → NP VP | 0.8 | S → NP VP | 0.8 |
| S → Aux NP VP | 0.1 | S → X1 VP | 0.1 |
| | | X1 → Aux NP | 1.0 |
| S → VP | 0.1 | S → book \| include \| prefer | |
| | | 0.01 0.004 0.006 | |
| | | S → Verb NP | 0.05 |
| | | S → VP PP | 0.03 |
| NP → Pronoun | 0.2 | NP → I \| he \| she \| me | |
| | | 0.1 0.02 0.02 0.06 | |
| NP → Proper-Noun | 0.2 | NP → Houston \| NWA | |
| | | 0.16 .04 | |
| NP → Det Nominal | 0.6 | NP → Det Nominal | 0.6 |
| Nominal → Noun | 0.3 | Nominal → book \| flight \| meal \| money | |
| | | 0.03 0.15 0.06 0.06 | |
| Nominal → Nominal Noun | 0.2 | Nominal → Nominal Noun | 0.2 |
| Nominal → Nominal PP | 0.5 | Nominal → Nominal PP | 0.5 |
| VP → Verb | 0.2 | VP → book \| include \| prefer | |
| | | 0.1 0.04 0.06 | |
| VP → Verb NP | 0.5 | VP → Verb NP | 0.5 |
| VP → VP PP | 0.3 | VP → VP PP | 0.3 |
| PP → Prep NP | 1.0 | PP → Prep NP | 1.0 |

## Lexicon

Det → that | this | a (0.6)

Noun → book | flight | meat | money (0.5)

Verb → book | include | prefer (0.5)

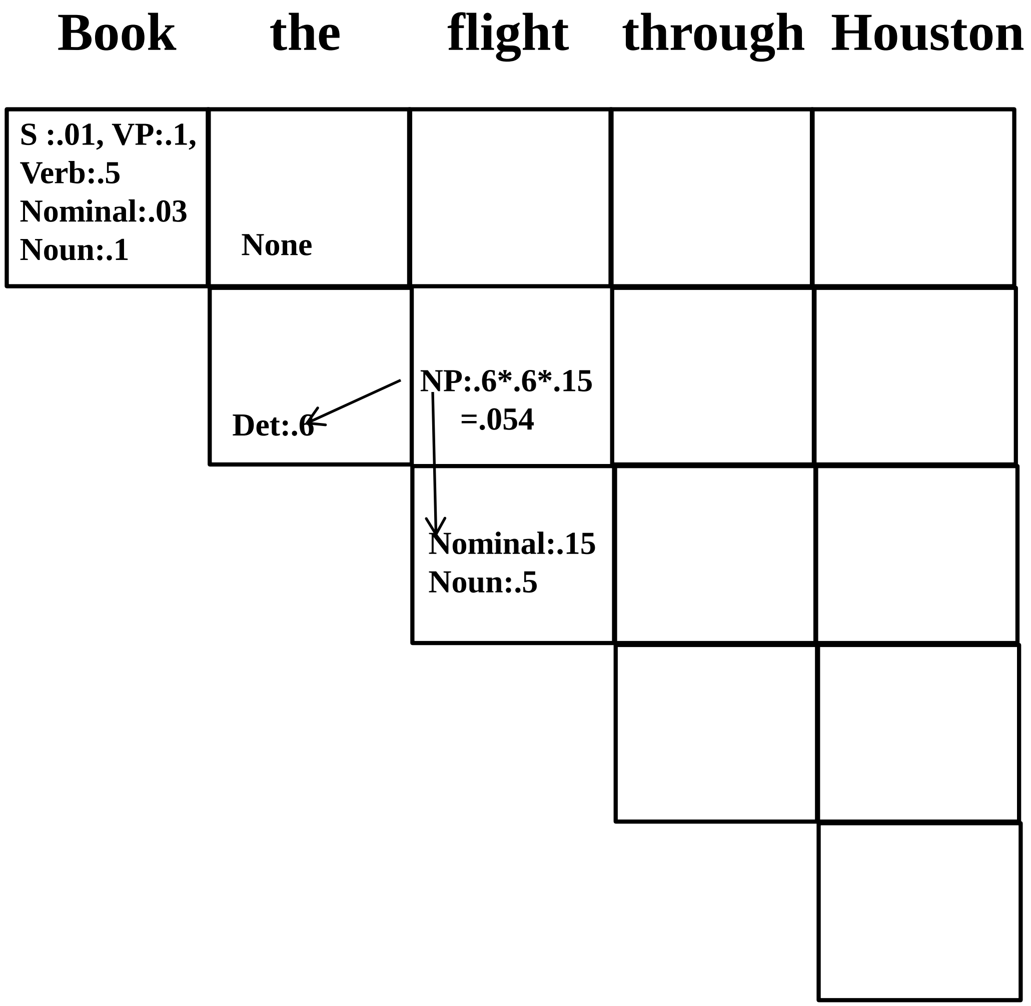
Aux → does

Prep → from | to | on

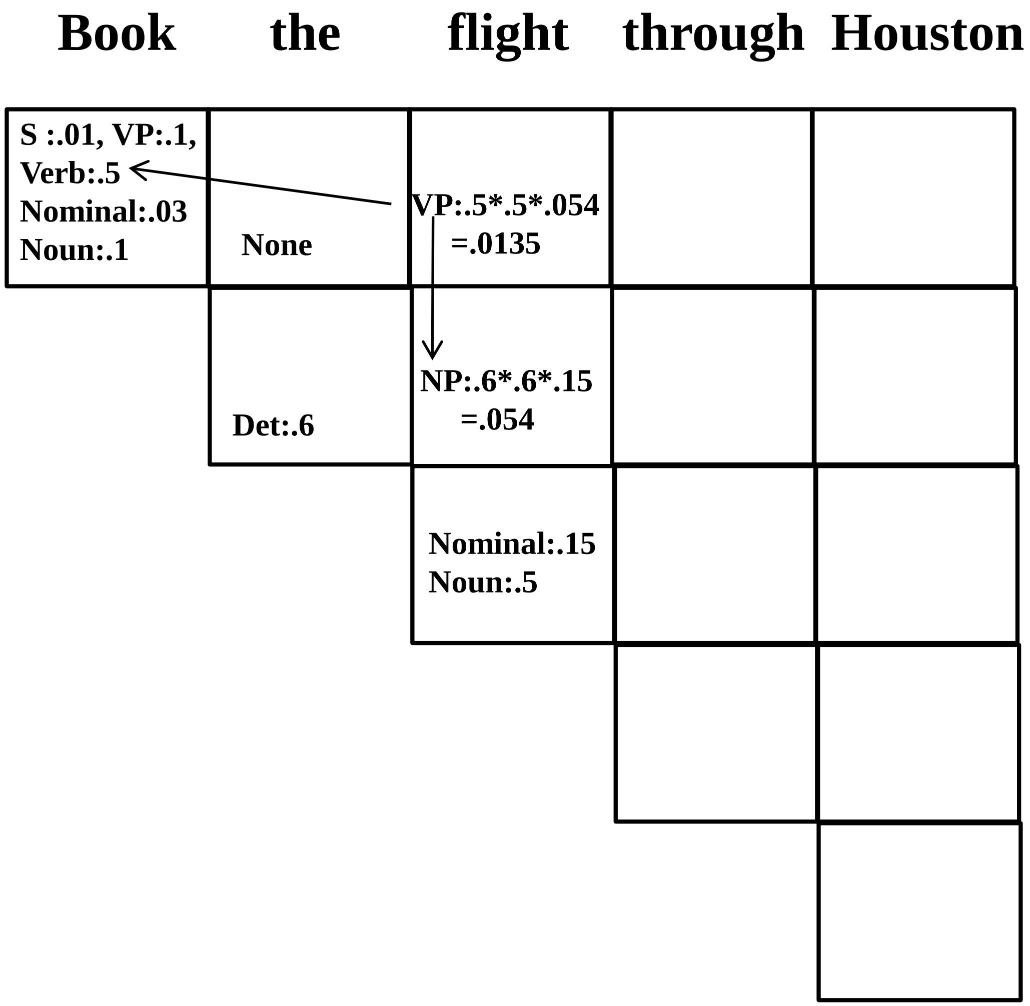
Proper-Noun → Houston | TWA

Nominal → Nominal PP

# Probabilistic CKY Parser

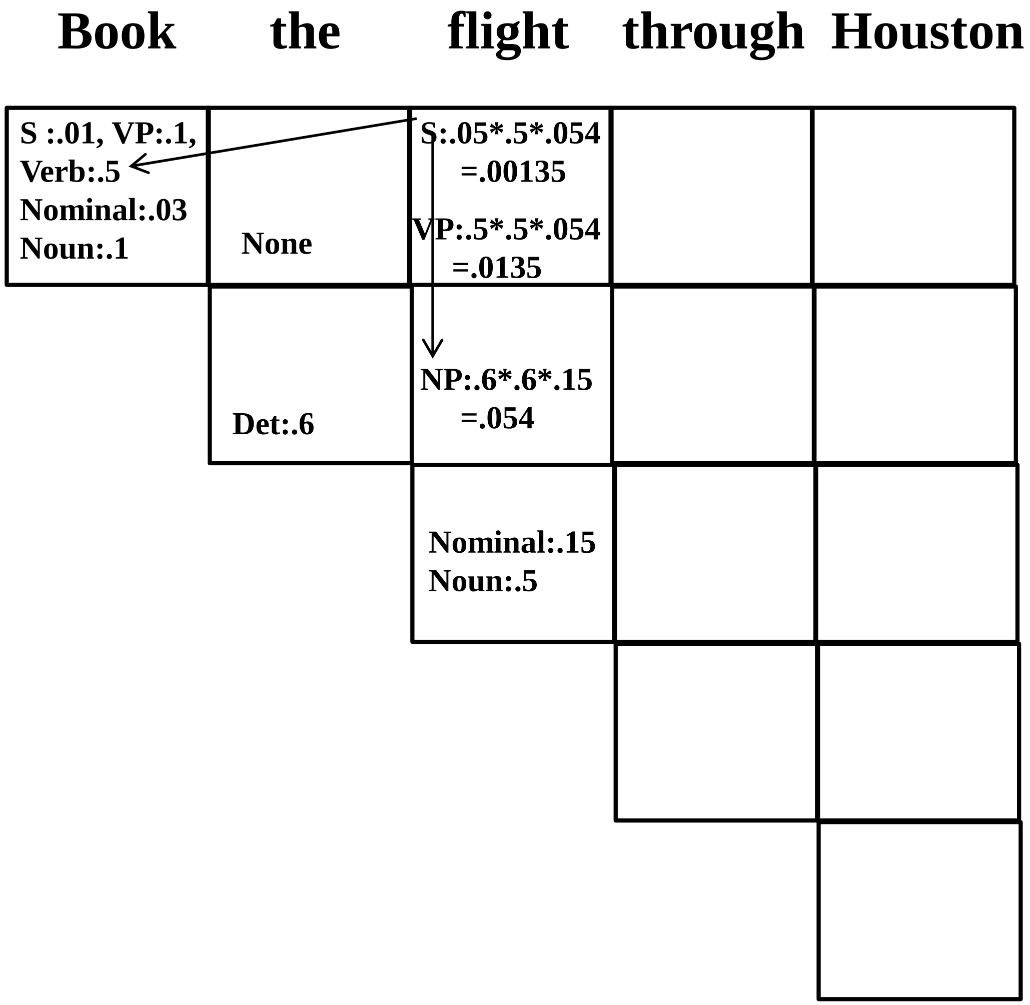


# Probabilistic CKY Parser



| Book | the | flight | through | Houston |
|---|---|---|---|---|
| S :.01, VP:.1, Verb:.5 Nominal:.03 Noun:.1 | None | VP:.5*.5*.054 =.0135 | | |
| | Det:.6 | NP:.6*.6*.15 =.054 | | |
| | | Nominal:.15 Noun:.5 | | |
| | | | | |
| | | | | |

# Probabilistic CKY Parser



| Book | the | flight | through | Houston |
|---|---|---|---|---|
| S :.01, VP:.1, Verb:.5 Nominal:.03 Noun:.1 | None | S:.05*.5*.054 =.00135 VP:.5*.5*.054 =.0135 | | |
| | Det:.6 | NP:.6*.6*.15 =.054 | | |
| | | Nominal:.15 Noun:.5 | | |
| | | | | |
| | | | | |

# Probabilistic CKY Parser

| Book | the | flight | through | Houston |
|---|---|---|---|---|
| S :.01, VP:.1, Verb:.5 Nominal:.03 Noun:.1 | None | S:.05*.5*.054 =.00135 VP:.5*.5*.054 =.0135 | None | |
| | Det:.6 | NP:.6*.6*.15 =.054 | None | |
| | | Nominal:.15 Noun:.5 | None | |
| | | | Prep:.2 | |
| | | | | |

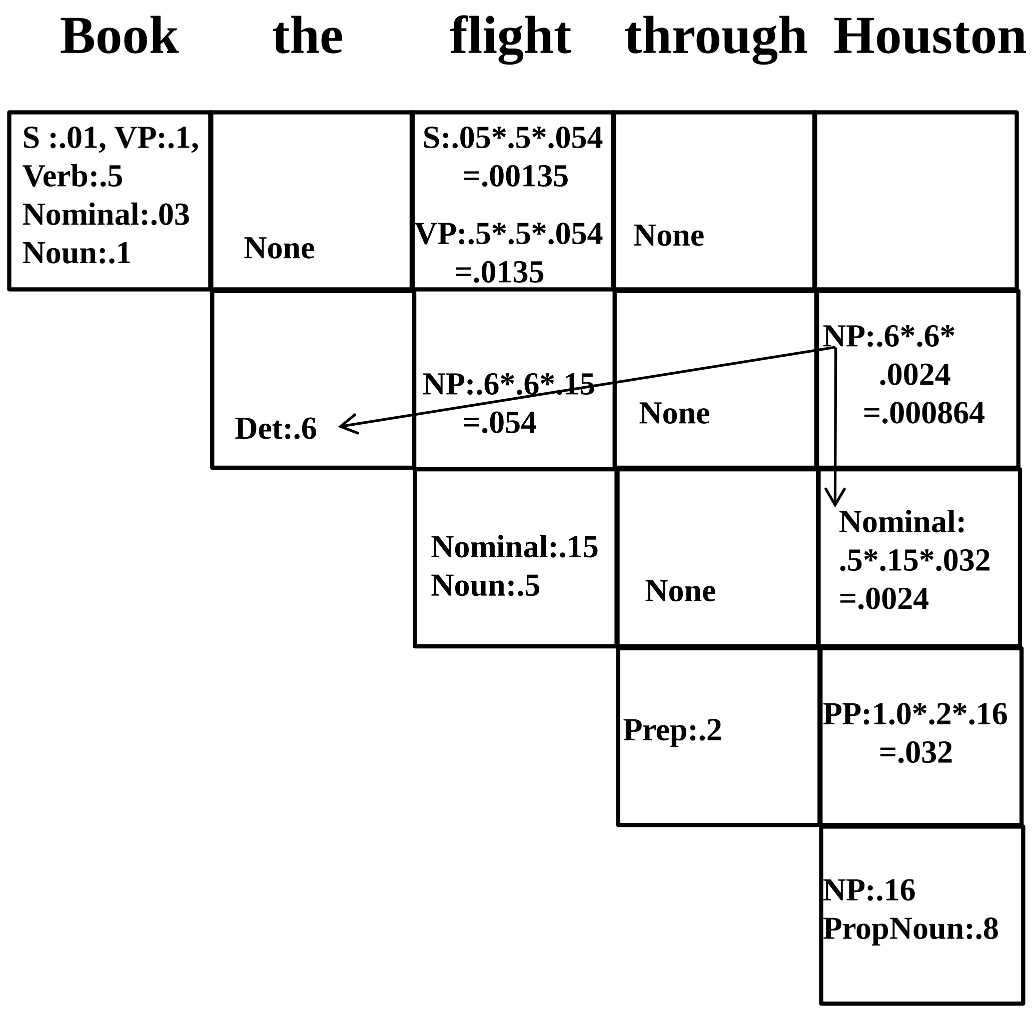
# Probabilistic CKY Parser

| Book | the | flight | through | Houston |
|---|---|---|---|---|
| S :.01, VP:.1, Verb:.5 Nominal:.03 Noun:.1 | None | S:.05*.5*.054 =.00135 VP:.5*.5*.054 =.0135 | None | |
| | Det:.6 | NP:.6*.6*.15 =.054 | None | |
| | | Nominal:.15 Noun:.5 | None | |
| | | | Prep:.2 | PP:1.0*.2*.16 =.032 |
| | | | | NP:.16 PropNoun:.8 |

# Probabilistic CKY Parser

| Book | the | flight | through | Houston |
|---|---|---|---|---|
| **S :.01, VP:.1, Verb:.5 Nominal:.03 Noun:.1** | **None** | **S:.05*.5*.054 =.00135 VP:.5*.5*.054 =.0135** | **None** | |
| | **Det:.6** | **NP:.6*.6*.15 =.054** | **None** | |
| | | **Nominal:.15 Noun:.5** | **None** | **Nominal: .5*.15*.032 =.0024** |
| | | | **Prep:.2** | **PP:1.0*.2*.16 =.032** |
| | | | | **NP:.16 PropNoun:.8** |

# Probabilistic CKY Parser



| Book | the | flight | through | Houston |
|---|---|---|---|---|
| S :.01, VP:.1, Verb:.5 Nominal:.03 Noun:.1 | None | S:.05*.5*.054 =.00135 VP:.5*.5*.054 =.0135 | None | |
| | Det:.6 | NP:.6*.6*.15 =.054 | None | NP:.6*.6* .0024 =.000864 |
| | | Nominal:.15 Noun:.5 | None | Nominal: .5*.15*.032 =.0024 |
| | | | Prep:.2 | PP:1.0*.2*.16 =.032 |
| | | | | NP:.16 PropNoun:.8 |

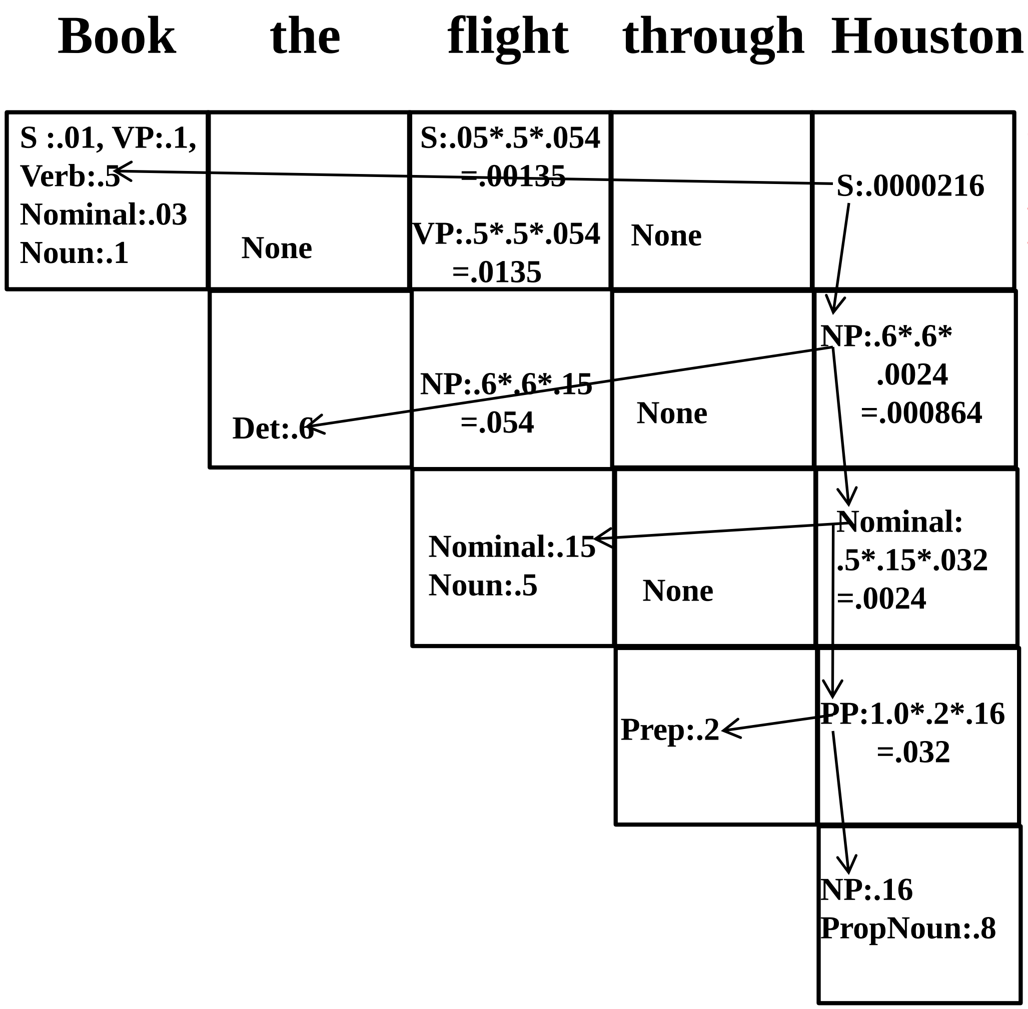
# Probabilistic CKY Parser

| Book | the | flight | through | Houston |
|---|---|---|---|---|
| S :.01, VP:.1, Verb:.5 Nominal:.03 Noun:.1 | None | S:.05*.5*.054 =.00135 VP:.5*.5*.054 =.0135 | None | S:.05*.5* .000864 =.0000216 |
| | Det:.6 | NP:.6*.6*.15 =.054 | None | NP:.6*.6* .0024 =.000864 |
| | | Nominal:.15 Noun:.5 | None | Nominal: .5*.15*.032 =.0024 |
| | | | Prep:.2 | PP:1.0*.2*.16 =.032 |
| | | | | NP:.16 PropNoun:.8 |

## Probabilistic CKY Parser

| Book | the | flight | through | Houston |
|---|---|---|---|---|
| S :.01, VP:.1, Verb:.5 Nominal:.03 Noun:.1 | None | S:.05*.5*.054 =.00135 VP:.5*.5*.054 =.0135 | None | S:.03*.0135* .032 =.00001296 S:.0000216 |
| | Det:.6 | NP:.6*.6*.15 =.054 | None | NP:.6*.6* .0024 =.000864 |
| | | Nominal:.15 Noun:.5 | None | Nominal: .5*.15*.032 =.0024 |
| | | | Prep:.2 | PP:1.0*.2*.16 =.032 |
| | | | | NP:.16 PropNoun:.8 |

# Probabilistic CKY Parser

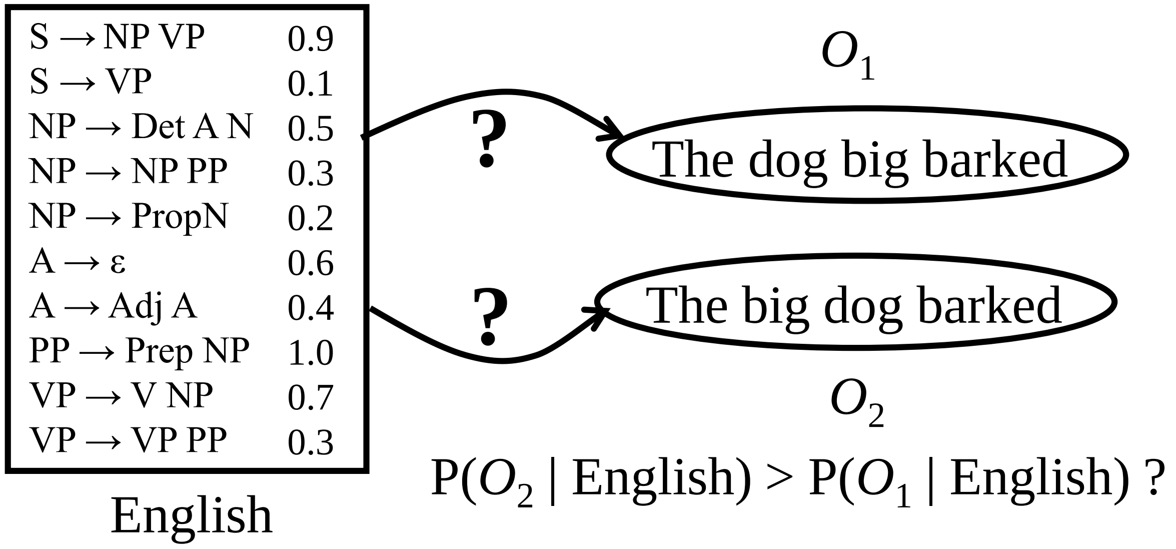


| Book | the | flight | through | Houston |
|---|---|---|---|---|
| S :.01, VP:.1, Verb:.5 Nominal:.03 Noun:.1 | None | S:.05*.5*.054 =.00135 VP:.5*.5*.054 =.0135 | None | S:.0000216 |
| | Det:.6 | NP:.6*.6*.15 =.054 | None | NP:.6*.6* .0024 =.000864 |
| | | Nominal:.15 Noun:.5 | None | Nominal: .5*.15*.032 =.0024 |
| | | | Prep:.2 | PP:1.0*.2*.16 =.032 |
| | | | | NP:.16 PropNoun:.8 |

**Pick most probable parse, i.e. take max to combine probabilities of multiple derivations of each constituent in each cell.**

## PCFG: Observation Likelihood

- There is an analog to Forward algorithm for HMMs called the Inside algorithm for efficiently determining how likely a string is to be produced by a PCFG
- Can use a PCFG as a language model to choose between alternative sentences for speech recognition or machine translation



| | |
|---|---|
| S → NP VP | 0.9 |
| S → VP | 0.1 |
| NP → Det A N | 0.5 |
| NP → NP PP | 0.3 |
| NP → PropN | 0.2 |
| A → ε | 0.6 |
| A → Adj A | 0.4 |
| PP → Prep NP | 1.0 |
| VP → V NP | 0.7 |
| VP → VP PP | 0.3 |

$P(O_2 \mid \text{English}) > P(O_1 \mid \text{English})$ ?

## Inside Algorithm

- Use CKY probabilistic parsing algorithm but combine probabilities of multiple derivations of any constituent using **addition** instead of **max**

## Probabilistic CKY Parser  for Inside Computation

| Book | the | flight | through | Houston |
|---|---|---|---|---|
| S :.01, VP:.1,<br>Verb:.5<br>Nominal:.03<br>Noun:.1 | None | S:.05*.5*.054<br>=.00135<br>VP:.5*.5*.054<br>=.0135 | None | S:..00001296<br>S:.0000216 |
| | Det:.6 | NP:.6*.6*.15<br>=.054 | None | NP:.6*.6*<br>.0024<br>=.000864 |
| | | Nominal:.15<br>Noun:.5 | None | Nominal:<br>.5*.15*.032<br>=.0024 |
| | | | Prep:.2 | PP:1.0*.2*.16<br>=.032 |
| | | | | NP:.16<br>PropNoun:.8 |

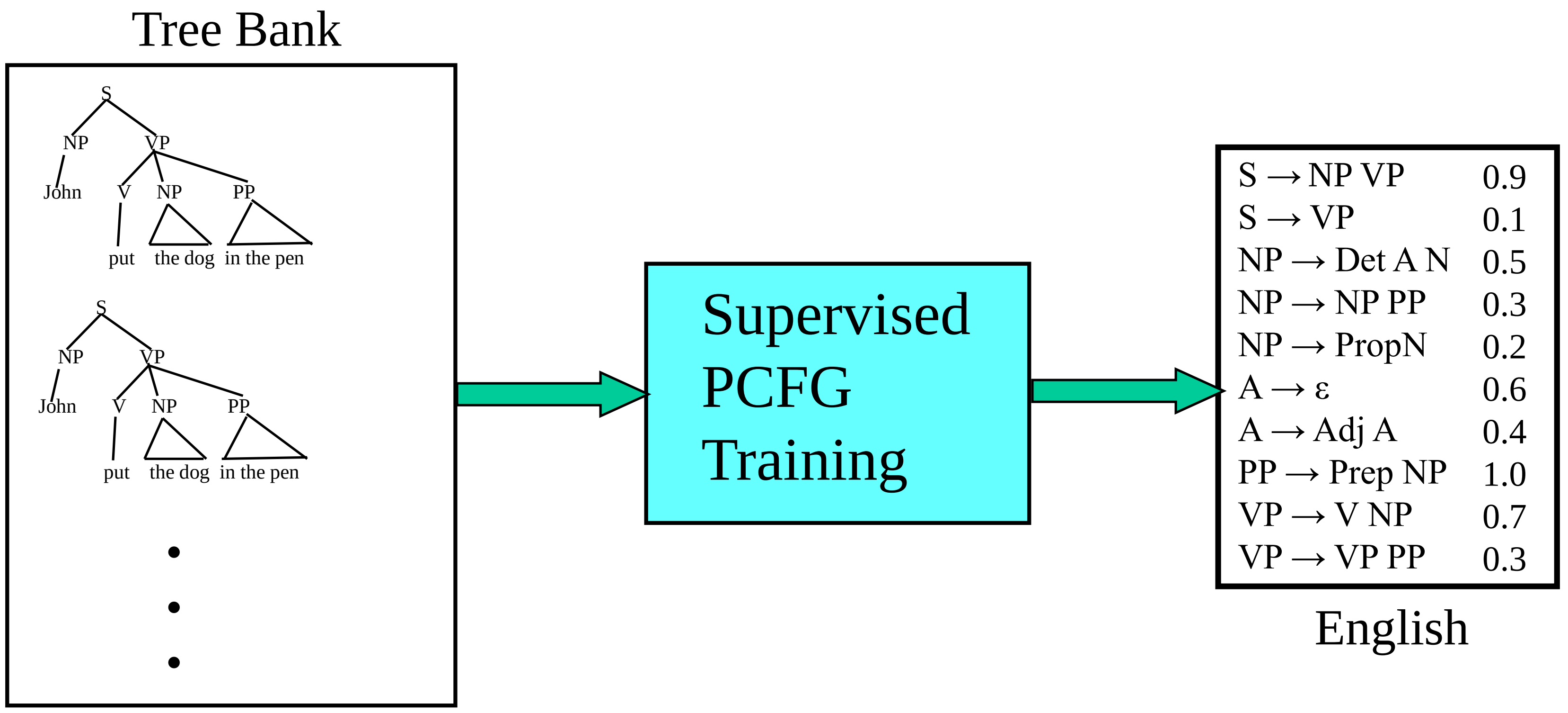
# Probabilistic CKY Parser  for Inside Computation

| Book | the | flight | through | Houston |
|---|---|---|---|---|
| S :.01, VP:.1, Verb:.5 Nominal:.03 Noun:.1 | None | S:.05*.5*.054 =.00135 VP:.5*.5*.054 =.0135 | None | S: .00001296 +.0000216 =.00003456 |
| | Det:.6 | NP:.6*.6*.15 =.054 | None | NP:.6*.6* .0024 =.000864 |
| | | Nominal:.15 Noun:.5 | None | Nominal: .5*.15*.032 =.0024 |
| | | | Prep:.2 | PP:1.0*.2*.16 =.032 |
| | | | | NP:.16 PropNoun:.8 |

**Sum probabilities of multiple derivations of each constituent in each cell**

## PCFG: Supervised Training

- If parse trees are provided for training sentences, a grammar and its parameters can all be estimated directly from counts accumulated from the tree-bank (with appropriate smoothing)



| | |
|---|---|
| S → NP VP | 0.9 |
| S → VP | 0.1 |
| NP → Det A N | 0.5 |
| NP → NP PP | 0.3 |
| NP → PropN | 0.2 |
| A → ε | 0.6 |
| A → Adj A | 0.4 |
| PP → Prep NP | 1.0 |
| VP → V NP | 0.7 |
| VP → VP PP | 0.3 |

## Estimating Production Probabilities

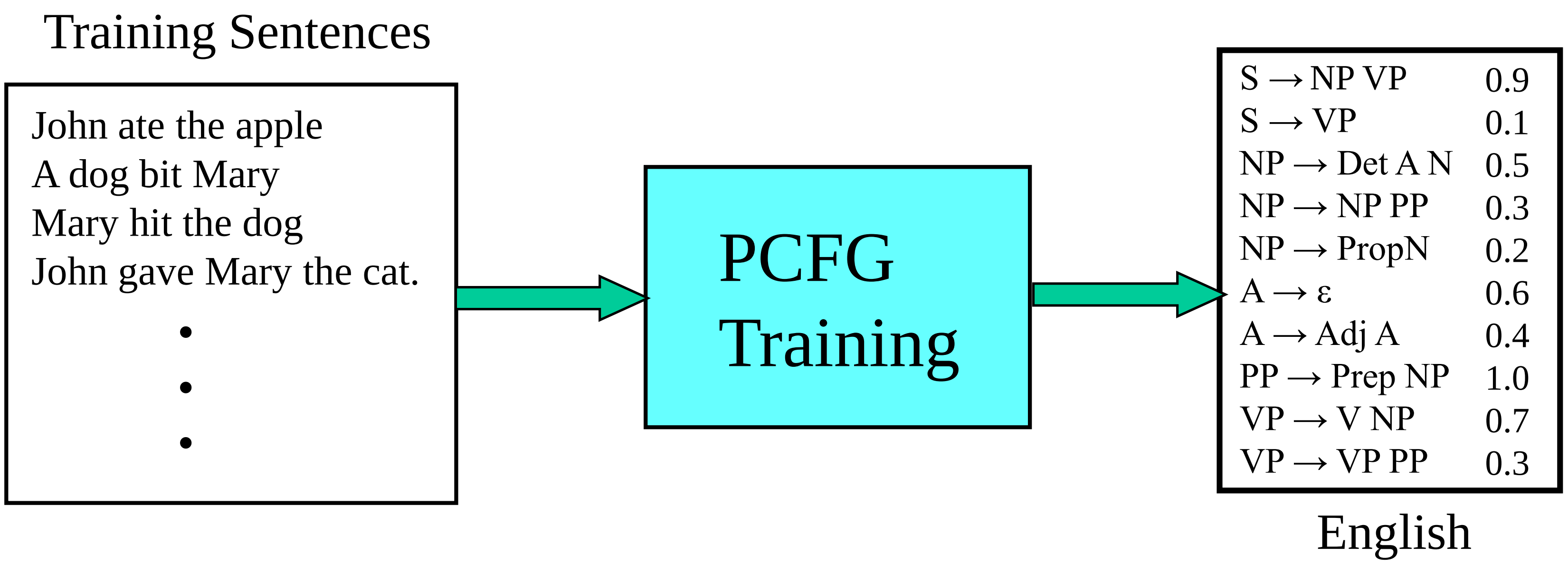
- Set of production rules can be taken directly from the set of rewrites in the treebank
- Parameters can be directly estimated from frequency counts in the treebank

$$P(\alpha \to \beta \mid \alpha) = \frac{\text{count}(\alpha \to \beta)}{\sum_{\gamma} \text{count}(\alpha \to \gamma)} = \frac{\text{count}(\alpha \to \beta)}{\text{count}(\alpha)}$$

## PCFG: Maximum Likelihood Training

- Given a set of sentences, induce a grammar that maximizes the probability that this data was generated from this grammar
- Assume the number of non-terminals in the grammar is specified
- Only need to have an un-annotated set of sequences generated from the model. Does not need correct parse trees for these sentences. In this sense, it is unsupervised

# PCFG: Maximum Likelihood Training



John ate the apple
A dog bit Mary
Mary hit the dog
John gave Mary the cat.
•
•
•

| | |
|---|---|
| $S \rightarrow NP\ VP$ | 0.9 |
| $S \rightarrow VP$ | 0.1 |
| $NP \rightarrow Det\ A\ N$ | 0.5 |
| $NP \rightarrow NP\ PP$ | 0.3 |
| $NP \rightarrow PropN$ | 0.2 |
| $A \rightarrow \varepsilon$ | 0.6 |
| $A \rightarrow Adj\ A$ | 0.4 |
| $PP \rightarrow Prep\ NP$ | 1.0 |
| $VP \rightarrow V\ NP$ | 0.7 |
| $VP \rightarrow VP\ PP$ | 0.3 |

## Inside-Outside

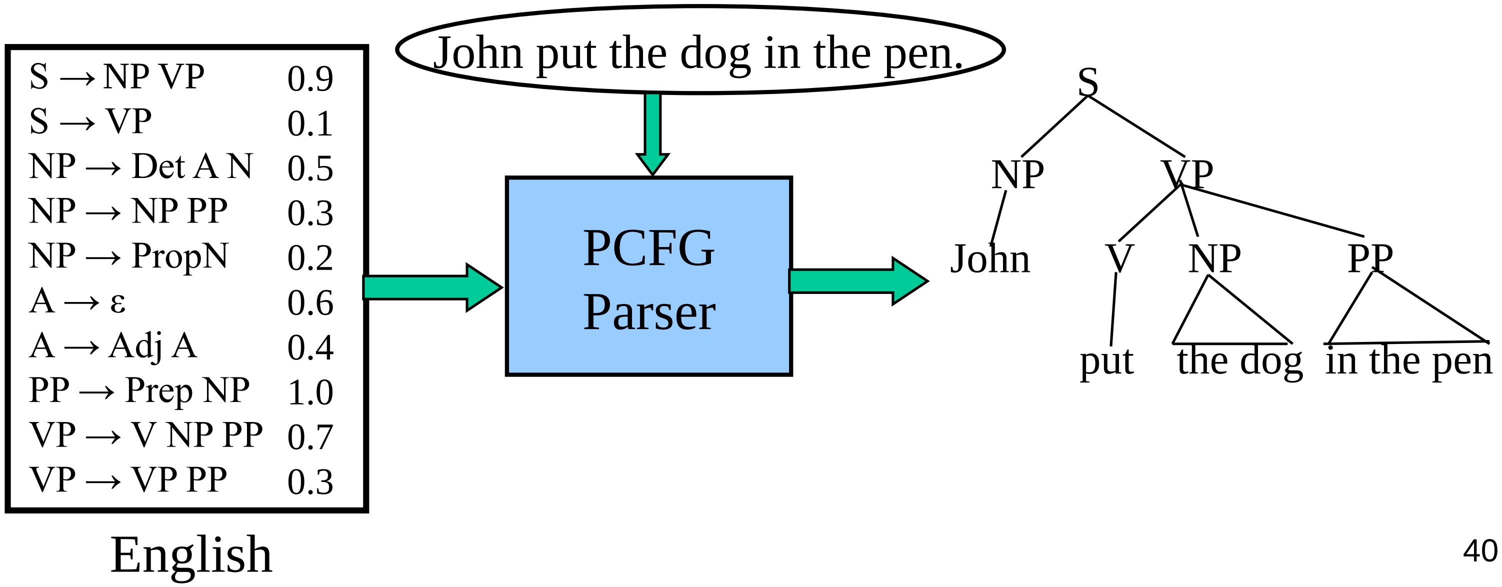
- The **Inside-Outside algorithm** is a version of EM for unsupervised learning of a PCFG
- Given the number of non-terminals, construct all possible CNF productions with these non-terminals and observed terminal symbols
- Use EM to iteratively train the probabilities of these productions to locally maximize the likelihood of the data
- Experimental results are not impressive, but recent work imposes additional constraints to improve unsupervised grammar learning

## Vanilla PCFG Limitations

- Since probabilities of productions do not rely on specific words or concepts, only general structural disambiguation is possible (e.g. prefer to attach PPs to Nominals)
- Consequently, vanilla PCFGs cannot resolve syntactic ambiguities that require semantics to resolve, e.g. *ate with fork* vs. *ate with sandwich*
- In order to work well, PCFGs must be lexicalized, i.e. productions must be specialized to specific words by including their head-word in their LHS non-terminals (e.g. VP-ate)

## Example of Importance of Lexicalization

- A general preference for attaching PPs to NPs rather than VPs can be learned by a vanilla PCFG
- But the desired preference can depend on specific words

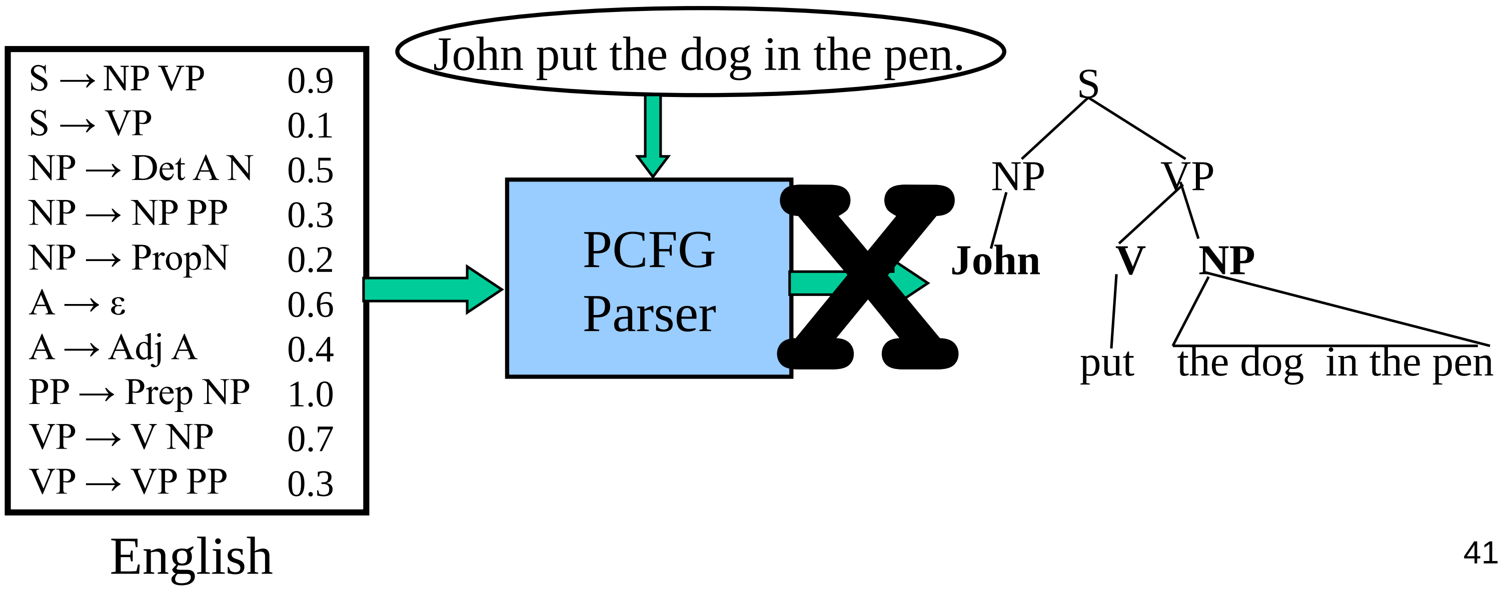


| | |
|---|---|
| S → NP VP | 0.9 |
| S → VP | 0.1 |
| NP → Det A N | 0.5 |
| NP → NP PP | 0.3 |
| NP → PropN | 0.2 |
| A → ε | 0.6 |
| A → Adj A | 0.4 |
| PP → Prep NP | 1.0 |
| VP → V NP PP | 0.7 |
| VP → VP PP | 0.3 |

English

# Example of Importance of Lexicalization

- A general preference for attaching PPs to NPs rather than VPs can be learned by a vanilla PCFG
- But the desired preference can depend on specific words



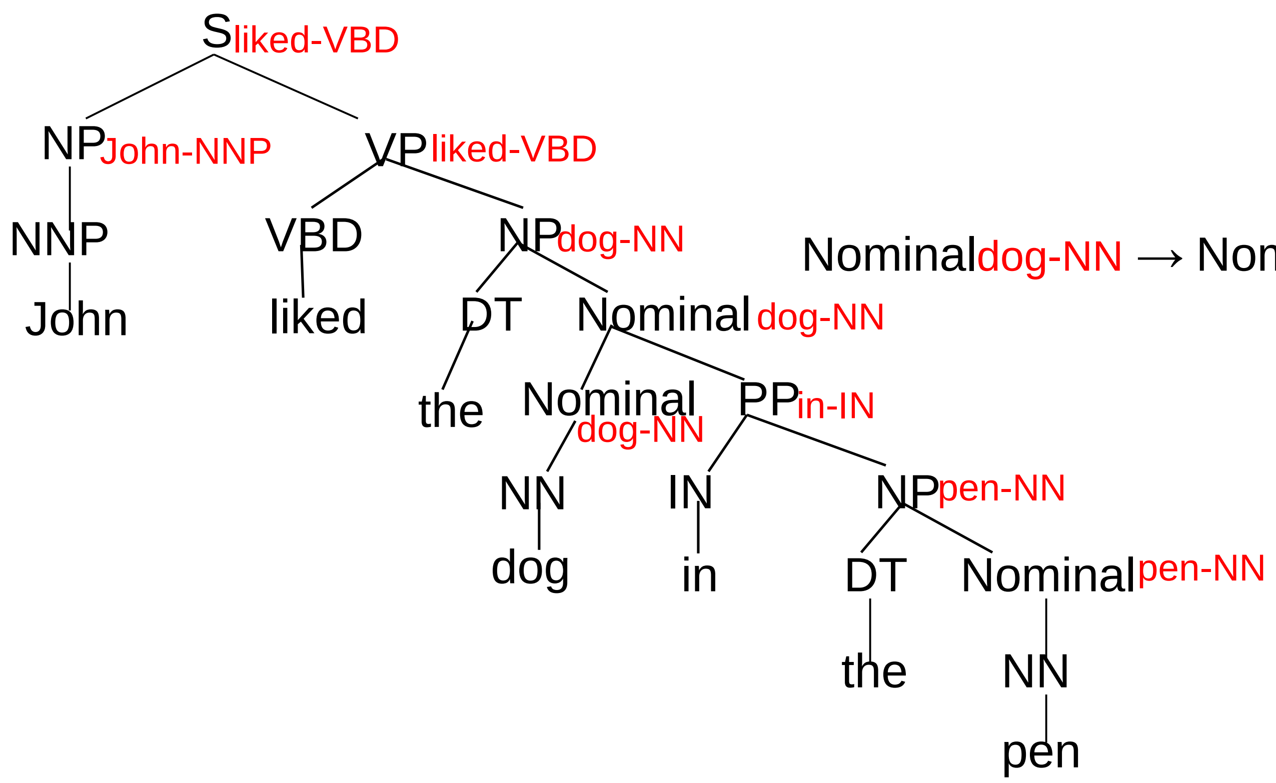
| | |
|---|---|
| S → NP VP | 0.9 |
| S → VP | 0.1 |
| NP → Det A N | 0.5 |
| NP → NP PP | 0.3 |
| NP → PropN | 0.2 |
| A → ε | 0.6 |
| A → Adj A | 0.4 |
| PP → Prep NP | 1.0 |
| VP → V NP | 0.7 |
| VP → VP PP | 0.3 |

## Head Words

- Syntactic phrases usually have a word in them that is most "central" to the phrase
- Linguists have defined the concept of a lexical **head** of a phrase
- Simple rules can identify the head of any phrase by percolating head words up the parse tree
  - Head of a VP is the main verb
  - Head of an NP is the main noun
  - Head of a PP is the preposition
  - Head of a sentence is the head of its VP

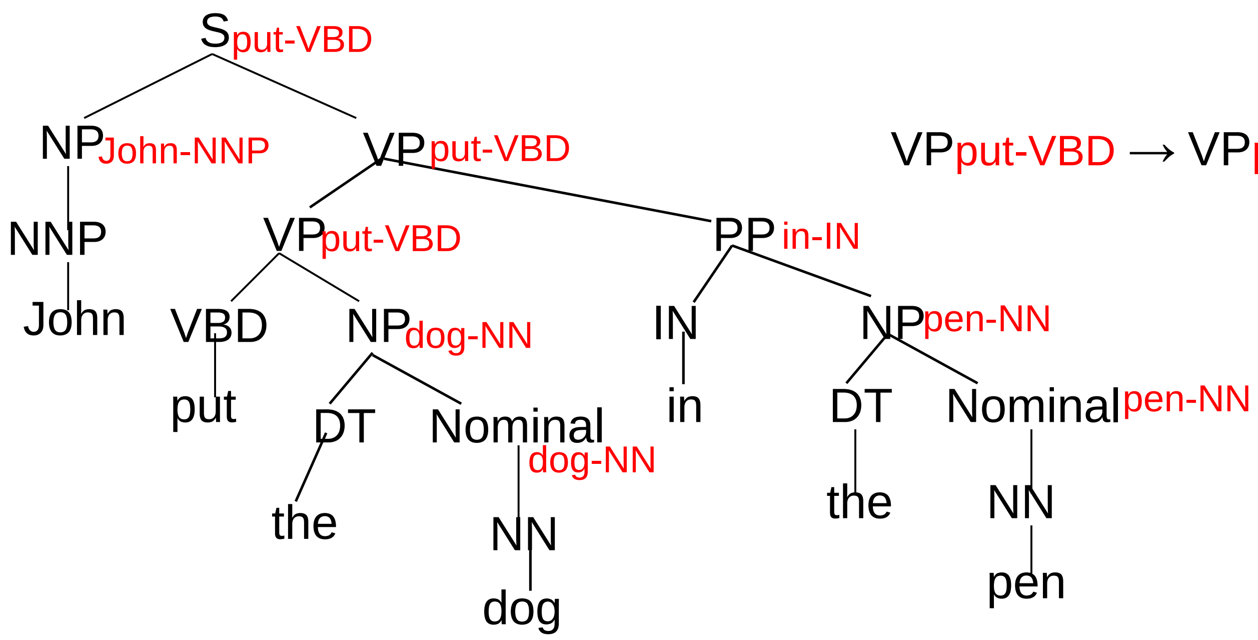
## Lexicalized Productions

- Specialized productions can be generated by including the head word and its POS of each non-terminal as part of that non-terminal's symbol



Nominal$_{\text{dog-NN}}$ → Nominal$_{\text{dog-NN}}$ PP$_{\text{in-IN}}$

# Lexicalized Productions



$\mathrm{VP}_{\text{put-VBD}} \rightarrow \mathrm{VP}_{\text{put-VBD}}\ \mathrm{PP}_{\text{in-IN}}$

## Parameterizing Lexicalized Productions

- Accurately estimating parameters on such a large number of very specialized productions could require enormous amount of treebank data

- Need some way of estimating parameters for lexicalized productions that make reasonable independence assumptions so that accurate probabilities for very specific rules can be learned

# Collins' Parser

- Collins' (1999) parser assumes a simple generative model of lexicalized productions
- Models productions based on context to the left and the right of the head daughter
  - $\mathrm{LHS} \longrightarrow \mathrm{L}_n \mathrm{L}_{n-1} \ldots \mathrm{L}_1 \mathrm{H}\ \mathrm{R}_1 \ldots \mathrm{R}_{m-1} \mathrm{R}_m$
- First generate the head (H) and then repeatedly generate left ($\mathrm{L}_i$) and right ($\mathrm{R}_i$) context symbols until the symbol STOP is generated

# Sample Production Generation

VPput-VBD → VBDput-VBD NPdog-NN PPin-IN

**Note:** Penn treebank tends to have fairly flat parse trees that produce long productions

VPput-VBD → STOP VBDput-VBD NPdog-NN PPin-IN STOP

$L_1$ H $R_1$ $R_2$ $R_3$

$P_L$(STOP | VPput-VBD) * $P_H$(VBD | Vpput-VBD)*
$P_R$(NPdog-NN | VPput-VBD)*
$P_R$(PPin-IN | VPput-VBD) * $P_R$(STOP | VPput-VBD)

## Estimating Production Generation Parameters

- Estimate $P_H$, $P_L$, and $P_R$ parameters from treebank data

$$P_R(\text{PPin-IN} \mid \text{VPput-VBD}) = \frac{\text{Count(PPin-IN right of head in a VPput-VBD production)}}{\text{Count(symbol right of head in a VPput-VBD)}}$$

$$P_R(\text{NPdog-NN} \mid \text{VPput-VBD}) = \frac{\text{Count(NPdog-NN right of head in a VPput-VBD production)}}{\text{Count(symbol right of head in a VPput-VBD)}}$$

- Smooth estimates by linearly interpolating with simpler models conditioned on just POS tag or no lexical info

$$\begin{aligned} smP_R(\text{PPin-IN} \mid \text{VPput-VBD}) = {} & \lambda_1 P_R(\text{PPin-IN} \mid \text{VPput-VBD}) \\ & + (1 - \lambda_1)\,(\lambda_2 P_R(\text{PPin-IN} \mid \text{VPVBD}) + \\ & (1 - \lambda_2)\, P_R(\text{PPin-IN} \mid \text{VP})) \end{aligned}$$

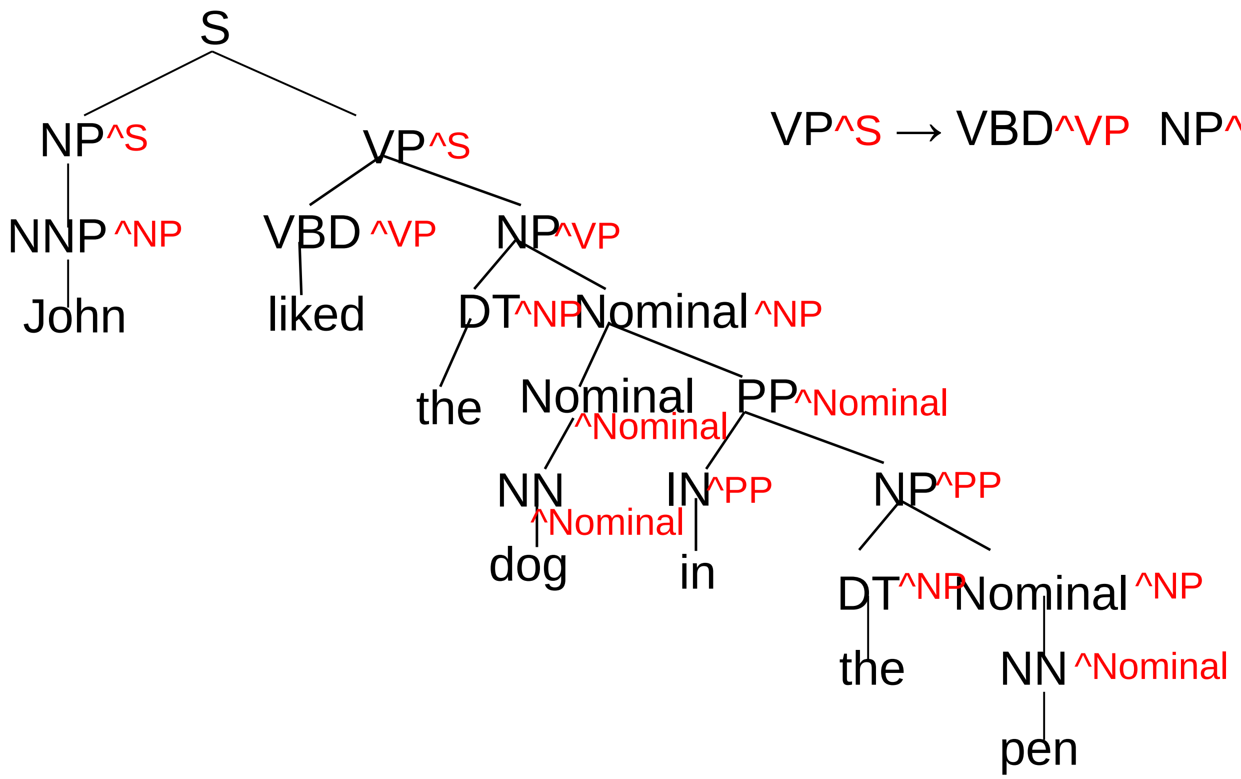
## Missed Context Dependence

- Another problem with CFGs is that which production is used to expand a non-terminal is independent of its context
- However, this independence is frequently violated for normal grammars
  - NPs that are subjects are more likely to be pronouns than NPs that are objects

## Splitting Non-Terminals

- To provide more contextual information, non terminals can be split into multiple new non terminals based on their parent in the parse tree using parent annotation
  - A subject NP becomes NP^S since its parent node is an S.
  - An object NP becomes NP^VP since its parent node is a VP

# Parent Annotation Example



VP^S → VBD^VP NP^VP

## Split and Merge

- Non-terminal splitting greatly increases the size of the grammar and the number of parameters that need to be learned from limited training data
  - Best approach is to only split non-terminals when it improves the accuracy of the grammar
- May also help to merge some non-terminals to remove some un-helpful distinctions and learn more accurate parameters for the merged productions
- Method: Heuristically search for a combination of splits and merges that produces a grammar that maximizes the likelihood of the training treebank

## Treebanks

- English Penn Treebank: Standard corpus for testing syntactic parsing consists of 1.2 M words of text from the Wall Street Journal (WSJ)
- Typical to train on about 40,000 parsed sentences and test on an additional standard disjoint test set of 2,416 sentences
- Chinese Penn Treebank: 100K words from the Xinhua news service
- Other corpora existing in many languages, see the Wikipedia article “Treebank”

## First WSJ Sentence

```
( (S
    (NP-SBJ
      (NP (NNP Pierre) (NNP Vinken) )
      (, ,)
      (ADJP
        (NP (CD 61) (NNS years) )
        (JJ old) )
      (, ,) )
    (VP (MD will)
      (VP (VB join)
        (NP (DT the) (NN board) )
        (PP-CLR (IN as)
          (NP (DT a) (JJ nonexecutive) (NN director) ))
        (NP-TMP (NNP Nov.) (CD 29) )))
    (. .) ))
```

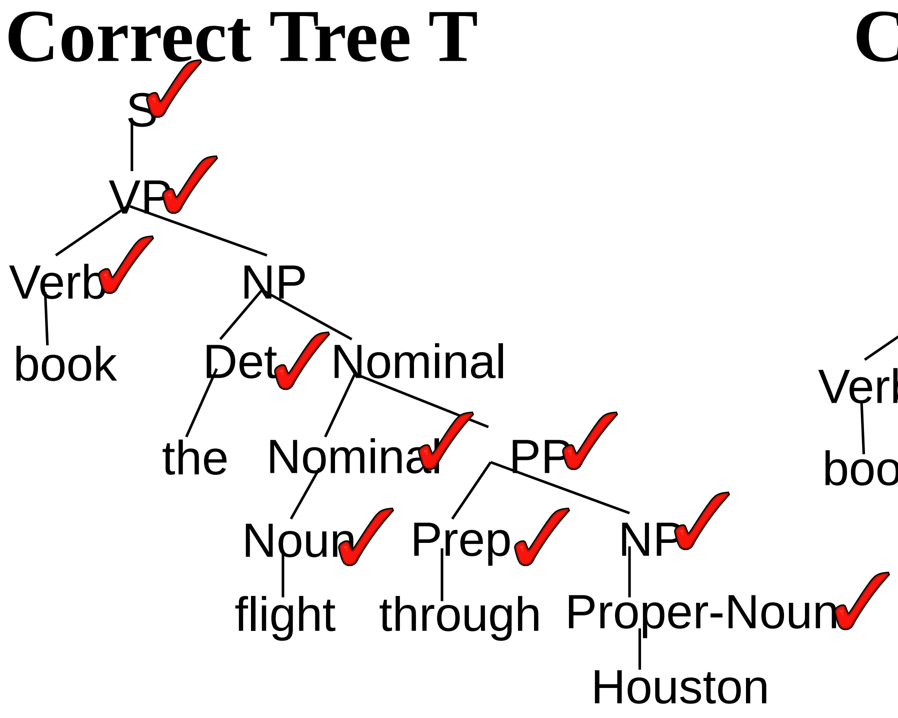
## WSJ Sentence with Trace (NONE)

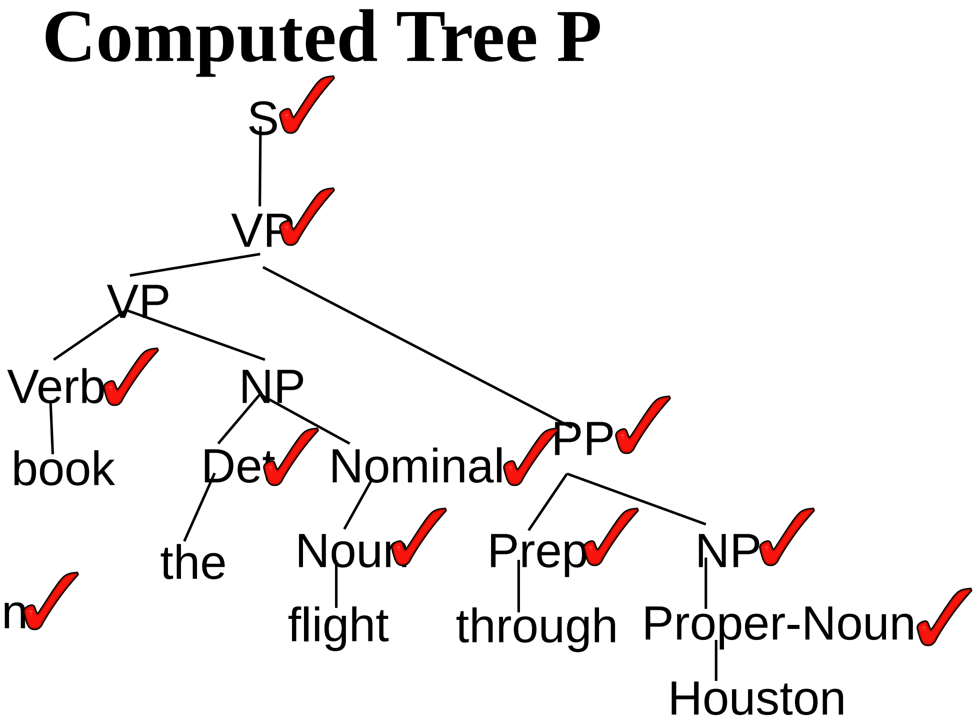
```
( (S
    (NP-SBJ (DT The) (NNP Illinois) (NNP Supreme) (NNP Court) )
    (VP (VBD ordered)
      (NP-1 (DT the) (NN commission) )
      (S
        (NP-SBJ (-NONE- *-1) )
        (VP (TO to)
          (VP
            (VP (VB audit)
              (NP
                (NP (NNP Commonwealth) (NNP Edison) (POS 's) )
                (NN construction) (NNS expenses) ))
            (CC and)
            (VP (VB refund)
              (NP (DT any) (JJ unreasonable) (NNS expenses) ))))))
    (. .) ))
```

## Parsing Evaluation Metrics

- PARSEVAL metrics measure the fraction of the constituents that match between the computed and human parse trees. If *P* is the system's parse tree and *T* is the human parse tree (the "gold standard"):
  - ***Recall*** = (# correct constituents in *P*) / (# constituents in *T*)
  - ***Precision*** = (# correct constituents in *P*) / (# constituents in *P*)
- ***Labeled Precision*** and ***labeled recall*** require getting the non-terminal label on the constituent node correct to count as correct
- $F_1$ is the harmonic mean of precision and recall

# Computing Evaluation Metrics





# Constituents: 12

# Constituents: 12

# Correct Constituents: 10

Recall = 10/12= 83.3% Precision = 10/12=83.3% $F_1$ = 83.3%

## Treebank Results

- Results of current state-of-the-art systems on the English Penn WSJ treebank are slightly greater than 90% labeled precision and recall

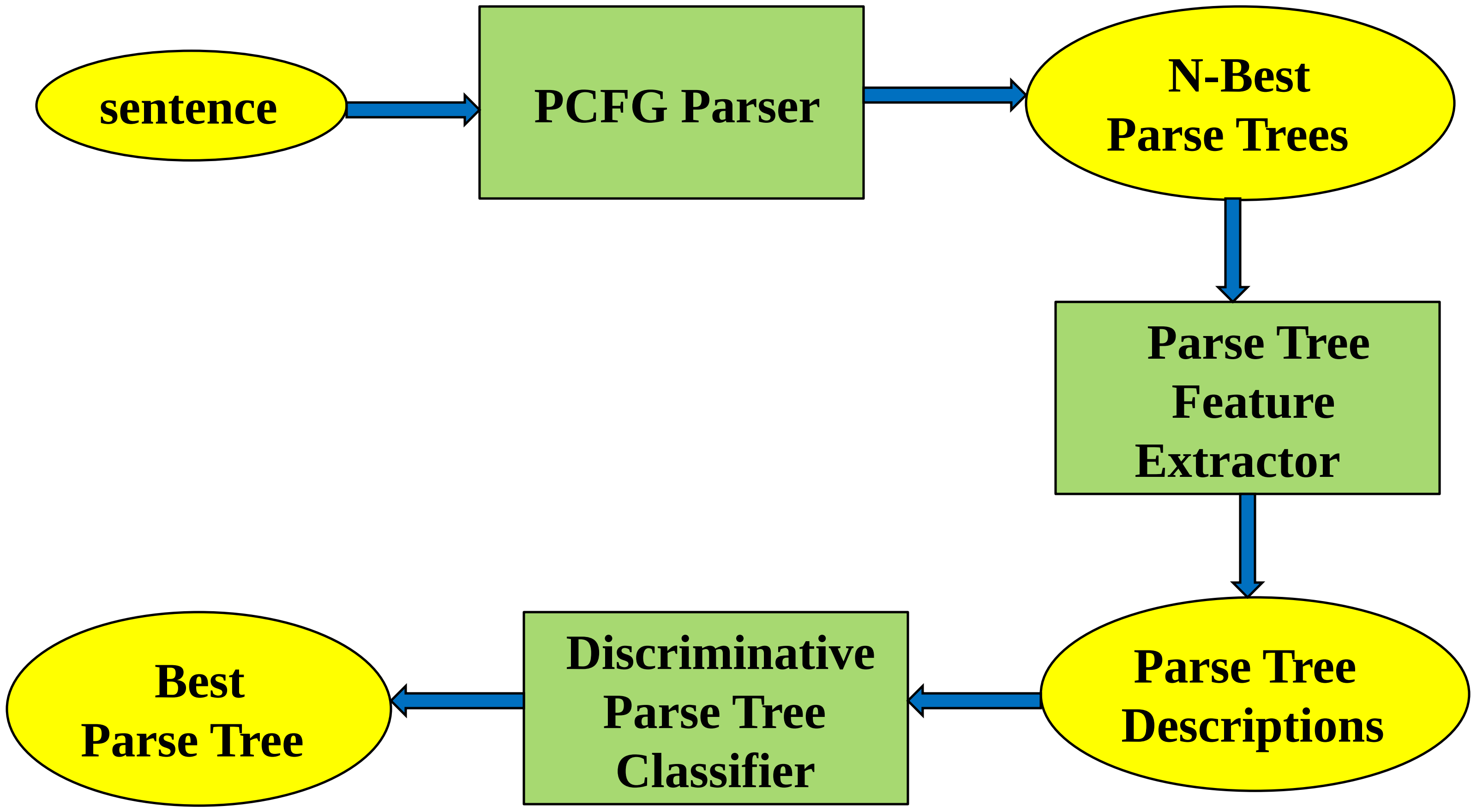
## Discriminative Parse Reranking

- **Motivation**: Even when the top-ranked parse not correct, frequently the correct parse is one of those ranked highly by a statistical parser
- Use a discriminative classifier that is trained to select the best parse from the N-best parses produced by the original parser
- Reranker can exploit global features of the entire parse whereas a PCFG is restricted to making decisions based on local info

## 2-Stage Reranking Approach

- Adapt the PCFG parser to produce an ***N-best list*** of the most probable parses in addition to the most-likely one
- Extract from each of these parses, a set of global features that help determine if it is a good parse tree
- Train a discriminative classifier (e.g. *logistic regression*) using the best parse in each N-best list as positive and others as negative

## Parse Reranking



## Sample Parse Tree Features

- Probability of the parse from the PCFG
- The number of parallel conjuncts
  - "the bird in the tree and the squirrel on the ground"
  - "the bird and the squirrel in the tree"
- The degree to which the parse tree is right branching
  - English parses tend to be right branching (cf. parse of "Book the flight through Houston", "Book the flight through Houston in the morning", Book the flight through Houston in the morning at 8 AM")
- Frequency of various tree fragments, i.e. specific combinations of 2 or 3 rules

## Evaluation of Reranking

- Reranking is limited by ***oracle accuracy***, i.e. the accuracy that results when an omniscient oracle picks the best parse from the N-best list

- Typical current oracle accuracy is around $F_1$=97%

- Reranking can generally improve test accuracy of current PCFG models a percentage point or two

## Other Discriminative Parsing

- There are also parsing models that move from generative PCFGs to a fully discriminative model, e.g. ***max margin parsing*** (Taskar *et al*., 2004)
- There is also a recent model that efficiently reranks all of the parses in the complete (compactly-encoded) parse forest, avoiding the need to generate an N-best list (***forest reranking***, Huang, 2008)

## Human Parsing

- Computational parsers can be used to predict human reading time as measured by tracking the time taken to read each word in a sentence
- Psycholinguistic studies show that words that are more probable given the preceding lexical and syntactic context are read faster
  - John put the dog in the pen with a lock.
  - John put the dog in the pen with a bone in the car.
  - John liked the dog in the pen with a bone.
- Modeling these effects requires an ***incremental*** statistical parser that incorporates one word at a time into a continuously growing parse tree

## Garden Path Sentences

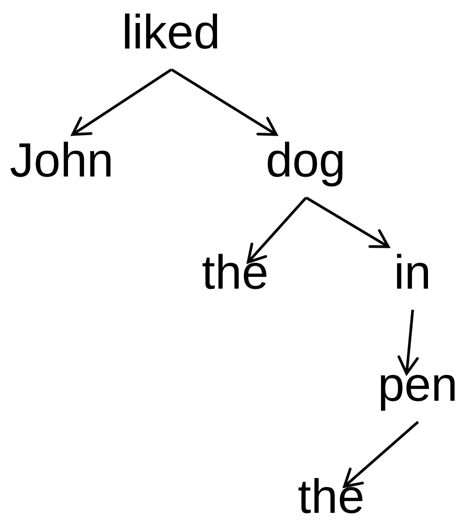
- People are confused by sentences that seem to have a particular syntactic structure but then suddenly violate this structure, so the listener is "lead down the garden path"
  - The horse raced past the barn fell.
    - vs. The horse raced past the barn broke his leg.
  - The complex houses married students.
  - The old man the sea.
  - While Anna dressed the baby spit up on the bed.
- Incremental computational parsers can try to predict and explain the problems encountered parsing such sentences

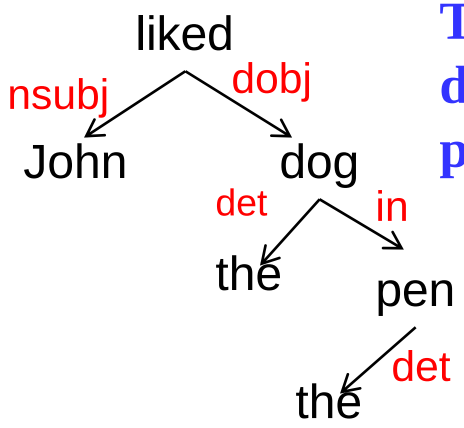
# Center Embedding

- Nested expressions are hard for humans to process beyond 1 or 2 levels of nesting
  - The rat the cat chased died.
  - The rat the cat the dog bit chased died.
  - The rat the cat the dog the boy owned bit chased died.
- Requires remembering and popping incomplete constituents from a stack and strains human short-term memory
- Equivalent "tail embedded" (tail recursive) versions are easier to understand since no stack is required
  - The boy owned a dog that bit a cat that chased a rat that died.

## Dependency Grammars

- An alternative to phrase-structure grammar is to define a parse as a directed graph between the words of a sentence representing ***dependencies*** between the words

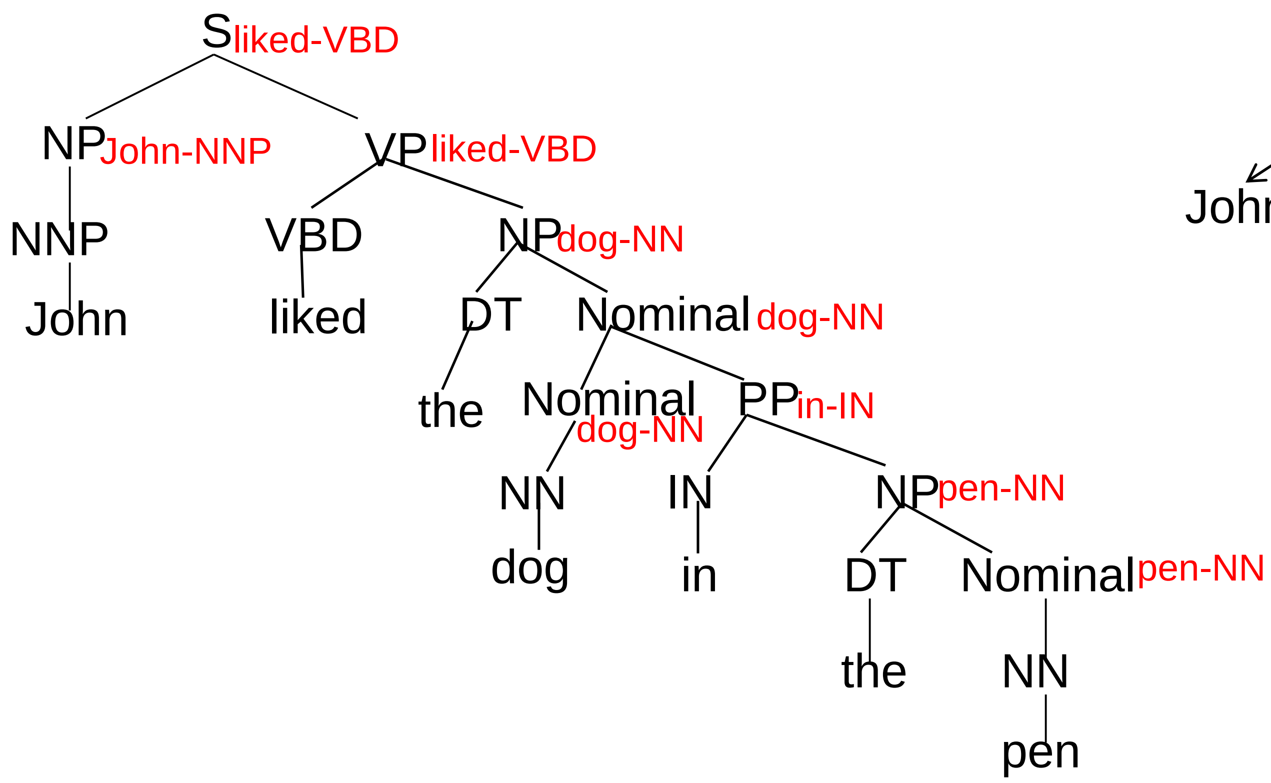


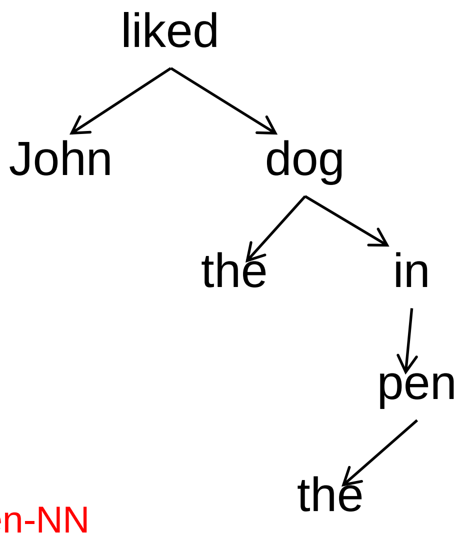


**Typed dependency parse**

## Dependency Graph from Parse Tree

- Can convert a phrase structure parse to a dependency tree by making the head of each non-head child of a node depend on the head of the head child





## Unification Grammars

- In order to handle agreement issues more effectively, each constituent has a list of features such as number, person, gender, etc. which may or may not be specified for a given constituent
- In order for two constituents to combine to form a larger constituent, their features must ***unify***, i.e. consistently combine into a merged set of features
- Expressive grammars and parsers (e.g. HPSG) have been developed using this approach and have been partially integrated with modern statistical models of disambiguation

# Mildly Context-Sensitive Grammars

- Some grammatical formalisms provide a degree of context-sensitivity that helps capture aspects of NL syntax that are not easily handled by CFGs
- Tree Adjoining Grammar (TAG) is based on combining tree fragments rather than individual phrases
- Combinatory Categorial Grammar (CCG) consists of:
  - Categorial Lexicon that associates a syntactic and semantic category with each word
  - Combinatory Rules that define how categories combine to form other categories

## Statistical Parsing Conclusions

- Statistical models such as PCFGs allow for probabilistic resolution of ambiguities
- PCFGs can be easily learned from treebanks
- Lexicalization and non-terminal splitting are required to effectively resolve many ambiguities
- Current statistical parsers are quite accurate but not yet at the level of human-expert agreement