Probabilistic Context-Free Grammars

A probabilistic context-free grammar (PCFG) is defined as:

- **W**: a set of terminal symbols ($w^1 ... w^n$)
- **N**: a set of non-terminal symbols ($N^1 ... N^m$)
- **S**: a start (sentence) symbol ($S \in N$)
- **G**: a set of context-free grammar rules

Each rule is of the form $(N^i \rightarrow \zeta^i)$, where $\zeta^i$ is sequence of terminal and/or non-terminal symbols.

Each grammar rule $(N^i \rightarrow \zeta^i)$ has a probability, which is the probability of expanding $N^i$ with this rule as opposed to other rules for $N^i$.

The probabilities for all rules that expand the same non-terminal must sum to 1!

Example

A simple PCFG would look like this:

<table>
<thead>
<tr>
<th>Rule</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>S \rightarrow NP VP</td>
<td>.80</td>
</tr>
<tr>
<td>S \rightarrow VP</td>
<td>.20</td>
</tr>
<tr>
<td>NP \rightarrow noun</td>
<td>.45</td>
</tr>
<tr>
<td>NP \rightarrow noun NP</td>
<td>.15</td>
</tr>
<tr>
<td>NP \rightarrow noun PP</td>
<td>.40</td>
</tr>
<tr>
<td>VP \rightarrow verb</td>
<td>.30</td>
</tr>
<tr>
<td>VP \rightarrow verb NP</td>
<td>.35</td>
</tr>
<tr>
<td>VP \rightarrow verb NP NP</td>
<td>.10</td>
</tr>
<tr>
<td>VP \rightarrow verb PP</td>
<td>.20</td>
</tr>
<tr>
<td>VP \rightarrow verb NP PP</td>
<td>.05</td>
</tr>
<tr>
<td>PP \rightarrow prep NP</td>
<td>1.0</td>
</tr>
</tbody>
</table>

Note: these probabilities are not realistic, would be much smaller!

Treebank Grammars

- A Treebank is a text corpus with a manually produced parse tree for every sentence.
- We can produce a PCFG by reverse engineering a grammar from the parse trees in the Treebank.
- Probabilities can be assigned by counting how often each rule occurs.

For example, if the rule NP → ART ADJ NOUN has frequency 50, and the frequency of all NP rules is 500, then $P(NP \rightarrow ART ADJ NOUN) = .10$

Strengths and Weaknesses of Treebank Grammars

- Initially there was skepticism that Treebank grammars would work well because many syntactic constructions won’t appear in a small manually annotated text corpus.
- But they work surprisingly well because:
  - the most common rules do appear
  - missing rules are likely to be uncommon, and often similar rules can be used to produce a “close” parse
  - rules exist for imperfect syntactic constructions that people use frequently, making treebank parsers more robust for real-world data than linguistically “perfect” grammars.
What is the Probability of a Parse Tree?

• We say that \( N \) dominates constituents \( t_{ij} \) if all of \( t_{ij} \) are derived (directly or indirectly) from \( N \).

• We write \( N_{kj} \) for a non-terminal \( N \) that dominates the words starting at \( k \) and ending at \( l \).

• We then make two independence assumptions:
  1. \( P(N_{kj} \rightarrow \zeta^m) = P(N_{kj} \rightarrow \zeta^m | \text{everything outside } k \text{ through } l) \)
  2. \( P(N_{kj} \rightarrow \zeta^m) = P(N_{kj} \rightarrow \zeta^m | \text{everything above } N_{kj} \text{ in the tree}) \)

• With these assumptions, the probability of a parse tree is the product of the rules in the parse tree.

  -- Intuitively, the probability of a tree is the probability that all of its constituents dominate the words under its subtree.

Parsing Accuracy Metrics

• The accuracy of an automatically generated parse tree is computed by measuring recall and precision with respect to a human generated parse tree.

• A constituent (non-terminal) is correct if:
  1. it dominates exactly the correct span of words
  2. it is the correct type of constituent

• Part-of-speech tag constituents may be treated as terminal symbols when evaluating parsers if the POS tagging is considered to be a separate subtask.

Example Parse Tree

The probability of this parse tree is:

\[
P(S \rightarrow NP VP) * \\
P(NP \rightarrow D N) * \\
P(D \rightarrow the) * \\
P(N \rightarrow chef) * \\
P(VP \rightarrow V NP) * \\
P(V \rightarrow cooks) * \\
P(NP \rightarrow D N) * \\
P(D \rightarrow the) * \\
P(N \rightarrow soup)
\]

Example: The most troublesome report may be the August merchandise trade deficit due out tomorrow.

Correct parse:

\[
(S (NP the (ADJP most troublesome) report)
  (VP may
    (VP be
      (NP (NP the August merchandise trade deficit) due)
      (ADJP (ADVP out (NP tomorrow))))))
\]

System parse:

\[
(S (NP the (ADJP most troublesome) report)
  (VP may
    (VP be
      (ADJP (NP the August merchandise trade deficit) due)
      (PP out (NP tomorrow))))
\]
Green constituents match, Red constituents do not match.
Recall = 7/10 = 70%  Precision = 7/9 = 78%

**Correct parse (10 constituents):**
(S (NP the (ADJP most troublesome) report)
 (VP may
 (VP be
 (NP (NP the August merchandise trade deficit)
 (ADJ due (ADVP out) (NP tomorrow))))))

**System parse (9 constituents):**
(S (NP the (ADJP most troublesome) report)
 (VP may
 (VP be
 (ADJP (NP the August merchandise trade deficit) due)
 (PP out (NP tomorrow)))))

---

**Problems with Ordinary PCFGs**

- So far, our grammars have not considered the specific words in syntactic constituents beyond POS tags. These *unlexicalized* grammars can struggle to rank parses well.

**Example:**
1. Alice bought a large plant with a credit card.
2. Alice bought a large plant with yellow leaves.

- These parses are identical except for PP attachment. Attachments are highly dependent on specific words.
- Without considering the words, the more common type of attachment (e.g., NP or VP) will always be chosen!

---

**Lexicalized Grammars**

- A *lexicalized grammar* associates a *lexical head* with each non-terminal node, and sometimes its POS tag (*head tag*).
- For example:

```
S (cooks, V)
```

```
NP (chef, N)  VP (cooks, V)
```

```
D  N  V  NP (soup, N)
```

```
the  chef  cooks  the  soup
```

---

**Lexicalized PCFGs**

- A *lexicalized PCFG* can be viewed as having many copies of each rule, one for each possible head word & tag.
- Lexicalized parsers often perform substantially better than unlexicalized parsers because they can model associations between words in syntactic constructions.
- Lexicalized rules will be sparse, however, so special methods are needed to get good probability estimates and to control the search (e.g., pruning).
The Challenge of Parsing

In a 1997 article by Eugene Charniak called “Statistical Techniques for Natural Language Parsing”, he discussed the performance of statistical parsers on the Penn Wall Street Journal corpus:

The average sentence length is 23 words and punctuation. I have not measured how many parses there typically are for these sentences, and, of course, it would depend on the grammar. But I would guess that for the kinds of grammars we discuss in the following pages, a million parses per sentence would be conservative.

Constituency vs. Dependency Parsers

Traditional parse trees represent sentence structure with respect to a phrase structure grammar. Phrase structure parsers generate constituency-based parses.

Another grammar formalism uses dependency grammars, which capture syntactic relations between words.

Dependency Parsing

- A dependency parse representation is essentially a directed graph of grammatical relations between words.
- The parse is often decomposed into pairwise dependencies based on the edges in the graph.
- Relations are between a word (a governing head) and its dependents.
- A dependency parse can be generated directly, or produced as a transformation from a phrase structure parse.

Types of Dependency Relations

- Different parsers represent different dependency relations, just like different constituency-based parsers use different phrase structure grammars.
- The Stanford dependency parser represents about 50 grammatical relations.
- Each dependency is a binary relation between a governor (head) and its dependent.
Examples of Dependency Relations

**nsubj**: nominal subject (syntactic subject of clause)

“Clinton defeated Dole” \( \text{nsubj}(\text{defeated, Clinton}) \)

“The baby is cute” \( \text{nsubj}(\text{cute, baby}) \)

**nsubjpass**: passive nominal subject (syntactic subject of passive clause)

“Dole was defeated by Clinton” \( \text{nsubjpass}(\text{defeated, Dole}) \)

**agent**: passive verb complement with preposition “by”

“He was killed by police” \( \text{agent}(\text{killed, police}) \)

Examples of Dependency Relations

**dobj**: direct object of a VP

“She gave Rover a treat” \( \text{dobj}(\text{gave, treat}) \)

**iobj**: indirect object of a VP

“She gave Rover a treat” \( \text{iobj}(\text{gave, Rover}) \)

**pobj**: object of a preposition (head of NP following the preposition)

“Rover ate the cookies on the kitchen table” \( \text{pobj}(\text{on, table}) \)

**prep**: head of phrase that the preposition attaches to

“Rover ate the cookies on the kitchen table” \( \text{prep}(\text{on, cookies}) \)

Examples of Dependency Relations

**det**: determiner and head of its NP

“the genetically modified food” \( \text{det}(\text{food, the}) \)

**amod**: adjectival modifier

“the genetically modified food” \( \text{amod}(\text{modified, genetically}) \)

**advmod**: adverbial modifier

“the genetically modified food” \( \text{advmod}(\text{modified, genetically}) \)

**appos**: NP immediately to the right of another NP in an appositive structure

“Steve Ballmer, CEO” \( \text{appos}(\text{Ballmer, CEO}) \)

“Steve Ballmer (CEO)” \( \text{appos}(\text{Ballmer, CEO}) \)

**infmod**: infinitive that modifies an NP

“a plan to graduate” \( \text{infmod}(\text{plan, graduate}) \)

**xcomp**: infinitive that modifies a VP or ADJP

“He likes to swim” \( \text{xcomp}(\text{likes, swim}) \)

“He is ready to swim” \( \text{xcomp}(\text{ready, swim}) \)
Examples of Stanford dependency parser’s output:

![Diagram of dependency parse graph]

Stanford parser’s collapsed dependencies:

- Many syntactic analysis tools are freely available, including part-of-speech taggers, chunkers, constituency-based parsers, and dependency parsers.
- Some of the most well-known toolkits are:
  - OpenNLP
  - Stanford NLP group
  - LingPipe
  - NLTK
  - GATE