Today: Probabilistic Parsing

Goal: Find the most likely parse.

1. Parsing with PCFGs
2. Problems
3. Probabilistic lexicalized CFGs

Slide CS474–1

Probabilistic CFGs

Augments each rule in $P$ with a conditional probability:

$A \rightarrow \beta | p$

where $p$ is the probability that the non-terminal $A$ will be expanded to the sequence $\beta$. Often referred to as

$P(A \rightarrow \beta)$ or

$P(A \rightarrow \beta | A)$.

Slide CS474–3

CFG’s

A context free grammar consists of:

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

Slide CS474–2

Example

<table>
<thead>
<tr>
<th>Production</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S \rightarrow NP \ VP$</td>
<td>0.80</td>
</tr>
<tr>
<td>$S \rightarrow Aux \ NP \ VP$</td>
<td>0.15</td>
</tr>
<tr>
<td>$S \rightarrow VP$</td>
<td>0.05</td>
</tr>
<tr>
<td>$NP \rightarrow Det \ Nom$</td>
<td>0.20</td>
</tr>
<tr>
<td>$NP \rightarrow Proper-Noun$</td>
<td>0.35</td>
</tr>
<tr>
<td>$NP \rightarrow Nom$</td>
<td>0.05</td>
</tr>
<tr>
<td>$NP \rightarrow Pronoun$</td>
<td>0.40</td>
</tr>
<tr>
<td>$Nom \rightarrow Nom$</td>
<td>0.75</td>
</tr>
<tr>
<td>$Nom \rightarrow Proper-Noun Nom$</td>
<td>0.05</td>
</tr>
<tr>
<td>$Nom \rightarrow Pronoun$</td>
<td>0.55</td>
</tr>
<tr>
<td>$VP \rightarrow Verb$</td>
<td>0.40</td>
</tr>
<tr>
<td>$VP \rightarrow Verb \ NP$</td>
<td>0.40</td>
</tr>
<tr>
<td>$VP \rightarrow Verb \ NP \ NP$</td>
<td>0.05</td>
</tr>
</tbody>
</table>

Slide CS474–4
Why are PCFGs useful?

- Assigns a probability to each parse tree $T$
- Useful in disambiguation
  - Choose the most likely parse
  - Computing the probability of a parse
    If we make independence assumptions, $P(T) = \prod_{n \in T} p(r(n))$.
- Useful in language modeling tasks

Where does the grammar come from?

1. developed manually
2. from a treebank

Where do the probabilities come from?

1. from a treebank:
   
   $$P(\alpha \rightarrow \beta|\alpha) = \frac{\text{Count}(\alpha \rightarrow \beta)}{\text{Count}(\alpha)}$$

2. use EM (forward-backward algorithm, inside-outside algorithm)
**Parsing with PCFGs**

Produce the most likely parse for a given sentence:

\[ \hat{T}(S) = \text{argmax}_{T \in \tau(S)} P(T) \]

where \( \tau(S) \) is the set of possible parse trees for \( S \).

- Augment the Earley algorithm to compute the probability of each of its parses.
  When adding an entry \( E \) of category \( C \) to the chart using rule \( i \) with \( n \) subconstituents, \( E_1, \ldots, E_n \):
  \[ P(E) = P(\text{rule } i \mid C) \times P(E_1) \times \ldots \times P(E_n) \]
- probabilistic CKY (Cocke-Kasami-Younger) algorithm

---

**Problems with PCFGs**

Do not model *structural dependencies*.

Often the choice of how a non-terminal expands depends on the location of the node in the parse tree.

E.g. Strong tendency in English for the syntactic subject of a spoken sentence to be a pronoun.

- 91% of declarative sentences in the Switchboard corpus are pronouns (vs. lexical).
- In contrast, 34% of direct objects in Switchboard are pronouns.

---

**Problems with PCFGs**

Do not adequately model *lexical dependencies*.

*Moscow sent more than 100,000 soldiers into Afghanistan...*

PP can attach to either the NP or the VP:

\( \text{NP} \rightarrow \text{NP PP} \quad \text{or} \quad \text{VP} \rightarrow \text{V NP PP} \)

Attachment choice depends (in part) on the verb: *send* subcategorizes for a destination (e.g. expressed via a PP that begins with *into* or *to* or ...).
**Example**

Slide CS474–13

**Probabilistic lexicalized CFGs**

Slide CS474–14

**Incorporating lexical dependency information**

Slide CS474–15

**Incorporating lexical dependency information**

Slide CS474–16

Incorporates lexical dependency information by:

1. relating the heads of phrases to the heads of their constituents;
2. including syntactic subcategorization information.

Syntactic subcategorization dependencies:

```
( r(n) | n, h(n) ).
```

Example: probability of expanding VP as VP → VBD NP PP will be 

```
p (r | VP, dumped).
```