Phrase structure grammar and phrase structure parsing

Natural Language Processing (5LN710)

2012-12-06

Marco Kuhlmann

Department of Linguistics and Philology
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

• **Generative syntax**
  Noam Chomsky (1928–)

• **Dependency syntax**
  Lucien Tesnière (1893–1954)

• **Categorial syntax**
  Kazimierz Ajdukiewicz (1890–1963)
Theoretical frameworks

Chomsky

Tesnière

Ajdukiewicz
Overview

• Phrase structure grammars
• Phrase structure parsing
• Dependency parsing
# Schedule

<table>
<thead>
<tr>
<th>Week</th>
<th>Schedule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Week 49</td>
<td>(2 hrs taught, 6 hrs reading, 8 hrs assignment)</td>
</tr>
<tr>
<td>06/12</td>
<td>10–12 Lecture 1</td>
</tr>
<tr>
<td>Week 50</td>
<td>(4 hrs taught, 8 hrs reading, 28 hrs assignment)</td>
</tr>
<tr>
<td>11/12</td>
<td>10–12 Lab (detailed discussion of the assignment)</td>
</tr>
<tr>
<td>13/12</td>
<td>10–12 Lecture 2</td>
</tr>
<tr>
<td>Week 51</td>
<td>(8 hrs assignment)</td>
</tr>
<tr>
<td>17/12</td>
<td>Deadline for the assignment (early bird)</td>
</tr>
<tr>
<td>Week 01</td>
<td></td>
</tr>
<tr>
<td>06/01</td>
<td>Deadline for the assignment (regular)</td>
</tr>
<tr>
<td>Week 03</td>
<td></td>
</tr>
<tr>
<td>20/01</td>
<td>Deadline for the assignment (completions and belated hand-ins)</td>
</tr>
</tbody>
</table>
Reading


Phrase structure grammars
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
Overview

• Constituency

• Context-free grammar (CFG)

• Ambiguity

• Probabilistic context-free grammar (PCFG)
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 \( S \) (‘sentence’) 
• a finite set of rules of the form \( A \to \alpha \), 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 \rightarrow NP \ VP )</td>
<td>I + want a morning flight</td>
</tr>
<tr>
<td>( NP \rightarrow ) Pronoun</td>
<td>I</td>
</tr>
<tr>
<td>( NP \rightarrow ) Proper-Noun</td>
<td>Los Angeles</td>
</tr>
<tr>
<td>( NP \rightarrow ) Det Nominal</td>
<td>a flight</td>
</tr>
<tr>
<td>Nominal ( \rightarrow ) Nominal Noun</td>
<td>morning flight</td>
</tr>
<tr>
<td>Nominal ( \rightarrow ) Noun</td>
<td>flights</td>
</tr>
<tr>
<td>( VP \rightarrow ) Verb</td>
<td>do</td>
</tr>
<tr>
<td>( VP \rightarrow ) Verb NP</td>
<td>want + a flight</td>
</tr>
<tr>
<td>( VP \rightarrow ) Verb NP PP</td>
<td>leave + Boston + in the morning</td>
</tr>
<tr>
<td>( VP \rightarrow ) Verb PP</td>
<td>leaving + on Thursday</td>
</tr>
<tr>
<td>( PP \rightarrow ) Preposition NP</td>
<td>from + Los Angeles</td>
</tr>
</tbody>
</table>
A sample phrase structure tree
A sample phrase structure tree
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.
Overview

- Constituency
- Context-free grammar (CFG)
- Ambiguity
- Probabilistic context-free grammar (PCFG)
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.
Combinatorial explosion

Ambiguity

- Linear
- Cubic
- Exponential
Overview

• Constituency
• Context-free grammar (CFG)
• Ambiguity
• Probabilistic context-free grammar (PCFG)
Probabilistic context-free grammar

• The number of possible parse trees grows rapidly with the length of the input.
• But not all parse trees are equally useful.
  
  *Example:* I booked a flight from Los Angeles.
• In many applications, we want the ‘best’ parse tree, or the first few best trees.
• Special case: ‘best’ = ‘most probable’
A probabilistic context-free grammar (PCFG) is a context-free grammar where

• each rule $r$ has been assigned a probability $p(r)$ between 0 and 1

• the probabilities of rules with the same left-hand side sum up to 1
## Probabilistic context-free grammar

### A sample PCFG

<table>
<thead>
<tr>
<th>Rule</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S \rightarrow NP\ VP$</td>
<td>1</td>
</tr>
<tr>
<td>$NP \rightarrow Pronoun$</td>
<td>1/3</td>
</tr>
<tr>
<td>$NP \rightarrow Proper-Noun$</td>
<td>1/3</td>
</tr>
<tr>
<td>$NP \rightarrow Det Nominal$</td>
<td>1/3</td>
</tr>
<tr>
<td>$Nominal \rightarrow Nominal PP$</td>
<td>1/3</td>
</tr>
<tr>
<td>$Nominal \rightarrow Noun$</td>
<td>2/3</td>
</tr>
<tr>
<td>$VP \rightarrow Verb NP$</td>
<td>8/9</td>
</tr>
<tr>
<td>$VP \rightarrow Verb NP PP$</td>
<td>1/9</td>
</tr>
<tr>
<td>$PP \rightarrow Preposition NP$</td>
<td>1</td>
</tr>
</tbody>
</table>
The probability of a parse tree

The probability of a parse tree is defined as the product of the probabilities of the rules that have been used to build the parse tree.
The probability of a parse tree

Probability: $16/729$
The probability of a parse tree

The probability of a parse tree is 6/729.

Probabilistic context-free grammar

Montag, 10. Dezember 12
Summary

• Context-free grammar can be used to provide a formal account of the syntax (of English).
• There are phenomena that context-free grammar is not able to handle gracefully (agreement, subcategorization).
• Natural language is ambiguous.
• One way to address the problem of ambiguity is to add probabilities to the rules of a grammar.
Phrase structure parsing
Parsing is the automatic analysis of a sentence with respect to its syntactic structure.
I prefer a morning flight.
Conventions

• We are given a probabilistic context-free grammar $G$ and a sentence $w = w_1 \ldots w_n$ where the $w_i$ are individual words.

• We write $S$ for the start symbol of $G$.

• We write $|G|$ for the number of rules in $G$.

• We write $|w|$ for the length of $w$, that is we have $|w| = n$. 
The CKY algorithm

- We are interested in an efficient algorithm that can compute a most probable parse tree for the sentence $w$ given the grammar $G$.
- This task is also known as Viterbi parsing.
- By ‘efficient’, we mean that the algorithm should run in polynomial time with respect to the combined size $|G| + |w|$.
The CKY algorithm

Combinatorial explosion

- Linear
- Cubic
- Exponential
Dynamic programming

General idea:
Do not work with individual objects, but group objects into classes that share relevant properties, and manipulate only these classes.
The CKY algorithm

Signatures

covers all words between \textit{min} and \textit{max}

\([\text{min}, \text{max}, C]\)
The number of signatures

- The number of parse trees grows exponentially with $|w|$.
- The number of signatures only grows polynomially with $|w|$.
Fencepost positions

We view the sentence $w$ as a fence with $n$ holes, one hole for each word $w_i$, and we number the fenceposts from 0 till $n$. 
The problem, reformulated

• Original problem:
  Find a most probable parse tree \( t \) for the sentence \( w \) according to the grammar \( G \).

• Reformulated problem:
  Find a most probable parse tree \( t \) with signature \([0, |w|, S]\).

• The CKY algorithm can solve this problem in time \( O(|G||w|^3) \).
The CKY algorithm

Restrictions

- The CKY algorithm as we present it here can only handle rules of one of two forms: $C \rightarrow w_i$ (preterminal) and $C \rightarrow C_1 C_2$ (binary).
- This restriction is not a problem theoretically, but requires pre- and postprocessing.
Preterminal rules and binary rules

• **preterminal rules:**
  rules that rewrite a part-of-speech tag to a token, i.e. rules of the form $C \rightarrow w_i$.
  
  Pro $\rightarrow$ I, Verb $\rightarrow$ booked, Noun $\rightarrow$ flight

• **binary rules:**
  rules that rewrite a syntactic category to exactly two other categories: $C \rightarrow C_1 C_2$.
  
  S $\rightarrow$ NP VP, NP $\rightarrow$ Det Nom, VP $\rightarrow$ V NP