Tutorial 11: Introduction to syntax

Practice with a simple grammar in NLTK

Here is the phrase structure grammar introduced at the end of Lecture 21:

S	→	NP, VP	Det	→	"the" | "a"
NP	→	PN	N	→	"dog" | "cat"
NP	→	Det, N	PN	→	"Fred" | "Jip"
VP	→	V0	V0	→	"slept" | "ran" | "snorted"
VP	→	V1, NP	V1	→	"bit" | "chased" | "caught"
S	→	S, Conj, S	Conj	→	"and" | "so"

In this exercise, you're going to experiment with this grammar in **NLTK**, the Python natural language toolkit. You must first set up Canopy to use a particular GUI (graphical user interface) called **TkInter**. In Canopy's Preferences dialog (Python tab), change the PyLab backend from Interactive (Qt4) to Inline (SVG), so it will not conflict with TkInter.

1. In Canopy, open the file `nltk_grammar1.py` from the tutorials page. This file defines the above grammar in NLTK notation. Try out two demo functions:

   (a) `demo_parse(string, grammar)` takes a string, tokenises it into a word sequence, and tries to parse the sequence using the specified grammar. Try out some examples, to see how it works. (N.B. if an input parses successfully, the tree structure is displayed in a separate window: you have to close the window to get back to the Python prompt.)

   (b) `create_sentences(grammar, max_depth, max_sentences)` produces sentences from `grammar`. Try different values of `max_depth` and `max_sentences`, to get a sense of the possible sentences your grammar can produce.

2. Add some more words to the grammar, to expand its coverage, and try parsing or generating some more sentences, to see how the expanded grammar works.

3. In Lecture 21, you saw an example of a **top-down parser**, that searches the space of possible sentences starting with the top-level node S. Run the function `top_down_parser_demo(string, grammar)` to explore how a top-down parser works.

(b) It can keep going without knowing there is no solution.
For example, My sentence is "Fred caught some dog murdered by Frank".

The reason cause that is the parser algorithm is depth first search, so when do the recursion it can happen that the depth is infinite. And depth first search is not complete when the tree's depth is infinite.

Add new gammer: S->Conj, VP->PN, NP->S, basically I want to try to try breadth first search.
Ask teacher: Is there a best adding rule? Or can we evaluate the good or bad of some adding rule?

   (a) First, try it on a sentence that's parseable by the grammar. (You have to hit 'step' at each iteration of the parser.)

   (b) Next, try it on a sentence that's *not* parseable by the grammar. What goes wrong? How could you change the grammar so the top-down parser can correctly reject ungrammatical sentences?

4. **Quiz:** Add some new rules to the grammar, that allow it to parse/generate sentences like the following:

   Fred thinks Jip bit the cat.
   The cat thinks Fred thinks Jip ran.
   A dog thinks the cat thinks Fred thinks Jip ran.

   Send your complete grammar as the answer to the question.

Syntax discussion questions

Having looked at a simple grammar, and tried out some sentence parsers and generators, have a go at these questions:

5. Speakers of any given language can produce and understand sentences *they have never heard before*. What does this tell us about human language?

6. Is the set of possible English sentences infinite? If it is not, explain why not. If it is, what does this tell us about our mechanism for generating sentences?

7. A grammar breaks up a well-formed sentence into **phrases**: hierarchically-strucured sequences of words that 'hang together'. When we're building a grammar for a given language, how do we decide which word sequences should be phrases?

8. Explain how the process of **parsing** a sequence of words, using a grammar, is a variety of **state space search**.

To find out more...

To find out more about grammar development and parsing in NLTK, see here:

`http://www.nltk.org/book/ch08.html`

`http://www.nltk.org/` is an excellent resource for natural language processing in general; we're only going to scratch the surface in COSC343.