UNIVERSITÄT WIEN
CSLEARN - EDUCATIONAL TECHNOLOGIES

Natural Language Processing

Exercise Sheet 9

Building Feature Based Grammars

Exercise 1

Take the following grammar:

```
S                      -> NP[AGR=?n] VP[AGR=?n]
NP[AGR=?n]             -> PropN[AGR=?n]
VP[TENSE=?t, AGR=?n] -> Cop[TENSE=?t, AGR=?n] Adj


Cop[TENSE=pres,  AGR=[NUM=sg, PER=3]] -> 'is'
PropN[AGR=[NUM=sg, PER=3]]            -> 'Kim'
Adj                                   -> 'happy'
```

as starting point to correctly parse word sequences like "I am happy" and "she is happy" but not " * you is happy" or " * they am happy".

Exercise 2

Write a Definite Clause Grammar in SWI-Prolog corresponding to Exercise 1, which produces the following output:

```
['I',am,happy]
[np([pro(I)]),vp([cop(am),adj(happy)])]

[she,is,happy]
[np([pro(she)]),vp([cop(is),adj(happy)])]

[you,is,happy]
incorrect sentence

[they,am,happy]
incorrect sentence
```

Exercise 3

Develop a variant of the grammar from Figure 1.1 that uses a feature `COUNT` to make the distinctions shown below:

1. a) the boy sings
   b) * boy sings
2. a) the boys sing
   b) boys sing
3. a) the water is precious
   b) water is precious

Exercise 4

Write a Definite Clause Grammar in SWI-Prolog corresponding to Exercise 3, which produces the following output:

```
[the,boy,sings]
[np([det(the),n(boy)]),vp([iv(sings)])]

[boy,sings]
incorrect sentence

[the,boys,sing]
[np([det(the),n(boys)]),vp([iv(sing)])]

[boys,sing]
[np([n(boys)]),vp([iv(sing)])]

[the,water,is,precious]
[np([det(the),n(water)]),vp([cop(is),adj(precious)])]

[water,is,precious]
[np([n(water)]),vp([cop(is),adj(precious)])]
```

Exercise 5

Extend the German grammar in Figure 3.2 so that it can handle so-called verb-second structures like “heute sieht der Hund die Katze” by using a slash category `S/TV` for the missing transitive verb in “der Hund die Katze”.

Exercise 6

Write a Definite Clause Grammar in SWI-Prolog corresponding to Exercise 5, which produces the following output:

```
[heute,sieht,der,'Hund',die,'Katze']

[adv(heute),s(tv(sieht,objcase=acc,agr=(gnd=masc,per=3,num=sg)),
[np([det(der,case=nom,agr=(gnd=masc,per=3,num=sg)),
n(Hund,case=nom,agr=(gnd=masc,per=3,num=sg))],
case=nom,agr=(gnd=masc,per=3,num=sg)),
vp([tv(),np([det(die,case=acc,agr=(gnd=fem,per=3,num=sg)),
n(Katze,case=acc,agr=(gnd=fem,per=3,num=sg))],
case=acc,agr=(gnd=fem,per=3,num=sg))],
agr=(gnd=masc,per=3,num=sg))])]
```

Exercise 7

Consider the patterns of grammaticality for the verbs "loaded", "filled", and "dumped" below. Write grammar productions to handle such data:

1. a) the farmer loaded the cart with sand
   b) the farmer loaded sand into the cart
2. a) the farmer filled the cart with sand
   b) * the farmer filled sand into the cart
3. a) * the farmer dumped the cart with sand
   b) the farmer dumped sand into the cart

Exercise 8

Write a Definite Clause Grammar in SWI-Prolog corresponding to Exercise 7, which produces the following output:

```
the farmer loaded the cart with sand
[np([det(the),n(farmer)]),vp([v(loaded,with),np([det(the),n(cart)]),pp([p(with),np([n(sand)])],with)])]

the farmer loaded sand into the cart
[np([det(the),n(farmer)]),vp([v(loaded,into),np([n(sand)]),pp([p(into),np([det(the),n(cart)])],into)])]

the farmer filled the cart with sand
[np([det(the),n(farmer)]),vp([v(filled,with),np([det(the),n(cart)]),pp([p(with),np([n(sand)])],with)])]

the farmer filled sand into the cart
incorrect sentence

the farmer dumped the cart with sand
incorrect sentence

the farmer dumped sand into the cart
[np([det(the),n(farmer)]),vp([v(dumped,into),np([n(sand)]),pp([p(into),np([det(the),n(cart)])],into)])]
```

Exercise 9

Consider the feature structures shown in Figure 6.1:

```
fs1 = nltk.FeatStruct("[A = ?x, B= [C = ?x]]")
fs2 = nltk.FeatStruct("[B = [D = d]]")
fs3 = nltk.FeatStruct("[B = [C = d]]")
fs4 = nltk.FeatStruct("[A = (1)[B = b], C->(1)]")
fs5 = nltk.FeatStruct("[A = (1)[D = ?x], C = [E -> (1), F = ?x] ]")
fs6 = nltk.FeatStruct("[A = [D = d]]")
fs7 = nltk.FeatStruct("[A = [D = d], C = [F = [D = d]]]")
fs8 = nltk.FeatStruct("[A = (1)[D = ?x, G = ?x], C = [B = ?x, E -> (1)] ]")
fs9 = nltk.FeatStruct("[A = [B = b], C = [E = [G = e]]]")
fs10 = nltk.FeatStruct("[A = (1)[B = b], C -> (1)]")
```

Work out on paper what the result is of the following unifications. (Hint: you might find it useful to draw the graph structures.)

1. `fs1` and `fs2`,
2. `fs1` and `fs3`,
3. `fs4` and `fs5`,
4. `fs5` and `fs6`,
5. `fs5` and `fs7`,
6. `fs8` and `fs9`,
7. `fs8` and `fs10`.

Check your answers using Python.