Announcements
Announcements

• Homework 10 due Wednesday 12/3 @ 11:59pm
Announcements

• Homework 10 due Wednesday 12/3 @ 11:59pm
• Quiz 3 released Wednesday, due Thursday 12/4 @ 11:59pm
Announcements

• Homework 10 due Wednesday 12/3 @ 11:59pm
• Quiz 3 released Wednesday, due Thursday 12/4 @ 11:59pm
• No videos for Lecture 38 on Friday 12/5
Announcements

- Homework 10 due Wednesday 12/3 @ 11:59pm
- Quiz 3 released Wednesday, due Thursday 12/4 @ 11:59pm
- No videos for Lecture 38 on Friday 12/5
  - Come to class and take the final survey
Announcements

• Homework 10 due Wednesday 12/3 @ 11:59pm

• Quiz 3 released Wednesday, due Thursday 12/4 @ 11:59pm

• No videos for Lecture 38 on Friday 12/5
  ▪ Come to class and take the final survey
  ▪ There will be a screencast of live lecture (http://goo.gl/hyUTca)
Announcements

• Homework 10 due Wednesday 12/3 @ 11:59pm
• Quiz 3 released Wednesday, due Thursday 12/4 @ 11:59pm
• No videos for Lecture 38 on Friday 12/5
  ▪ Come to class and take the final survey
  ▪ There will be a screencast of live lecture (http://goo.gl/hyUTca)
• Final exam held on Thursday 12/18 3pm–6pm
Announcements

• Homework 10 due Wednesday 12/3 @ 11:59pm
• Quiz 3 released Wednesday, due Thursday 12/4 @ 11:59pm
• No videos for Lecture 38 on Friday 12/5
  ▪ Come to class and take the final survey
  ▪ There will be a screencast of live lecture (http://goo.gl/hyUTca)
• Final exam held on Thursday 12/18 3pm–6pm
  ▪ 30 hours of review sessions next week! Monday – Friday 11am–6pm (mostly in 271 Soda)
Ambiguity
Syntactic Ambiguity in English

Programs must be written for people to read
Syntactic Ambiguity in English

Programs must be written for people to read\(^1\)

\(^1\)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
Programs must be written for people to read\(^1\)

\(^1\)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

Programs must be written for people to read

1Preface 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

Programs must be written for people to read

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

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

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

buffalo
Representing Syntactic Structure

Noun

buffalo
Representing Syntactic Structure

Photo by Vince O'Sullivan licensed under http://creativecommons.org/licenses/by-nc-nd/2.0/

**Noun**

buffalo
Representing Syntactic Structure

Photo by Vince O'Sullivan licensed under http://creativecommons.org/licenses/by-nc-nd/2.0/
Representing Syntactic Structure

Photo by Vince O'Sullivan licensed under
http://creativecommons.org/licenses/by-nc-nd/2.0/
Representing Syntactic Structure

Photo by Vince O'Sullivan licensed under http://creativecommons.org/licenses/by-nc-nd/2.0/
Representing Syntactic Structure

Sentence

Noun buffalo Verb buffalo Noun buffalo

Photo by Vince O'Sullivan licensed under http://creativecommons.org/licenses/by-nc-nd/2.0/
Representing Syntactic Structure

Sentence

Phrase

Noun

Verb

Noun

buffalo

buffalo

buffalo

Photo by Vince O'Sullivan licensed under http://creativecommons.org/licenses/by-nc-nd/2.0/
Representing Syntactic Structure

Photo by Vince O'Sullivan licensed under http://creativecommons.org/licenses/by-nc-nd/2.0/
Representing Syntactic Structure

Photo by Vince O'Sullivan licensed under http://creativecommons.org/licenses/by-nc-nd/2.0/
A Tree represents a phrase:
Representing Syntactic Structure

A Tree represents a phrase:

- **tag** — What kind of phrase (e.g., S, NP, VP)

Photo by Vince O'Sullivan licensed under
http://creativecommons.org/licenses/by-nc-nd/2.0/
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
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:
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)
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

---

Photo by **Vince O'Sullivan** licensed under [http://creativecommons.org/licenses/by-nc-nd/2.0/](http://creativecommons.org/licenses/by-nc-nd/2.0/)
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

---

Photo by Vince O'Sullivan licensed under [http://creativecommons.org/licenses/by-nc-nd/2.0/](http://creativecommons.org/licenses/by-nc-nd/2.0/)
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

Photo by Vince O'Sullivan licensed under http://creativecommons.org/licenses/by-nc-nd/2.0/
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

[Image of a tree diagram with tags and components labeled.]
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

Photo by Vince O'Sullivan licensed under http://creativecommons.org/licenses/by-nc-nd/2.0/
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

Photo by Vince O'Sullivan licensed under http://creativecommons.org/licenses/by-nc-nd/2.0/
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
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

Photo by Vince O’Sullivan licensed under [http://creativecommons.org/licenses/by-nc-nd/2.0/](http://creativecommons.org/licenses/by-nc-nd/2.0/)
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

beasts = Leaf('N', 'buffalo')
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

```
beasts = Leaf('N', 'buffalo')
imidate = Leaf('V', 'buffalo')
```
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

beasts = Leaf('N', 'buffalo')
intimidate = Leaf('V', 'buffalo')
S, NP, VP = 'S', 'NP', 'VP'
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

```
beasts = Leaf('N', 'buffalo')
imintidate = Leaf('V', 'buffalo')
S, NP, VP = 'S', 'NP', 'VP'
Tree(S, [Tree(NP, [beasts])])
```
### 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

---

**beasts** = Leaf('N', 'buffalo')

**intimidate** = Leaf('V', 'buffalo')

S, NP, VP = 'S', 'NP', 'VP'

Tree(S, [Tree(NP, [beasts]),
       Tree(VP, [intimidate,}
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

```plaintext
beasts = Leaf('N', 'buffalo')
intimidate = Leaf('V', 'buffalo')
S, NP, VP = 'S', 'NP', 'VP'
Tree(S, [Tree(NP, [beasts]),
        Tree(VP, [intimidate,
                 Tree(NP, [beasts])])])
```
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

**beasts** = Leaf('N', 'buffalo')
**intimidate** = Leaf('V', 'buffalo')
S, NP, VP = 'S', 'NP', 'VP'
Tree(S, [Tree(NP, [beasts]),
       Tree(VP, [intimidate,
              Tree(NP, [beasts])])])

(Demo)
Grammars
Context-Free Grammar Rules
Context-Free Grammar Rules

A grammar rule describes how a tag can be expanded as a sequence of tags or words.
Context-Free Grammar Rules

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

\[ S \rightarrow NP \ VP \]
Context-Free Grammar Rules

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

A Sentence ...

S → NP VP
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 ...

\[ S \rightarrow NP \ VP \]
A grammar rule describes how a tag can be expanded as a sequence of tags or words.

**Context-Free Grammar Rules**

A Sentence ...

... can be expanded as ...

... a Noun Phrase then a Verb Phrase.
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.

Grammar
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.

Grammar

\[
S \rightarrow NP \text{ VP}
\]
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.

Grammar

\[
S \rightarrow NP \ VP
\]
A grammar rule describes how a tag can be expanded as a sequence of tags or words. For example:

**Grammar**

\[ S \rightarrow NP \rightarrow VP \]

A *Sentence* ...

... can be expanded as ...

... a *Noun Phrase* then a *Verb Phrase*.

Diagram:

```
  S -> NP -> VP
```

```
  NP     S     VP
```

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.

Grammar

\[
S \rightarrow NP \ VP \\
NP \rightarrow N
\]
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.

Grammar:

\[
S \rightarrow \text{NP VP} \\
\text{NP} \rightarrow \text{N}
\]
A grammar rule describes how a tag can be expanded as a sequence of tags or words.

**Context-Free Grammar Rules**

A sentence ...

... can be expanded as ...

... a Noun Phrase then a Verb Phrase.

**Grammar**

\[
\begin{align*}
S & \rightarrow \text{NP} \quad \text{VP} \\
\text{NP} & \rightarrow \text{N} \\
\text{N} & \rightarrow \text{buffalo}
\end{align*}
\]
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.

Grammar

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>→</td>
<td>NP</td>
</tr>
<tr>
<td>NP</td>
<td>→</td>
<td>N</td>
</tr>
<tr>
<td>N</td>
<td>→</td>
<td>buffalo</td>
</tr>
</tbody>
</table>

buffalo
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.

Grammar

\[
\begin{align*}
S & \rightarrow \text{NP} \ \text{VP} \\
\text{NP} & \rightarrow \text{N} \\
\text{N} & \rightarrow \text{buffalo}
\end{align*}
\]
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.

Grammar

<table>
<thead>
<tr>
<th>S</th>
<th>→</th>
<th>NP</th>
<th>VP</th>
</tr>
</thead>
<tbody>
<tr>
<td>NP</td>
<td>→</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>→</td>
<td>buffalo</td>
<td></td>
</tr>
<tr>
<td>VP</td>
<td>→</td>
<td>V</td>
<td>NP</td>
</tr>
</tbody>
</table>

buffalo
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**.

### Grammar

<table>
<thead>
<tr>
<th>Rule</th>
<th>Symbol(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>S  →  NP  VP</td>
<td></td>
</tr>
<tr>
<td>NP  →  N</td>
<td></td>
</tr>
<tr>
<td>N  →  buffalo</td>
<td></td>
</tr>
<tr>
<td>VP  →  V  NP</td>
<td></td>
</tr>
</tbody>
</table>

"buffalo"
Context-Free Grammar Rules

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

\[
S \rightarrow \text{NP} \quad \text{VP}
\]

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 → buffalo</td>
</tr>
<tr>
<td>VP → V NP</td>
</tr>
<tr>
<td>V → buffalo</td>
</tr>
</tbody>
</table>
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.

Grammar

\[
\begin{align*}
S & \rightarrow NP \ VP \\
NP & \rightarrow N \\
N & \rightarrow \text{buffalo} \\
VP & \rightarrow V \ NP \\
V & \rightarrow \text{buffalo}
\end{align*}
\]
A grammar rule describes how a tag can be expanded as a sequence of tags or words:

**Grammar**

- **S** → **NP** **VP**
- **NP** → **N**
- **N** → *buffalo*
- **VP** → **V** **NP**
- **V** → *buffalo*

A Sentence ...

... can be expanded as ...

... a Noun Phrase then a Verb Phrase.
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 → buffalo</td>
</tr>
<tr>
<td>VP → V NP</td>
</tr>
<tr>
<td>V → buffalo</td>
</tr>
</tbody>
</table>
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.

Grammar

\[
\begin{align*}
S & \rightarrow \text{NP} \; \text{VP} \\
\text{NP} & \rightarrow \text{N} \\
\text{N} & \rightarrow \text{buffalo} \\
\text{VP} & \rightarrow \text{V} \; \text{NP} \\
\text{V} & \rightarrow \text{buffalo}
\end{align*}
\]
Context-Free Grammar Rules

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

Grammar

\[
S \rightarrow NP \ VP \\
NP \rightarrow \ N \\
N \rightarrow \text{buffalo} \\
VP \rightarrow \ V \ NP \\
V \rightarrow \text{buffalo}
\]

(Demo)
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

buffalo  buffalo  buffalo  buffalo
Exhaustive Parsing

Expand all tags recursively, but constrain words to match input

```
buffalo  buffalo  buffalo  buffalo
```
Exhaustive Parsing

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

Expand all tags recursively, but constrain words to match input

```
S
```

```
buffalo  buffalo  buffalo  buffalo
```

```
Exhaustive Parsing

Expand all tags recursively, but constrain words to match input

---

buffalo 1 buffalo 2 buffalo 3 buffalo 4
Exhaustive Parsing

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

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

Expand all tags recursively, but constrain words to match input

```
S
NP  buffalo  buffalo  buffalo  buffalo  VP
  0   1       2       3       4
```
Exhaustive Parsing

Expand all tags recursively, but constrain words to match input

```
Exhaustive Parsing

Expand all tags recursively, but constrain words to match input

<table>
<thead>
<tr>
<th>buffalo</th>
<th>buffalo</th>
<th>buffalo</th>
<th>buffalo</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>
```
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

Constraint: A Leaf must match the input word

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

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

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

Expand all tags recursively, but constrain words to match input
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
Scoring a Tree Using Relative Frequencies

Not all syntactic structures are equally common

teacher strikes idle kids
Scoring a Tree Using Relative Frequencies

Not all syntactic structures are equally common

```
S
```

teacher strikes idle kids
Scoring a Tree Using Relative Frequencies

Not all syntactic structures are equally common

teacher strikes idle kids
Scoring a Tree Using Relative Frequencies

Not all syntactic structures are equally common

teacher strikes idle kids
Scoring a Tree Using Relative Frequencies

Not all syntactic structures are equally common

teacher strikes idle kids
Scoring a Tree Using Relative Frequencies

Not all syntactic structures are equally common

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

```
teacher strikes idle kids
```

Rule frequency per 100,000 tags

- NN → teacher
- NNS → strikes
- VB → idle
- NNS → kids
Scoring a Tree Using Relative Frequencies

Not all syntactic structures are equally common

teacher strikes idle kids

Rule frequency per 100,000 tags

\[
\begin{align*}
S & \rightarrow \text{NP VP} & 25372 & \text{NN} & \rightarrow & \text{teacher} \\
\text{NP} & \rightarrow \text{NN NNS} & & \text{NNS} & \rightarrow & \text{strikes} \\
\text{VP} & \rightarrow \text{VB NP} & & \text{VB} & \rightarrow & \text{idle} \\
\text{NP} & \rightarrow \text{NNS} & & \text{NNS} & \rightarrow & \text{kids}
\end{align*}
\]
Scoring a Tree Using Relative Frequencies

Not all syntactic structures are equally common

```
Scoring a Tree Using Relative Frequencies

Not all syntactic structures are equally common

```

```
Rule frequency per 100,000 tags

S → NP VP  25372  NN →  teacher
NP → NN NNS  1335  NNS →  strikes
VP → VB NP  1335  VB →  idle
NP → NNS  1335  NNS →  kids
```
Scoring a Tree Using Relative Frequencies

Not all syntactic structures are equally common

```
S   NP   VP
```

```
NP   NN   NNS   VB   NNS
```

**teacher strikes idle kids**

**Rule frequency per 100,000 tags**

- $S \rightarrow NP\ VP$ 25372 $NN \rightarrow$ teacher
- $NP \rightarrow NN\ NNS$ 1335 $NNS \rightarrow$ strikes
- $VP \rightarrow VB\ NP$ 6679 $VB \rightarrow$ idle
- $NP \rightarrow NNS$ $NNS \rightarrow$ kids
Scoring a Tree Using Relative Frequencies

Not all syntactic structures are equally common

```
S  ---  NP  ---  VP
    |  /     |  |
    |     NP  |
    |    /    |
    |   /     |
    |  /      |
    | /       |
    |/        |
  NN  NNS  VB  NNS
```

teacher strikes idle kids

Rule frequency per 100,000 tags

|  |  |  |
| S --> NP VP | 25372 | NN --> teacher |
| NP --> NN NNS | 1335 | NNS --> strikes |
| VP --> VB NP | 6679 | VB --> idle |
| NP --> NNS | 4282 | NNS --> kids |
### Scoring a Tree Using Relative Frequencies

Not all syntactic structures are equally common

```
S ————
|       |
NP ———— VP ———— NP
     |       |
  NN   NNS   VB   NNS

teacher strikes idle kids
```

**Rule frequency per 100,000 tags**

<table>
<thead>
<tr>
<th>Rule</th>
<th>Frequency</th>
<th>Tag</th>
<th>Value</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></td>
</tr>
<tr>
<td>VP ———— VB ———— NP</td>
<td>6679</td>
<td>idle</td>
<td></td>
</tr>
<tr>
<td>NP ———— NNS</td>
<td>4282</td>
<td>kids</td>
<td></td>
</tr>
</tbody>
</table>
Scoring a Tree Using Relative Frequencies

Not all syntactic structures are equally common.

```
S ————> NP  VP
NP  ————> NN  NNS
VP  ————> VB  NP
NP  ————> NNS
```

**teacher strikes idle kids**

**Rule frequency per 100,000 tags**

<table>
<thead>
<tr>
<th>Rule</th>
<th>Frequency</th>
<th>Tag</th>
<th>Word</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>S  ————&gt; NP  VP</td>
<td>25372</td>
<td>NN</td>
<td>teacher</td>
<td>5</td>
</tr>
<tr>
<td>NP  ————&gt; NN  NNS</td>
<td>1335</td>
<td>NNS</td>
<td>strikes</td>
<td>25</td>
</tr>
<tr>
<td>VP  ————&gt; VB  NP</td>
<td>6679</td>
<td>VB</td>
<td>idle</td>
<td></td>
</tr>
<tr>
<td>NP  ————&gt; NNS</td>
<td>4282</td>
<td>NNS</td>
<td>kids</td>
<td></td>
</tr>
</tbody>
</table>
Scoring a Tree Using Relative Frequencies

Not all syntactic structures are equally common

teacher strikes idle kids

Rule frequency per 100,000 tags

- $S \rightarrow NP \ VP$: 25372
- $NP \rightarrow NN \ NNS$: 1335
- $VP \rightarrow VB \ NP$: 6679
- $NP \rightarrow NNS$: 4282

- $NN \rightarrow$ teacher: 5
- $NNS \rightarrow$ strikes: 25
- $VB \rightarrow$ idle: 26
- $NNS \rightarrow$ kids
Scoring a Tree Using Relative Frequencies

Not all syntactic structures are equally common

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

teacher strikes idle kids

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>
Scoring a Tree Using Relative Frequencies

Not all syntactic structures are equally common

```
S ⟷ NP ⟷ VP
```

```
NN ⟷ VBZ ⟷ 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>NN ⟷ teacher</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>NP ⟷ NN</td>
<td>1335</td>
<td>VBZ ⟷ strikes</td>
<td>25</td>
<td>19</td>
</tr>
<tr>
<td>VP ⟷ VBZ ⟷ NP</td>
<td>6679</td>
<td>JJ ⟷ idle</td>
<td>26</td>
<td>18</td>
</tr>
<tr>
<td>NP ⟷ JJ ⟷ NNS</td>
<td>4282</td>
<td>NNS ⟷ kids</td>
<td>32</td>
<td></td>
</tr>
</tbody>
</table>
Scoring a Tree Using Relative Frequencies

Not all syntactic structures are equally common

```
S ── NP ── VP ── NP
  |       |       |
  NN  VBZ   JJ  NNS
```

teacher strikes idle kids

**Rule frequency per 100,000 tags**

<table>
<thead>
<tr>
<th>Rule</th>
<th>Frequency 1</th>
<th>Frequency 2</th>
<th>Target</th>
<th>Frequency 3</th>
<th>Frequency 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>S → NP VP</td>
<td>25372</td>
<td></td>
<td>teacher</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>NP → NN</td>
<td>1335</td>
<td>4358</td>
<td>VBZ</td>
<td>25</td>
<td>19</td>
</tr>
<tr>
<td>VP → VBZ NP</td>
<td>6679</td>
<td>3160</td>
<td>JJ</td>
<td>26</td>
<td>18</td>
</tr>
<tr>
<td>NP → JJ NNS</td>
<td>4282</td>
<td>2526</td>
<td>NNS</td>
<td>32</td>
<td></td>
</tr>
</tbody>
</table>

(Demo)