exp8

October 31, 2021

Neha Kulkarni

A060 BTech. IT SEM 7

Computational Linguistics and Natural Language Processing Lab Experiment 8

[1]:
```
import nltk
```

[2]:
```
groucho_grammar = nltk.CFG.fromstring("""
S -> NP VP
PP -> P NP
NP -> Det N | Det N PP | 'I'
VP -> V NP | VP PP
Det -> 'an' | 'my'
N -> 'elephant' | 'pajamas'
V -> 'shot'
P -> 'in'
""")
```

[3]:
```
sent = ['I', 'shot', 'an', 'elephant', 'in', 'my', 'pajamas']
parser = nltk.ChartParser(groucho_grammar)
for tree in parser.parse(sent):
    print(tree)
```

```
(S
  (NP I)
  (VP
    (VP (V shot) (NP (Det an) (N elephant)))
    (PP (P in) (NP (Det my) (N pajamas)))))
(S
  (NP I)
  (VP
    (V shot)
    (NP (Det an) (N elephant) (PP (P in) (NP (Det my) (N pajamas))))))
```

[4]:
```
grammar1 = nltk.CFG.fromstring("""
S -> NP VP
VP -> V NP | V NP PP
PP -> P NP
V -> "saw" | "ate" | "walked"
NP -> "John" | "Mary" | "Bob" | Det N | Det N PP
Det -> "a" | "an" | "the" | "my"
N -> "man" | "dog" | "cat" | "telescope" | "park"
P -> "in" | "on" | "by" | "with"
""")
```

[5]:
```
sent = "Mary saw Bob".split()
rd_parser = nltk.RecursiveDescentParser(grammar1)
for tree in rd_parser.parse(sent):
    print(tree)
```

```
(S (NP Mary) (VP (V saw) (NP Bob)))
```

[7]:
```
rd_parser = nltk.RecursiveDescentParser(grammar1)
sent = 'Mary saw a dog'.split()
for tree in rd_parser.parse(sent):
    print(tree)
```

```
(S (NP Mary) (VP (V saw) (NP (Det a) (N dog))))
```

[8]:
```
sr_parser = nltk.ShiftReduceParser(grammar1)
sent = 'Mary saw a dog'.split()
for tree in sr_parser.parse(sent):
    print(tree)
```

```
(S (NP Mary) (VP (V saw) (NP (Det a) (N dog))))
```

[9]:
```
text = ['I', 'shot', 'an', 'elephant', 'in', 'my', 'pajamas']
groucho_grammar.productions(rhs=text[1])
```

[9]:
```
[V -> 'shot']
```

[15]:
```
def init_wfst(tokens, grammar):
    numtokens = len(tokens)
    wfst = [[None for i in range(numtokens+1)] for j in range(numtokens+1)]
    for i in range(numtokens):
        productions = grammar.productions(rhs=tokens[i])
        wfst[i][i+1] = productions[0].lhs()
    return wfst



def complete_wfst(wfst, tokens, grammar, trace=False):
    index = dict((p.rhs(), p.lhs()) for p in grammar.productions())
    numtokens = len(tokens)
    for span in range(2, numtokens+1):
        for start in range(numtokens+1-span):
            end = start + span
            for mid in range(start+1, end):
                nt1, nt2 = wfst[start][mid], wfst[mid][end]
                if nt1 and nt2 and (nt1,nt2) in index:
                    wfst[start][end] = index[(nt1,nt2)]
                    if trace:
                        print("[%s] %3s [%s] %3s [%s] ==> [%s] %3s [%s]" % \
                        (start, nt1, mid, nt2, end, start, index[(nt1,nt2)],␣
↪end))
    return wfst

    def display(wfst, tokens):
        print('\nWFST ' + ' '.join(("%-4d" % i) for i in range(1, len(wfst))))
        for i in range(len(wfst)-1):
            print("%d   " % i, end=" ")
            for j in range(1, len(wfst)):
                print("%-4s" % (wfst[i][j] or '.'), end=" ")
            print()

tokens = "I shot an elephant in my pajamas".split()
wfst0 = init_wfst(tokens, groucho_grammar)
display(wfst0, tokens)
```

```
[[None, NP, None, None, None, None, None, None],
 [None, None, V, None, None, None, None, None],
 [None, None, None, Det, None, None, None, None],
 [None, None, None, None, N, None, None, None],
 [None, None, None, None, None, P, None, None],
 [None, None, None, None, None, None, Det, None],
 [None, None, None, None, None, None, None, N],
 [None, None, None, None, None, None, None, None]]

['I', 'shot', 'an', 'elephant', 'in', 'my', 'pajamas']
```

[16]:
```
wfst1 = complete_wfst(wfst0, tokens, groucho_grammar, trace=True)
```

```
[2] Det [3]   N [4] ==> [2]  NP [4]
[5] Det [6]   N [7] ==> [5]  NP [7]
[1]   V [2]  NP [4] ==> [1]  VP [4]
[4]   P [5]  NP [7] ==> [4]  PP [7]
[0]  NP [1]  VP [4] ==> [0]   S [4]
[1]  VP [4]  PP [7] ==> [1]  VP [7]
[0]  NP [1]  VP [7] ==> [0]   S [7]
```