The path so far

• Originally, we treated language as a *sequence of words*
  → n-gram language models

• Then, we introduced the notion of *syntactic properties of words*
  → part-of-speech tags

• Now, we look at *syntactic relations* between words
  → syntax trees
A simple sentence

I like the interesting lecture

Part-of-speech tags

I like the interesting lecture

PRO VB DET JJ NN
Syntactic relations

I like the interesting lecture
PRO VB DET JJ NN

- The adjective *interesting* gives more information about the noun *lecture*
- The determiner *the* says something about the noun *lecture*
- The noun *lecture* is the object of the verb *like*, specifying *what* is being liked
- The pronoun *I* is the subject of the verb *like*, specifying *who* is doing the liking

Dependency structure

I like the interesting lecture
PRO VB DET JJ NN
↓ ↓ ↓ ↓
like lecture lecture like

This can also be visualized as a dependency tree:
Dependency structure (2)

The dependencies may also be labeled with the type of dependency

\[
\begin{array}{cccccc}
I & \text{like} & \text{the} & \text{interesting} & \text{lecture} \\
\text{PRO} & \text{VB} & \text{DET} & \text{JJ} & \text{NN} \\
\downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
\text{subject} & \text{adjunct} & \text{adjunct} & \text{object} \\
\downarrow & \downarrow & \downarrow & \downarrow \\
\text{like} & \text{lecture} & \text{lecture} & \text{like} \\
\end{array}
\]

Phrase-structure tree

A popular grammar formalism is phrase structure grammar

Internal nodes combine leaf nodes into phrases, such as noun phrases (NP)
Building phrase-structure trees

- Our task for this week: parsing
  - given: an input sentence with part-of-speech tags
  - wanted: the right syntax tree for it

- Formalism: context-free grammars
  - non-terminal nodes such as NP, S appear inside the tree
  - terminal nodes such as like, lecture appear at the leafs of the tree
  - rules such as NP → DET JJ NN

Applying the rules

<table>
<thead>
<tr>
<th>Input</th>
<th>Rule</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>S → NP VP</td>
<td>NP VP</td>
</tr>
<tr>
<td>NP VP</td>
<td>NP → PRO</td>
<td>PRO VP</td>
</tr>
<tr>
<td>PRO VP</td>
<td>PRO → I</td>
<td>I VP</td>
</tr>
<tr>
<td>I VP</td>
<td>VP → VP NP</td>
<td>I VP NP</td>
</tr>
<tr>
<td>I VP NP</td>
<td>VP → VB</td>
<td>I VB</td>
</tr>
<tr>
<td>I VB NP</td>
<td>VB → like</td>
<td>I like NP</td>
</tr>
<tr>
<td>I like  NP</td>
<td>NP → DET JJ NN</td>
<td>I like DET JJ NN</td>
</tr>
<tr>
<td>I like DET JJ NN</td>
<td>DET → the</td>
<td>I like the JJ NN</td>
</tr>
<tr>
<td>I like the JJ NN</td>
<td>JJ → interesting</td>
<td>I like the interesting NN</td>
</tr>
<tr>
<td>I like the interesting NN</td>
<td>NN → lecture</td>
<td>I like the interesting lecture</td>
</tr>
</tbody>
</table>
Recursion

Rules can be applied recursively, for example the rule $VP \rightarrow NP \ VP$

Context-free grammars in context

- **Chomsky hierarchy** of formal languages (terminals in caps, non-terminal lowercase)
  - **regular**: only rules of the form $A \rightarrow a$, $A \rightarrow B$, $A \rightarrow Ba$ (or $A \rightarrow aB$)
    Cannot generate languages such as $a^n b^n$
  - **context-free**: left-hand side of rule has to be single non-terminal, anything goes on right hand-side. Cannot generate $a^n b^n c^n$
  - **context-sensitive**: rules can be restricted to a particular context, e.g. $\alpha A \beta \rightarrow \alpha a B c \beta$, where $\alpha$ and $\beta$ are strings of terminal and non-terminals

- Moving up the hierarchy, languages are more expressive and parsing becomes computationally more expensive

- Is natural language context-free?
Why is parsing hard?

**Prepositional phrase attachment:** Who has the *telescope*?

Why is parsing hard?

**Scope:** Is *Jim* also from *Hoboken*?
CYK Parsing

• We have input sentence:

   \[ I \text{ like the interesting lecture} \]

• We have a set of context-free rules:

   \[
   \begin{align*}
   S & \rightarrow \text{NP VP, NP} \rightarrow \text{PRO, PRO} \rightarrow I, \text{VP} \rightarrow \text{VP NP, VP} \rightarrow \text{VB, VB} \rightarrow \text{like}, \\
   \text{NP} & \rightarrow \text{DET JJ NN, DET} \rightarrow \text{the, JJ} \rightarrow, \text{NN} \rightarrow \text{lecture}
   \end{align*}
   \]

• **Cocke-Younger-Kasami (CYK)** parsing
  
  – a **bottom-up** parsing algorithm
  – uses a **chart** to store intermediate result

---

Example

Initialize chart with the words

\[
\begin{array}{cccc}
I & \text{like} & \text{the} & \text{interesting lecture} \\
1 & 2 & 3 & 4 & 5
\end{array}
\]
Example (2)

Apply first terminal rule $\text{PRO} \rightarrow I$

Example (3)

... and so on ...

\[\text{PRO} \quad I \quad \text{like} \quad \text{the} \quad \text{interesting} \quad \text{lecture}\]
Example (4)

Try to apply a non-terminal rule to the first word
The only matching rule is $\text{NP} \rightarrow \text{PRO}$

Example (5)

Recurse: try to apply a non-terminal rule to the first word
No rule matches
Example (6)

Try to apply a non-terminal rule to the second word
The only matching rule is $VP \rightarrow VB$
No recursion possible, no additional rules match

Example (7)

Try to apply a non-terminal rule to the third word
No rule matches
Example (8)

Try to apply a non-terminal rule to the first two words.
The only matching rule is $S \rightarrow NP\ VP$.
No other rules match for spans of two words.

Example (9)

One rule matches for a span of three words: $NP \rightarrow DET\ JJ\ NN$.
Example (10)

One rule matches for a span of four words: \( VP \rightarrow VP \ NP \)

Example (11)

One rule matches for a span of five words: \( S \rightarrow NP \ VP \)
CYK algorithm for binarized grammars

- for all words \( w_i \): // terminal rules
  - for all rules \( A \rightarrow w_i \): add new chart entry \( A \) at span \([i, i]\)
- for length \= 1 to sentence length \( n \) // non-terminal rules
  - for start \= 1 to \( n - (\text{length} - 1) \)
    end \= start + length - 1
    - for middle \= start to end - 1: // binary rules
      for all non-terminals \( X \) in \([\text{start}, \text{middle}]\):
      for all non-terminals \( Y \) in \([\text{middle} + 1, \text{end}]\):
        for all rules \( A \rightarrow X \ Y \):
          add new chart entry \( A \) at position \([\text{start}, \text{end}]\)
  - for all non-terminals \( X \) in \([\text{start}, \text{end}]\): // unary rules
    for all rules \( A \rightarrow X \):
      add new chart entry \( A \) at position \([\text{start}, \text{end}]\)