CS 6840: Natural Language Processing

Syntactic Parsing

Razvan C. Bunescu
School of Electrical Engineering and Computer Science
bunescu@ohio.edu
Syntactic Parsing

- **Syntactic Parsing** = assigning a syntactic structure to a sentence.
  - For CFGs: assigning a *phrase-structure tree* to a sentence.

<table>
<thead>
<tr>
<th>Grammar</th>
<th>Lexicon</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S \rightarrow NP \ VP$</td>
<td>$Det \rightarrow that \mid this \mid a$</td>
</tr>
<tr>
<td>$S \rightarrow Aux \ NP \ VP$</td>
<td>$Noun \rightarrow book \mid flight \mid meal \mid money$</td>
</tr>
<tr>
<td>$S \rightarrow VP$</td>
<td>$Verb \rightarrow book \mid include \mid prefer$</td>
</tr>
<tr>
<td>$NP \rightarrow Pronoun$</td>
<td>$Pronoun \rightarrow I \mid she \mid me$</td>
</tr>
<tr>
<td>$NP \rightarrow Proper-Noun$</td>
<td>$Proper-Noun \rightarrow Houston \mid NWA$</td>
</tr>
<tr>
<td>$NP \rightarrow Det \ Nominal$</td>
<td>$Aux \rightarrow does$</td>
</tr>
<tr>
<td>$Nominal \rightarrow Noun$</td>
<td>$Preposition \rightarrow from \mid to \mid on \mid near \mid through$</td>
</tr>
</tbody>
</table>

```
Book that flight.
```
Syntactic Parsing as Search

- Parsing $\equiv$ search through the space of all possible parse trees such that:
  1. The leaves of the final parse tree coincide with the words in the input sentence.
  2. The root of the parse tree is the symbol S, i.e. complete parse tree.

$\Rightarrow$ 2 search strategies:
- **Top-Down** parsing (goal-directed search).
- **Bottom-Up** parsing (data-directed search).
Top-Down Parsing

- Build the parse tree from the root $S$ down to the leaves:
  - Expand tree nodes $N$ by using CFG rules $N \rightarrow N_1 \ldots N_k$.
  - Grow trees downward until reaching the POS categories at the bottom of the tree.
  - Reject trees that do not match all the words in the input.
Bottom-Up Parsing

• Build the parse tree from the leaf words up to the root S:
  – Find root nodes $N_1 \ldots N_k$ in the current forest such that they match a CFG rule $N \rightarrow N_1 \ldots N_k$.
  – Reject sub-trees that cannot lead to the start symbol S.
Top-Down vs. Bottom-Up

• **Top-down:**
  – Only searches for trees that are complete (i.e. S’s)
  – But also suggests trees that are not consistent with any of the words.

• **Bottom-up:**
  – Only forms trees consistent with the words.
  – But also suggests trees that make no sense globally.

• How expensive is the entire search process?
Syntactic Parsing as Search

• How to keep track of the search space and how to make choices:
  – Which node to try to expand next.
  – Which grammar rule to use to expand a node.

• Backtracking (naïve implementation of parsing):
  – Expand the search space incrementally, choose a state to expand in the search space (depth-first, breadth-first, or other strategies).
  – If strategy arrives at an inconsistent tree, backtrack to an unexplored search on the agenda.
  – Doomed because of large search space and redundant work due to shared subproblems.
Large Search Space

- **Global Ambiguity:**
  - coordination: *old men and women*
  - attachment: *we saw the Eiffel Tower flying to Paris*

- **Local Ambiguity**
• Parse the sentence:
  “a flight from Indianapolis to Houston on NWA”

• Use backtracking with a top-down, depth-first, left-to-right strategy:
  – Assume a top-down parse making choices among the various Nominal rules, in particular, between these two:
    • Nominal → Noun
    • Nominal → Nominal PP
  – Staticaly choosing the rules in this order leads to the following bad results, in which every part of the final tree is derived more than once:
Shared Subproblems
Syntactic Parsing using Dynamic Programming

- Shared subproblems ⇒ **dynamic programming** could help.

- Dynamic Programming:
  - **CKY** algorithm (bottom-up search).
    - Need to transform the CFG into Chomsky Normal Form (CNF).
    - Any CFG can be transformed into CNF automatically.
  - **Earley** algorithm (top-down search).
    - does not require a normalized grammar.
    - a single left-to-right pass that fills an array/chart of size $n + 1$.
    - more complex than CKY.
  - **Chart parsing**:
    - more general, retain completed phrases in a chart, can combine top-down and bottom-up search.
CKY Parsing: Chomsky Normal Form

- All rules should be of one of two forms:
  \[ A \rightarrow B C \text{ or } A \rightarrow w \]

- CNF conversion procedure:
  1. Convert terminals to dummy non-terminals:
     \[ \text{INF-VP} \rightarrow to \ VP \Leftrightarrow \text{INF-VP} \rightarrow \text{TO VP} \text{ and } \text{TO} \rightarrow to \]
  2. Convert unit productions
     \[ \text{Nominal} \rightarrow \text{Noun} \]
     \[ \text{Noun} \rightarrow book \mid flight \]
     \[ \Leftrightarrow \text{Nominal} \rightarrow book \mid flight \]
  3. Make all rules binary by adding new non-terminals:
     \[ \text{VP} \rightarrow \text{Verb NP PP} \Leftrightarrow \text{VP} \rightarrow \text{VX PP} \]
     \[ \text{VX} \rightarrow \text{Verb NP} \]
# $L_1$ Grammar

<table>
<thead>
<tr>
<th>Grammar</th>
<th>Lexicon</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S \rightarrow NP \ VP$</td>
<td>$Det \rightarrow that</td>
</tr>
<tr>
<td>$S \rightarrow Aux \ NP \ VP$</td>
<td>$Noun \rightarrow book</td>
</tr>
<tr>
<td>$S \rightarrow VP$</td>
<td>$Verb \rightarrow book</td>
</tr>
<tr>
<td>$NP \rightarrow Pronoun$</td>
<td>$Pronoun \rightarrow I</td>
</tr>
<tr>
<td>$NP \rightarrow Proper-Noun$</td>
<td>$Proper-Noun \rightarrow Houston</td>
</tr>
<tr>
<td>$NP \rightarrow Det Nominal$</td>
<td>$Aux \rightarrow does$</td>
</tr>
<tr>
<td>Nominal $\rightarrow$ Noun</td>
<td>$Preposition \rightarrow from</td>
</tr>
<tr>
<td>Nominal $\rightarrow$ Nominal Noun</td>
<td></td>
</tr>
<tr>
<td>Nominal $\rightarrow$ Nominal PP</td>
<td></td>
</tr>
<tr>
<td>$VP \rightarrow$ Verb</td>
<td></td>
</tr>
<tr>
<td>$VP \rightarrow$ Verb NP</td>
<td></td>
</tr>
<tr>
<td>$VP \rightarrow$ Verb NP PP</td>
<td></td>
</tr>
<tr>
<td>$VP \rightarrow$ Verb PP</td>
<td></td>
</tr>
<tr>
<td>$VP \rightarrow$ VP PP</td>
<td></td>
</tr>
<tr>
<td>$PP \rightarrow$ Preposition NP</td>
<td></td>
</tr>
<tr>
<td>$\mathcal{L}_1$ Grammar</td>
<td>$\mathcal{L}_1$ in CNF</td>
</tr>
<tr>
<td>------------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>$S \rightarrow NP \ VP$</td>
<td>$S \rightarrow NP \ VP$</td>
</tr>
<tr>
<td>$S \rightarrow Aux \ NP \ VP$</td>
<td>$S \rightarrow X1 \ VP$</td>
</tr>
<tr>
<td>$S \rightarrow VP$</td>
<td>$X1 \rightarrow Aux \ NP$</td>
</tr>
<tr>
<td>$S \rightarrow book \mid include \mid prefer$</td>
<td>$S \rightarrow Verb \ NP$</td>
</tr>
<tr>
<td>$S \rightarrow X2 \ PP$</td>
<td>$S \rightarrow Verb \ PP$</td>
</tr>
<tr>
<td>$S \rightarrow VP \ PP$</td>
<td>$S \rightarrow VP \ PP$</td>
</tr>
<tr>
<td>$NP \rightarrow Pronoun$</td>
<td>$NP \rightarrow I \mid she \mid me$</td>
</tr>
<tr>
<td>$NP \rightarrow Proper-Noun$</td>
<td>$NP \rightarrow TWA \mid Houston$</td>
</tr>
<tr>
<td>$NP \rightarrow Det \ Nominal$</td>
<td>$NP \rightarrow Det \ Nominal$</td>
</tr>
<tr>
<td>Nominal $\rightarrow Noun$</td>
<td>Nominal $\rightarrow book \mid flight \mid meal \mid money$</td>
</tr>
<tr>
<td>Nominal $\rightarrow Nominal \ Noun$</td>
<td>Nominal $\rightarrow Nominal \ Noun$</td>
</tr>
<tr>
<td>Nominal $\rightarrow Nominal \ PP$</td>
<td>Nominal $\rightarrow Nominal \ PP$</td>
</tr>
<tr>
<td>$VP \rightarrow Verb$</td>
<td>$VP \rightarrow book \mid include \mid prefer$</td>
</tr>
<tr>
<td>$VP \rightarrow Verb \ NP$</td>
<td>$VP \rightarrow Verb \ NP$</td>
</tr>
<tr>
<td>$VP \rightarrow Verb \ NP \ PP$</td>
<td>$VP \rightarrow X2 \ PP$</td>
</tr>
<tr>
<td>$VP \rightarrow Verb \ PP$</td>
<td>$X2 \rightarrow Verb \ NP$</td>
</tr>
<tr>
<td>$VP \rightarrow VP \ PP$</td>
<td>$VP \rightarrow VP \ PP$</td>
</tr>
<tr>
<td>$PP \rightarrow Preposition \ NP$</td>
<td>$PP \rightarrow Preposition \ NP$</td>
</tr>
</tbody>
</table>
CKY Parsing: Dynamic Programming

• Use indeces to point at gaps between words:
  
  _0 Book_ 1 _the_ 2 _flight_ 3 _through_ 4 _Houston_ 5

• A sentence with _n_ words ⇒ _n + 1_ positions.

• Define a (_n + 1_×(_n + 1)) matrix _T_:  
  – _T_[i,j] = the set of non-terminals that can generate the sequence of words between gaps _i_ and _j_.
  – _T_[0,n] contains _S_ ⇒ the sentence can be generated by the CFG.

• How can we compute _T_[i,j]?
  – Only interested in the upper-triangular portion (i.e. _i_ < _j_).
CKY: Dynamic Programming

- Recursively define the table values:
  1. $A \in T[i-1,i]$ if and only if there is a rule $A \rightarrow \text{words}[i]$.
  2. $A \in T[i,j]$ if and only if $\exists \ k, \ i < k < j$, such that:
     - $B \in T[i,k]$ and $C \in T[k,j]$.
     - There is a rule $A \rightarrow B C$ in the CFG.

- Bottom-up computation:
  - In order to compute the set $T[i,j]$, the sets $T[i,k]$ and $T[k,j]$ need to have been computed already, for all $i < k < j$.
  $\Rightarrow$ (at least) two possible orderings:
    - which one is more “natural”?
CKY: Bottom-Up Computation

\[
A[i,k] 

A[i,j] 

A[k,j] 
\]

0 1 2 3 4 5 6 7

i = 1

0 1 2 3 4 5 6 7

j = 6

1 2 3 4 5 6 7
CKY Parsing

- Fill the table a column at a time, left to right, bottom to top.

```
function CKY-PARSE(words, grammar) returns table

for j ← from 1 to LENGTH(words) do
    table[j - 1, j] ← \{A \mid A \rightarrow words[j] \in grammar\}

for i ← from j - 2 downto 0 do
    for k ← i + 1 to j - 1 do
        table[i,j] ← table[i,j] \cup \{A \mid A \rightarrow BC \in grammar, B \in table[i,k], C \in table[k,j]\}
```
### CKY Parsing: Example

<table>
<thead>
<tr>
<th></th>
<th>Book</th>
<th>the</th>
<th>flight</th>
<th>through</th>
<th>Houston</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0, 1]</td>
<td>S, VP, Verb Nominal, Noun</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[1, 2]</td>
<td>Det</td>
<td>NP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[2, 3]</td>
<td>Nominal, Noun</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[3, 4]</td>
<td>Prep</td>
<td>PP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[4, 5]</td>
<td>NP, Proper-Noun</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The diagram shows the CKY parsing process for the sentence "Book the flight through Houston." The arrows indicate the order and structure of the parsing process.
S → NP VP
S → X1 VP
X1 → Aux NP
S → book | include | prefer
S → Verb NP
S → X2 NP
X2 → Verb NP
S → VP PP
NP → I | he | she | me
NP → Houston | NWA
NP → Det Nominal
Nominal → book | flight | meal | money
Nominal → Nominal Noun
Nominal → Nominal PP
VP → book | include | prefer
VP → Verb NP
VP → VP PP
VP → X2 PP
PP → Prep NP
S → NP VP
S → X1 VP
X1 → Aux NP
S → book | include | prefer
S → Verb NP
S → X2 NP
X2 → Verb NP
S → VP PP
S → NP PP
NP → I | he | she | me
NP → Houston | NWA
NP → Det Nominal
Nominal → book | flight | meal | money
Nominal → Nominal Noun
Nominal → Nominal PP
VP → book | include | prefer
VP → Verb NP
VP → VP PP
VP → X2 PP
PP → Prep NP
S → NP VP
S → X1 VP
X1 → Aux NP
S → book | include | prefer
S → Verb NP
S → X2 NP
X2 → Verb NP
S → VP PP
NP → I | he | she | me
NP → Houston | NWA
NP → Det Nominal
Nominal → book | flight | meal | money
Nominal → Nominal Noun
Nominal → Nominal PP
VP → book | include | prefer
VP → Verb NP
VP → VP PP
VP → X2 PP
PP → Prep NP

<table>
<thead>
<tr>
<th>0</th>
<th>Book</th>
<th>1</th>
<th>the</th>
<th>2</th>
<th>flight</th>
<th>3</th>
<th>through</th>
<th>4</th>
<th>Houston</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>S, VP, Verb, Nominal, Noun</td>
<td></td>
<td>S, VP, X2</td>
<td></td>
<td>Noun, Nominal</td>
<td></td>
<td>Noun</td>
<td></td>
<td>Nominal</td>
<td></td>
</tr>
<tr>
<td>[0,1]</td>
<td>[0,2]</td>
<td>[0,3]</td>
<td>[0,4]</td>
<td>[0,5]</td>
<td>[1,2]</td>
<td>[1,3]</td>
<td>[1,4]</td>
<td>[1,5]</td>
<td>[2,3]</td>
<td>[2,4]</td>
</tr>
<tr>
<td>Det</td>
<td>NP</td>
<td>NP</td>
<td>Nominal</td>
<td>Nominal</td>
<td>Prep</td>
<td>PP</td>
<td>NP, Proper-Noun</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
S → NP VP
S → X1 VP
X1 → Aux NP
S → book | include | prefer
S → Verb NP
S → X2 NP
X2 → Verb NP
S → VP PP
S → X2 PP
PP → Prep NP
NP → I | he | she | me
NP → Houston | NWA
NP → Det Nominal
Nominal → book | flight | meal | money
Nominal → Nominal Noun
Nominal → Nominal PP
VP → book | include | prefer
VP → Verb NP
VP → VP PP
VP → X2 PP
PP → Prep NP
S → NP VP
S → X1 VP
X1 → Aux NP
S → book | include | prefer
S → Verb NP
S → X2 NP
X2 → Verb NP
S → VP PP
NP → I | he | she | me
NP → Houston | NWA
NP → Det Nominal
Nominal → book | flight | meal | money
Nominal → Nominal Noun
Nominal → Nominal PP
VP → book | include | prefer
VP → Verb NP
VP → VP PP
VP → X2 PP
PP → Prep NP
CKY Parsing

• How do we change the algorithm to output the parse trees?
• Time complexity:
  – for computing the table?
  – for computing all parses?

function CKY-PARSE(words, grammar) returns table

for j ← from 1 to LENGTH(words) do
  table[j - 1, j] ← \{A | A → words[j] ∈ grammar\}
  for i ← from j - 2 downto 0 do
    for k ← i + 1 to j - 1 do
      table[i,j] ← table[i,j] ∪
      \{A | A → BC ∈ grammar, B ∈ table[i,k], C ∈ table[k,j]\}
The parse trees correspond to the CNF grammar, not the original CFG:

\[ \rightarrow \] complicates subsequent syntax-direct semantic analysis.

Post-processing of the parse tree:

- For binary productions:
  - delete the new dummy non-terminals and promote their daughters to restore the original tree.

- For unit productions:
  - alter the basic CKY algorithm to handle them directly.
    - homework exercise 13.3
CKY Parsing

• Does CKY solve ambiguity?
  – Book the flight through Houston.

  Use *probabilistic* CKY parsing, output *highest probability* tree.

• Will probabilistic CKY solve all ambiguity?
  – One morning I shot an elephant in my pajamas.
    – How he got into my pajamas I don’t know.
Statistical Parsing

• Define a probabilistic model of syntax $P(T | S)$:
  • Probabilistic Context Free Grammars (PCFG).
  • Lexicalized PCFGs:
    – Collins’ parser, Charniak’s parser, …

• Use probabilistic model for:
  – Statistical parsing $\equiv$ choose the most probable parse:
    $$\hat{T}(S) = \arg\max_{T: \text{yield}(T) = S} P(T | S)$$
  – Language Modeling $\equiv$ compute the probability of a sentence:
    $$P(S) = \sum_{T: \text{yield}(T) = S} P(T, S)$$
Probabilistic CFG (PCFG)

- Augment each rule in a CFG with a conditional probability:

\[
A \rightarrow \beta \ [p]
\]

\[
p = p(A \rightarrow \beta) = p(\beta \mid A)
\]

\[
\sum_{\beta} p(A \rightarrow \beta) = 1
\]

\[N\] a set of non-terminal symbols (or variables)
\[\Sigma\] a set of terminal symbols (disjoint from \(N\))
\[R\] a set of rules or productions, each of the form \(A \rightarrow \beta \ [p]\),
where \(A\) is a non-terminal,
\(\beta\) is a string of symbols from the infinite set of strings \((\Sigma \cup N)^*\),
and \(p\) is a number between 0 and 1 expressing \(P(\beta \mid A)\)
\[S\] a designated start symbol
<table>
<thead>
<tr>
<th>Grammar</th>
<th>Lexicon</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S \rightarrow NP \ VP$</td>
<td>$Det \rightarrow that [.10]</td>
</tr>
<tr>
<td>$S \rightarrow Aux \ NP \ VP$</td>
<td>$Noun \rightarrow book [.10]</td>
</tr>
<tr>
<td>$S \rightarrow VP$</td>
<td>$\quad</td>
</tr>
<tr>
<td>$NP \rightarrow Pronoun$</td>
<td>$\quad</td>
</tr>
<tr>
<td>$NP \rightarrow Proper-Noun$</td>
<td>$Verb \rightarrow book [.30]</td>
</tr>
<tr>
<td>$NP \rightarrow Det Nominal$</td>
<td>$\quad</td>
</tr>
<tr>
<td>$NP \rightarrow Nominal$</td>
<td>$Pronoun \rightarrow I [.40]</td>
</tr>
<tr>
<td>Nominal $\rightarrow Noun$</td>
<td>$\quad</td>
</tr>
<tr>
<td>Nominal $\rightarrow Nominal Noun$</td>
<td>$Proper-Noun \rightarrow Houston [.60]</td>
</tr>
<tr>
<td>Nominal $\rightarrow Nominal PP$</td>
<td>$Aux \rightarrow does [.60]</td>
</tr>
<tr>
<td>$VP \rightarrow Verb$</td>
<td>$Preposition \rightarrow from [.30]</td>
</tr>
<tr>
<td>$VP \rightarrow Verb NP$</td>
<td>$\quad</td>
</tr>
<tr>
<td>$VP \rightarrow Verb NP PP$</td>
<td>$\quad</td>
</tr>
<tr>
<td>$VP \rightarrow Verb PP$</td>
<td>$\quad</td>
</tr>
<tr>
<td>$VP \rightarrow VP PP$</td>
<td>$\quad</td>
</tr>
<tr>
<td>$PP \rightarrow Preposition NP$</td>
<td>$\quad</td>
</tr>
</tbody>
</table>
Probability of Parse Trees

- Assume rewriting rules are chosen independently:
  \[ P(T) = P(T,S) = \prod_{i=1}^{n} P(RHS_i \mid LHS_i) \]

- **Statistical parsing** ≡ choose the most probable parse:
  \[ \hat{T}(S) = \arg \max_{T : \text{yield}(T) = S} P(T \mid S) = \arg \max_{T : \text{yield}(T) = S} P(T) \]

- **Language Modeling** ≡ compute the probability of a sentence:
  \[ P(S) = \sum_{T : \text{yield}(T) = S} P(T,S) = \sum_{T : \text{yield}(T) = S} P(T) \]
\[
P(T_1) = 0.05 \times 0.20 \times 0.20 \times 0.75 \times 0.30 \times 0.60 \times 0.10 \times 0.40 = 2.2 \times 10^{-6}
\]
\[
P(T_2) = 0.05 \times 0.10 \times 0.20 \times 0.15 \times 0.75 \times 0.75 \times 0.30 \times 0.60 \times 0.10 \times 0.40 = 6.1 \times 10^{-7}
\]
PCFGs

- **Statistical parsing** \(\equiv\) choose the most probable parse:
  \[
  \hat{T}(S) = \arg \max_{T:yield(T)=S} P(T) = T_1
  \]

- **Language Modeling** \(\equiv\) the probability of a sentence:
  \[
  P(S) = \sum_{T:yield(T)=S} P(T) = 2.2 \times 10^{-6} + 6.1 \times 10^{-7}
  \]
HMMs: Inference and Training

• **Three fundamental questions:**

  1) Given a model $\mu = (A, B, \Pi)$, compute the probability of a given observation sequence i.e. $p(O|\mu)$ (*Forward/Backward*).

  2) Given a model $\mu$ and an observation sequence $O$, compute the most likely hidden state sequence (*Viterbi*).

  $\hat{X} = \arg \max_{X} P(X | O, \mu)$

  3) Given an observation sequence $O$, find the model $\mu = (A, B, \Pi)$ that best explains the observed data (*EM*).

• Given observation and state sequence $O, X$, find $\mu$ (*ML*).
PCFGs: Inference and Training

• Three fundamental questions:
  1) Given a model $\mu$, compute the probability of a given sentence $S$ i.e. $p(S|\mu)$ (Inside/Outside).
     
     $$P(S) = \sum_{T: \text{yield}(T)=S} P(T, S) = \sum_{T: \text{yield}(T)=S} P(T)$$

  2) Given a model $\mu$ and a sentence $S$, compute the most likely parse tree ($pCKY$).
     
     $$\hat{T}(S) = \arg \max_{T: \text{yield}(T)=S} P(T | S) = \arg \max_{T: \text{yield}(T)=S} P(T)$$

  3) Given a set of sentences $\{S\}$, find the model $\mu$ that best explains the observed data ($EM$).
    
    • Given sentences and parses $\{S, T\}$ find $\mu$ ($ML$).
2) Probabilistic CKY (pCKY) Parsing

2) Given a model $\mu$ and a sentence $S$, compute the most likely parse tree ($pCKY$):

$$\hat{T}(S) = \arg \max_{T: \text{yield}(T)=S} P(T \mid S) = \arg \max_{T: \text{yield}(T)=S} P(T)$$

- CKY can be modified for PCFG parsing by including in each cell a probability for each non-terminal.
- $T[i,j]$ 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.
<table>
<thead>
<tr>
<th>Original Grammar</th>
<th>Chomsky Normal Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>S → NP VP</td>
<td>0.8 S → NP VP</td>
</tr>
<tr>
<td>S → Aux NP VP</td>
<td>0.1 S → X1 VP</td>
</tr>
<tr>
<td></td>
<td>X1 → Aux NP</td>
</tr>
<tr>
<td>S → VP</td>
<td>0.1 S → book</td>
</tr>
<tr>
<td></td>
<td>0.01 0.004 0.006</td>
</tr>
<tr>
<td></td>
<td>S → Verb NP</td>
</tr>
<tr>
<td></td>
<td>0.05 S → VP PP</td>
</tr>
<tr>
<td></td>
<td>0.03</td>
</tr>
<tr>
<td>NP → Pronoun</td>
<td>0.2 NP → I</td>
</tr>
<tr>
<td></td>
<td>0.1 0.02 0.02 0.06</td>
</tr>
<tr>
<td>NP → Proper-Noun</td>
<td>0.2 NP → Houston</td>
</tr>
<tr>
<td></td>
<td>0.16 .04</td>
</tr>
<tr>
<td>NP → Det Nominal</td>
<td>0.3 NP → Det Nominal</td>
</tr>
<tr>
<td>Nominal → Noun</td>
<td>0.6 Nominal → book</td>
</tr>
<tr>
<td></td>
<td>0.03 0.15 0.06 0.06</td>
</tr>
<tr>
<td>Nominal → Nominal Noun</td>
<td>0.2 Nominal → Nominal Noun</td>
</tr>
<tr>
<td>Nominal → Nominal PP</td>
<td>0.5 Nominal → Nominal PP</td>
</tr>
<tr>
<td>VP → Verb</td>
<td>0.2 VP → book</td>
</tr>
<tr>
<td></td>
<td>0.1 0.04 0.06</td>
</tr>
<tr>
<td>VP → Verb NP</td>
<td>0.5 VP → Verb NP</td>
</tr>
<tr>
<td>VP → VP PP</td>
<td>0.3 VP → VP PP</td>
</tr>
<tr>
<td>PP → Prep NP</td>
<td>1.0 PP → Prep NP</td>
</tr>
</tbody>
</table>
Probabilistic CKY (pCKY) Parsing

Book the flight through Houston

S:.01, VP:.1, Verb:.5, Nominal:.03, Noun:.1

None

Det:.6

NP:.6*.6*.15 = .054

Nominal:.15, Noun:.5
# Probabilistic CKY (pCKY) Parsing

![Animation by Ray Mooney](animation.png)

<table>
<thead>
<tr>
<th>Book</th>
<th>the</th>
<th>flight</th>
<th>through</th>
<th>Houston</th>
</tr>
</thead>
<tbody>
<tr>
<td>S : .01, VP : .1, Verb : .5, Nominal : .03, Noun : .1</td>
<td>None</td>
<td>VP : .5 * .5 * .054 = .0135</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>Det : .6</td>
<td>None</td>
<td>VP : .5 * .5 * .054 = .0135</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>Nominal : .15, Noun : .5</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>

- **S**: Start symbol
- **VP**: Verb Phrase
- **Nominal**: Noun Phrase
- **Noun**: Noun
- **Det**: Determiner
- **NP**: Noun Phrase
- **VP**: Verb Phrase
Probabilistic CKY (pCKY) Parsing

Animation by Ray Mooney

Book       the        flight    through  Houston

S :.01, VP:.1, Verb:.5 ← Nominal:.03 Noun:.1

None

Det:.6

NP:.6*.6*.15 =.054

Nominal:.15 Noun:.5

S:.05*.5*.054 =.00135

VP:.5*.5*.054 =.0135

S:.05*.5*.054 =.00135
Probabilistic CKY (pCKY) Parsing

![Animation by Ray Mooney]

<table>
<thead>
<tr>
<th></th>
<th>Book</th>
<th>the</th>
<th>flight</th>
<th>through</th>
<th>Houston</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>.01, VP:.1, Verb:.5, Nominal:.03, Noun:.1</td>
<td>None</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VP</td>
<td>.05*.5*.054 = .00135</td>
<td>None</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S</td>
<td>.05*.5*.054 = .0135</td>
<td>None</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NP</td>
<td>.6*.6*.15 = .054</td>
<td>None</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nominal</td>
<td>.15</td>
<td>None</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Noun</td>
<td>.5</td>
<td>None</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Prep</td>
<td>.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Probabilistic CKY (pCKY) Parsing

[Animation by Ray Mooney]

<table>
<thead>
<tr>
<th></th>
<th>Book</th>
<th>the</th>
<th>flight</th>
<th>through</th>
<th>Houston</th>
</tr>
</thead>
<tbody>
<tr>
<td>S :.01, VP:.1, Verb:.5, Nominal:.03, Noun:.1</td>
<td>None</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>S :.05*.5*.054 = .00135</td>
<td>None</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>VP :.5*.5*.054 = .0135</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Det :.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NP :.6*.6*.15 = .054</td>
<td>None</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Nominal :.15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Noun :.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Prep :.2</td>
<td></td>
<td>PP :1.0*.2*.16 = .032</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NP :.16</td>
<td>PropNoun :.8</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Probabilistic CKY (pCKY) Parsing

[Animation by Ray Mooney]

<table>
<thead>
<tr>
<th>Book</th>
<th>the</th>
<th>flight</th>
<th>through</th>
<th>Houston</th>
</tr>
</thead>
<tbody>
<tr>
<td>S (.01, VP:.1, Verb:.5, Nominal:.03, Noun:.1)</td>
<td>None</td>
<td>S (.05<em>5</em>.054 =.00135)</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>Det:.6</td>
<td>NP(.6<em>6</em>.15 =.054)</td>
<td>None</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nominal:.15</td>
<td>Noun:.5</td>
<td>None</td>
<td>Nominal(.5<em>15</em>.032 =.0024)</td>
<td></td>
</tr>
<tr>
<td>Prep:.2</td>
<td>PP(1.0<em>2</em>.16 =.032)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NP(.16)</td>
<td>PropNoun(.8)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Probabilistic CKY (pCKY) Parsing

[Animation by Ray Mooney]

```
<table>
<thead>
<tr>
<th>Book</th>
<th>the</th>
<th>flight</th>
<th>through</th>
<th>Houston</th>
</tr>
</thead>
<tbody>
<tr>
<td>S :.01, VP:.1, Verb:.5, Nominal:.03, Noun:.1</td>
<td>None</td>
<td>S :.05*.5*.054 = .00135</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Det:.6</td>
<td>VP :.5*.5*.054 = .0135</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Nominal:.15, Noun:.5</td>
<td>None</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Prep:.2</td>
<td>PP :1.0*.2*.16 = .032</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NP:.16</td>
<td>PropNoun:.8</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Animation by Ray Mooney
Probabilistic CKY (pCKY) Parsing

<table>
<thead>
<tr>
<th>Book</th>
<th>the</th>
<th>flight</th>
<th>through</th>
<th>Houston</th>
</tr>
</thead>
<tbody>
<tr>
<td>S :.01, VP:.1, Verb:.5&lt; Nominal:.03 Noun:.1</td>
<td>None</td>
<td>S:.05*.5*.054 =.00135</td>
<td>None</td>
<td>S:.05*.5* .000864 =.0000216</td>
</tr>
<tr>
<td>Det:.6</td>
<td>NP:.6*.6*.15 =.054</td>
<td>None</td>
<td>NP:.6*.6* .0024 =.000864</td>
<td></td>
</tr>
<tr>
<td>Nominal:.15 Noun:.5</td>
<td>None</td>
<td>Nominal: .5*.15*.032 =.0024</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Prep:.2</td>
<td>PP:1.0*.2*.16 =.032</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NP:.16 PropNoun:.8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Probabilistic CKY (pCKY) Parsing

Book the flight through Houston

<table>
<thead>
<tr>
<th></th>
<th>the</th>
<th>flight</th>
<th>through</th>
<th>Houston</th>
</tr>
</thead>
<tbody>
<tr>
<td>S :01, VP:.1, Verb:.5, Nominal:.03, Noun:.1</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>Det:.6</td>
<td>NP:.6*.6*.15 =.054</td>
<td>None</td>
<td>None</td>
<td>NP:.6*.6* .024 =.000864</td>
</tr>
<tr>
<td>Nominal:.15, Noun:.5</td>
<td>None</td>
<td>None</td>
<td>Nominal:.5*.15*.032 =.0024</td>
<td></td>
</tr>
<tr>
<td>Prep:.2</td>
<td>PP:1.0*.2*.16 =.032</td>
<td>None</td>
<td>None</td>
<td>NP:.16 PropNoun:.8</td>
</tr>
</tbody>
</table>

Animation by Ray Mooney
Probabilistic CKY (pCKY) Parsing

<table>
<thead>
<tr>
<th>Book</th>
<th>the</th>
<th>flight</th>
<th>through</th>
<th>Houston</th>
</tr>
</thead>
<tbody>
<tr>
<td>S :.01, VP:.1, Verb:.5, Nominal:.03, Noun:.1</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>S:.0000216</td>
</tr>
<tr>
<td>Det:.6</td>
<td>NP:.6*.6*.15 = .054</td>
<td>None</td>
<td>None</td>
<td>NP:.6*.6* .0024 = .000864</td>
</tr>
<tr>
<td>Nominal:.15, Noun:.5</td>
<td>None</td>
<td>Nominal:.5*.15*.032 = .0024</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Prep:.2</td>
<td>PP:1.0*.2*.16 = .032</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NP:.16</td>
<td>PropNoun:.8</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Most probable parse.
Probabilistic CKY Algorithm

function PROBABILISTIC-CKY(words, grammar) returns most probable parse and its probability

for \( j \leftarrow 1 \) to \( \text{LENGTH}(words) \) do
  for all \( \{ A \mid A \rightarrow \text{words}[j] \in \text{grammar} \} \)
    \( \text{table}[j-1, j, A] \leftarrow P(A \rightarrow \text{words}[j]) \)
  for \( i \leftarrow j-2 \) downto 0 do
    for \( k \leftarrow i+1 \) to \( j-1 \) do
      for all \( \{ A \mid A \rightarrow BC \in \text{grammar}, \)
        \( \text{and} \ \text{table}[i, k, B] > 0 \ \text{and} \ \text{table}[k, j, C] > 0 \} \)
        if \( (\text{table}[i, j, A] < P(A \rightarrow BC) \times \text{table}[i, k, B] \times \text{table}[k, j, C]) \) then
          \( \text{table}[i, j, A] \leftarrow P(A \rightarrow BC) \times \text{table}[i, k, B] \times \text{table}[k, j, C] \)
          \( \text{back}[i, j, A] \leftarrow \{ k, B, C \} \)
      return BUILD_TREE(back[1, \text{LENGTH}(words), S]), table[1, \text{LENGTH}(words), S]
1) Observation Probability using pCKY

1) Given a model $\mu$, compute the probability of a given sentence $S$ i.e. $p(S|\mu)$ ($\text{Inside/Outside}$):

- Use Inside probabilities, the analogue of Backward probabilities in HMMs:
  \[ \beta_j(p,q) = p(w_{pq} | N_{pq}^j, G) \]
  - Compute Inside probabilities by replacing $\text{max}$ with $\text{sum}$ inside the pCKY algorithm.

- Or use Outside probs, the analogue of Forward probs in HMMs.
Probabilistic CKY (pCKY) Parsing: Sum

<table>
<thead>
<tr>
<th></th>
<th>Book</th>
<th>the</th>
<th>flight</th>
<th>through</th>
<th>Houston</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>:01, VP:.1, Verb:.5, Nominal:.03, Noun:.1</td>
<td>None</td>
<td>S:.05<em>5</em>0.054 =.00135</td>
<td>None</td>
<td>S:.00001296</td>
</tr>
<tr>
<td>VP</td>
<td>:.5<em>5</em>0.054 =.0135</td>
<td>None</td>
<td>VP:.5<em>5</em>0.054 =.0135</td>
<td>None</td>
<td>S:.0000216</td>
</tr>
<tr>
<td>NP</td>
<td>:6<em>6</em>0.15 =.054</td>
<td>None</td>
<td>None</td>
<td>NP:.6<em>6</em> .0024 =.000864</td>
<td></td>
</tr>
<tr>
<td>Nominal</td>
<td>:.15 Noun:.5</td>
<td>None</td>
<td>Nominal: .5*.15*.032 =.0024</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Prep</td>
<td>:.2</td>
<td>PP:1.0<em>2</em>.16 =.032</td>
<td>None</td>
<td>NP:.16 PropNoun:.8</td>
<td></td>
</tr>
</tbody>
</table>
Probabilistic CKY (pCKY) Parsing: Sum

<table>
<thead>
<tr>
<th>Book</th>
<th>the</th>
<th>flight</th>
<th>through</th>
<th>Houston</th>
</tr>
</thead>
<tbody>
<tr>
<td>S : .01, VP:.1, Verb:.5, Nominal:.03, Noun:.1</td>
<td>None</td>
<td>S:.05*.5*.054 =.00135</td>
<td>None</td>
<td>S: .00001296 +.0000216 =.00003456</td>
</tr>
<tr>
<td>Det:.6</td>
<td>NP:.6*.6*.15 =.054</td>
<td>None</td>
<td>NP:.6*.6* .0024 =.000864</td>
<td></td>
</tr>
<tr>
<td>Nominal:.15 Noun:.5</td>
<td>None</td>
<td>Nominal:.5*.15*.032 =.0024</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Prep:.2</td>
<td>PP:1.0*.2*.16 =.032</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NP:.16 PropNoun:.8</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Sum probabilities of each derivation.
PCFG Training: Unsupervised

3) Given a set of sentences \( \{S\} \), find the model \( \mu \) that best explains the observed data (EM).
   - Use **Inside-Outside**, a generalization of **Forward-Backward**.
     1. Begin with a grammar with equal rule probabilities / random.
     2. For each sentence, compute the probability of each parse.
     3. Re-estimate the rule probabilities by using the parse probabilities as weights for the Counts.
     4. Repeat from 2, until probabilities converge.
   - Problems: each iteration is slow \( O(m^3n^3) \), sensitive to initialization
     \( \Rightarrow \) many local maxima, no guarantee learned non-terminals correspond to linguistic intuitions / constituents.
   - Exact algorithm in M&S, pages 398 – 402.
3) Given sentences and parses \( \{S, T\} \) find \( \mu \) (ML):

- estimate parameters directly from counts in the treebank.

\[
P(\alpha \rightarrow \beta | \alpha) = \frac{\text{count}(\alpha \rightarrow \beta)}{\sum_{\gamma} \text{count}(\alpha \rightarrow \gamma)} = \frac{\text{count}(\alpha \rightarrow \beta)}{\text{count}(\alpha)}
\]
Limitations of Vanilla PCFGs

- **Poor Independence Assumptions:**
  - cannot model the fact that:
    - NPs that are syntactic subjects are far more likely to be pronouns.
    - NPs that are syntactic objects are far more likely to be non-pronominal.

<table>
<thead>
<tr>
<th></th>
<th>Pronoun</th>
<th>Non-Pronoun</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subject</td>
<td>91%</td>
<td>9%</td>
</tr>
<tr>
<td>Object</td>
<td>34%</td>
<td>66%</td>
</tr>
</tbody>
</table>

- **Lack of Lexical Conditioning:**
  - can only model general preference for PP attachment to NPs vs VPs.
    - but PPs sometimes attach to NPs, sometimes attach to VPs, depending on the actual Verb, Preposition, Noun.
Splitting Non-terminals

- Split the NP into two versions: one for subjects, one for objects.
- **Parent Annotation:**
  - annotate each node with its parent in the parse tree.
    - NP subject ⇒ annotated as NP^S
    - NP object ⇒ annotated as NP^VP
Splitting Non-terminals

- Split pre-terminals to allow *if* to prefer a sentential complements:
Split and Merge [Petrov et al., 2006]

- Node splitting increases the size of the grammar ⇒ need to find the right level of granularity:
  - automatically search for the optimal splits.
  - start with a simple X-bar grammar.
  - alternate between splitting and merging non-terminals.
  - stop when likelihood of training treebank is maximized.

- Alternatively, use hand-written rules to find an optimal number of non-terminals:
  - [Klein and Manning, 2003]
If preference is given to verb attachment, then the PCFG get the wrong parse for “fishermen caught tons of herring”.

\[ \text{VP} \rightarrow \text{VBD NP PP} \]

\[ \text{VP} \rightarrow \text{VBD NP} \]
\[ \text{NP} \rightarrow \text{NP PP} \]
Lexicalized PCFGs

- **Lexicalized Grammar** ≡ in every rule, associate each non-terminal symbol with its lexical **head** and **head tag**.

\[
\begin{align*}
\text{VP}(\text{dumped}, \text{VBD}) & \rightarrow \\
\text{VBD}(\text{dumped}, \text{VBD}) \text{ NP}(\text{sacks}, \text{NNS}) \text{ PP}(\text{into}, \text{IN})
\end{align*}
\]
Internal Rules

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOP</td>
<td>$S(dumped,VBD)$</td>
</tr>
<tr>
<td>$S(dumped,VBD)$</td>
<td>$NP(workers,NNS)$, $VP(dumped,VBD)$</td>
</tr>
<tr>
<td>$NP(workers,NNS)$</td>
<td>$NNS(workers,NNS)$</td>
</tr>
<tr>
<td>$NNS(workers,NNS)$</td>
<td>$VBD(dumped,VBD)$, $VP(dumped,VBD)$</td>
</tr>
<tr>
<td>$VBD(dumped,VBD)$</td>
<td>$NP(sacks,NNS)$, $PP(into,P)$</td>
</tr>
<tr>
<td>$PP(into,P)$</td>
<td>$NP(bin,NN)$</td>
</tr>
<tr>
<td>$NP(bin,NN)$</td>
<td>$DT(a,DT), NN(bin,NN)$</td>
</tr>
</tbody>
</table>

Lexical Rules

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$NNS(workers,NNS)$</td>
<td>$workers$</td>
</tr>
<tr>
<td>$VBD(dumped,VBD)$</td>
<td>$dumped$</td>
</tr>
<tr>
<td>$NNS(sacks,NNS)$</td>
<td>$sacks$</td>
</tr>
<tr>
<td>$PP(into,P)$</td>
<td>$into$</td>
</tr>
<tr>
<td>$NP(bin,NN)$</td>
<td>$bin$</td>
</tr>
</tbody>
</table>
Lexicalized PCFGs

- **Lexicalized Grammar**  
  - in every rule, associate each non-terminal symbol with its lexical **head** and **head tag**.
  - important to have rules for head identification.

  \[
  \text{VP}(\text{dumped, VBD}) \rightarrow \\
  \text{VBD}(\text{dumped, VBD}) \text{ NP}(\text{sacks, NNS}) \text{ PP}(\text{into, IN})
  \]

- Estimating the corresponding probabilities is not feasible, due to sparse counts.
- Need to make further independence assumptions ⇒ Collins’ Parser.
Collins’ Parser

• All rules are expressed as:
  - $P(h) \rightarrow L_{n+1} L_n(l_n) \ldots L_1(l_1) H(h) R_1(r_1) \ldots R_m(r_m) R_{m+1}$
  - where $L_{n+1} = \text{STOP}$, $R_{m+1} = \text{STOP}$

• Generative story:
  1. Generate the head label of the phrase: $P_h(H|P,h)$
  2. Generate modifiers to the left of the head, independently given the head info: $P_L(L_i(l_i)|P,h,H)$
     • stop when STOP is generated.
  3. Generate modifiers to the right of the head, independently given the head info: $P_R(R_i(r_i)|P,h,H)$
     • stop when STOP is generated.
Workers \([_{VP} \text{ dumped sacks into bins}].\)

\[ VP(\text{dumped, VBD}) \rightarrow \]
\[ \text{STOP} \ VBD(\text{dumped, VBD}) \ NP(\text{sacks, NNS}) \ PP(\text{into, IN}) \ \text{STOP} \]

\[ P(h) \rightarrow L_{n+1} \ L_n(l_n) \ \ldots \ L_1(l_1) \ H(h) \ R_1(r_1) \ \ldots \ R_m(r_m) \ R_{m+1} \]
\[ n = 0, \ m = 2 \]
\[ P = VP, \ H = VBD, \ L_1 = \text{STOP}, \ R_1 = NP, \ R_2 = PP, \ R_3 = \text{STOP} \]
\[ h = \langle \text{dumped, VBD} \rangle, \ r_1 = \langle \text{sacks, NNS} \rangle, \ r_2 = \langle \text{dumped, VBD} \rangle \]

\[ P_H(VBD \mid VP, \text{dumped}) \times P_L(\text{STOP} \mid VP, VBD, \text{dumped}) \]
\[ \times P_R(NP(\text{sacks, NNS}) \mid VP, VBD, \text{dumped}) \]
\[ \times P_R(PP(\text{into, IN}) \mid VP, VBD, \text{dumped}) \]
\[ \times P_R(\text{STOP} \mid VP, VBD, \text{dumped}) \]
Collins’ Parser: Training

• Estimate $P_H$, $P_L$ and $P_R$ from treebank data:

$$P_R(PP\text{into-IN} \mid VP\text{dumped-VBD}) = \frac{\text{Count}(PP\text{into-IN right of head in a VP\text{dumped-VBD production})}}{\text{Count(symbol right of head in a VP\text{dumped-VBD})}}$$

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

$$P_R(PP\text{into-IN} \mid VP\text{dumped-VBD}) = \lambda_1 P_R(PP\text{into-IN} \mid VP\text{dumped-VBD}) + (1-\lambda_1) \left( \lambda_2 P_R(PP\text{into-IN} \mid VP\text{VBD}) + (1-\lambda_2) P_R(PP\text{into-IN} \mid VP) \right)$$

Witten-Bell discounting.
Collins’ Parser

- Model 1 also conditions on a distance feature:
  - distance as a function of words between modifier and head:
    - is the distance 0?
    - do the words contain a verb?

- Model 2 adds more sophisticated features:
  - condition on the subcategorization frames for each verb.
  - distinguish arguments from adjuncts.
    - *IBM bought* Lotus *yesterday*.

- Parsing algorithm is an extension of pCKY.
Shallow Parsing: Chunking

- **Chunking** = find all non-recursive major types of phrases:
  - \([\text{NP} \text{ The morning flight}] [\text{PP from}] [\text{NP Denver}] [\text{VP has arrived}]\)
  - \([\text{NP The morning flight}] \text{ from } [\text{NP Denver}] \text{ has arrived}\)

- Chunking can be approached as **Sequence Labeling**.

- Evaluation:

  \[
  \text{Precision (P)} = \frac{\# \text{correct chunks found}}{\text{total \# chunks found}}
  \]

  \[
  \text{Recall (R)} = \frac{\# \text{correct chunks found}}{\text{total \# actual chunks}}
  \]

  \[
  F = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}
  \]

  \[
  F_1 = \frac{2PR}{P + R}
  \]

Currently, best NP chunking system obtains \(F_1=96\%\).