Human Communication 1
Lecture 6

Language models

- Two kinds of language models
  - Grammar-based models
  - Statistical models
- We still look at grammar-based models today

Grammar

- Symbols
- Rules
- Procedure of rule application

Formal grammar

- More technically, a formal grammar consists of
  - a finite set of terminal symbols
  - a finite set of nonterminal symbols
  - a set of rules (also called *production rules*)
    - with a left- and a right-handed side
    - each consisting of a word
  - a start symbol

Special symbols

- Formal grammars usually have two special symbols
  - S: the start symbol
  - ε: the empty string (sometimes: λ)

Terminology

- **Alphabet**: A set of (terminal and nonterminal) symbols
- **Word**: A string of symbols from an alphabet (what we also called 'sentence')
- **Grammar**: A set of rules defined on a alphabet
- **Language**: The set of words defined by a grammar

Formal definition

A grammar $G = \langle \Phi, \sum, R, S \rangle$ consists of

1. An alphabet Φ of **nonterminal** symbols,
2. An alphabet $\sum$ of **terminal** symbols,
3. A set $R \subseteq \Gamma^* \times \Gamma^*$ of **rules** (where $\Gamma = \Phi \cup \sum$),
4. A **start symbol** $S \in \Phi$.

Representing formal grammar

- Nonterminals are usually represented by upper-case letters {S, A, B}
- Terminals by lower case letters {a, b, c}
- The start symbols by S

Example

- Grammar
  - Alphabet: a, b
  - Start symbol: S
  - Rules:
    1. S → aSb
    2. S → ba
- What words are covered by this grammar?

Solution

- Apply rewrite rules until the result contains only symbols from the alphabet.
- We can rewrite
  - S to aSb by replacing S with aSb (rule 1);
  - aSb to aaSbb (rule 1)
  - aSb to abab (rule 2)
- For example: S → aSb → aaSbb → aababb
- The language of this grammar consists of the words $a^n bab^n$ (where n are 0 or more occurrences of the symbol, but the number of a's and b's is the same)

Chomsky hierarchy

- 4 types of grammars (Type-0 to Type-3)
  - Type-0: recursively enumerable
  - Type-1: context sensitive
  - Type-2: context free (CFG)
  - Type-3: regular

Type-3: regular

- LHS: 1 nonterminal
- RHS: 1 terminal and 0 or 1 nonterminals
- Pattern:
  $N \rightarrow t$
  $N_1 \rightarrow t\ N_2$ OR $N_1 \rightarrow N_2\ t$

Type-2: context free (CFG)

- LHS: 1 nonterminal
- RHS: terminals and nonterminals
- Pattern:
  $N \rightarrow \gamma$
  (where N is a nonterminal; γ is a string of terminals and nonterminals)

Type-1: context sensitive

- LHS: at least 1 nonterminal
- RHS: terminals and nonterminals
- Pattern:
  $\alpha N \beta \rightarrow \alpha \gamma \beta$
  (where N is a nonterminal; α, β, γ are strings of terminals and nonterminals; γ is not empty)

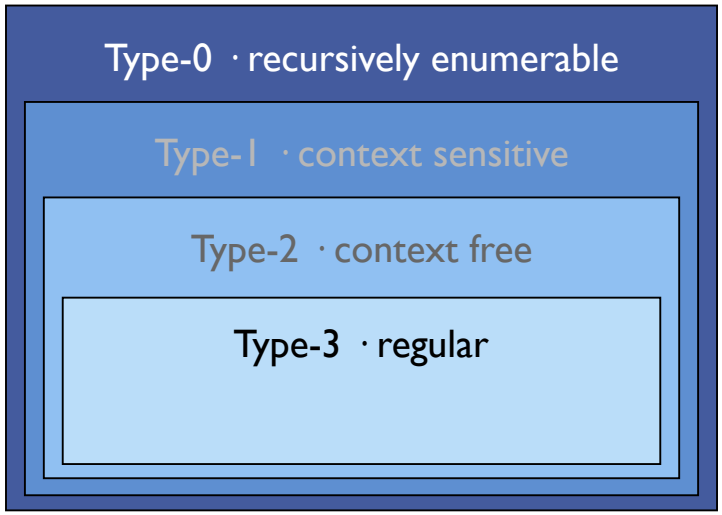
Type-1: context sensitive

- $\alpha N \beta \rightarrow \alpha \gamma \beta$
- α and β are the context in which N can be replaced by γ

Type-0: recursively enumerable

- All grammars and languages
- (Those that can be recognised by a Turing Machine.)
- Pattern:
  $\alpha \rightarrow \beta$
  (where α and β are any string of terminals and nonterminals, including the empty string)

Chomsky hierarchy



What type?

S → aS

S → abS

S → c

What type?

S → aS

S → abS

S → Sb

S → c

What type?

S → aS

S → abS

S → Bb

B → ε

S → cngw

What type?

$S \rightarrow aS$

$S \rightarrow aABb$

$A \rightarrow Aa$

$A \rightarrow a$

$B \rightarrow Bb$

$B \rightarrow b$

What type?

$S \rightarrow aS$

$S \rightarrow aABb$

$aA \rightarrow Aa$

$Aa \rightarrow a$

$B \rightarrow Bb$

$B \rightarrow b$

What type?

$S \rightarrow aS$

$S \rightarrow aABb$

$aAb \rightarrow Aa$

$Aa \rightarrow a$

$B \rightarrow Bb$

$Bb \rightarrow b$

Resource

- The Wikipedia page on Chomsky Hierarchies is a good starting point for formal grammars: http://en.wikipedia.org/wiki/Chomsky_grammar

Why the distinction?

- Why is this distinction relevant?
- Answer: different computational complexity
- This means: different amount of resources needed
- This means in particular: different execution times
- Generally: The simpler the grammar type, the faster the parsing and generation

Human grammar

- What type of grammar is human grammar?
- Probably in between context free and context sensitive: *mildly context-sensitive grammars* (MCSG)
- MCSGs
  - allow certain kinds of context dependencies
  - have low computational complexity (they have polynomial complexity)

Parsing algorithms

- In addition to complexity of the grammar, there are also different parsing algorithms
- Parallel instead of serial processing:
  If multiple rules apply, investigate all possibilities at once.
- Problem:
  A large number of possible analyses must be stored (probably exponentially many: $\textit{length-of-sentence}^{\textit{some-constant}}$).

What is the problem?

- The simple parsing models have difficulties
  - Serial model requires too much time
  - Parallel models requires too much storage
- Solution: Often a combination of serial and parallel parsing as well as top-down and bottom-up is used