Empirical Methods in Natural Language Processing
Lecture 9
Parsing (I): Context-free grammars and chart parsing

Philipp Koehn

4 February 2008

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

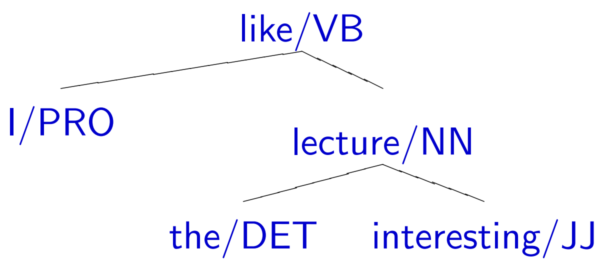
I	like	the	interesting	lecture
PRO	VB	DET	JJ	NN

- 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**:



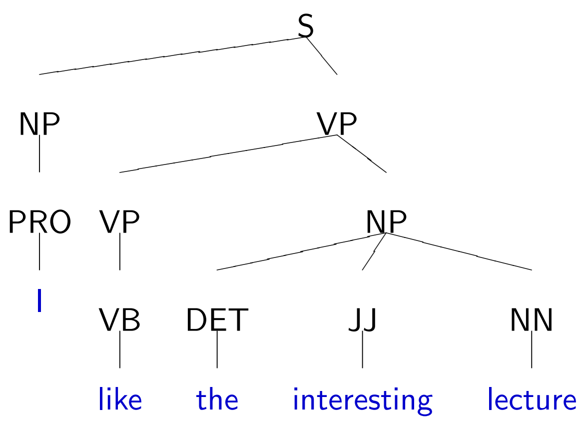
Dependency structure (2)

The dependencies may also be **labeled** with the type of dependency

I	like	the	interesting	lecture
PRO	VB	DET	JJ	NN
↓		↓	↓	↓
subject		adjunct	adjunct	object
↓		↓	↓	↓
like		lecture	lecture	like

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

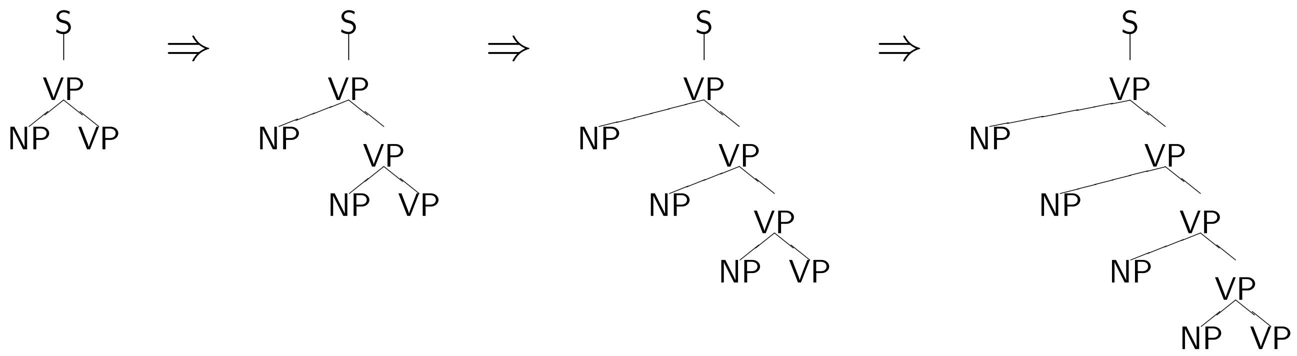
- Our task for this week: **parsing**
  - *given:* an input sentence with part-of-speech tags
  - *wanted:* the right syntax tree for it
- Formalism: **context-free grammars**
  - **non-terminal nodes** such as *NP*, *S* appear inside the tree
  - **terminal nodes** such as *like*, *lecture* appear at the leafs of the tree
  - **rules** such as *NP → DET JJ NN*

Applying the rules

Input	Rule	Output
S	S → NP VP	NP VP
NP VP	NP → PRO	PRO VP
PRO VP	PRO → *I*	*I* VP
I VP	VP → VP NP	*I* VP NP
I VP NP	VP → VB	*I* VB
I VB NP	VB → *like*	*I like* NP
I like NP	NP → DET JJ NN	*I like* DET JJ NN
I like DET JJ NN	DET → *the*	*I like the* JJ NN
I like the JJ NN	JJ → *interesting*	*I like the interesting* NN
I like the interesting NN	NN → *lecture*	*I like the interesting lecture*

Recursion

Rules can be applied **recursively**, for example the rule *VP → NP VP*



Context-free grammars in context

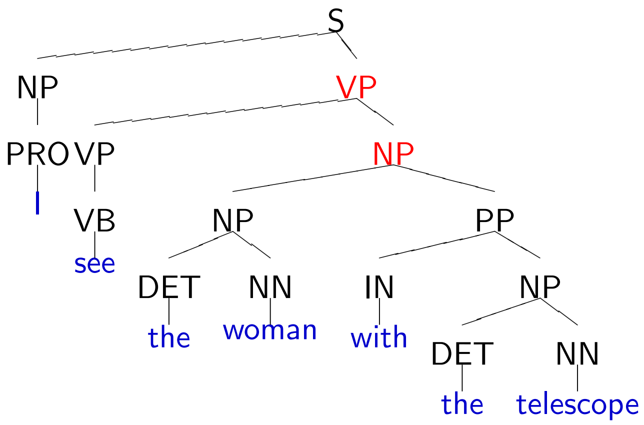
- **Chomsky hierarchy** of **formal languages**
  (terminals in caps, non-terminal lowercase)
  - **regular**: only rules of the form $A \rightarrow a, A \rightarrow B, A \rightarrow Ba$ (or $A \rightarrow aB$)
    Cannot generate languages such as $a^n b^n$
  - **context-free**: left-hand side of rule 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 α and β are strings of terminal and non-terminals

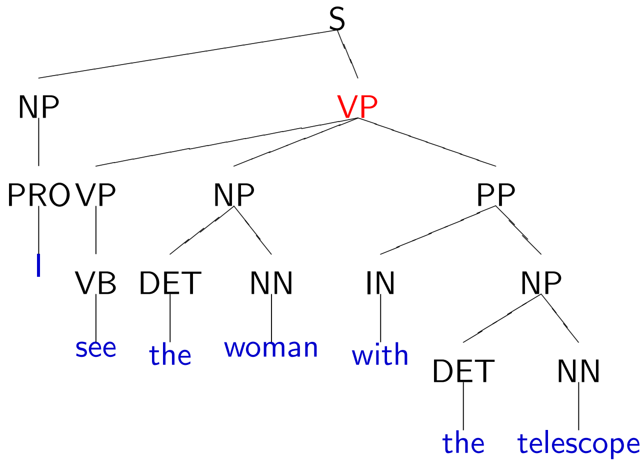
- Moving up the hierarchy, languages are more expressive and parsing becomes computationally more expensive

- Is natural language context-free?

Why is parsing hard?

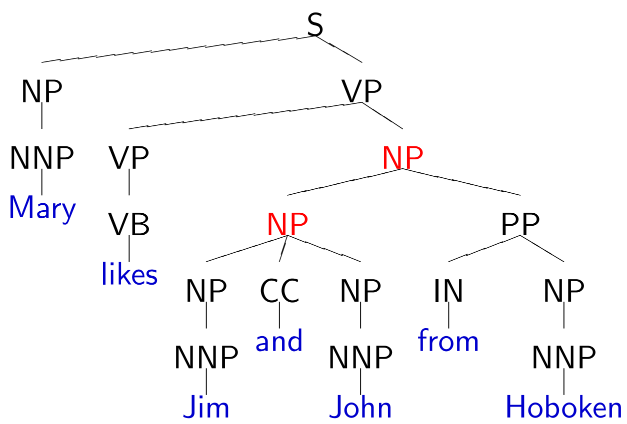
Prepositional phrase attachment: Who has the *telescope*?

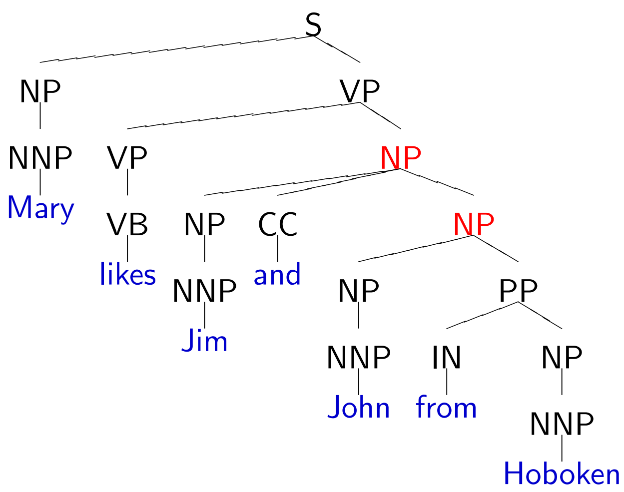




Why is parsing hard?

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





CYK Parsing

- We have input sentence:
  I like the interesting lecture

- We have a set of context-free rules:
  S → NP VP, NP → PRO, PRO → *I*, VP → VP NP, VP → VB, VB → *like*, NP → DET JJ NN, DET → *the*, JJ →, NN → *lecture*

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

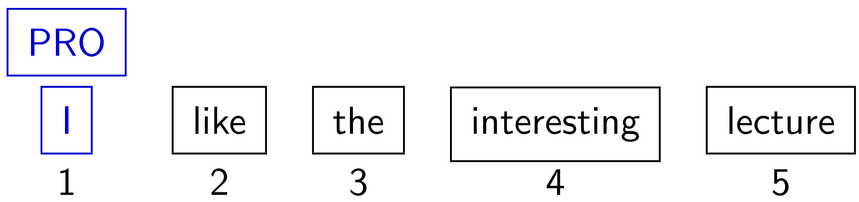
Example

Initialize chart with the words

I	like	the	interesting	lecture
1	2	3	4	5

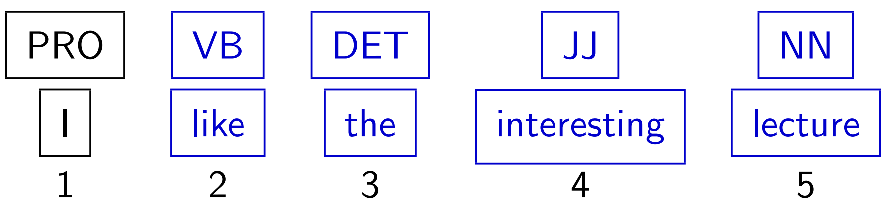
Example (2)

Apply first terminal rule PRO → *I*



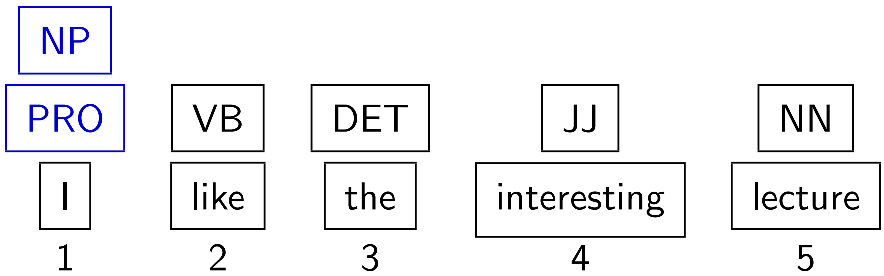
Example (3)

... and so on ...



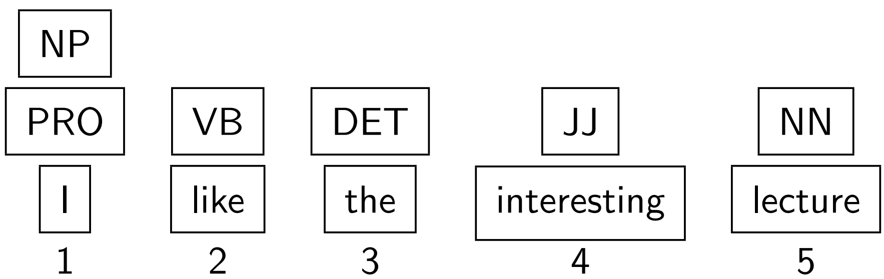
Example (4)

Try to apply a non-terminal rule to the first word
The only matching rule is NP → PRO



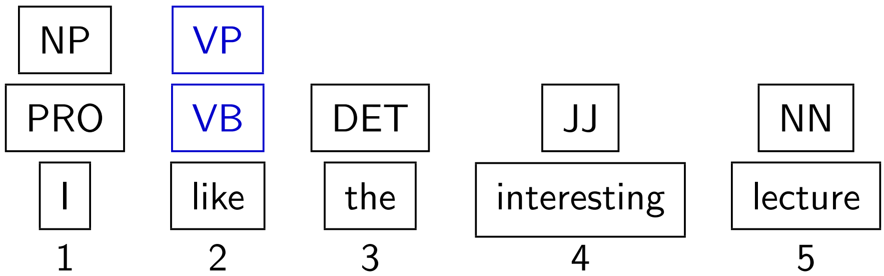
Example (5)

Recurse: try to apply a non-terminal rule to the first word
No rule matches



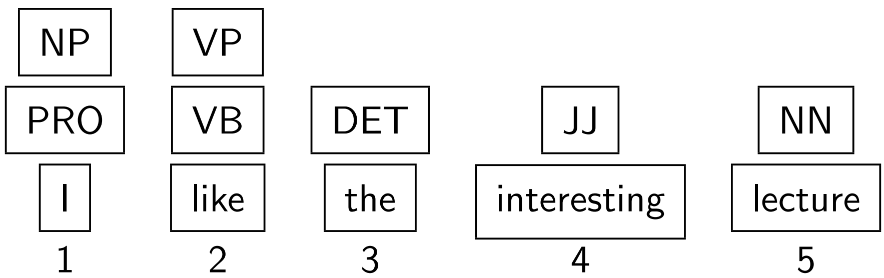
Example (6)

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



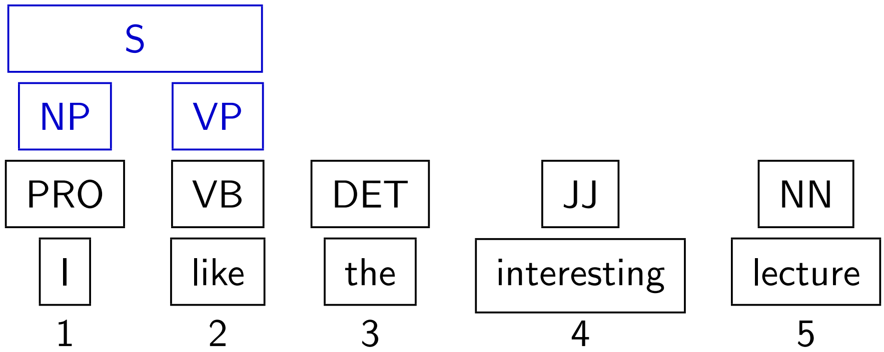
Example (7)

Try to apply a non-terminal rule to the third word
No rule matches



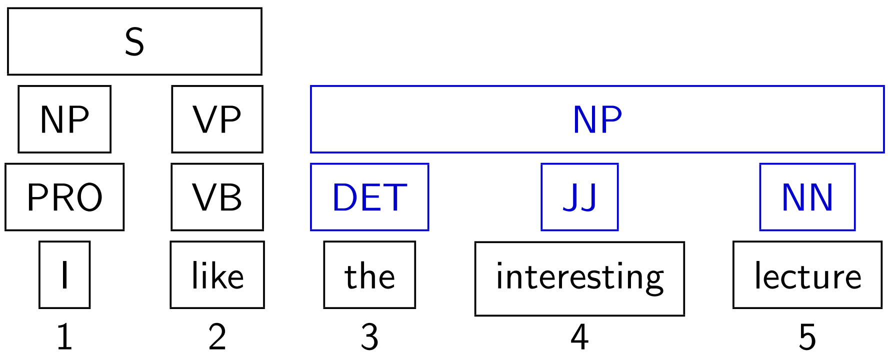
Example (8)

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



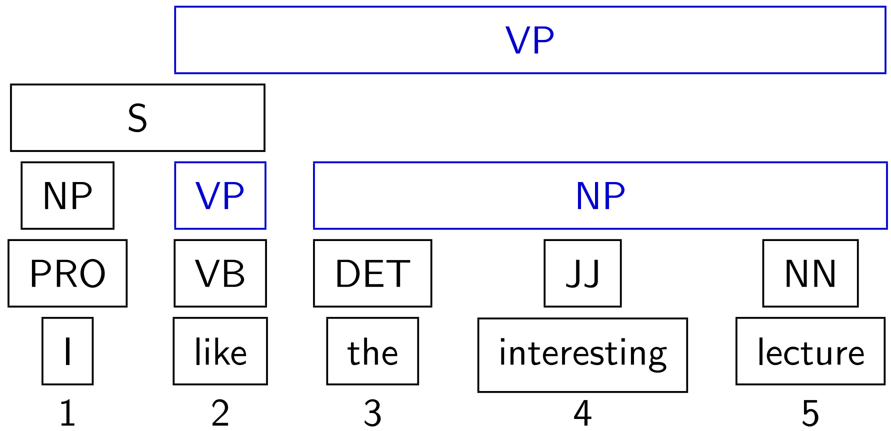
Example (9)

One rule matches for a span of three words: NP → DET JJ NN



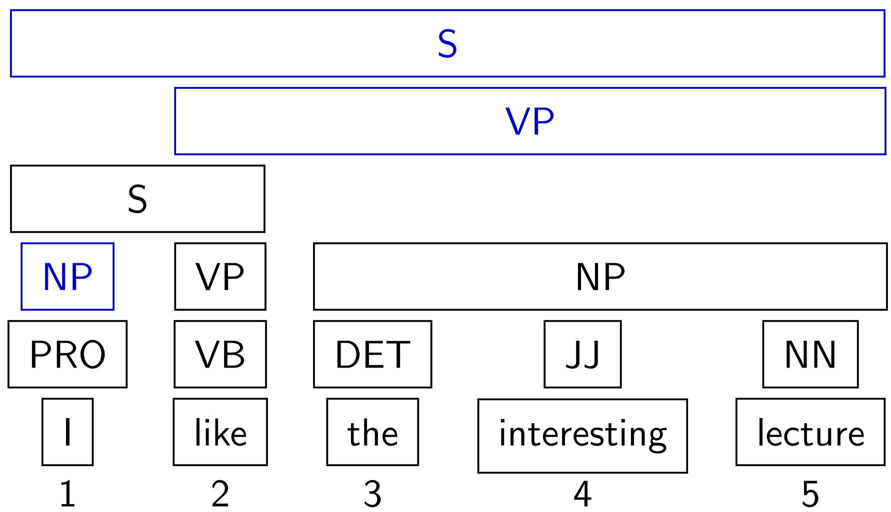
Example (10)

One rule matches for a span of four words: VP → VP NP



Example (11)

One rule matches for a span of five words: S → 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 $n - (length - 1)$
    $end = start + length - 1$
    - for $middle = start$ to $end - 1$: // binary rules
      for all non-terminals X in $[start, middle]$:
      for all non-terminals Y in $[middle + 1, end]$:
      for all rules $A \rightarrow X\ Y$:
      add new chart entry A at position $[start, end]$
    - for all non-terminals X in $[start, end]$: // unary rules
      for all rules $A \rightarrow X$:
      add new chart entry A at position $[start, end]$