Lab 5: Recursive Descent Parser

Running the code

- What is the type of the parsed sentence object (Hint: type command)?
  
  ```python
type(psents[0])
give 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.
  
  ```python
psents[0].leaves() and psents[0].pos()' will gives 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?
  
  ```python
lex_prods, nonlex_prods = production_distribution(psents)
```

10 most frequent productions

```python
sorted(lex_prods.items(), key=lambda x : x[1], reverse=True)[0:10]
sorted(nonlex_prods.items(), key=lambda x : x[1], reverse=True)[0:10]
```

10 least frequent productions

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

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 where 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.

Ungrammatical sentences

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

  ```python
Change VP -> V | V NP | V NP PP to VP -> Vi | Vt NP | Vp NP PP. This will handle the subcategorization information of verbs correctly.
```
Going further

1. What is the percentage of the 10 most frequent grammatical and lexical productions, with respect to the total number of productions?

   \[
   100 \times \frac{\text{sum}(\text{sorted}(\text{lex_prods.values()})[-10:])}{\text{sum}(\text{lex_prods.values()})}
   \]

   \[
   100 \times \frac{\text{sum}(\text{sorted}(\text{nonlex_prods.values()})[-10:])}{\text{sum}(\text{nonlex_prods.values()})}
   \]

2. Run the parser on the sentence "John ate salad". Add required grammatical and lexical productions to handle this sentence.

   We should add rules to handle proper nouns as below

   \[
   \text{NP } \rightarrow \text{PropN} ; \text{PropN } \rightarrow \text{"John"}
   \]

3. Does the rule-ordering trick work if you add a rule of the form: \text{Det } \rightarrow \text{NP 's'}

   This leads into the left recursion problem again.

4. In recursive descent parsing, you should have observed same production being used multiple types. How can we handle this problem?

   This can be handled using a Chart parser with a redundancy check.

5. Our grammar ignores the concept of number agreement. Change the grammar to handle number agreement and parse the following sentences:

   Use \text{grammar2} in \text{lab5-sol.py}

3. The grammar which handles number agreement still parses ungrammatical sentences like "i sleeps". Adding more agreement rules is a messy process. What is the better way to deal with this problem?

   We can generalize the rules using features.