NLP_lab-11_205229118_Mahalakshmi.S

May 30, 2021

0.1 Lab11. Building Parse Trees

0.1.1 EXERCISE-1

Build the following three tree objects as np, aux, and vp.

[1]:
```
import nltk,re,pprint
from nltk.tree import Tree
from nltk.tokenize import word_tokenize
from nltk.tag import pos_tag
from nltk.chunk import ne_chunk
import numpy as npt
```

[2]:
```
np =nltk.Tree.fromstring('(NP (N Marge))')    #### np
np.pretty_print()
```

```
  NP
  |
  N
  |
Marge
```

[4]:
```
aux= nltk.Tree.fromstring('(AUX will)')       #### aux
aux.pretty_print()
```

```
AUX
 |
will
```

[3]:
```
vp = nltk.Tree.fromstring('(VP (V make) (NP (DET a) (N ham) (N sandwich)))')␣
 ↪#### vp
vp.pretty_print()
```

```
          VP
  ________|___
 |            NP
 |     _______|_____
 V    DET      N     N
 |    |       |     |
```

```
make  a      ham sandwich
```

0.1.2 EXERCISE-3

Using them, build two tree objects, named s1 and s2, for the following sentences. The trees should look exactly like the ones shown below

(s1) Marge will make a ham sandwich

[5]:
```
s1 = nltk.Tree.fromstring('(S (NP (N Marge)) (AUX will) (VP (V make) (NP (DET␣
 ↪a) (N ham) (N sandwich))))')
s1.pretty_print()
```

```
              S
   ___________|________
  |     |              VP
  |     |     _________|___
  NP    |    |             NP
  |     |    |      _______|_____
  N    AUX   V    DET       N     N
  |     |    |     |        |     |
Marge will make   a       ham sandwich
```

(s2) will Marge make a ham sandwich

[6]:
```
s2 = nltk.Tree.fromstring('(S (AUX will) (NP (N Marge)) (VP (V make) (NP (DET␣
 ↪a) (N ham) (N sandwich))))')
s2.pretty_print()
```

```
              S
   ___________|________
  |      |              VP
  |      |     _________|___
  |      NP   |             NP
  |      |    |      _______|_____
 AUX     N    V    DET       N     N
  |      |    |     |        |     |
will Marge make   a       ham sandwich
```

0.1.3 EXERCISE-4

Build a tree object named s3 for the following sentence, using its full-sentence string representation.

(s3) Homer ate the donut on the table

[7]:
```

```

```
s3 = nltk.Tree.fromstring('(S (NP (N Homer)) (VP (V ate) (NP (DET the) (N␣
↪donut))))')
s3.pretty_print()
```

```
             S
   ________|___
  |            VP
  |     _______|___
  NP    |           NP
  |     |        ___|____
  N     V      DET       N
  |     |       |        |
Homer ate     the     donut
```

0.1.4 EXERCISE-5

Build tree objects named s4 and s5 for the following sentences.

(s4) my old cat died on Tuesday

[8]:
```
s4 = nltk.Tree.fromstring('(S (NP (DET my)(ADJ old)(N cat))(VP(VP(V died))(PP(P␣
↪on)(NP(N Tuesday)))))')
s4.pretty_print()
```

```
          S
       ___|________
      |            VP
      |         ____|___
      |        |        PP
      |        |     ___|_____
      NP       VP    |         NP
   ___|___     |     |         |
DET ADJ   N    V     P         N
 |   |    |    |     |         |
 my old cat died    on      Tuesday
```

(s5) children must play in the park with their friends

[9]:
```
s5 = nltk.Tree.fromstring('(S(NP (N children))(AUX must)(VP(VP(V play))(PP(P␣
↪in)(NP(DET the)(N park)))(PP (P with)(NP (DET their)(N friends)))))')
s5.pretty_print()
```

```
                        S
    ____________________|_______
   |       |                    VP
   |       |     _________________|__________
   |       |    |                PP                PP
   |       |    |            ____|___          _____|____
```

```
   NP     |    VP    |        NP        |           NP
   |      |    |     |     ___|___      |       ____|_____
   N     AUX   V     P  DET       N     P     DET          N
   |      |    |     |   |        |     |      |           |
children must play  in the     park with their     friends
```

0.1.5 EXERCISE-6

Once a tree is built, you can extract a list of context-free rules, generally called production rules, from it using the .productions() method. Each CF rule in the list is either lexical, i.e, contains a lexical word on its right-hand side, or not:

[10]:
```
print(vp)
```

```
(VP (V make) (NP (DET a) (N ham) (N sandwich)))
```

[11]:
```
vp_rules = vp.productions()
vp_rules
```

[11]:
```
[VP -> V NP,
 V -> 'make',
 NP -> DET N N,
 DET -> 'a',
 N -> 'ham',
 N -> 'sandwich']
```

[12]:
```
vp_rules[0]
```

[12]:
```
VP -> V NP
```

[13]:
```
vp_rules[1]
```

[13]:
```
V -> 'make'
```

[14]:
```
vp_rules[0].is_lexical()
```

[14]:
```
False
```

[15]:
```
vp_rules[0].is_lexical()
```

[15]:
```
False
```

[16]:
```
 vp_rules[1].is_lexical()
```

[16]:
```
True
```

Explore the CF rules of s5. Include in your script the answers to the following:

[17]:
```
print(s5)
```

```
(S
  (NP (N children))
  (AUX must)
  (VP
    (VP (V play))
    (PP (P in) (NP (DET the) (N park)))
    (PP (P with) (NP (DET their) (N friends)))))
```

[18]:
```
cf_s5_rules=s5.productions()
cf_s5_rules
```

[18]:
```
[S -> NP AUX VP,
 NP -> N,
 N -> 'children',
 AUX -> 'must',
 VP -> VP PP PP,
 VP -> V,
 V -> 'play',
 PP -> P NP,
 P -> 'in',
 NP -> DET N,
 DET -> 'the',
 N -> 'park',
 PP -> P NP,
 P -> 'with',
 NP -> DET N,
 DET -> 'their',
 N -> 'friends']
```

a. How many CF rules are used in s5?

[22]:
```
print("How Many CF values are used in s5  ",len(cf_s5_rules))
```

```
How Many CF values are used in s5   17
```

b. How many unique CF rules are used in s5?

[23]:
```
x = npt.array(cf_s5_rules)
print("How Many unique CF rules are used in s5  ",len(npt.unique(x)))
```

```
How Many unique CF rules are used in s5   15
```

c. How many of them are lexical?

[25]:
```
n=0
for x in cf_s5_rules:
  if x.is_lexical():
    n = n+1
print("How many of them are lexical?  ",n)
```

```
How many of them are lexical?   9
```