Assignment 1

Mariya Lazarova
DD2418 Language Engineering
KTH

April 3, 2020

1 CFG trees

Find all the trees derivable from the sentence *He built the shed with a hammer in the yard behind the house*
In the first interpretation of this sentence, which is shown in figure 1, he builds a shed, the shed is with a hammer, the hammer is in the yard, and the yard in behind the house. So there are more than 1 shed, more than 1 hammers, more than 1 yards, and 1 house, and he builds a particular shed, which has a particular hammer, and this hammer is the one in the particular yard, which is behind the house.

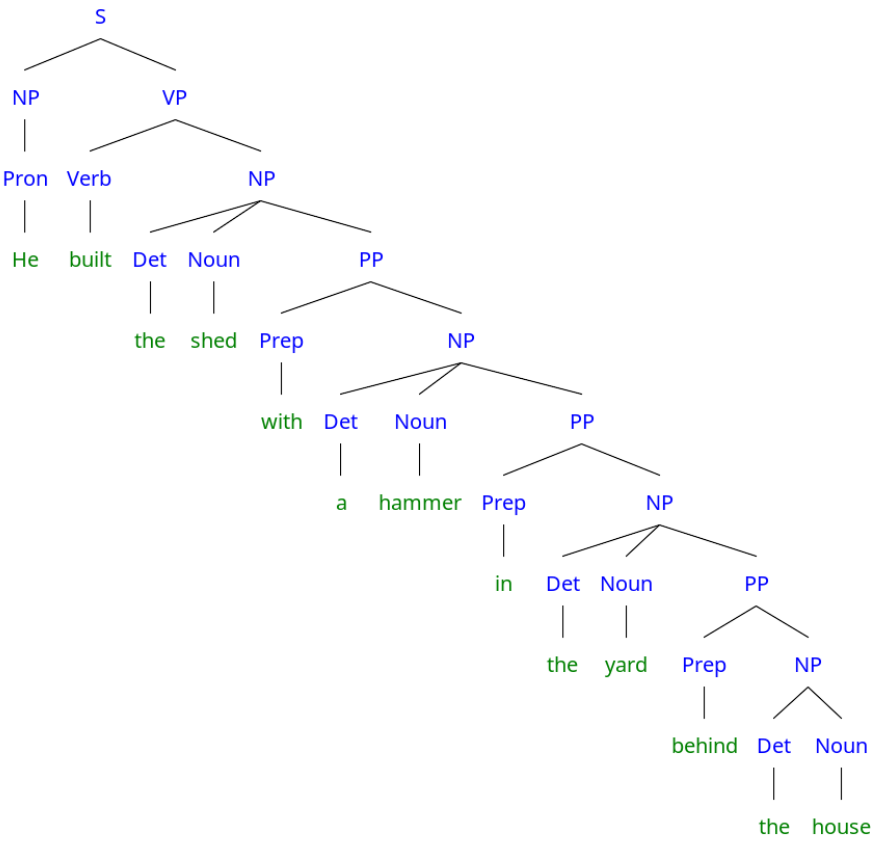


Figure 1: Tree 1

In the second interpretation of this sentence, which is shown in figure 2, he builds a shed and he uses hammer to do that, the hammer is in the yard, and the yard in behind the house. So there are more than 1 hammers, more than 1 yards, 1 shed, and 1 house, and he builds the shed, and to do this, he uses the particular hammer that is in the particular yard, that is behind the house.

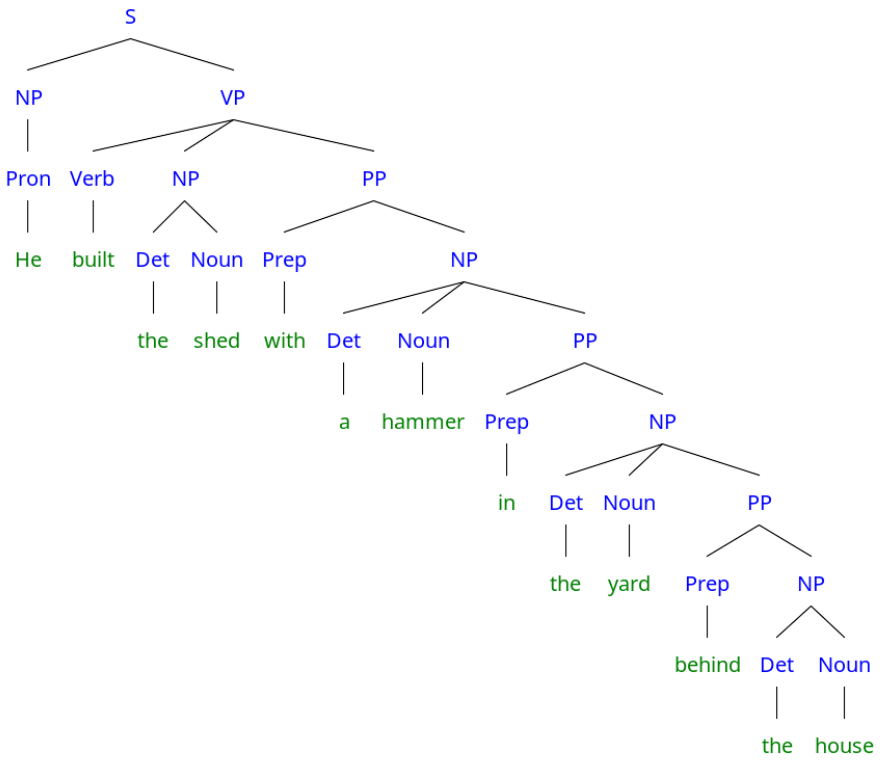


Figure 2: Tree 2

In the third interpretation of this sentence, which is shown in figure 3, he builds the shed, in which there is a hammer, and he does that in the yard behind the house. There are more than 1 sheds, more than 1 yards, 1 house, and 1 hammer, and he built the shed that has a hammer, and he has done this in the yard that is behind the house.

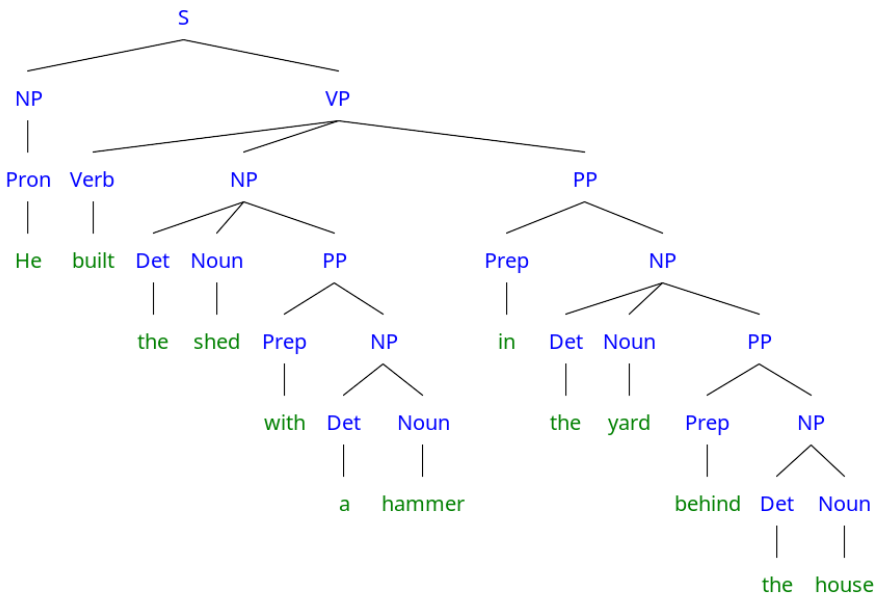


Figure 3: Tree 3

In the fourth interpretation of this sentence, which is shown in figure 4, he builds the shed, in which there is a hammer, and this hammer is in the yard, and he does the building behind the house. There are more than 1 sheds, more than 1 hammers, 1 house, and 1 yard, and he built the shed that has the particular hammer that is in the yard, and he does this behind the house.

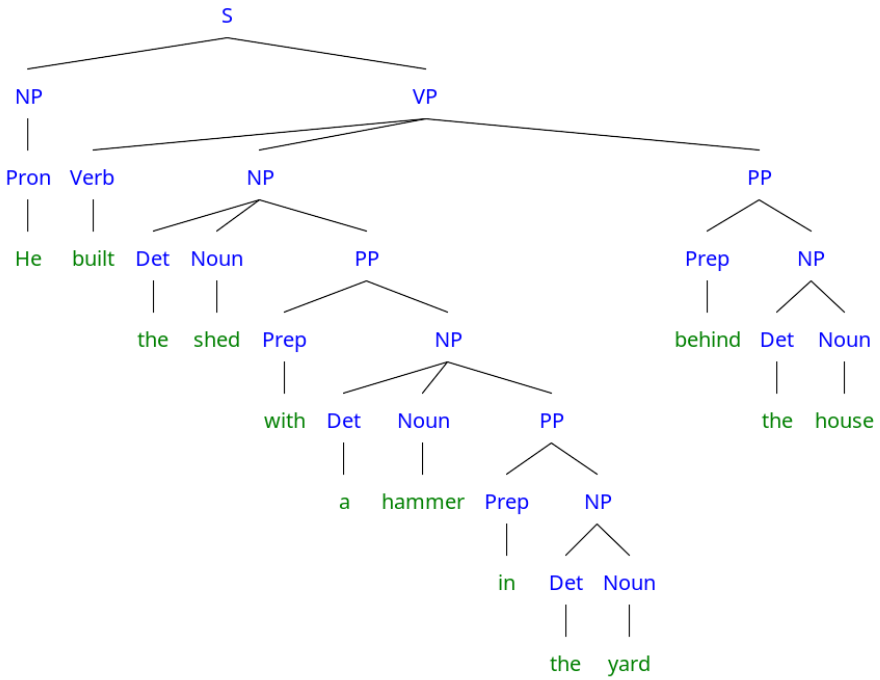


Figure 4: Tree 4

2 Chomsky Normal Form and CKY by hand

a

S → NP, VP
S → VP, NP
NP → Noun, Noun
NP → Noun, PP
NP → Det, Noun
NP → time
NP → flies
NP → arrow
VP → Verb NP
VP → Verb, PP
VP → time
VP → flies
VP → like
PP → Prep, NP
Noun → time
Noun → flies
Noun → arrow
Verb → time
Verb → flies
Verb → like
Prep → like
Det → an

b

time	flies	like	an	arrow
NP, Noun, VP, Verb	S, S, NP, VP	S		S, VP S, S
	NP, Noun, VP, Verb	S		S, VP NP
		VP, Verb, Prep		VP, PP, S
			Det	NP
				NP, Noun

Table 1: Result of CKY algorithm for the sentence *time flies like an arrow*

c

There 3 sentences and one VP. I can see that from the to right box in table **??**. To retrieve the parse trees, I have to backtrack to the boxes on the diagonal.

d

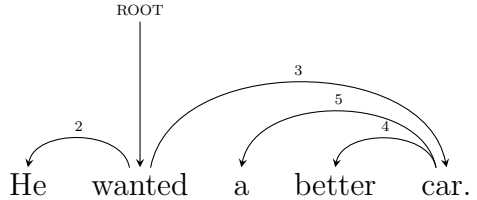
Here are the trees that start from S.
S(NP(time), VP(Verb(flies), PP(Prep(like), NP(Det(an), Noun(arrow)))))
S(VP(time), NP(Noun(flies), PP(Prep(like), NP(Det(an), Noun(arrow)))))
S(NP(Noun(time), Noun(flies)), VP(Verb(like), NP(Det(an), Noun(arrow))))

There is also one tree that is rooted at VP¿ VP(Verb(time), NP(Noun(flies), PP(Prep(like), NP(Det(an), Noun(arrow)))))

3 Dependency parsing mistakes

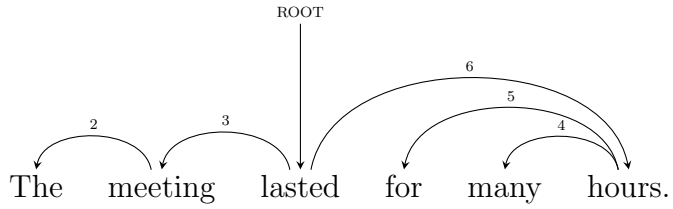
a

The wrong label is number 3. The article *a* is a determiner of *car*. The correct dependency parsing is below:



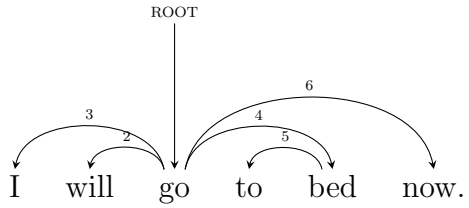
b

The wrong label is number 2.The article *the* is a determiner to *meeting*. The correct dependency parsing is below:



c

The wrong label is number 6. The arrow to *now* has to come from *go*. The correct dependency parsing is below:

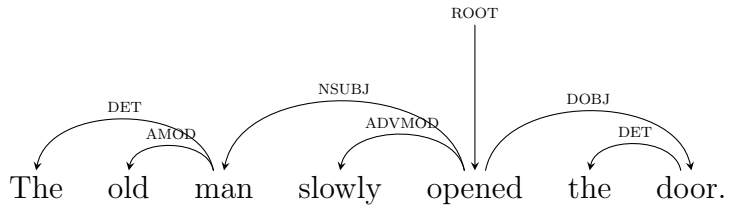


4 Dependency parsing labels

The labels for the number inputs are:

1. ROOT
2. NSUBJ
3. AMOD
4. DET
5. ADVMOD
6. DOBJ
7. DET

The dependency tree looks like this:



5 Dependency parsing

I have completed the `valid_moves`, `move`, and `compute_correct_move`. To verify that I outputted that result from my implementation into a text file `mimi_output.conllu` and did diff with the provided file `correct_moves_en-ud-dev.conllu`. It showed no differences between the two files. I am attaching my output, as well as my code.

6 CKY parsing

a

For this part, I implemented the `parse`. Result from my implementation - CKY table, using `grammar.txt` and parsing *giant cuts in welfare*:

giant	cuts	in	welfare
['NP', 'JJ']	['NP']	[]	['S', 'NP', 'NP']
[]	['NP', 'Verb']	[]	['NP', 'VP']
[]	[]	['Prep']	['PP']
[]	[]	[]	['NP']

With the second provided grammar, `grammar2.txt` and parsing the sentence *time flies like and arrow*, the results are as follows:

time	flies	like	an	arrow
['Noun', 'NP', 'Verb', 'VP']	['NP', 'S', 'VP', 'S']	['S']	[]	['S', 'VP', 'S', 'S']
[]	['Noun', 'NP', 'Verb', 'VP']	['S']	[]	['NP', 'S', 'VP']
[]	[]	['Verb', 'VP', 'Prep']	[]	['VP', 'S', 'PP']
[]	[]	[]	['Det']	['NP']
[]	[]	[]	[]	['Noun', 'NP']

b

I also implemented the function `print_tree`. I also changed the function `parse` to fill in the variable `backtrack` to keep track of the parse trees `print_table` to print the backtracking table. Running using `grammar.txt` and parsing *giant cuts in welfare*, the backtracking tree looks like that:

Backrtracking table

giant	cuts	in	welfare
[]	[(('JJ', 0, 0), ('NP', 1, 1))]	[]	[(('NP', 0, 0), ('VP', 1, 3)), (('JJ', 0, 0), ('NP', 1, 3)), (('NP', 0, 1), ('PP', 2, 3))]
[]	[]	[]	[(('NP', 1, 1), ('PP', 2, 3)), (('Verb', 1, 1), ('PP', 2, 3))]
[]	[]	[]	[(('Prep', 2, 2), ('NP', 3, 3))]
[]	[]	[]	[]

What this table shows is, at each cell of the table, there is a list with the same size as the corresponding resulting table from the CKY algorithm. For example, the NP in cell (0, 1) of the parse table for `giant cuts in welfare` (indexing starts from 0), backtracks to (('JJ', 0, 0), ('NP', 1, 1)), which is the first and only element in in cell (0, 1) of backtracking table.

This means that here the use rule is NP $\rightarrow$ JJ, NP, the JJ comes from cell (0, 0), and the NP on the right-hand side comes from cell (0, 0).
The resulting parse trees for grammar `grammar.txt` and sentence *giant cuts in welfare* are:
S(NP(giant), VP(Verb(cuts), PP(Prep(in), NP(welfare))))

I also added option to print all possible parse tree, regardless of whether they are rooted at S or at something else. The other parse trees are:
NP(JJ(giant), NP(NP(cuts), PP(Prep(in), NP(welfare))))
NP(NP(JJ(giant), NP(cuts)), PP(Prep(in), NP(welfare)))

The resulting parse trees for grammar `grammar2.txt` and sentence *time flies like an arrow* are:

VP(Verb(time), NP(Noun(flies), PP(Prep(like), NP(Det(an), Noun(arrow)))))
S(NP(time), VP(Verb(flies), PP(Prep(like), NP(Det(an), Noun(arrow)))))
S(VP(time), NP(Noun(flies), PP(Prep(like), NP(Det(an), Noun(arrow)))))
S(NP(Noun(time), Noun(flies)), VP(Verb(like), NP(Det(an), Noun(arrow))))

The code for this will be attached.