Natural Language Processing

Grammars and Parsing
What is syntax?

- **Syntax** addresses the question how sentences are constructed in particular languages.
- A **grammar** is a set of rules that govern the composition of sentences.
- **Parsing** refers to the process of analyzing an utterance in terms of its syntactic structure.
Why should you care?

Syntactic information is important for many tasks:

• **Question answering**
  
  *What books did he like?*

• **Grammar checking**
  
  *He is friend of mine.*

• **Information extraction**
  
  *Oracle acquired Sun.*
Theoretical frameworks

- **Phrase structure grammar**
  Noam Chomsky (1928–)
  Immediate constituent analysis

- **Dependency grammar**
  Lucien Tesnière (1893–1954)
  Functional dependency relations

- **Categorial grammar**
  Kazimierz Ajdukiewicz (1890–1963)
  Logical derivations
Constituency

- A basic observation about syntactic structure is that groups of words can act as single units

Los Angeles, a high-class spot such as Mindy’s, three parties from Brooklyn, they.

- Such groups of words are called constituents

- Constituents tend to have similar internal structure, and behave similarly with respect to other units
Examples of constituents

- **noun phrases** (NP)
  
  she, the house, Robin Hood and his merry men,
  a high-class spot such as Mindy’s

- **verb phrases** (VP)
  
  blushed, loves Mary, was told to sit down
  and be quiet, lived happily ever after

- **prepositional phrases** (PP)
  
  on it, with the telescope, through the foggy dew,
  apart from everything I have said so far
Context-free grammar

- Simple yet powerful formalism to describe the syntactic structure of natural languages
- Developed in the mid-1950s by Noam Chomsky
- Allows one to specify rules that state how a constituent can be segmented into smaller and smaller constituents, up to the level of individual words
A context-free grammar (CFG) consists of

- a finite set of nonterminal symbols
- a finite set of terminal symbols
- a distinguished nonterminal symbol $S$
- a finite set of rules of the form $A \rightarrow \alpha$, where $A$ is a nonterminal and $\alpha$ is a possibly empty sequence of nonterminal and terminal symbols
A sample context-free grammar

<table>
<thead>
<tr>
<th>Grammar rule</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>S → NP VP</td>
<td>I + want a morning flight</td>
</tr>
<tr>
<td>NP → Pronoun</td>
<td>I</td>
</tr>
<tr>
<td>NP → Proper-Noun</td>
<td>Los Angeles</td>
</tr>
<tr>
<td>NP → Det Nominal</td>
<td>a flight</td>
</tr>
<tr>
<td>Nominal → Nominal Noun</td>
<td>morning flight</td>
</tr>
<tr>
<td>Nominal → Noun</td>
<td>flights</td>
</tr>
<tr>
<td>VP → Verb</td>
<td>do</td>
</tr>
<tr>
<td>VP → Verb NP</td>
<td>want + a flight</td>
</tr>
<tr>
<td>VP → Verb NP PP</td>
<td>leave + Boston + in the morning</td>
</tr>
<tr>
<td>VP → Verb PP</td>
<td>leaving + on Thursday</td>
</tr>
<tr>
<td>PP → Preposition NP</td>
<td>from + Los Angeles</td>
</tr>
</tbody>
</table>
Derivations

• A derivation is a sequence of rule applications that derive a terminal string \( w = w_1 \ldots w_n \) from \( S \)

• For example:

\[
\begin{align*}
S \\
NP \to VP \\
Pro \to VP \\
I \to VP \\
I \to Verb \to NP \\
I \to prefer \to NP \\
I \to prefer \to Det \to Nom \\
I \to prefer \to a \to Nom \\
I \to prefer \to a \to Nom \to Noun \\
I \to prefer \to a \to Noun \to Noun \\
I \to prefer \to a \to morning \to Noun \\
I \to prefer \to a \to morning \to flight
\end{align*}
\]
A sample phrase structure tree
A sample phrase structure tree

Context-free grammar

root (top)

leaves (bottom)
Formal languages

• The language defined by a CFG is the set of terminal strings derivable from S

• The context-free languages constitute a proper superset of the regular languages

• For example, the language $a^n b^n$ is context-free but not regular

• The syntax of natural languages is generally assumed to require at least context-free capacity
Problems with context-free grammars

• While context-free grammar can account for much of the syntactic structure of English, it is not a perfect solution.

• Two problems for context-free grammar:
  • agreement constraints
  • subcategorization constraints
The term **agreement** refers to constraints that hold between constituents that take part in a rule or a set of rules.

In English, subject and verb agree in number:

- [This flight] [leaves on Monday]
  - * [These flights] [leaves on Monday]

Our earlier rules are deficient in the sense that they do not capture this constraint. They **overgenerate**
Agreement

• One possible solution: fold agreement information into the rules

<table>
<thead>
<tr>
<th>Grammar rule</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>S → NP[sg] VP[sg]</td>
<td>this flight + leaves on Monday</td>
</tr>
<tr>
<td>VP[sg] → Verb[sg] PP</td>
<td>leaves + on Monday</td>
</tr>
<tr>
<td>NP[pl] → Det[pl] NP[pl]</td>
<td>these + flights</td>
</tr>
</tbody>
</table>

• While this approach is sound, it is practically infeasible: the grammars get too large
Subcategorization

- English VPs consist of a main verb along with zero or more constituents that we can call arguments.

<table>
<thead>
<tr>
<th>Grammar rule</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>VP → Verb</td>
<td>sleep</td>
</tr>
<tr>
<td>VP → Verb NP</td>
<td>want + a flight</td>
</tr>
<tr>
<td>VP → Verb NP PP</td>
<td>leave + Boston + in the morning</td>
</tr>
<tr>
<td>VP → Verb PP</td>
<td>leave + on Thursday</td>
</tr>
</tbody>
</table>

- But not all verbs are allowed to participate in all of those rules: we need to subcategorize them.
Modern grammars may have several hundreds of subcategories. Examples:

<table>
<thead>
<tr>
<th>Verb</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>sleep</td>
<td>John slept.</td>
</tr>
<tr>
<td>find + NP</td>
<td>Please find [a flight to New York].</td>
</tr>
<tr>
<td>give + NP + NP</td>
<td>Give [me] [a cheaper fare].</td>
</tr>
<tr>
<td>help + NP + PP</td>
<td>Can you help [me] [with a flight]?</td>
</tr>
<tr>
<td>prefer + TO-VP</td>
<td>I prefer [to leave earlier].</td>
</tr>
<tr>
<td>told + S</td>
<td>I was told [United has a flight].</td>
</tr>
</tbody>
</table>
Problems with context-free grammars

• CFGs appear to be adequate for describing a lot of the syntactic structure of English
  The situation for other languages is less clear …

• Problems include agreement and subcategorization
  These problems can be solved within the CFG formalism, but the resulting grammars are not practical.

• There are other, more elegant formalisms
  But these formalisms have more formal power than CFG, and are harder to parse.
Treebanks

- Treebanks are corpora where each sentence is annotated with a parse tree.
- Treebanks are generally created by:
  - parsing texts with an existing parser
  - having human annotators correct the result
- This requires detailed annotation guidelines for annotating different grammatical constructions.
The Penn Treebank

- **Penn Treebank** is a popular treebank for English
- Wall Street Journal section
- 1 million words from WSJ 1987–1989

(S
  (NP-SBJ
    (NP (NNP Pierre) (NNP Vinken))
    (, ,)
    (ADJP
      (NP (CD 61) (NNS years))
      (JJ old))
    (, ,))
  (VP (MD will)
    (VP (VB join)
      (NP (DT the) (NN board))
      (PP-CLR (IN as)
        (NP (DT a) (JJ nonexecutive) (NN director))
        (NP-TMP (NNP Nov.) (CD 29)))))
  (.. ..)))
Treebank grammars

• A treebank implicitly defines a grammar for the language covered in the treebank
• Simply take the set of rules needed to generate all the trees in the treebank
• Coverage of the language depends on the size of the treebank (but never complete)
Treebank grammars tend to be very flat because they avoid recursive rules (and hard distinctions).

The Penn Treebank has 4500 different rules for verb phrases.

For example:

\[
\begin{align*}
\text{VP} & \rightarrow \text{VBD PP} \\
\text{VP} & \rightarrow \text{VBD PP PP} \\
\text{VP} & \rightarrow \text{VBD PP PP PP} \\
\text{VP} & \rightarrow \text{VBD PP PP PP PP} \\
\end{align*}
\]
I booked a flight from LA.

• This sentence is ambiguous. In what way?
• What should happen if we parse the sentence?
I booked a flight from LA.
Ambiguity

S
   /\                      /
  NP  VP                   NP  PP
     |                       |
     Pro  Verb              from LA
      |                    /\            /
      I   booked      Det  Nom  flight
      |                  |  |
      a  Noun
Ambiguity

Combinatorial explosion

1600
1200
800
400
0

linear  cubic  exponential

1  2  3  4  5  6  7  8
Parsing is the automatic analysis of a sentence with respect to its syntactic structure. Given a CFG, this means deriving a phrase structure tree assigned to the sentence by the grammar. With ambiguous grammars, each sentence may have many valid parse trees.

- Should we retrieve all of them or just one?
- If the latter, how do we know which one?
Phrase structure trees

S
  / \  
NP  VP
  /   |
Pro Verb

I  prefer

Det Nom
  /   |
Nom Noun
  /   |
Noun flight
  /   |
morning
Basic concepts of parsing

- Two problems for grammar $G$ and string $w$:
  - **Recognition**: determine if $G$ accepts $w$
  - **Parsing**: retrieve (all or some) parse trees assigned to $w$ by $G$

- Two basic search strategies:
  - **Top-down**: start at the root of the tree
  - **Bottom-up**: start at the leaves
Top-down parsing

- Basic idea
  - Start at the root node, expand tree by matching the left-hand side of rules
  - Derive a tree whose leaves match the input
- Potential problems:
  - Uses rules that could never match the input
  - May loop on recursive rules: $VP \rightarrow VP PP$
Bottom-up parsing

• Basic idea:
  • Start with the leaves, build tree by matching the right-hand side of rules
  • Build a tree with S at the root

• Potential problems
  • Builds structures that could never be in a tree
  • May loop on epsilon productions: $NP \rightarrow \varepsilon$
Dealing with ambiguity

• The number of possible parse trees grows exponentially with sentence length

• A naive backtracking approach is too inefficient

• Key observation:
  • Alternative parse trees share substructures
  • We can use dynamic programming (again)

• Next time: the CKY algorithm
Summary

• Context-free grammar can be used to provide a formal account of the syntax (of English)
• There are phenomena that context-free grammar cannot handle well (agreement, subcategorization).
• Grammars can be written by hand or extracted from treebanks (which are annotated by hand)
• Parsing is the process of pairing sentences with parse trees under massive ambiguity