Today: Probabilistic Parsing

Goal: Find the most likely parse.

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

$$P(A \rightarrow \beta) \text{ or } P(A \rightarrow \beta | A).$$
Example

<table>
<thead>
<tr>
<th>Production</th>
<th>Probability</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S \rightarrow NP \ VP )</td>
<td>0.80</td>
<td>( Det \rightarrow that )</td>
</tr>
<tr>
<td>( S \rightarrow Aux \ NP \ VP )</td>
<td>0.15</td>
<td>( Noun \rightarrow book )</td>
</tr>
<tr>
<td>( S \rightarrow VP )</td>
<td>0.05</td>
<td>( Noun \rightarrow flights )</td>
</tr>
<tr>
<td>( NP \rightarrow Det \ Nom )</td>
<td>0.20</td>
<td>( Noun \rightarrow meal )</td>
</tr>
<tr>
<td>( NP \rightarrow Proper-Noun )</td>
<td>0.35</td>
<td>( Verb \rightarrow book )</td>
</tr>
<tr>
<td>( NP \rightarrow Nom )</td>
<td>0.05</td>
<td>( Verb \rightarrow include )</td>
</tr>
<tr>
<td>( NP \rightarrow Pronoun )</td>
<td>0.40</td>
<td>( Verb \rightarrow want )</td>
</tr>
<tr>
<td>( Nom \rightarrow Noun )</td>
<td>0.75</td>
<td>( Aux \rightarrow can )</td>
</tr>
<tr>
<td>( Nom \rightarrow Noun \ Nom )</td>
<td>0.20</td>
<td>( Aux \rightarrow does )</td>
</tr>
<tr>
<td>( Nom \rightarrow Proper-Noun \ Nom )</td>
<td>0.05</td>
<td>( Aux \rightarrow do )</td>
</tr>
<tr>
<td>( VP \rightarrow Verb )</td>
<td>0.55</td>
<td>( Proper-Noun \rightarrow TWA )</td>
</tr>
<tr>
<td>( VP \rightarrow Verb \ NP )</td>
<td>0.40</td>
<td>( Proper-Noun \rightarrow Denver )</td>
</tr>
<tr>
<td>( VP \rightarrow Verb \ NP \ NP )</td>
<td>0.05</td>
<td>( Pronoun \rightarrow you )</td>
</tr>
</tbody>
</table>
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
Example

(a) S → Aux NP VP .15
    Aux → Pro .40
    NP → V NP .05
    VP → V NP Nom .05
    NP → PNoun Nom .35
    Nom → Noun .75
    Aux → Can .40
    NP → Pro .40
    Pro → you .40
    Verb → book .30
    PNoun → TWA .40
    Noun → flights .50

(b) S → Aux NP VP .15
    Aux → Pro .40
    NP → V NP .05
    VP → V NP Nom .05
    NP → PNoun Nom .35
    Nom → Noun .75
    Aux → Can .40
    NP → Pro .40
    Pro → you .40
    Verb → book .30
    PNoun → TWA .40
    Noun → flights .50
Where does the grammar come from?

1. developed manually
2. from a treebank
Treebanks

- Corpus with sentence - parse tree (presumably the right one) pairs.
  - (S ('' ' ')
    (S-TPC-2
      (NP-SBJ-1 (PRP We) )
      (VP (MD would)
        (VP (VB have)
          (S
            (NP-SBJ (-NONE- *-1) )
            (VP (TO to)
              (VP (VB wait)
                (SBAR-TMP (IN until)
                  (S
                    (NP-SBJ (PRP we) )
                    (VP (VBP have)
                      (VP (VBN collected)
                        (PP-CLR (IN on)
                          (NP (DT those)(NNS assets)))))))))))
            (. .) ('' ' ')
            (NP-SBJ (PRP he) )
            (VP (VBD said)
              (S (-NONE- *T*-2) ))
            (. .) ))
      (. .) )}

- Penn TreeBank a widely used treebank.

  - Most well known is the Wall Street Journal section of the Penn TreeBank.
Treebanks

• How are they created?
  – Parse the collection with an automatic parser
  – Manually correct each parse as necessary.

• Requires detailed annotation guidelines that provide
  – a POS tagset
  – a grammar
  – instructions for how to deal with particular grammatical constructions.
Treebank Grammars

• Treebanks implicitly define a grammar.
• Simply take the local rules that make up the sub-trees in all the trees in the collection and you have a grammar.
• Not complete, but if you have decent size corpus, you’ll have a grammar with decent coverage.
Treebank Grammars

• Tend to be very flat due to the fact that they tend to avoid recursion.
  – To ease the annotators burden
• For example, the Penn Treebank has 4500 different rules for VPs. Among them...

```
VP → VBD PP
VP → VBD PP PP
VP → VBD PP PP PP
VP → VBD PP PP PP PP
```
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:

$$\hat{T}(S) = \arg\max_{T \in \tau(S)} P(T)$$

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, $E_1, \ldots, E_n$:
  
  $$P(E) = P(rule\ i \mid C) \times P(E_1) \times \ldots \times P(E_n)$$

- probabilistic CKY (Cocke-Kasami-Younger) algorithm

Slide CS474–9
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:
NP $\rightarrow$ NP PP or VP $\rightarrow$ 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 ...).
Probabilistic lexicalized CFGs

- Each non-terminal is associated with its head.
- Each PCFG rule needs to be augmented to identify one rhs constituent to be the head daughter.
- Headword for a node in the parse tree is set to the headword of its head daughter.
Example

workers dumped sacks into a bin
Noun Phrases

NP

PreDet

| all

Det

| the

NP

Noun

flights

Nom

from Denver

Nom

morning

Gerundive VP

leaving before 10

PP

to Tampa
Probabilistic lexicalized CFGs

View a lexicalized (P)CFG as a simple (P)CFG with a lot more rules.

\[
\begin{align*}
VP(\text{dumped}) & \rightarrow \text{VBD(}\text{dumped} \text{)} \text{ NP(}\text{sacks} \text{)} \text{ PP(}\text{into} \text{)} [3 \times 10^{-10}] \\
VP(\text{dumped}) & \rightarrow \text{VBD(}\text{dumped} \text{)} \text{ NP(}\text{cats} \text{)} \text{ PP(}\text{into} \text{)} [8 \times 10^{-10}] \\
VP(\text{dumped}) & \rightarrow \text{VBD(}\text{dumped} \text{)} \text{ NP(}\text{sacks} \text{)} \text{ PP(}\text{above} \text{)} [1 \times 10^{-12}] \\
\ldots
\end{align*}
\]

Problem?
Evaluation Measures and State of the Art

• labeled recall: \( \frac{\# \text{ correct constituents in candidate parse of } s}{\# \text{ correct constituents in treebank parse of } s} \)

• labeled precision: \( \frac{\# \text{ correct constituents in candidate parse of } s}{\text{total } \# \text{ of constituents in candidate parse of } s} \)

• crossing brackets: the number of crossed brackets

State of the art: 90% recall, 90% precision, 1% crossed bracketed constituents per sentence (WSJ treebank)