Last Class: Part-of-Speech Tagging

1. HMM Tagger

Today: Parsing

1. Grammars and parsing
2. Top-down and bottom-up parsing
3. Bottom-up chart parsing

Syntax

syntax: from the Greek *syntaxis*, meaning "setting out together or arrangement."
Refers to the way words are arranged together.

Why worry about syntax?

- The boy ate the frog.
- The frog was eaten by the boy.
- The frog that the boy ate died.
- The boy whom the frog was eaten by died.

Syntactic Analysis

Key ideas:

- **constituency**: groups of words may behave as a single unit or phrase
- **grammatical relations**: refer to the SUBJECT, OBJECT, INDIRECT OBJECT, etc.
- **subcategorization and dependencies**: refer to certain kinds of relations between words and phrases, e.g. *want* can be followed by an infinitive, but *find* cannot.

All can be modeled by various kinds of grammars that are based on context-free grammars.

Grammars and Parsing

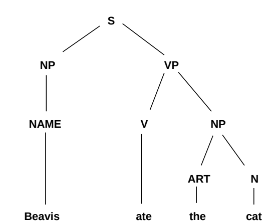
Need a **grammar:** a formal specification of the structures allowable in the language.

Need a **parser**: algorithm for assigning syntactic structure to an input sentence.

Sentence

Beavis ate the cat.

Parse Tree



CFG example

CFG's are also called phrase-structure grammars.
Equivalent to Backus-Naur Form (BNF).

1. S → NP VP
2. VP → V NP
3. NP → NAME
4. NP → ART N
5. NAME → Beavis
6. V → ate
7. ART → the
8. N → cat

- CFG's are *powerful* enough to describe most of the structure in natural languages.
- CFG's are *restricted* enough so that efficient parsers can be built.

Slide CS674–5

CFG's

A context free grammar consists of:

1. a set of non-terminal symbols N
2. a set of terminal symbols Σ (disjoint from N)
3. a set of productions, P, each of the form $A \rightarrow \alpha$, where A is a non-terminal and α is a string of symbols from the infinite set of strings $(\Sigma \cup N)^*$
4. a designated start symbol S

Slide CS674–6

Derivations

- If the rule $A \rightarrow \beta \in P$, and α and γ are strings in the set $(\Sigma \cup N)^*$, then we say that $\alpha A \gamma$ **directly derives** $\alpha\beta\gamma$, or $\alpha A \gamma \Rightarrow \alpha\beta\gamma$
- Let $\alpha_1, \alpha_2, \ldots, \alpha_m$ be strings in $(\Sigma \cup N)^*$, $m > 1$, such that

$$\alpha_1 \Rightarrow \alpha_2, \alpha_2 \Rightarrow \alpha_3, \ldots, \alpha_{m-1} \Rightarrow \alpha_m,$$

then we say that α_1 **derives** α_m or $\alpha_1 \stackrel{*}{\Rightarrow} \alpha_m$

Slide CS674–7

L_G

The language L_G generated by a grammar G is the set of strings composed of terminal symbols that can be derived from the designated start symbol S.

$L_G = \{w | w \in \Sigma^*, S \stackrel{*}{\Rightarrow} w\}$

Parsing: the problem of mapping from a string of words to its parse tree according to a grammar G.

Slide CS674–8

General Parsing Strategies

Grammar	Top-Down	Bottom-Up
1. S → NP VP	S → NP VP	→ NAME ate the cat
2. VP → V NP	→ NAME VP	→ NAME V the cat
3. NP → NAME	→ Beav VP	→ NAME V ART cat
4. NP → ART N	→ Beav V NP	→ NAME V ART N
5. NAME → Beavis	→ Beav ate NP	→ NP V ART N
6. V → ate	→ Beav ate ART N	→ NP V NP
7. ART → the	→ Beav ate the N	→ NP VP
8. N → cat	→ Beav ate the cat	→ S

Efficient Parsing

Have the first year Phd students in the computer science department take the Q-exam.

Have the first year Phd students in the computer science department taken the Q-exam?

Chart Parsers

chart: data structure that stores partial results of the parsing process in such a way that they can be reused. The chart for an n-word sentence consists of:

- $n + 1$ **vertices**
- a number of **edges** that connect vertices

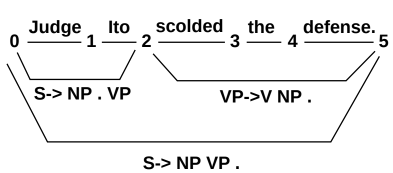


Chart Parsing: The General Idea

The process of parsing an n-word sentence consists of forming a chart with $n + 1$ vertices and adding edges to the chart one at a time.

- Goal: To produce a complete edge that spans from vertex 0 to n and is of category S.
- There is no backtracking.
- Everything that is put in the chart stays there.
- Chart contains all information needed to create parse tree.

Bottom-UP Chart Parsing Algorithm

Do until there is no input left:

1. If the agenda is empty, get next word from the input, look up word categories, add to agenda (as constituent spanning two postions).
2. Select a constituent from the agenda: constituent C from p_1 to p_2.
3. Insert C into the chart from position p_1 to p_2.
4. For each rule in the grammar of form $X \rightarrow C\ X_1 \ldots X_n$, add an active edge of form $X \rightarrow C \circ X_1 \ldots X_n$ from p_1 to p_2.

5. Extend existing edges that are looking for a C.
   (a) For any active edge of form $X \rightarrow X_1 \ldots \circ C X_n$ from p_0 to p_1, add a new active edge $X \rightarrow X_1 \ldots C \circ X_n$ from p_0 to p_2.
   (b) For any active edge of form $X \rightarrow X_1 \ldots X_n \circ C$ from p_0 to p_1, add a new (completed) constituent of type X from p_0 to p_2 to the agenda.

Grammar and Lexicon

Grammar:

1. S → NP VP
2. NP → ART N
3. NP → ART ADJ N
4. VP → V NP

Lexicon:

the: ART

old: ADJ, N

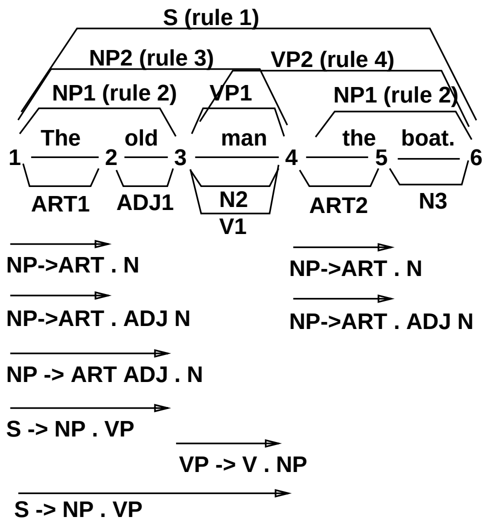
man: N, V

boat: N

Sentence: $_1$ The $_2$ old $_3$ man $_4$ the $_5$ boat $_6$

Example

[See .ppt slides]



Efficient Parsing

n = sentence length
Time complexity for bottom-up chart parser: $\bigcirc(n^3)$

1. Don't do twice what you can do once.