Handout 15: Parsing

NLTK chart parser

1. A toy grammar

a. Create a file called `g1.cfg` containing:

```
% start S

S -> NP VP
VP -> V NP | V NP PP
PP -> P NP
NP -> Det N | Det N PP | Name

V -> "saw" | "ate" | "walked"
Name -> "John" | "Mary" | "Bob"
Det -> "a" | "an" | "the" | "my"
N -> "man" | "dog" | "cat" | "telescope" | "park"
P -> "in" | "on" | "by" | "with"
```

b. In Python:

```
>>> s = open(’g1.cfg’).read()
>>> from nltk import CFG
>>> g = CFG.fromstring(s)
>>> type(g)
<class ’nltk.grammar.CFG’>
>>> print(g)
Grammar with 26 productions (start state = S)
    S -> NP VP
    VP -> V NP
    ...
```

2. Create and use a parser

```
>>> from nltk import ChartParser
>>> p = ChartParser(g)
>>> s = ’the dog saw the cat’.split()
>>> s
[’the’, ’dog’, ’saw’, ’the’, ’cat’]
>>> for t in p.parse(s):
...     print(t)
...
(S (NP (Det the) (N dog)) (VP (V saw) (NP (Det the) (N cat))))
```

3. That's a lot of steps to remember. Assignment: package them up as a function called `make_parser`:

```
>>> p = make_parser('g1.cfg')
>>> print(next(p.parse(s)))
(S (NP (Det the) (N dog)) (VP (V saw) (NP (Det the) (N cat))))
```

4. Note: if you have a parser `p`, you can get the grammar:

```
>>> p.grammar()
<Grammar with 26 productions>
```

5. What if the sentence fails to parse?

a. Not very useful:

```
>>> s = 'the dog with the telescope walked'.split()
>>> p.parse(s).next()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
StopIteration
```

b. Getting a trace:

```
>>> p.chart_parse(s, trace=2)
|. the  . dog  . with . the  .telesc.walked.|
...
Single Edge Fundamental Rule:
|[---------------------------------]       .| [0:5] NP -> Det N PP *
...
Bottom Up Predict Combine Rule:
|.      .      .      .      .      [------>| [5:6] VP -> V * NP
|.      .      .      .      .      [------>| [5:6] VP -> V * NP PP
<nltk.parse.chart.Chart object at 0x101c9f850>
```

Trees

6. Anatomy of an NLTK tree

a. Creating a tree manually

```
>>> from nltk import Tree
>>> t = Tree.fromstring('(S (NP Fido) (V barked))')
>>> print(t)
(S (NP Fido) (V barked))
```

b. The structure

```
>>> t
Tree('S', [Tree('NP', ['Fido']), Tree('V', ['barked'])])
```

7. Tree = node

a. Label

```
>>> t.label()
’S’
```

b. The children of the node: tree behaves like a list

```
>>> len(t)
2
>>> print(t[0])
(NP Fido)
>>> print(t[1])
(V barked)
```

8. Leaf nodes

a. The leaves are strings, not trees

```
>>> np = t[0]
>>> print(np)
(NP Fido)
>>> len(np)
1
>>> n = np[0]
>>> n
’Fido’
```

b. Testing for a leaf

```
>>> isinstance(np, str)
False
>>> isinstance(n, str)
True
```

9. Creating a tree

```
>>> subj = Tree(’NP’, [’Garfield’])
>>> obj = Tree(’NP’, [’lasagna’])
>>> v = Tree(’V’, [’likes’])
>>> vp = Tree(’VP’, [v, obj])
>>> s = Tree(’S’, [subj, vp])
>>> print(s)
(S (NP Garfield) (VP (V likes) (NP lasagna)))
```

10. Functions on trees

a. Comparison of trees

```
>>> t == Tree.fromstring(’(S (NP Spot) (V ran))’)
False
>>> t == Tree.fromstring(’(S (NP Fido) (V barked))’)
True
```

b. Subtrees

```
>>> for st in t.subtrees(): print(st)
...
(S (NP Fido) (V barked))
(NP Fido)
(V barked)
```

c. Leaves

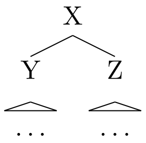
```
>>> t.leaves()
['Fido', 'barked']
>>> t.pos()
[('Fido', 'NP'), ('barked', 'V')]
```

11. Defining recursive functions on tree structure

a. Write a function that counts the nodes in a tree. For example:

```
>>> count_nodes(t)
3
>>> count_nodes(np)
1
```

b. Answer: how many nodes in a tree? One for X, plus the number in the Y subtree, plus the number in the Z subtree.



```
n = 1
for child in tree:
    n += count_nodes(child)
```

c. “Bottoming out”: when you get to a terminal node (string), the answer is 0 (not a node!).

```
def count_nodes (tree):
    if isinstance(tree, str):
        return 0
    else:
        n = 1
        for child in tree:
            n += count_nodes(child)
        return n
```

12. Another example: make a mirror image of a tree in which all expansions are reversed from the original.

13. Saving trees

a. Load g1.cfg:

```
>>> p = make_parser(’g1.cfg’)
>>> s = ’the dog saw the cat’.split()
>>> trees = list(p.parse(s))
```

b. Converting a tree to a string:

```
>>> str(trees[0])
’(S (NP (Det the) (N dog)) (VP (V saw) (NP (Det the) (N cat))))’
```

c. We know how to write a file

```
>>> with open(’sents1.txt’, ’w’) as f:
...     f.write(str(trees[0]) + ’\n’)
...
```

14. Loading trees

a. Reload

```
>>> f = open(’sents1.txt’)
>>> line = next(f).rstrip(’\r\n’)
>>> t = Tree.fromstring(line)
>>> print(t)
(S (NP (Det the) (N dog)) (VP (V saw) (NP (Det the) (N cat))))
```

b. Suppose g1.cfg has been modified. Are we still getting the same result as before?

```
>>> p = make_parser(’g1.cfg’)
>>> trees = list(p.parse(s))
>>> t == trees[0]
True
```

15. Penn treebank (sample)

```
>>> from nltk.corpus import treebank
>>> trees = treebank.parsed_sents()
>>> print(trees[0])
(S
  (NP-SBJ
    (NP (NNP Pierre) (NNP Vinken))
    ...
```