**Lab 6: CFG & Parse Tree**

**References:**

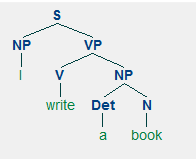
1. Natural Language Processing with Python, by Steven Bird, Ewan Klein and Edward Loper, 2014.

Quick Review

CFG has been the most influential grammar formalism for describing language syntax. This is not because CFG has been generally adopted as such for linguistic description, but rather because most grammar formalisms are derived from or can somehow be related to CFG. For this reason, CFG is often used as a base formalism when parsing algorithms are described.

The standard way to represent the syntactic structure of a grammatical sentence is as a syntax tree, or a parse tree, which is a representation of all the steps in the derivation of the sentence from the root node. This means that each internal node in the tree represents an application of a grammar rule.

Practices

**Parse Tree 01**

import nltk

text2 = nltk.CFG.fromstring("""

S -> NP VP

PP -> P NP

NP -> Det N | PP NP | Det N PP | 'I'

VP -> V NP | VP PP | V

Det -> 'a'

N -> 'book'

V -> 'write'

""")

text1 = nltk.tokenize.word\_tokenize("I write a book")

print(text1)

parser = nltk.ChartParser(text2)

for tree in parser.parse(text1):

print(tree)

tree.draw()

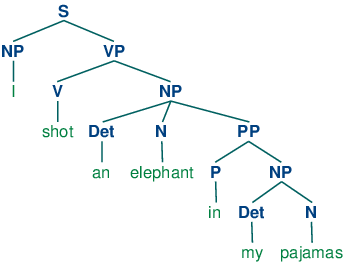
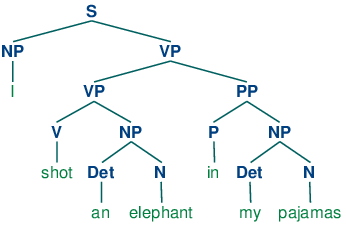
**Output**

['I', 'write', 'a', 'book']

(S (NP I) (VP (V write) (NP (Det a) (N book))))

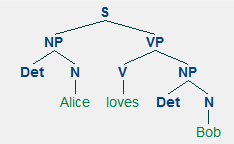
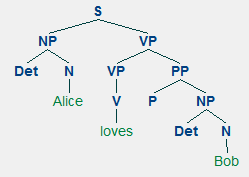
**Parse Tree 02**

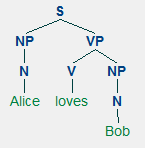
**import** nltk  
groucho\_grammar = nltk.CFG.fromstring(**"""  
S -> NP VP   
PP -> P NP   
NP -> Det N | Det N PP | 'I'   
VP -> V NP | VP PP   
Det -> 'an' | 'my'   
N -> 'elephant' | 'pajamas'   
V -> 'shot'   
P -> 'in'   
"""**)  
  
sent = [**'I'**, **'shot'**, **'an'**, **'elephant'**, **'in'**, **'my'**, **'pajamas'**]  
parser = nltk.ChartParser(groucho\_grammar)  
**for** tree **in** parser.parse(sent):  
 tree.draw()  
print(tree)



**Parse Tree 03**

**import** nltk  
  
text2 = nltk.CFG.fromstring(**"""  
S -> NP VP   
PP -> P NP   
NP -> Det N | PP NP | Det N PP  
VP -> V NP | VP PP | V   
N -> 'Alice' | 'Bob'  
V -> 'loves'  
Det ->   
P ->   
"""**)  
text1 = nltk.tokenize.word\_tokenize(**"Alice loves Bob"**)  
print(text1)  
print()  
parser = nltk.ChartParser(text2)  
**for** tree **in** parser.parse(text1):  
 print(tree)  
tree.draw()



**Parse Tree 04 – Adjective Phrase**

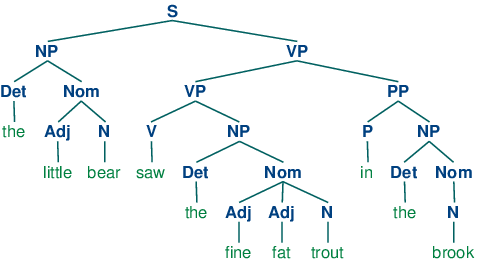
The little bear saw the fine fat trout in the brook

**Clue:**

**NP 🡪 DT Nom**

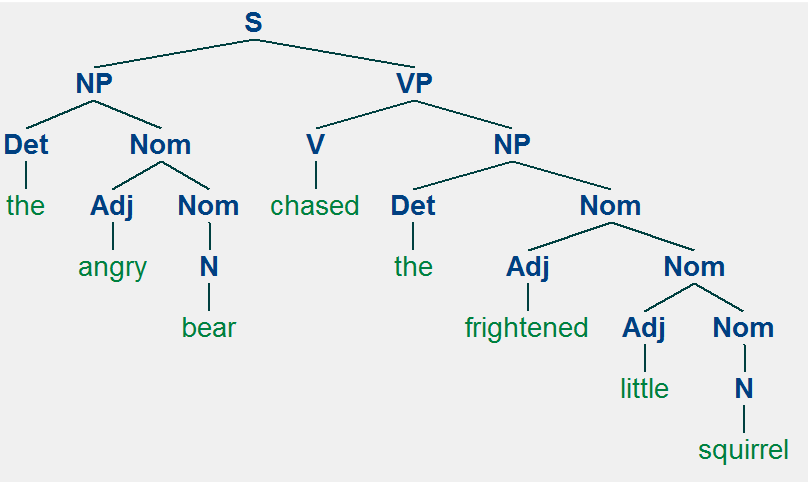
**Nom 🡪 Adj N | Adj Adj N**

**import** nltk  
text2 = nltk.CFG.fromstring(**"""  
S -> NP VP   
PP -> P NP   
NP -> Det N | Det N PP | Det Nom | 'the'  
VP -> V NP | VP PP   
Nom -> Adj N | Adj Adj N  
Det -> 'the'  
N -> 'bear' | 'trout' | 'brook'  
V -> 'saw'   
P -> 'in'   
Adj -> 'little' | 'fine' | 'fat'  
"""**)  
  
text1 = nltk.tokenize.word\_tokenize(**"the little bear saw the fine fat trout in the brook"**)  
print(text1)  
print()  
parser = nltk.ChartParser(text2)  
**for** tree1 **in** parser.parse(text1):  
 tree1.draw()  
print(tree1)



**Parse Tree 05 – Adjective Phrase**

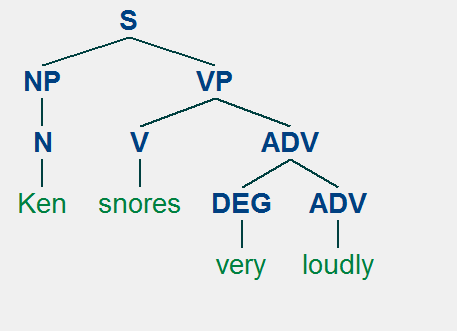
**import** nltk  
grammar2 = nltk.CFG.fromstring(**"""  
 S -> NP VP  
 NP -> Det Nom | Det N | PropN  
 Nom -> Adj Nom | N  
 VP -> V Adj | V NP | V S | V NP PP  
 PP -> P NP  
 PropN -> 'Buster' | 'Chatterer' | 'Joe'  
 Det -> 'the' | 'a'  
 N -> 'bear' | 'squirrel' | 'tree' | 'fish' | 'log'  
 Adj -> 'angry' | 'frightened' | 'little' | 'tall'  
 V -> 'chased' | 'saw' | 'said' | 'thought' | 'was' | 'put'  
 P -> 'on'  
 """**)  
  
sent = [**'the'**, **'angry'**, **'bear'**, **'chased'**, **'the'**, **'frightened'**, **'little'**, **'squirrel'**]  
parser = nltk.ChartParser(grammar2)  
**for** tree **in** parser.parse(sent):  
 tree.draw()  
print(tree)



**Parse Tree 06 – Adverb Phrases (AdvP)**

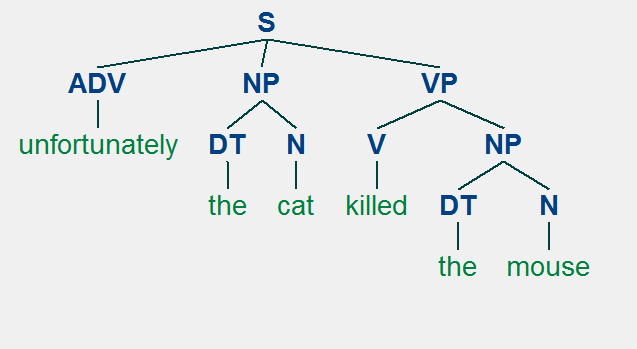
E.g.: Ken snores very loudly

**import** nltk  
  
sentence = **"Ken snores very loudly"**gram = nltk.CFG.fromstring(**"""  
S -> NP VP  
NP -> N  
VP -> V ADV  
N -> 'Ken'  
V -> 'snores'  
DEG -> 'very'  
ADV -> DEG ADV | 'loudly'  
"""**)  
  
token = nltk.tokenize.word\_tokenize(sentence)  
print(token)  
parser = nltk.ChartParser(gram)  
**for** tree **in** parser.parse(token):  
 print(tree)  
tree.draw()

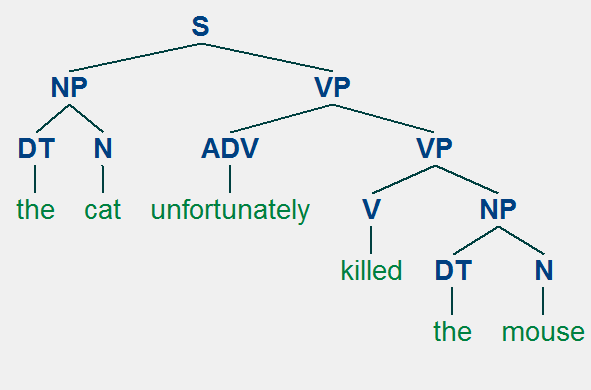


**import** nltk  
**from** nltk.tokenize **import** word\_tokenize  
  
sents = [  
 **"unfortunately the cat killed the mouse"**,  
 **"the cat unfortunately killed the mouse"**,  
 **"the cat killed the mouse unfortunately"**]  
  
grammar = nltk.CFG.fromstring(**"""  
S -> ADV NP VP | NP VP  
NP -> DT N  
VP -> ADV VP | VP ADV | V NP  
DT -> 'the'  
N -> 'cat' | 'mouse'  
V -> 'killed'  
ADV -> 'unfortunately'  
"""**)  
  
parser = nltk.ChartParser(grammar)  
  
**for** sent **in** sents:  
 print(sent)  
 **for** tree **in** parser.parse(word\_tokenize(sent)):  
 tree.draw()  
 print(tree)

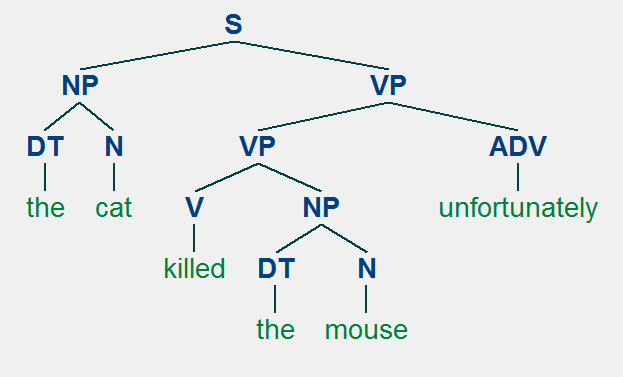
**Unfortunately the cat killed the mouse**



**The cat unfortunately killed the mouse**



**The cat killed the mouse unfortunately**



**Draw Parse Tree using COLAB**

**Run the following set of codes to set the COLAB platform**

import nltk

nltk.download('punkt')

### CREATE VIRTUAL DISPLAY ###

!apt-get install -y xvfb # Install X Virtual Frame Buffer

import os

os.system('Xvfb :1 -screen 0 1600x1200x16  &')    # create virtual display with size 1600x1200 and 16 bit color. Color can be changed to 24 or 8

os.environ['DISPLAY']=':1.0'    # tell X clients to use our virtual DISPLAY :1.0.

%matplotlib inline

### INSTALL GHOSTSCRIPT (Required to display NLTK trees) ###

!apt install ghostscript python3-tk

**Example Program ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~**

import nltk

from IPython.display import display

text2 = nltk.CFG.fromstring("""

S -> NP VP

PP -> P NP

NP -> Det N | PP NP | Det N PP | 'I'

VP -> V NP | VP PP | V

Det -> 'a'

N -> 'book'

V -> 'write'

""")

text1 = nltk.tokenize.word\_tokenize("I write a book")

print(text1)

parser = nltk.ChartParser(text2)

for tree in parser.parse(text1):

  display(tree) # tree.draw()

# print(tree)