Announcements
Ambiguity
Programs must be written for people to read

Preface of *Structure and Interpretation of Computer Programs* by Harold Abelson and Gerald Sussman with Julie Sussman
Programs must be written for people to read

Preface of *Structure and Interpretation of Computer Programs* by Harold Abelson and Gerald Sussman with Julie Sussman
Syntactic Ambiguity in English

Programs must be written for people to read

Preface of *Structure and Interpretation of Computer Programs* by Harold Abelson and Gerald Sussman with Julie Sussman
Syntactic Ambiguity in English

**program** (noun)
- a series of coded software instructions

**program** (verb)
- provide a computer with coded instructions

**must** (verb)
- be obliged to

**must** (noun)
- dampness or mold

Programs must be written for people to read

Definitions from the New Oxford American Dictionary
Syntax Trees
Representing Syntactic Structure

A Tree represents a phrase:
- **tag**  -- What kind of phrase (e.g., *S*, *NP*, *VP*)
- **branches**  -- Sequence of Tree or Leaf components

A Leaf represents a single word:
- **tag**  -- What kind of word (e.g., *N*, *V*)
- **word**  -- The word

\[
\text{cows} = \text{Leaf}('N', 'cows') \\
\text{intimidate} = \text{Leaf}('V', 'intimidate') \\
\text{S, NP, VP} = 'S', 'NP', 'VP' \\
\text{Tree}(S, [\text{Tree}(NP, [\text{cows}]), \\
\text{Tree}(VP, [\text{intimidate}, \\
\text{Tree}(NP, [\text{cows}]))]))
\]
Grammars
Context-Free Grammar Rules

A grammar rule describes how a tag can be expanded as a sequence of tags or words.

A Sentence ...

... can be expanded as ...

... a Noun Phrase then a Verb Phrase.

<table>
<thead>
<tr>
<th>Grammar</th>
</tr>
</thead>
<tbody>
<tr>
<td>S → NP VP</td>
</tr>
<tr>
<td>NP → N</td>
</tr>
<tr>
<td>N → cows</td>
</tr>
<tr>
<td>VP → V NP</td>
</tr>
<tr>
<td>V → intimidate</td>
</tr>
</tbody>
</table>
Parsing
Exhaustive Parsing

Expand all tags recursively, but constrain words to match input
Exhaustive Parsing

Expand all tags recursively, but constrain words to match input

Constraint: A **Leaf** must match the input word
Exhaustive Parsing

Expand all tags recursively, but constrain words to match input

```
<table>
<thead>
<tr>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>NP</td>
</tr>
<tr>
<td>VP</td>
</tr>
</tbody>
</table>
```

```
0  buffalo  1  buffalo  2  buffalo  3  buffalo  4
```
Exhaustive Parsing

Expand all tags recursively, but constrain words to match input

(Demo)
Learning

(Demo)
Scoring a Tree Using Relative Frequencies

Not all syntactic structures are equally common

```
S → NP VP
NP → NN NNS
VP → VB NP
NP → NNS
```

Rule frequency per 100,000 tags

<table>
<thead>
<tr>
<th>Rule</th>
<th>Frequency</th>
<th>Token</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>S → NP VP</td>
<td>25372</td>
<td>teacher</td>
<td>5</td>
</tr>
<tr>
<td>NP → NN NNS</td>
<td>1335</td>
<td>strikes</td>
<td>25</td>
</tr>
<tr>
<td>VP → VB NP</td>
<td>6679</td>
<td>idle</td>
<td>26</td>
</tr>
<tr>
<td>NP → NNS</td>
<td>4282</td>
<td>kids</td>
<td>32</td>
</tr>
</tbody>
</table>

```
teacher strikes idle kids
```
Scoring a Tree Using Relative Frequencies

Not all syntactic structures are equally common

```
S → NP  VP
NP → NN
VP → VBZ  NP
NP → JJ  NNS
```

- teacher strikes idle kids

Rule frequency per 100,000 tags

<table>
<thead>
<tr>
<th>Rule</th>
<th>Frequency</th>
<th>Tag</th>
<th>Frequency</th>
<th>Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>S → NP VP</td>
<td>25372</td>
<td>teacher</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>NP → NN</td>
<td>1335 4358</td>
<td>strikes</td>
<td>25 19</td>
<td></td>
</tr>
<tr>
<td>VP → VBZ NP</td>
<td>6679 3160</td>
<td>idle</td>
<td>26 18</td>
<td></td>
</tr>
<tr>
<td>NP → JJ NNS</td>
<td>4282 2526</td>
<td>kids</td>
<td>32</td>
<td></td>
</tr>
</tbody>
</table>

(Demo)
Translation
Help you, I can!
Yes! Mm!

When 900 years old you reach,
look as good, you will not. Hm.