Syntax

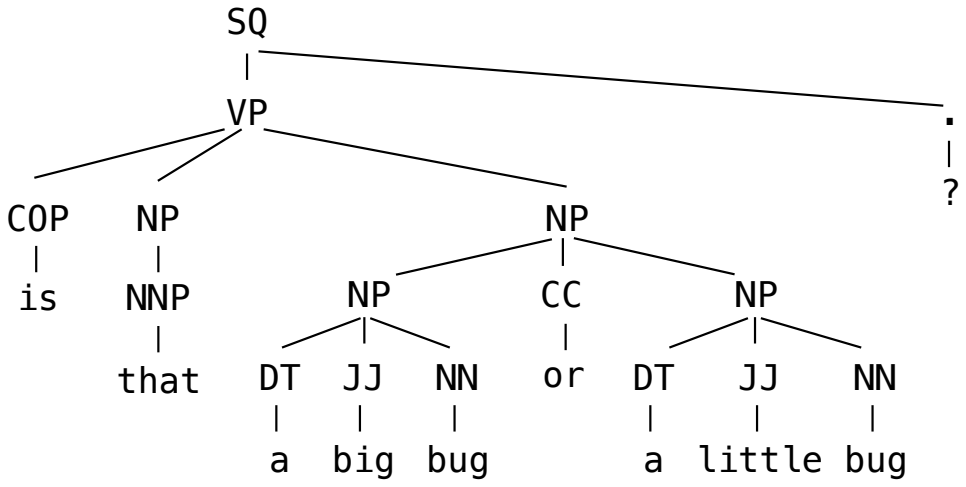
Announcements

Natural Language Syntax

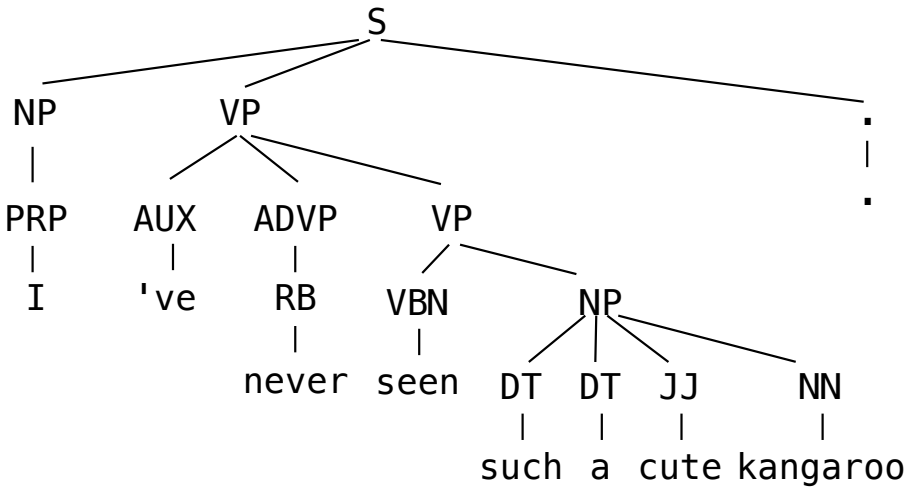
English Syntax

Programming languages and natural languages both have compositional syntax.

Is that a big bug or a little bug?



I've never seen such a cute kangaroo.



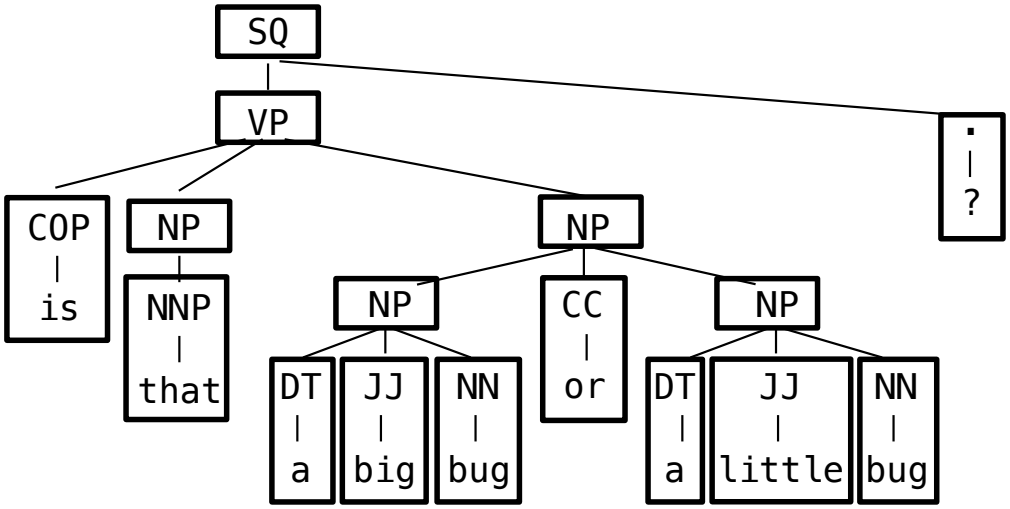
Utterances from the Suppes subject in the "Child Language Data Exchange System (CHILDES)" project

MacWhinney, B. (2000). The CHILDES Project: Tools for analyzing talk. Third Edition. Mahwah, NJ: Lawrence Erlbaum Associates.

Data Abstraction for Syntax

Representing English Syntax

The tree data abstraction can represent the structure of a sentence.



The label of a phrase is its tag: `'SQ'`

The label of a word (a leaf) contains both its tag and text: `['JJ', 'big']`

(Demo)

Parsing

Files, Strings, and Lists

Some files are plain text and can be read into Python as either:

- One string containing the whole contents of the file: **open('/some/file.txt').read()**
- A list of strings, each containing one line: **open('/some/file.txt').readlines()**

Useful string methods for processing the contents of a file:

.strip() returns a string without whitespace (spaces, tabs, etc.) on the ends

```
>>> ' hello '.strip()
'hello'
```

.split() returns a list of strings that were separated by whitespace

```
>>> 'hi  there'.split()
['hi', 'there']
```

.replace(a, b) returns a string with all instances of string **a** replaced by string **b**

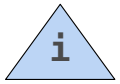
```
>>> '2+2'.replace('+', ' + ')
'2 + 2'
```

(Demo)

Converting a List Describing a Constituent (Phrase or Word) to a Tree

```
['(', 'ROOT', '(', 'SQ', '(', 'VP', '(', 'COP', 'is', ')', '(', 'NP', '(', 'NN', 'that', ')',
')', '(', 'NP', '(', 'NP', '(', 'DT', 'a', ')', '(', 'JJ', 'big', ')', '(', 'NN', 'bug', ')',
')', '(', 'CC', 'or', ')', '(', 'NP', '(', 'DT', 'a', ')', '(', 'JJ', 'little', ')', '(',
'NN', 'bug', ')', ')', ')', ')', '(', '.', '?', ')', ')', ')']

[..., '(', 'NP', '(', 'DT', 'a', ')', '(', 'JJ', 'big', ')', '(', 'NN', 'bug', ')', ')', ...]
```




```
def read_parse_tree(tokens, i):
    Read the tag, which is tokens[i], then advance i.
    While the current item is a '(', call read_parse_tree to construct a branch.
    Once the current item is a ')', return a phrase from the tag and branches.
    Base case: there is no '(' or ')' because there is just text after the tag.
```

read_parse_tree needs to return the tree it read and what to read next.

(Demo)

Generating Language

Language Models

A statistical (or probabilistic) language model describes how likely some text would be.

What word do you think appears at the end of this ____?

Sampling from a statistical language model uses that description to generate language.

A useful language model needs to generalize from examples.

E.g., Substitute any phrase in an utterance with any other phrase that has the same tag.

(S (NP (DT the) (NN dog)) (VP (VBD ran)))

(S (NP (DT the) (NN water)) (VP (VBD evaporated)))

Sampling Strategy

Starting with the branches of the root node, flip a coin for each branch. If it comes up heads, swap that branch for another constituent (phrase or word) that has the same tag. Then, apply this procedure to all of the branches.

```
def gen_tree(t, flip):
    """Return a version of t in which constituents are randomly replaced."""
    if is_leaf(t):
        return t
    new_branches = []
    for b in branches(t):
        if flip():
            b = random.choice(all_constituents_with_the_same_tag)
        new_branches.append(gen_tree(b, flip))
    return phrase(tag(t), new_branches)
```

First create a dictionary from tags to lists of constituents

(Demo)

Changing the Data Representation

Data Abstraction Makes It Possible to Change the Data Representation

```
>>> lines
['(ROOT (S (NP (NN this)) (VP (COP is) (NP (DT a) (NN book)))', '  (. ?)))']

>>> read_sentences(lines)[0]
['(', 'ROOT', '(', 'S', '(', 'NP', '(', 'NN', 'this', ')', ')', ... , ')']

>>> tokens_to_parse_tree(read_sentences(lines)[0])
['ROOT', ['S', ['NP', [['NN', 'this']]], ['VP', [['COP', 'is']], ... ]]]]
```

(Demo)