Natural Language Processing

Lecture 13: The Chomsky Hierarchy
Learning Objectives

• Be able to recognize the different kinds of formal grammars and the languages that they generate
• Be able to state where in the Chomsky Hierarchy a particular grammar/language lies
• Be able to reason about which kind of language is relevant to a particular NLP task
Formal Grammars have Four Components

- Vocabulary of terminal symbols, $\Sigma$
- Set of nonterminal symbols, $N$
- Special start symbol $S \in N$
- Production rules

Restrictions on the production rules determine what kind of grammar you have
A formal grammar $G$ defines a formal language, usually denoted $L(G)$. 
Regular Grammars Have Two Types of Rules

• Regular grammars can be either right-linear or left-linear.

• For right-linear grammars (NT=non-terminal, T=terminal):
  – NT → T
  – NT → T NT

• For left-linear grammars:
  – NT → T
  – NT → NT T
For every right-linear grammar, there is a left-linear grammar that generates the same language. For every left-linear grammar, there is an equivalent right-linear grammar.
Example of a Regular Grammar

• aaaa...bbbb...
  o S $\rightarrow$ a AS
  o S $\rightarrow$ a BS
  o AS $\rightarrow$ a AS
  o AS $\rightarrow$ a BS
  o BS $\rightarrow$ b
  o BS $\rightarrow$ b BS

• To what regular expression does this correspond?
Regular Grammars are Equivalent to FSAs and Regular Expressions

• L(RG) can be recognized by FSM
  – Can be **determinized**
    • Each state has at most one arc per terminal
    • But might need $2^n$ new states
  – Can be **minimized**
    • Can find a minimal set of states/arcs that accept the same language

• Used in regular expressions
Definition of Context Free Grammars

Production rules of the type $X \rightarrow \alpha$

- $X$ is an NT
- $\alpha$ is any number of NTs and Ts, in any order
- Examples
  - $NT \rightarrow T$
  - $NT \rightarrow NT$
  - $NT \rightarrow T T$
  - $NT \rightarrow T NT T$
  - $NT \rightarrow T T T$
Context Free Grammar Activity

+, −, (, and ) are Ts; S is the start symbol (NT)

• $S \rightarrow S + S$
• $S \rightarrow S - S$
• $S \rightarrow ( S )$
• $S \rightarrow a, b$

Give some expressions that this grammar can generate. Give some strings of Ts that it cannot generate
Properties of Context Free Grammars

- $L(G)$ recognized by push-down automata
- Can be normalized into Chomsky Normal Form
  - $NT \rightarrow T$
  - $NT \rightarrow \varepsilon$  ($\varepsilon$ is the empty string)
  - $NT \rightarrow NT NT$
Most programming languages are context free languages
Definition of Context Sensitive Grammars

• Rewrite a string of symbols in a context (without changing the context)
  – $\alpha A \beta \rightarrow \alpha \gamma \beta$
    • $\alpha$ and $\beta$ are strings of NTs and Ts
    • $A$ is a NT
    • $\gamma$ is a non-empty string of NTs and Ts
Phonologists use Context Sensitive Rules

• n is rewritten as m before b
  – n → m / _ b
  – nb → mb

• z is rewritten as s between t and a word boundary
  – z → s / t _ #
  – tz# → ts#
Irony: Phonology and morphology can usually be modelled with finite-state machinery.
Properties of Context Sensitive Grammars

• $L(G)$ recognized by linear bounded automata

• Can be **harder to process**
  – Parsing is expensive
  – Spurious ambiguity
Defining Generalized Re-write Rules

- \([T \ NT]^* \rightarrow [T \ NT]^*\)
- Any number of symbols of either type on either side
- Equivalent to Turing Machines
- Can be intractable
Chomsky Hierarchy

<table>
<thead>
<tr>
<th>language class</th>
<th>automaton</th>
</tr>
</thead>
<tbody>
<tr>
<td>recursively enumerable</td>
<td>Turing machine</td>
</tr>
<tr>
<td>context-sensitive</td>
<td>linear bounded automaton</td>
</tr>
<tr>
<td>mildly context-sensitive</td>
<td>thread automaton</td>
</tr>
<tr>
<td>context-free</td>
<td>pushdown automaton</td>
</tr>
<tr>
<td>regular</td>
<td>finite-state automaton</td>
</tr>
</tbody>
</table>
Pumping Lemma for Regular Languages: Intuition

• An intuition (from Jurafsky & Martin, p. 533): “...if a regular language has any long strings (longer than the number of states in the automaton), there must be some sort of loop in the automaton for the language. We can use this fact by showing that if a language doesn’t have such a loop, then it can’t be regular.”
Pumping Lemma for Regular Languages

If \( L \) is an infinite regular language, then there are strings \( x, y, \) and \( z \) such that \( y \neq \varepsilon \) and \( xy^nz \in L \), for all \( n \geq 0 \).
Is English Regular?

• The cat likes tuna fish.
• The cat the dog chased likes tuna fish.
• The cat the dog the rat bit chased likes tuna fish.
• The cat the dog the rat the elephant admired bit chased likes tuna fish.
Is English Regular?

$L_1 = (\text{the cat|dog|mouse|...}^* (\text{chased|bit|ate|...})^* \text{ likes tuna fish}$

$L_2 = \text{English}$

$L_1 \cap L_2 = (\text{the cat|dog|mouse|...}^n (\text{chased|bit|ate|...})^{n-1} \text{ likes tuna fish}$
More Examples

• The cat likes tuna fish
• The cat the dog chased likes tuna fish
• The cat the dog the mouse scared chased likes tuna fish
• The cat the dog the mouse the elephant squashed scared chased likes tuna fish
• The cat the dog the mouse the elephant the flea bit squashed scared chased likes tuna fish
• The cat the dog the mouse the elephant the flea the virus infected bit squashed scared chased likes tuna fish
Chomsky Hierarchy

<table>
<thead>
<tr>
<th>language class</th>
<th>automaton</th>
</tr>
</thead>
<tbody>
<tr>
<td>recursively enumerable</td>
<td>Turing machine</td>
</tr>
<tr>
<td>context-sensitive</td>
<td>linear bounded automaton</td>
</tr>
<tr>
<td>mildly context-sensitive</td>
<td>thread automaton</td>
</tr>
<tr>
<td>context-free</td>
<td>pushdown automaton</td>
</tr>
<tr>
<td>regular</td>
<td>finite-state automaton</td>
</tr>
</tbody>
</table>
Swiss German

dative-Np  accusative-Nq  dative-taking-Vp  accusative-taking-Vq

- Jan säit das mer em Hans es huus hälfd aastriiche]
  Jan says that we Hans the house helped paint
  ‘Jan says that we helped Hans paint the house.’

- Jan säit das mer d’chind em Hans es huus haend wele laa hälfe aastriiche
  Jan says that we the children Hans the house have wanted to let help paint
  ‘Jan says that we have wanted to let the children help Hans paint the house’
Is Swiss German Context-Free?

$L_1 =$
Jan säit das mer (d’chind)$^*$ (em Hans)$^*$ es huus
haend wele (laa)$^*$ (hälfe)$^*$ aastriiche

$L_2 =$ Swiss German

$L_1 \cap L_2 =$
Jan säit das mer (d’chind)$^n$ (em Hans)$^m$ es huus
haend wele (laa)$^n$ (hälfe)$^m$ aastriiche
Swiss German Has Cross-Serial Dependencies

- Swiss German has sequences like \([A B C A' B' C']\) where \(A\) is licit just in case \(A'\) appears and vice versa (and likewise for \(B/B'\) and \(C/C'\))
- Cross-serial dependencies
Can Swiss German Be Recognized by a Context Free Grammar?
Context Sensitive English

“respectively”

Alice, Bob and Carol will have a beer, a wine and a coffee respectively

A B C ... a b c ...
Chomsky Hierarchy

• Natural Language is mildly context sensitive
  – This may not be true of English
    • English is largely context-free
    • There are some exceptional constructions, though
  – This is true of Swiss German, and some other languages
    • The frequency of context-sensitive constructions is relatively low
    • They tend to be bounded in depth by memory constraints
Are CFGs Sufficient?

For your application, CFGs might be adequate, even if you are modeling Swiss German

• Tractable
• Interpretable
• Easy to use for parsing
Are Regular Grammars Sufficient?

• For many applications, regular grammars are actually sufficient
  – Flat structure (bad!)
  – Efficient (good!)
Questions?