Natural Language Processing CSCI 4152/6509 — Lecture 22 Natural Language Syntax

Instructors: Vlado Keselj
Time and date: 16:05 – 17:25, 23-Nov-2023
Location: Rowe 1011

Previous Lecture

- Neural language model, RNN, stacked and bidirectional RNN
- LSTM, self-attention, transformers

  Part IV: Parsing (Syntactic Processing)
- Prolog introduction
  - unification and backtracking
  - variables, lists; examples: factorial, member

Natural Language Syntax

- Syntax — NLP level of processing
  - Syntax = sentence structure; i.e., study of the phrase structure
- *sýntaxis* (Greek) — "setting out together, arrangement"
- Words are not randomly ordered — word order is important and non-trivial
- There are "free-order" languages (e.g., Latin, Russian), but they are not completely order free.
- Reading: Chapter 12 (JM book) or Ch.17 (JM on-line)

Phrase Structure and Dependency Structure

- Two ways of organizing sentence structure:
  - phrase structure
  - dependency structure
- Phrase structure
  - nested consecutive groupings of words
- Dependency structure
  - dependency relations between words
- The main NLP task at the syntax level: *parsing*
  - given a sentence, find the correct structure

Phrase Structure

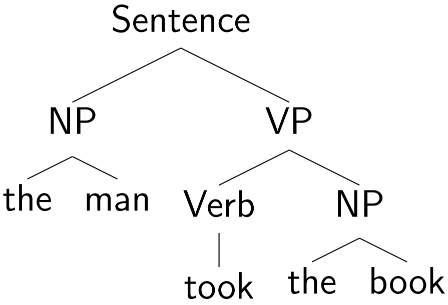
- Phrase Structure Grammars or Context-Free Grammars
- A hierarchical view of sentence structure:
  - words form phrases
  - phrases form clauses
  - clauses form sentences
- Parsing: given a sentence find the context-free parse tree; a.k.a. phrase structure parse tree

Example Sentence

the man took the book

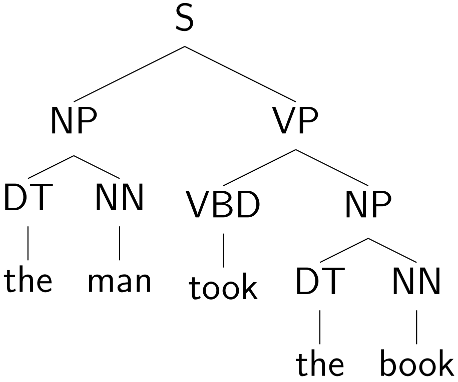
Phrase Structure Parse Tree Examples

- Phrase Structure parse trees are also called Context-Free parse trees
- This example is from the seminal Noam Chomsky's paper in 1956:



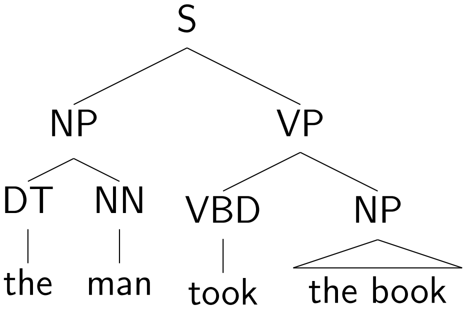
Parse Tree Examples (Penn treebank tagset)

- Using Penn treebank tagset:



Parse Tree Examples ('triangle' notation)

- Sometimes we simplify a parse tree by ignoring a part of the structure, as in:

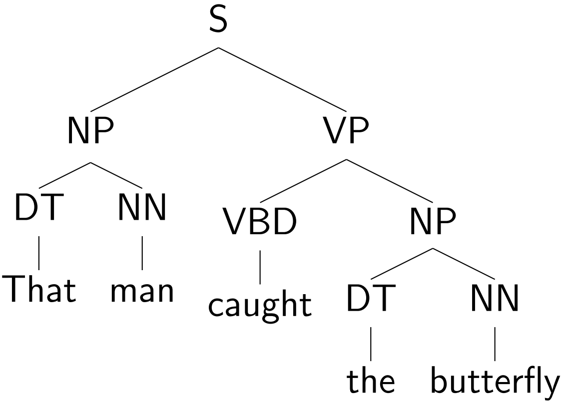


Parse Tree Example 2 ('butterfly' sentence)

That man caught the butterfly with a net

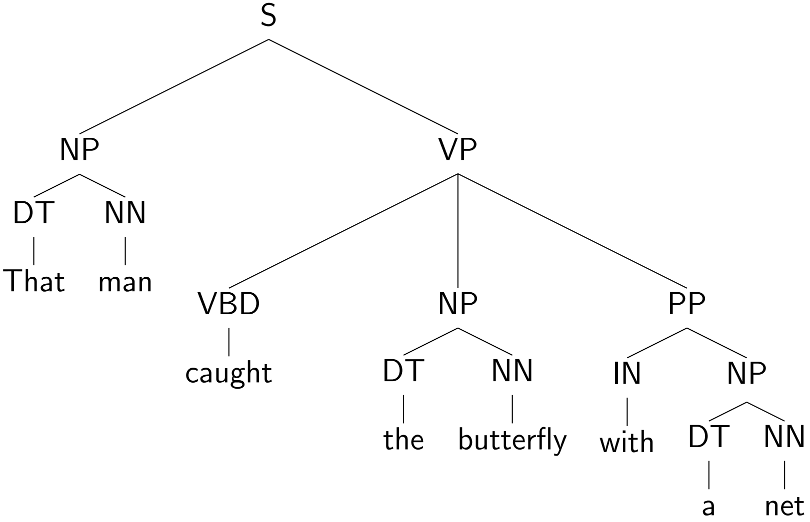
Parse Tree Example 2 (‘butterfly’)

- Another example:



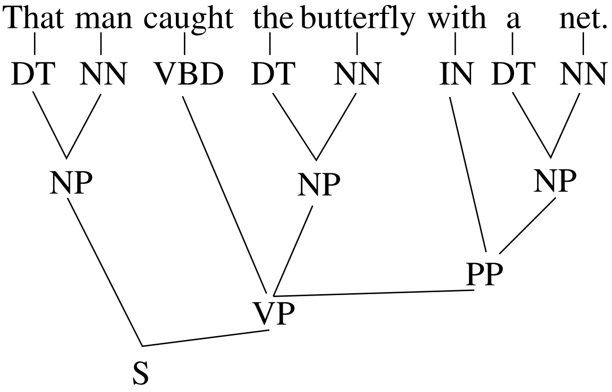
Parse Tree Example3 ('butterfly' extended)

- Extending the previous example:



Parse Tree Example (root bottom)

- Representing parse trees in the bottom-up direction:



Some Basic Notions in Context-Free Trees

- Context-free trees, also called phrase structure trees, parse trees, syntactic trees
- Node relations: root, leaf, parent (mother), child (daughter), sibling, ancestor, descendant, dominate
- Context-free grammar
- Consider for example the context-free grammar induced by the last parse tree shown

Context-Free Grammars (CFG) Review

CFG is a tuple (V, T, P, S), where

- V is a finite set of **variables** or **non-terminals**; e.g., $V = \{\mathit{S}, \mathit{NP}, \mathit{DT}, \mathit{NN}, \mathit{VP}, \mathit{VBD}, \mathit{PP}, \mathit{IN}\}$
- T is a finite set of **terminals**, words, or lexemes; e.g., $T = \{$That, man, caught, the, butterfly, with, a, net$\}$
- P is a set of **rules** or **productions** in the form $X \rightarrow \alpha$, where $X \in V$ and $\alpha \in (V \cup T)^*$; e.g., $P = \{\mathit{S} \rightarrow \mathit{NP}\ \mathit{VP},\ \ \mathit{NP} \rightarrow \mathit{DT}\ \mathit{NN},\ \ \mathit{DT} \rightarrow \mathrm{That},\ \ \mathit{NP} \rightarrow \epsilon\}$
- S is the **start symbol** $S \in V$

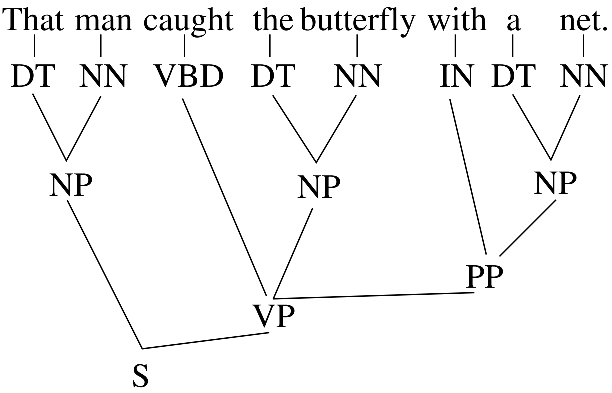
Some Notions about CFGs

- CFG, also known as Phrase-Structure Grammar (PSG)
- Equivalent to BNF (Backus-Naur form)
- Idea from Wundt (1900), formally defined by Chomsky (1956) and Backus (1959)
- Typical notation (V, T, P, S); also (N, Σ, R, S)

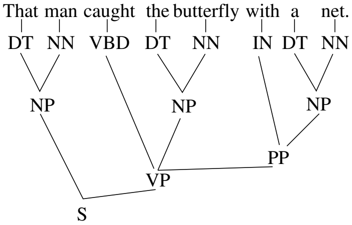
CFG Derivations

- Direct derivation, derivation
- Example of a direct derivation: $S \Rightarrow NP\ VP$
- Example of a derivation (beginning of):
  $S \Rightarrow NP\ VP \Rightarrow DT\ NN\ VP \Rightarrow \text{That}\ NN\ VP \Rightarrow \ldots$
- Left-most and right-most derivation

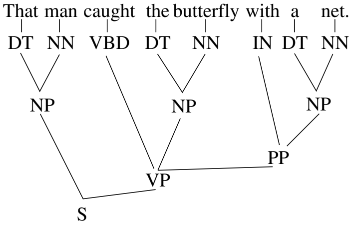
Parse Tree Example (revisited)



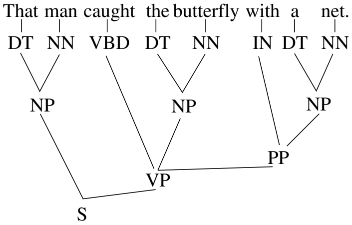
A Derivation Example (random)



Leftmost Derivation Example



Rightmost Derivation Example



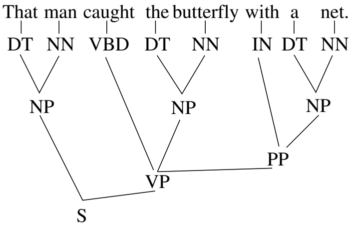
Leftmost Derivation Example

$$
\begin{aligned}
S &\Rightarrow NP\ VP \Rightarrow DT\ NN\ VP \Rightarrow \text{That}\ NN\ VP \Rightarrow \text{That man}\ VP \\
&\Rightarrow \text{That man}\ VBD\ NP\ PP \\
&\Rightarrow \text{That man caught}\ NP\ PP \\
&\Rightarrow \text{That man caught}\ DT\ NN\ PP \\
&\Rightarrow \text{That man caught the}\ NN\ PP \\
&\Rightarrow \text{That man caught the butterfly}\ PP \\
&\Rightarrow \text{That man caught the butterfly}\ IN\ NP \\
&\Rightarrow \text{That man caught the butterfly with}\ NP \\
&\Rightarrow \text{That man caught the butterfly with}\ DT\ NN \\
&\Rightarrow \text{That man caught the butterfly with a}\ NN \\
&\Rightarrow \text{That man caught the butterfly with a net}
\end{aligned}
$$

Some Notions about CFGs (continued)

- Language generated by a CFG
- Context-Free languages
- Parsing task
- Ambiguous sentences
- Ambiguous grammars
- Inherently ambiguous languages

Bracket Representation of a Parse Tree



Bracket Representation of a Parse Tree

```
(S (NP (DT That)
       (NN man))
   (VP (VBD caught)
       (NP (DT the)
           (NN butterfly))
       (PP (IN with)
           (NP (DT a)
               (NN net)
)  )   )   )
```

Some Notes on CFGs

- Left-hand side (lhs) and right-had side (rhs) of a production

$$\underbrace{S}_{lhs} \rightarrow \underbrace{NP\ VP}_{rhs}$$

- Empty rule (epsilon rule, epsilon production): $V \rightarrow \epsilon$
- Unit production: $A \rightarrow B$, where A and B are non-terminals
- Notational variations:
  - use of '|': $P \rightarrow N \mid A\,P$, instead of $P \rightarrow N,\ \ P \rightarrow A\ P$
  - BNF notation: $P ::= N \mid A\,P$
  - use of word 'opt': $NP\ ::=\ DT\ NN\ PP_{opt}$
  - or Kleene star: $NP\ ::=\ DT\ NN\ PP^{*}$

Using Prolog to Parse NL

Example: Consider a simple CFG to parse the following two sentences: "the dog runs" and "the dogs run"
The grammar is:

```
S  -> NP VP          N  -> dog
NP -> D N            N  -> dogs
D  -> the            VP -> run
                     VP -> runs
```

`How to parse: the dog runs`

Using Difference Lists: Idea

Consider rule: `S -> NP VP` and sentence [the,dog,runs]

Using Difference Lists to Parse CFG

The problem of parsing using this grammar can be expressed in the following way in Prolog:

```
s(S,R)  :- np(S,I), vp(I, R).
np(S,R) :- d(S,I), n(I,R).
d([the|R], R).
n([dog|R], R).
n([dogs|R], R).
vp([run|R], R).
vp([runs|R], R).
```

Parsing using Difference Lists

Save this in file `parse.prolog`. On Prolog prompt we type: `?- ['parse.prolog'].`

```
% parse.prolog compiled 0.00 sec, 1,888 bytes

Yes
?- s([the,dog,runs],[]).

Yes
?- s([runs,the,dog],[]).

No
```