Lab for week 5: Recursive Descent Parser -- Answers

Author: Henry S. Thompson
Date: 2015-10-20, updated 2016-10-17

Adding productions to the grammar

- What production should be added to handle the sentence "He ate salad" ?

  Add the production `Pro -> 'He'`

- Is there a problem with either of the parse trees ?

  The parse tree for "he ate salad with a fork" is correct: the preposition phrase "with a fork" is correctly attached to the verb ('high' attachment). For the sentence "he ate salad with mushrooms", the preposition phrase "with mushrooms" is wrongly attached to the verb, whereas it should be attached to the noun phrase "salad" ('low' attachment).

- Change the order of the rules "NP -> N" and "NP -> NP PP"

  This leads into a standard problem of left recursion as "NP -> NP PP" production is applied infinitely.

- How do you think this behaviour depends on the particular way this recursive descent parser chooses which rule to expand when there are multiple options?

  Clearly it does: the infinite recursion depends on the left-recursive rule being chosen before other options (although note that in the case of some *unparseable* strings the recursion would happen no matter what the order was.

Ungrammatical sentences

- Though the second sentence is ungrammatical, it parsed by our grammar. Modify the grammar to handle such cases

  Change `VP -> V | V NP | V NP PP` to `VP -> Vi | Vt NP | Vp NP PP`. This will exploit the subcategorization information in the alternative set of productions for the verbs.

Number agreement (optional)

Change the grammar to handle number agreement and parse the following sentences:

Use `grammar2` in lab5-sol.py

[1] http://creativecommons.org/licenses/by-nc/4.0/.
[2] lab5-sol.py

Exploring a treebank grammar

- What is the type of the parsed sentence object (Hint: *type* command) ?

  `type(psents[0])` gives the object type which is `nltk.tree.Tree`

- Extract the list of words and the list of word,pos-tag tuples from `psents[0]` using some of the other available methods.

  `psents[0].leaves()` and *psents[0].pos()*‘ will give, respectively, the list of words and word, pos-tag tuples

Distribution of Productions

- What are the 10 most frequent and least frequent lexical and grammatical productions ?

  First download the answer code[2] and look at the definition of `production_distribution`

  Then do `%run lab5-sol.py`

  ```
      lex_prods, nonlex_prods = production_distribution(psents)
  ```

  For the 10 most frequent productions

  ```
      sorted(lex_prods.items(), key=lambda x :  x[1], reverse=True)[:10]
      sorted(nonlex_prods.items(), key=lambda x :  x[1], reverse=True)[:10]
  ```

  10 least frequent productions

  ```
      sorted(lex_prods.items(), key=lambda x :  x[1])[:10]
      sorted(nonlex_prods.items(), key=lambda x :  x[1])[:10]
  ```