Lecture 17: Statistical Parsing with PCFGs

Julia Hockenmaier
juliahmr@illinois.edu
3324 Siebel Center
Back to the lecture
Recap: CKY algorithm

1. Create the chart
   (an \(n \times n\) upper triangular matrix for an sentence with \(n\) words)
   - Each cell chart\([i][j]\) corresponds to the substring \(w^i \ldots w^j\)

2. Initialize the chart (fill the diagonal cells chart\([i][i]\)):
   - For all rules \(X \rightarrow w^i\), add an entry \(X\) to chart\([i][i]\)

3. Fill in the chart:
   - Fill in all cells chart\([i][i+1]\), then chart\([i][i+2]\), ..., until you reach chart\([1][n]\) (the top right corner of the chart)
   - To fill chart\([i][j]\), consider all binary splits \(w^i \ldots w^k|w^{k+1} \ldots w^j\)
   - If the grammar has a rule \(X \rightarrow YZ\), chart\([i][k]\) contains a \(Y\) and chart\([k+1][j]\) contains a \(Z\), add an \(X\) to chart\([i][j]\) with two backpointers to the \(Y\) in chart\([i][k]\) and the \(Z\) in chart\([k+1][j]\)

4. Extract the parse trees from the \(S\) in chart\([1][n]\).
What are the terminals in NLP?

Are the “terminals” words or POS tags?

For toy examples (e.g. on slides), it’s typically the words

With POS-tagged input, we may either treat the POS tags as
the terminals, or we assume that the unary rules in our
grammar are of the form

   POS-tag → word

(so POS tags are the only nonterminals that can be rewritten
as words; some people call POS tags “preterminals”)

Additional unary rules

In practice, we may allow other unary rules, e.g.
   \[ NP \rightarrow \text{Noun} \]
(where Noun is also a nonterminal)

In that case, we apply all unary rules to the entries
in chart[i][j] after we have checked all binary splits
(chart[i][k], chart[k+1][j])

Unary rules are fine as long as there are no “loops”
that could lead to an infinite chain of unary
productions, e.g.:
   \[ X \rightarrow Y \quad \text{and} \quad Y \rightarrow X \]
   \[ \text{or: } X \rightarrow Y \quad \text{and} \quad Y \rightarrow Z \quad \text{and} \quad Z \rightarrow X \]
CKY so far…

Each entry in a cell $chart[i][j]$ is associated with a nonterminal $X$.

If there is a rule $X \rightarrow YZ$ in the grammar, and there is a pair of cells $chart[i][k]$, $chart[k+1][j]$ with a $Y$ in $chart[i][k]$ and a $Z$ in $chart[k+1][j]$, we can add an entry $X$ to cell $chart[i][j]$, and associate one pair of backpointers with the $X$ in cell $chart[i][k]$

Each entry might have multiple pairs of backpointers.

When we extract the parse trees at the end, we can get all possible trees.
We will need probabilities to find the single best tree!
How do you count the **number of parse trees** for a sentence?

1. For each **initial item** (e.g. \( V \rightarrow \text{eat} \)): \#trees = 1
   
   \[
   \text{trees}(V_V \rightarrow \text{eat}) = 1
   \]

2. For each **pair of backpointers** (e.g. \( \text{VP} \rightarrow V \ NP \)): **multiply** \#trees of children
   
   \[
   \text{trees}(\text{VP}_{\text{VP}} \rightarrow V_{\text{NP}}) = \text{trees}(V) \times \text{trees}(\text{NP})
   \]

3. For each **list of pairs of backpointers** (e.g. \( \text{VP} \rightarrow V \ NP \) and \( \text{VP} \rightarrow \text{VP} \ PP \)): **sum** \#trees
   
   \[
   \text{trees}(\text{VP}) = \text{trees}(\text{VP}_{\text{VP}} \rightarrow V_{\text{NP}}) + \text{trees}(\text{VP}_{\text{VP}} \rightarrow \text{VP}_{\text{PP}})
   \]
Exercise: CKY parser

I eat sushi with chopsticks with you

S → NP VP
NP → NP PP
NP → sushi
NP → I
NP → chopsticks
NP → you
VP → VP PP
VP → Verb NP
Verb → eat
PP → Prep NP
Prep → with
Dealing with ambiguity: Probabilistic Context-Free Grammars (PCFGs)
Grammars are ambiguous

A grammar might generate multiple trees for a sentence:

What’s the most likely parse $\tau$ for sentence $S$?

**We need a model of** $P(\tau \mid S)$
Computing $P(\tau \mid S)$

Using Bayes’ Rule:

$$\arg\max_{\tau} P(\tau \mid S) = \arg\max_{\tau} \frac{P(\tau, S)}{P(S)}$$

$$= \arg\max_{\tau} P(\tau, S)$$

$$= \arg\max_{\tau} P(\tau) \text{ if } S = \text{yield}(\tau)$$

The **yield of a tree** is the string of terminal symbols that can be read off the leaf nodes

```
yield(  V       VP
      NP    PP  NP
    eat   sushi with tuna ) = eat sushi with tuna
```
Computing $P(\tau)$

$T$ is the (infinite) set of all trees in the language:

$$L = \{ s \in \Sigma^* | \exists \tau \in T : \text{yield}(\tau) = s \}$$

The set $T$ is generated by a context-free grammar:

$$
\begin{align*}
S & \rightarrow \ NP \ VP \\
VP & \rightarrow \ Verb \ NP \\
NP & \rightarrow \ Det \ Noun \\
S & \rightarrow \ S \ \text{conj} \ S \\
VP & \rightarrow \ VP \ PP \\
NP & \rightarrow \ NP \ PP \\
S & \rightarrow \ \ldots \\
VP & \rightarrow \ \ldots \\
NP & \rightarrow \ \ldots
\end{align*}
$$

We need to define $P(\tau)$ such that:

$$\forall \tau \in T : \quad 0 \leq P(\tau) \leq 1$$

$$\sum_{\tau \in T} P(\tau) = 1$$
Probabilistic Context-Free Grammars

For every nonterminal \( X \), define a probability distribution \( P(X \to \alpha | X) \) over all rules with the same LHS symbol \( X \):

<table>
<thead>
<tr>
<th>Rule</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S \to NP \ VP )</td>
<td>0.8</td>
</tr>
<tr>
<td>( S \to S \ conj \ S )</td>
<td>0.2</td>
</tr>
<tr>
<td>( NP \to Noun )</td>
<td>0.2</td>
</tr>
<tr>
<td>( NP \to Det \ Noun )</td>
<td>0.4</td>
</tr>
<tr>
<td>( NP \to NP \ PP )</td>
<td>0.2</td>
</tr>
<tr>
<td>( NP \to NP \ conj \ NP )</td>
<td>0.2</td>
</tr>
<tr>
<td>( VP \to Verb )</td>
<td>0.4</td>
</tr>
<tr>
<td>( VP \to Verb \ NP )</td>
<td>0.3</td>
</tr>
<tr>
<td>( VP \to Verb \ NP \ NP )</td>
<td>0.1</td>
</tr>
<tr>
<td>( VP \to VP \ PP )</td>
<td>0.2</td>
</tr>
<tr>
<td>( PP \to P \ NP )</td>
<td>1.0</td>
</tr>
</tbody>
</table>
Computing $P(\tau)$ with a PCFG

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

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

$$= 0.00384$$
Learning the parameters of a PCFG

If we have a treebank (a corpus in which each sentence is associated with a parse tree), we can just count the number of times each rule appears, e.g.:

\[ S \rightarrow \text{NP VP} \ . \ (1000) \quad S \rightarrow \text{S conj S} \ . \ (220) \]

etc.

and then we divide the observed frequency of each rule \( X \rightarrow Y Z \) by the sum of the frequencies of all rules with the same LHS \( X \) to turn these counts into probabilities:

\[ S \rightarrow \text{NP VP} \ . \quad (p = 1000/1220) \]
\[ S \rightarrow \text{S conj S} \ . \quad (p = 220/1220) \]
More on probabilities:

**Computing** $P(s)$:
If $P(\tau)$ is the probability of a tree $\tau$, the probability of a sentence $s$ is the sum of the probabilities of all its parse trees:

$$P(s) = \sum_{\tau: \text{yield}(\tau) = s} P(\tau)$$

**How do we know that** $P(L) = \sum_{\tau} P(\tau) = 1$?
If we have learned the PCFG from a corpus via MLE, this is guaranteed to be the case.
If we just set the probabilities by hand, we could run into trouble, as in the following example:

$$S \rightarrow S S \ (0.9) \quad S \rightarrow w \ (0.1)$$
PCFG parsing (decoding): Probabilistic CKY
Probabilistic CKY: Viterbi

Like standard CKY, but with probabilities. Finding the most likely tree is similar to Viterbi for HMMs:

**Initialization:** every chart entry that corresponds to a **terminal** (entries $X$ in $\text{cell}[i][i]$) has a Viterbi probability $P_{VIT}(X_{[i][i]}) = 1$

**Recurrence:** For every entry that corresponds to a **non-terminal** $X$ in $\text{cell}[i][j]$, keep only the highest-scoring pair of backpointers to any pair of children ($Y$ in $\text{cell}[i][k]$ and $Z$ in $\text{cell}[k+1][j]$): $P_{VIT}(X_{[i][j]}) = \arg\max_{Y,Z,k} P_{VIT}(Y_{[i][k]}) \times P_{VIT}(Z_{[k+1][j]}) \times P(X \rightarrow YZ \mid X)$

**Final step:** Return the Viterbi parse for the start symbol $S$ in the top $\text{cell}[1][n]$. 
# Probabilistic CKY

## 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</td>
<td>S</td>
<td>S</td>
<td>S</td>
<td>0.8<em>0.2</em>0.4</td>
</tr>
<tr>
<td></td>
<td>V</td>
<td>VP</td>
<td>VP</td>
<td>VP</td>
<td>max(0.008<em>0.2, 0.06</em>0.2*0.2)</td>
</tr>
<tr>
<td>N</td>
<td>NP</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>0.2<em>0.2</em>0.2</td>
</tr>
<tr>
<td>P</td>
<td>PP</td>
<td></td>
<td></td>
<td></td>
<td>1*0.2</td>
</tr>
</tbody>
</table>

<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>S</td>
<td>NP</td>
<td>VP</td>
<td>VP</td>
<td>VP</td>
<td>0.4<em>0.3</em>0.2</td>
</tr>
<tr>
<td>NP</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>0.2<em>0.2</em>0.2</td>
</tr>
<tr>
<td>P</td>
<td>PP</td>
<td></td>
<td></td>
<td></td>
<td>1*0.2</td>
</tr>
</tbody>
</table>

<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>S</td>
<td>NP</td>
<td>VP</td>
<td>VP</td>
<td>VP</td>
<td>max(0.008<em>0.2, 0.06</em>0.2*0.2)</td>
</tr>
<tr>
<td>NP</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>0.2<em>0.2</em>0.2</td>
</tr>
<tr>
<td>P</td>
<td>PP</td>
<td></td>
<td></td>
<td></td>
<td>1*0.2</td>
</tr>
</tbody>
</table>

<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>S</td>
<td>NP</td>
<td>VP</td>
<td>VP</td>
<td>VP</td>
<td>max(0.008<em>0.2, 0.06</em>0.2*0.2)</td>
</tr>
<tr>
<td>NP</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>0.2<em>0.2</em>0.2</td>
</tr>
<tr>
<td>P</td>
<td>PP</td>
<td></td>
<td></td>
<td></td>
<td>1*0.2</td>
</tr>
</tbody>
</table>

- 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
How well can a PCFG model the distribution of trees?

PCFGs make **independence assumptions**: Only the label of a node determines what children it has.

Factors that influence these assumptions:

**Shape** of the trees:
A corpus with flat trees (i.e. few nodes/sentence) results in a model with few independence assumptions.

**Labeling** of the trees:
A corpus with many node labels (nonterminals) results in a model with few independence assumptions.
Example 1: flat trees

I eat sushi with tuna

I eat sushi with chopsticks

What sentences would a PCFG estimated from this corpus generate?
Example 2: deep trees, few labels

What sentences would a PCFG estimated from this corpus generate?
Example 3: deep trees, many labels

What sentences would a PCFG estimated from this corpus generate?
Aside: Bias/Variance tradeoff

A probability model has **low bias** if it makes few independence assumptions.
⇒ It can capture the structures in the training data.

This typically leads to a more fine-grained partitioning of the training data.

Hence, fewer data points are available to estimate the model parameters.

This **increases the variance** of the model.
⇒ This yields a poor estimate of the distribution.
Parser evaluation
Precision and recall

Precision and recall were originally developed as evaluation metrics for information retrieval:

- **Precision**: What percentage of retrieved documents are relevant to the query?
- **Recall**: What percentage of relevant documents were retrieved?

In NLP, they are often used in addition to accuracy:

- **Precision**: What percentage of items that were assigned label X do actually have label X in the test data?
- **Recall**: What percentage of items that have label X in the test data were assigned label X by the system?

Particularly useful when there are more than two labels.
True vs. false positives, false negatives

- True positives: Items that were labeled X by the system, and should be labeled X.
- False positives: Items that were labeled X by the system, but should not be labeled X.
- False negatives: Items that were not labeled X by the system, but should be labeled X,

\[\text{False Positives (FP)} = \text{Items labeled X by the system} = \text{TP} + \text{FP}\n\]

\[\text{False Negatives (FN)} = \text{Items labeled X in the gold standard (‘truth’)} = \text{TP} + \text{FN}\n\]
Precision, recall, f-measure

Items labeled X in the gold standard ('truth')
= TP + FN

Items labeled X by the system
= TP + FP

False Negatives (FN)
True Positives (TP)
False Positives (FP)

Precision: \( P = \frac{TP}{TP + FP} \)
Recall: \( R = \frac{TP}{TP + FN} \)
F-measure: harmonic mean of precision and recall
\( F = \frac{2 \cdot P \cdot R}{P + R} \)
False Negatives (FN)  True Positives (TP)  False Positives (FP)

Items labeled X in the gold standard ('truth') = TP + FN
Items labeled X by the system = TP + FP

Precision: \( P = \frac{TP}{TP + FP} \)
Recall: \( R = \frac{TP}{TP + FN} \)
F-measure: harmonic mean of precision and recall
\[ F = \frac{2 \cdot P \cdot R}{P + R} \]
Evalb ("Parseval")

Measures recovery of phrase-structure trees.

**Unlabeled**: span of nodes has to be right

**Labeled**: span and label (NP, PP,...) has to be right.

Two aspects of evaluation

**Precision**: How many of the predicted nodes are correct?

**Recall**: How many of the correct nodes were predicted?

*Usually combined into one metric (F-measure):*

\[
P = \frac{\text{#correctly predicted nodes}}{\text{#predicted nodes}}
\]

\[
R = \frac{\text{#correctly predicted nodes}}{\text{#correct nodes}}
\]

\[
F = \frac{2PR}{P + R}
\]
parseval in practice

Gold standard

Parser output

eat sushi with tuna: Precision: 4/5 Recall: 4/5

eat sushi with chopsticks: Precision: 4/5 Recall: 4/5