Probabilistic CFG

- normalization
  - $\sum_{\beta} p(A \rightarrow \beta) = 1$
- where are these probs from?
  - “gold-stand” trees -- Treebank
  - $p(A \rightarrow \beta) = \#(A \rightarrow \beta) / \#(A)$
- what’s the most likely tree? *easy*
- what’s the most likely string? *hard*
- given string $w$, what’s the most likely tree for $w$?
  - this is called “parsing” (like decoding)

<table>
<thead>
<tr>
<th>Rule</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S \rightarrow NP\ VP$</td>
<td>0.8</td>
</tr>
<tr>
<td>$S \rightarrow S\ conj\ S$</td>
<td>0.2</td>
</tr>
<tr>
<td>$NP \rightarrow Noun$</td>
<td>0.2</td>
</tr>
<tr>
<td>$NP \rightarrow Det\ Noun$</td>
<td>0.4</td>
</tr>
<tr>
<td>$NP \rightarrow NP\ PP$</td>
<td>0.2</td>
</tr>
<tr>
<td>$NP \rightarrow NP\ conj\ NP$</td>
<td>0.2</td>
</tr>
<tr>
<td>$VP \rightarrow Verb$</td>
<td>0.4</td>
</tr>
<tr>
<td>$VP \rightarrow Verb\ NP$</td>
<td>0.3</td>
</tr>
<tr>
<td>$VP \rightarrow Verb\ NP\ NP$</td>
<td>0.1</td>
</tr>
<tr>
<td>$VP \rightarrow VP\ PP$</td>
<td>0.2</td>
</tr>
<tr>
<td>$PP \rightarrow P\ NP$</td>
<td>1.0</td>
</tr>
</tbody>
</table>
Probability of a tree

The probability of a tree $\tau$ is the product of the probabilities of all its rules:

$$P(\tau) = 0.8 \times 0.3 \times 0.2 \times 1.0 \times 0.2^3$$

$$= 0.00384$$
Most Likely Tree - Knuth 77

- highest-probability tree out of a PCFG
- idea 1: dynamic programming (DP)
  - optimize for each nonterminal $X$
  - problem: cyclic updates
    - if $S \rightarrow NP \ VP$ and $VP \rightarrow VB \ S$
    - can’t use Viterbi (which relies on topological ordering)
- idea 2: best-first DP: Dijkstra! (b/c prob $\leq 1$, cf. non-negative)
  - every time choose the best nonterminal in the queue to expand
  - but not always possible to combine w/ others to update LHS
  - update LHS only when all RHS nonterminals are ready (popped)
Knuth 77 Example

- initial queue = (NP: 0.4, VP: 0.4)
- say pop VP: 0.4
  - which rules can be used for updates?
    - S -> NP VP 0.8; VP -> VP PP 0.2
    - can we use these rules now?
      - No, b/c NP and PP are not ready
- next, pop NP: 0.4
  - which rules can be used for updates?
    - S -> NP VP 0.8; PP -> P NP 1.0
    - update S to be 0.8x0.4x0.4=0.128 and PP to be 1.0x0.4=0.4
- next, pop PP: 0.4; but can’t update anything (NP/VP already popped)
- next, pop S: 0.128, and finishes here since S is the start symbol
Most likely tree given string

- Parsing is to search for the best tree $t^*$ that:
  - $t^* = \arg \max_t p(t \mid w) = \arg \max_t p(t) p(w \mid t)$
  - $= \arg \max_{\{t: \text{yield}(t)=w\}} p(t)$
- Analogous to HMM decoding
- Is it related to “intersection” or “composition” in FSTs?

\[
\text{yield} \left( \begin{array}{c}
  V \\
  \text{eat} \\
  \end{array} \right) \\
\begin{array}{c}
  \text{NP} \\
  \text{sushi} \\
  \end{array} \\
\begin{array}{c}
  \text{PP} \\
  \text{with tuna} \\
  \end{array} \\
\right) = \text{eat sushi with tuna}
\]
CKY Algorithm

- For each diff (\( \leq n \))
  - For each i (\( \leq n \))
    - For each rule \( X \rightarrow Y Z \)
      - For each split point k
        \[
        \text{score}[X][i][j] = \max \text{ score}[X][i][j], \text{ score}(X\rightarrowYZ) \times \text{ score}[Y][i][k] \times \text{ score}[Z][k][j]
        \]

CS 544 - Syntax and Parsing
CKY Algorithm

- For each diff ($\leq n$)
  - For each $i$ ($\leq n$)
    - For each rule $X \rightarrow Y Z$
      - For each split point $k$
        \[
        \text{score}[X][i][j] = \max \text{score}[X][i][j], \text{score}(X \rightarrow YZ) \ast \text{score}[Y][i][k] \ast \text{score}[Z][k][j]
        \]

flies like a flower

S $\rightarrow$ NP VP
NP $\rightarrow$ DT NN
NP $\rightarrow$ NNS
NP $\rightarrow$ NP PP
VP $\rightarrow$ VB NP
VP $\rightarrow$ VP PP
VP $\rightarrow$ VB
PP $\rightarrow$ P NP

VB $\rightarrow$ flies
NNS $\rightarrow$ flies
VB $\rightarrow$ like
P $\rightarrow$ like
DT $\rightarrow$ a
NN $\rightarrow$ flower
CKY Algorithm

- For each diff (\( \leq n \))
  - For each i (\( \leq n \))
    - For each rule \( X \rightarrow Y Z \)
      - For each split point \( k \)
        
        \[
        \text{score}[X][i][j] = \max \text{score}[X][i][j], \\
        \text{score}(X \rightarrow YZ) \times \\
        \text{score}[Y][i][k] \times \\
        \text{score}[Z][k][j]
        \]

- \( S \rightarrow \text{NP VP} \)
- \( \text{NP} \rightarrow \text{DT NN} \)
- \( \text{NP} \rightarrow \text{NNS} \)
- \( \text{NP} \rightarrow \text{NP PP} \)
- \( \text{VP} \rightarrow \text{VB NP} \)
- \( \text{VP} \rightarrow \text{VP PP} \)
- \( \text{VP} \rightarrow \text{VB} \)
- \( \text{PP} \rightarrow \text{P NP} \)

flies like a flower

note: unary rules
## CKY Example

**Input: POS-tagged sentence**

John\_N eats\_V pie\_N with\_P cream\_N

```
<table>
<thead>
<tr>
<th></th>
<th>John</th>
<th>eats</th>
<th>pie</th>
<th>with</th>
<th>cream</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>NP 0.2</td>
<td>S 0.8<em>0.2</em>0.4</td>
<td>S 0.8<em>0.2</em>0.08</td>
<td>S 0.2<em>0.0024</em>0.8</td>
<td>John</td>
</tr>
<tr>
<td>V</td>
<td>VP 0.4</td>
<td>VP 0.3*0.2</td>
<td>VP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>NP 0.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P</td>
<td></td>
<td></td>
<td></td>
<td>PP 1*0.2</td>
<td></td>
</tr>
</tbody>
</table>
```

- **Rules:**
  - S → NP VP 0.8
  - S → S conj S 0.2
  - NP → Noun 0.2
  - NP → Det Noun 0.4
  - NP → NP PP 0.2
  - NP → NP conj NP 0.2
  - VP → Verb 0.4
  - VP → Verb NP 0.3
  - VP → Verb NP NP 0.1
  - VP → VP PP 0.2
  - PP → P NP 1.0

---

CS 498 JH: *Introduction to NLP* (Fall 08)
Chomsky Normal Form

- wait! how can you assume a CFG is binary-branching?
- well, we can always convert a CFG into Chomsky-Normal Form (CNF)
  - A → B C
  - A → a
- how to deal with epsilon-removal?
- how to do it with PCFG?