lab12_nlp_viviyan_33

June 8, 2021

0.1 Lab12. Building and Parsing Context Free Grammars

[1]:
```
import nltk
nltk.download("punkt")
from nltk.tree import Tree
from nltk.tokenize import word_tokenize
from IPython.display import display
import nltk,re,pprint
from nltk.tag import pos_tag
from nltk.chunk import ne_chunk
import numpy as npt

!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
```

```
[nltk_data] Downloading package punkt to /root/nltk_data…
[nltk_data]   Unzipping tokenizers/punkt.zip.
Reading package lists… Done
Building dependency tree
Reading state information… Done
The following package was automatically installed and is no longer required:
  libnvidia-common-460
Use 'apt autoremove' to remove it.
The following NEW packages will be installed:
  xvfb
0 upgraded, 1 newly installed, 0 to remove and 34 not upgraded.
Need to get 784 kB of archives.
After this operation, 2,270 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu bionic-updates/universe amd64 xvfb amd64
2:1.19.6-1ubuntu4.9 [784 kB]
Fetched 784 kB in 1s (958 kB/s)
Selecting previously unselected package xvfb.
```

```
(Reading database … 160706 files and directories currently installed.)
Preparing to unpack …/xvfb_2%3a1.19.6-1ubuntu4.9_amd64.deb …
Unpacking xvfb (2:1.19.6-1ubuntu4.9) …
Setting up xvfb (2:1.19.6-1ubuntu4.9) …
Processing triggers for man-db (2.8.3-2ubuntu0.1) …
Reading package lists… Done
Building dependency tree
Reading state information… Done
python3-tk is already the newest version (3.6.9-1~18.04).
The following package was automatically installed and is no longer required:
  libnvidia-common-460
Use 'apt autoremove' to remove it.
The following additional packages will be installed:
  fonts-droid-fallback fonts-noto-mono gsfonts libcupsfilters1 libcupsimage2
  libgs9 libgs9-common libijs-0.35 libjbig2dec0 poppler-data
Suggested packages:
  fonts-noto ghostscript-x poppler-utils fonts-japanese-mincho
  | fonts-ipafont-mincho fonts-japanese-gothic | fonts-ipafont-gothic
  fonts-arphic-ukai fonts-arphic-uming fonts-nanum
The following NEW packages will be installed:
  fonts-droid-fallback fonts-noto-mono ghostscript gsfonts libcupsfilters1
  libcupsimage2 libgs9 libgs9-common libijs-0.35 libjbig2dec0 poppler-data
0 upgraded, 11 newly installed, 0 to remove and 34 not upgraded.
Need to get 14.1 MB of archives.
After this operation, 49.9 MB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu bionic/main amd64 fonts-droid-fallback
all 1:6.0.1r16-1.1 [1,805 kB]
Get:2 http://archive.ubuntu.com/ubuntu bionic/main amd64 poppler-data all
0.4.8-2 [1,479 kB]
Get:3 http://archive.ubuntu.com/ubuntu bionic/main amd64 fonts-noto-mono all
20171026-2 [75.5 kB]
Get:4 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libcupsimage2
amd64 2.2.7-1ubuntu2.8 [18.6 kB]
Get:5 http://archive.ubuntu.com/ubuntu bionic/main amd64 libijs-0.35 amd64
0.35-13 [15.5 kB]
Get:6 http://archive.ubuntu.com/ubuntu bionic/main amd64 libjbig2dec0 amd64
0.13-6 [55.9 kB]
Get:7 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libgs9-common
all 9.26~dfsg+0-0ubuntu0.18.04.14 [5,092 kB]
Get:8 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libgs9 amd64
9.26~dfsg+0-0ubuntu0.18.04.14 [2,265 kB]
Get:9 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 ghostscript
amd64 9.26~dfsg+0-0ubuntu0.18.04.14 [51.3 kB]
Get:10 http://archive.ubuntu.com/ubuntu bionic/main amd64 gsfonts all
1:8.11+urwcyr1.0.7~pre44-4.4 [3,120 kB]
Get:11 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64
libcupsfilters1 amd64 1.20.2-0ubuntu3.1 [108 kB]
Fetched 14.1 MB in 1s (10.5 MB/s)
```

```
Selecting previously unselected package fonts-droid-fallback.
(Reading database … 160713 files and directories currently installed.)
Preparing to unpack …/00-fonts-droid-fallback_1%3a6.0.1r16-1.1_all.deb …
Unpacking fonts-droid-fallback (1:6.0.1r16-1.1) …
Selecting previously unselected package poppler-data.
Preparing to unpack …/01-poppler-data_0.4.8-2_all.deb …
Unpacking poppler-data (0.4.8-2) …
Selecting previously unselected package fonts-noto-mono.
Preparing to unpack …/02-fonts-noto-mono_20171026-2_all.deb …
Unpacking fonts-noto-mono (20171026-2) …
Selecting previously unselected package libcupsimage2:amd64.
Preparing to unpack …/03-libcupsimage2_2.2.7-1ubuntu2.8_amd64.deb …
Unpacking libcupsimage2:amd64 (2.2.7-1ubuntu2.8) …
Selecting previously unselected package libijs-0.35:amd64.
Preparing to unpack …/04-libijs-0.35_0.35-13_amd64.deb …
Unpacking libijs-0.35:amd64 (0.35-13) …
Selecting previously unselected package libjbig2dec0:amd64.
Preparing to unpack …/05-libjbig2dec0_0.13-6_amd64.deb …
Unpacking libjbig2dec0:amd64 (0.13-6) …
Selecting previously unselected package libgs9-common.
Preparing to unpack …/06-libgs9-common_9.26~dfsg+0-0ubuntu0.18.04.14_all.deb
…
Unpacking libgs9-common (9.26~dfsg+0-0ubuntu0.18.04.14) …
Selecting previously unselected package libgs9:amd64.
Preparing to unpack …/07-libgs9_9.26~dfsg+0-0ubuntu0.18.04.14_amd64.deb …
Unpacking libgs9:amd64 (9.26~dfsg+0-0ubuntu0.18.04.14) …
Selecting previously unselected package ghostscript.
Preparing to unpack …/08-ghostscript_9.26~dfsg+0-0ubuntu0.18.04.14_amd64.deb
…
Unpacking ghostscript (9.26~dfsg+0-0ubuntu0.18.04.14) …
Selecting previously unselected package gsfonts.
Preparing to unpack …/09-gsfonts_1%3a8.11+urwcyr1.0.7~pre44-4.4_all.deb …
Unpacking gsfonts (1:8.11+urwcyr1.0.7~pre44-4.4) …
Selecting previously unselected package libcupsfilters1:amd64.
Preparing to unpack …/10-libcupsfilters1_1.20.2-0ubuntu3.1_amd64.deb …
Unpacking libcupsfilters1:amd64 (1.20.2-0ubuntu3.1) …
Setting up libgs9-common (9.26~dfsg+0-0ubuntu0.18.04.14) …
Setting up fonts-droid-fallback (1:6.0.1r16-1.1) …
Setting up gsfonts (1:8.11+urwcyr1.0.7~pre44-4.4) …
Setting up poppler-data (0.4.8-2) …
Setting up fonts-noto-mono (20171026-2) …
Setting up libcupsfilters1:amd64 (1.20.2-0ubuntu3.1) …
Setting up libcupsimage2:amd64 (2.2.7-1ubuntu2.8) …
Setting up libjbig2dec0:amd64 (0.13-6) …
Setting up libijs-0.35:amd64 (0.35-13) …
Setting up libgs9:amd64 (9.26~dfsg+0-0ubuntu0.18.04.14) …
Setting up ghostscript (9.26~dfsg+0-0ubuntu0.18.04.14) …
Processing triggers for man-db (2.8.3-2ubuntu0.1) …
```

```
Processing triggers for fontconfig (2.12.6-0ubuntu2) …
Processing triggers for libc-bin (2.27-3ubuntu1.2) …
/sbin/ldconfig.real: /usr/local/lib/python3.7/dist-
packages/ideep4py/lib/libmkldnn.so.0 is not a symbolic link
```

0.1.1 EXERCISE-1: Build Grammar and Parser

[2]:
```
Grammar_1 = nltk.CFG.fromstring("""
S -> NP VP | NP VP
NP -> N | Det N | PRO | N N
VP -> V NP CP | VP ADVP | V NP
ADVP -> ADV ADV
CP -> COMP S
N -> 'Lisa' | 'brother' | 'peanut' | 'butter'
V -> 'told' | 'liked'
COMP -> 'that'
Det -> 'her'
PRO -> 'she'
ADV -> 'very' | 'much'

S -> NP VP
NP -> NP CONJ NP | N | NP PP | Det N | N | Det N
VP -> VP PP | VP CONJ VP | V | V
PP -> P NP | P NP
N -> 'Homer' | 'friends' | 'work' | 'bar'
V -> 'drank' | 'sang'
CONJ -> 'and' | 'and'
Det -> 'his' | 'the'
P -> 'from' | 'in'

S -> NP VP
NP -> NP CONJ NP | N | N
VP -> V ADJP
ADJP -> ADJP CONJ ADJP | ADJ | ADV ADJ
N -> 'Homer' | 'Marge'
V -> 'are'
CONJ -> 'and' | 'but'
ADJ -> 'poor' | 'happy'
ADV -> 'very'

S -> NP VP | NP AUX VP
NP -> PRO | NP CP | Det N | PRO | PRO | PRO | N |Det N
VP -> V NP PP | V NP NP
CP -> COMP S
PP -> P NP
Det -> 'the' | 'his'
```

```
PRO -> 'he' | 'I' | 'him'
N -> 'book' | 't' | 'sister'
V -> 'gave' | 'given'
COMP -> 'that'
AUX -> 'had'
P -> 'to'

S -> NP VP
NP -> PRO | Det N | Det N
VP -> V NP PP
PP -> P NP
Det -> 'the' | 'his'
PRO -> 'he'
N -> 'book' | 'sister'
V -> 'gave'
P -> 'to'

S -> NP VP
NP -> Det ADJ N | Det ADJ ADJ N | N
VP -> V NP|VP PP
PP -> P NP
Det -> 'the' | 'the'
ADJ -> 'big' | 'tiny' | 'nerdy'
N -> 'bully' | 'kid' | 'school'
V -> 'punched'
P -> 'after'
""")
```

1.Using NLTK's nltk.CFG.fromstring() method, build a CFG named grammar1. The grammar should cover all of the sentences below and their tree structure as presented on this page. The grammar's start symbol should be 'S': make sure that an S rule (ex. S -> NP VP) is the very top rule in your list of rules.

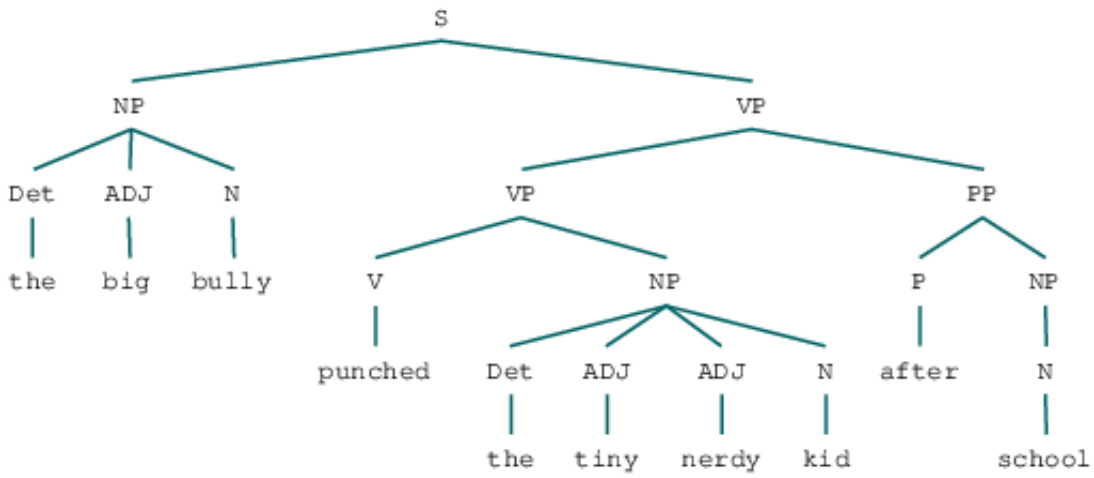
0.1.2 (s6)the big bully punched the tiny nerdy kid after school

[16]:
```
s6_grammar1 = nltk.CFG.fromstring("""
S -> NP VP
NP -> Det ADJ N | Det ADJ ADJ N | N
VP -> V NP|VP PP
PP -> P NP
Det -> 'the' | 'the'
ADJ -> 'big' | 'tiny' | 'nerdy'
N -> 'bully' | 'kid' | 'school'
V -> 'punched'
P -> 'after'
""")
```

[17]:
```
sentence6 = word_tokenize("the big bully punched the tiny nerdy kid after␣
 ↪school")
parser = nltk.ChartParser(s6_grammar1)
for tree in parser.parse(sentence6):
    print(tree)
```

```
(S
  (NP (Det the) (ADJ big) (N bully))
  (VP
    (VP (V punched) (NP (Det the) (ADJ tiny) (ADJ nerdy) (N kid)))
    (PP (P after) (NP (N school)))))
```

[18]:
```
np6 =nltk.Tree.fromstring('(S(NP (Det the) (ADJ big) (N bully))(VP(VP (V␣
 ↪punched) (NP (Det the) (ADJ tiny) (ADJ nerdy) (N kid)))(PP (P after) (NP (N␣
 ↪school)))))')
display(np1)
```



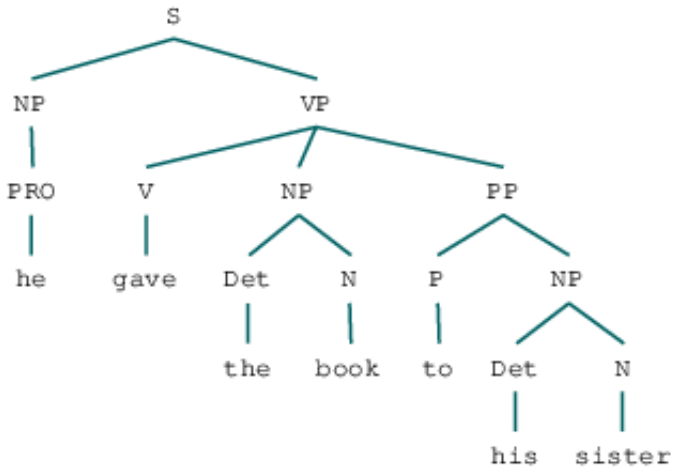
0.1.3 (s7)he gave the book to his sister

[19]:
```
s7_grammar1 = nltk.CFG.fromstring("""
S -> NP VP
NP -> PRO | Det N | Det N
VP -> V NP PP
PP -> P NP
Det -> 'the' | 'his'
PRO -> 'he'
N -> 'book' | 'sister'
V -> 'gave'
P -> 'to'
""")
```

[20]:
```
sentence7 = word_tokenize("he gave the book to his sister")
parser = nltk.ChartParser(s7_grammar1)
for i in parser.parse(sentence7):
    print(i)
```

```
(S
  (NP (PRO he))
  (VP
    (V gave)
    (NP (Det the) (N book))
    (PP (P to) (NP (Det his) (N sister)))))
```

[21]:
```
np7 =nltk.Tree.fromstring('(S(NP (PRO he))(VP(V gave)(NP (Det the) (N book))(PP␣
↪(P to) (NP (Det his) (N sister)))))')
display(np7)
```



0.1.4 (s8)he gave the book that I had given him t to his sister

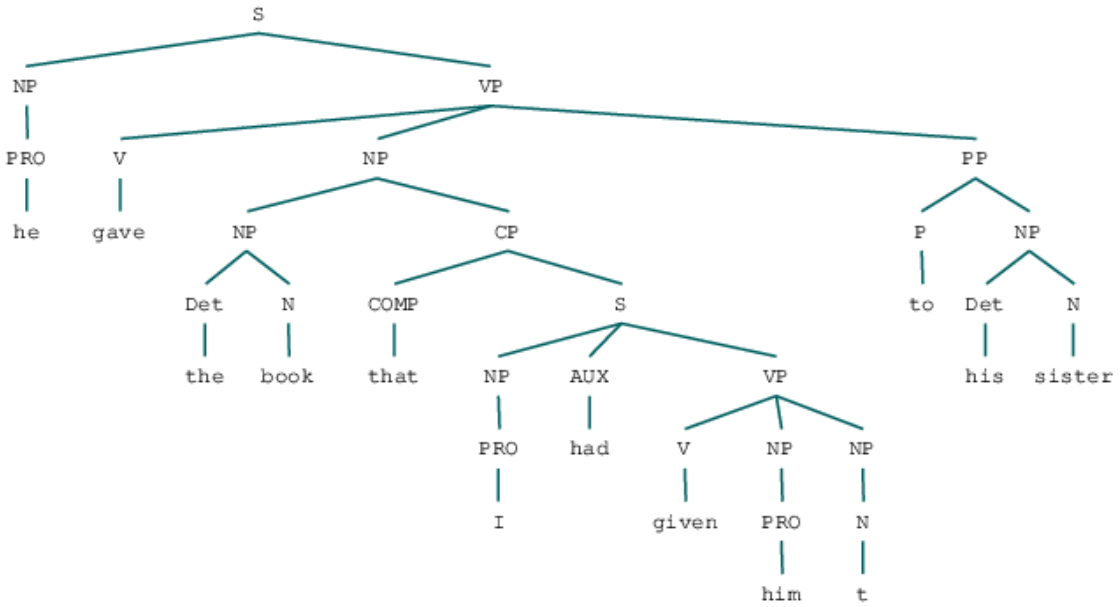
[22]:
```
s8_grammar1 = nltk.CFG.fromstring("""
S -> NP VP | NP AUX VP
NP -> PRO | NP CP | Det N | PRO | PRO | PRO | N |Det N
VP -> V NP PP | V NP NP
CP -> COMP S
PP -> P NP
Det -> 'the' | 'his'
PRO -> 'he' | 'I' | 'him'
N -> 'book' | 't' | 'sister'
V -> 'gave' | 'given'
COMP -> 'that'
AUX -> 'had'
```

```
P -> 'to'
""")
```

[23]:
```
sentence8 = word_tokenize("he gave the book that I had given him t to his␣
 ↪sister")
parser = nltk.ChartParser(s8_grammar1)
for i in parser.parse(sentence8):
    print(i)
```

```
(S
  (NP (PRO he))
  (VP
    (V gave)
    (NP
      (NP (Det the) (N book))
      (CP
        (COMP that)
        (S
          (NP (PRO I))
          (AUX had)
          (VP (V given) (NP (PRO him)) (NP (N t))))))
    (PP (P to) (NP (Det his) (N sister)))))
```

[24]:
```
np8 =nltk.Tree.fromstring('(S(NP (PRO he))(VP(V gave)(NP(NP (Det the) (N␣
 ↪book))(CP(COMP that)(S(NP (PRO I))(AUX had)(VP (V given) (NP (PRO him)) (NP␣
 ↪(N t))))))(PP (P to) (NP (Det his) (N sister)))))')
display(np8)
```



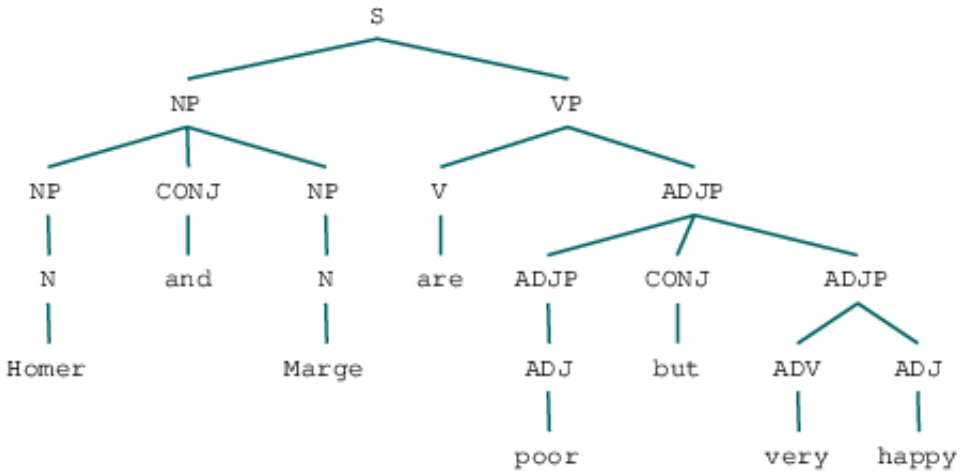
0.1.5 (s9)Homer and Marge are poor but very happy

[25]:
```
s9_grammar1 = nltk.CFG.fromstring("""
S -> NP VP
NP -> NP CONJ NP | N | N
VP -> V ADJP
ADJP -> ADJP CONJ ADJP | ADJ | ADV ADJ
N -> 'Homer' | 'Marge'
V -> 'are'
CONJ -> 'and' | 'but'
ADJ -> 'poor' | 'happy'
ADV -> 'very'
""")
```

[26]:
```
sentence9 = word_tokenize("Homer and Marge are poor but very happy")
parser = nltk.ChartParser(s9_grammar1)
for i in parser.parse(sentence9):
    print(i)
```

```
(S
  (NP (NP (N Homer)) (CONJ and) (NP (N Marge)))
  (VP
    (V are)
    (ADJP (ADJP (ADJ poor)) (CONJ but) (ADJP (ADV very) (ADJ happy)))))
```

[27]:
```
np9 =nltk.Tree.fromstring('(S(NP (NP (N Homer)) (CONJ and) (NP (N Marge)))(VP(V␣
↪are)(ADJP (ADJP (ADJ poor)) (CONJ but) (ADJP (ADV very) (ADJ happy)))))')
display(np9)
```



0.1.6 (s10)Homer and his friends from work drank and sang in the bar

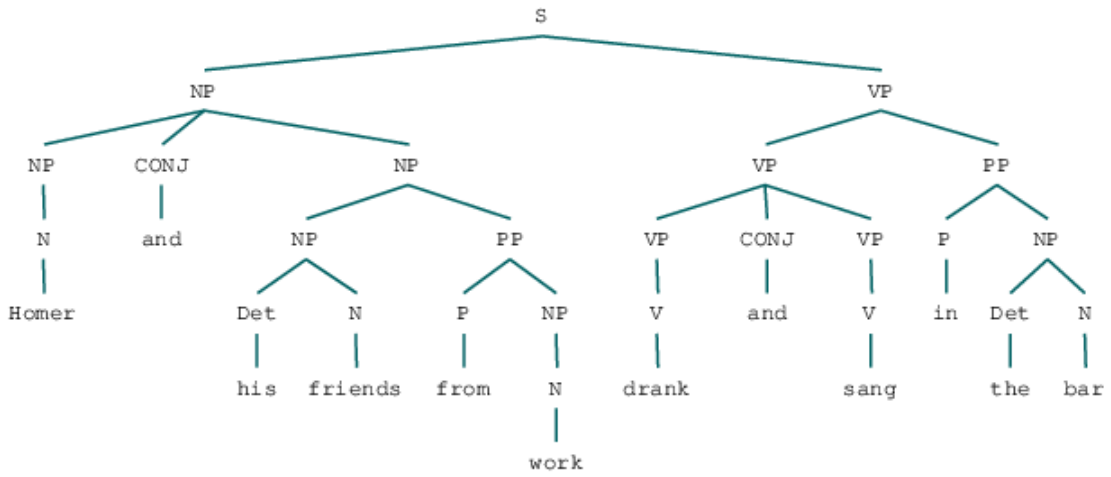
[28]:
```
s10_grammar1 = nltk.CFG.fromstring("""
S -> NP VP
NP -> NP CONJ NP | N | NP PP | Det N | N | Det N
VP -> VP PP | VP CONJ VP | V | V
PP -> P NP | P NP
N -> 'Homer' | 'friends' | 'work' | 'bar'
V -> 'drank' | 'sang'
CONJ -> 'and' | 'and'
Det -> 'his' | 'the'
P -> 'from' | 'in'
""")
```

[29]:
```
sentence10 = word_tokenize("Homer and his friends from work drank and sang in␣
 ↪the bar")
parser = nltk.ChartParser(s10_grammar1)
for i in parser.parse(sentence10):
    print(i)
```

```
(S
  (NP
    (NP (NP (N Homer)) (CONJ and) (NP (Det his) (N friends)))
    (PP (P from) (NP (N work))))
  (VP
    (VP (VP (V drank)) (CONJ and) (VP (V sang)))
    (PP (P in) (NP (Det the) (N bar)))))
(S
  (NP
    (NP (N Homer))
    (CONJ and)
    (NP (NP (Det his) (N friends)) (PP (P from) (NP (N work)))))
  (VP
    (VP (VP (V drank)) (CONJ and) (VP (V sang)))
    (PP (P in) (NP (Det the) (N bar)))))
(S
  (NP
    (NP (NP (N Homer)) (CONJ and) (NP (Det his) (N friends)))
    (PP (P from) (NP (N work))))
  (VP
    (VP (V drank))
    (CONJ and)
    (VP (VP (V sang)) (PP (P in) (NP (Det the) (N bar))))))
(S
  (NP
    (NP (N Homer))
    (CONJ and)
    (NP (NP (Det his) (N friends)) (PP (P from) (NP (N work)))))
```

```
  (VP
    (VP (V drank))
    (CONJ and)
    (VP (VP (V sang)) (PP (P in) (NP (Det the) (N bar))))))
```

[30]:
```
np10 =nltk.Tree.fromstring('(S(NP(NP (N Homer))(CONJ and)(NP (NP (Det his) (N␣
↪friends)) (PP (P from) (NP (N work)))))(VP(VP (VP (V drank)) (CONJ and) (VP␣
↪(V sang)))(PP (P in) (NP (Det the) (N bar)))))')
display(np10)
```



0.1.7 (s11)Lisa told her brother that she liked peanut butter very much

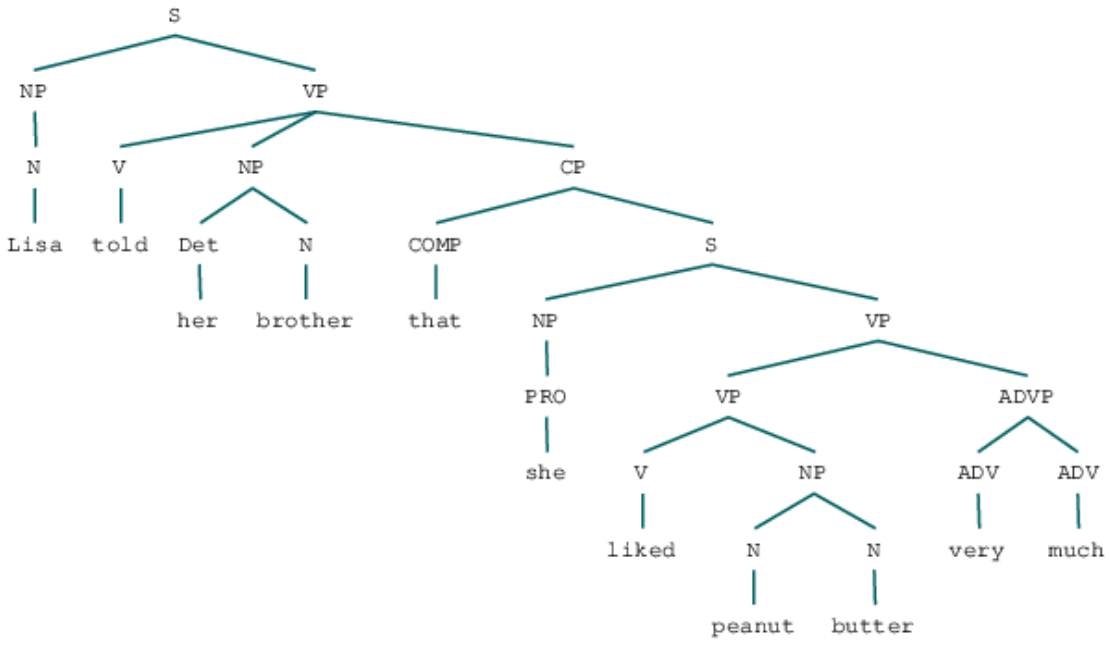
[31]:
```
s11_grammar1 = nltk.CFG.fromstring("""
S -> NP VP | NP VP
NP -> N | Det N | PRO | N N
VP -> V NP CP | VP ADVP | V NP
ADVP -> ADV ADV
CP -> COMP S
N -> 'Lisa' | 'brother' | 'peanut' | 'butter'
V -> 'told' | 'liked'
COMP -> 'that'
Det -> 'her'
PRO -> 'she'
ADV -> 'very' | 'much'
""")
```

[32]:
```
sentence11 = word_tokenize("Lisa told her brother that she liked peanut butter␣
↪very much")
parser = nltk.ChartParser(s11_grammar1)
for i in parser.parse(sentence11):
```

```
    print(i)
```

```
(S
  (NP (N Lisa))
  (VP
    (VP
      (V told)
      (NP (Det her) (N brother))
      (CP
        (COMP that)
        (S (NP (PRO she)) (VP (V liked) (NP (N peanut) (N butter))))))
    (ADVP (ADV very) (ADV much))))
(S
  (NP (N Lisa))
  (VP
    (V told)
    (NP (Det her) (N brother))
    (CP
      (COMP that)
      (S
        (NP (PRO she))
        (VP
          (VP (V liked) (NP (N peanut) (N butter)))
          (ADVP (ADV very) (ADV much)))))))
```

[33]:
```
np11 =nltk.Tree.fromstring('(S(NP (N Lisa))(VP(V told)(NP (Det her) (N␣
↪brother))(CP(COMP that)(S(NP (PRO she))(VP(VP (V liked) (NP (N peanut) (N␣
↪butter)))(ADVP (ADV very) (ADV much)))))))')
display(np11)
```



2.Once a grammar is built, you can print it. Also, you can extract a set of production rules with the .productions() method. Unlike the .productions() method called on a Tree object, the resulting list should be duplicate-free. As before, each rule in the list is a production rule type. A rule has a left-hand side node (the parent node), which you can getto using the .lhs() method; the actual string label for the node can be accessed by calling .symbol() on the node object.

[ ]:
```
grammer3 = nltk.CFG.fromstring("""
S -> NP VP
NP -> N
VP -> V
N -> 'Homer'
V -> 'sleeps'
""")
```

[ ]:
```
print(grammer3)
```

```
Grammar with 5 productions (start state = S)
    S -> NP VP
    NP -> N
    VP -> V
    N -> 'Homer'
    V -> 'sleeps'
```

[ ]:
```
grammer3.productions()
```

[ ]:
```
[S -> NP VP, NP -> N, VP -> V, N -> 'Homer', V -> 'sleeps']
```

[ ]:
```
last_rule = grammer3.productions()[-1]
```

[ ]:
```
last_rule
```

[ ]:
```
V -> 'sleeps'
```

[ ]:
```
last_rule.is_lexical()
```

[ ]:
```
True
```

[ ]:
```
last_rule.lhs()
```

[ ]:
```
V
```

[ ]:
```
last_rule.lhs().symbol()
```

[ ]:
```
'V'
```

0.2 3.Explore the rules and answer the following questions.

[34]:
```
Grammar_all = nltk.CFG.fromstring("""
S -> NP VP | NP AUX VP
NP -> Det ADJ N | N | PRO | Det N | PRO | NP CP | PRO | NP CONJ | NP PP | N N
VP -> V NP | VP PP | V NP PP | V NP | V ADJP | VP PP | VP CONJ | V NP CP | VP␣
 ↪ADVP
CP -> COMP S
PP -> P NP
Det -> 'the' | 'his' | 'her'
ADJ -> 'big' | 'tiny' | 'nerdy' | 'poor' | 'happy'
ADV -> 'very' | 'much'
PRO -> 'he' | 'I' | 'him' | 'she'
ADJP -> ADJP CONJ | ADJ
ADVP -> ADV
N -> 'bully' | 'kid' | 'school' | 'book' | 'sister' | 't' | 'Homer' | 'Marge'|␣
 ↪'friends' | 'work' | 'bar' | 'Lisa' | 'brother' | 'peanut' | 'butter'
V -> 'punched' | 'gave' | 'given' | 'are' | 'drank' | 'sang' | 'told' | 'liked'
CONJ -> 'and' | 'but'
COMP -> 'that'
AUX -> 'had'
P -> 'after' | 'to' | 'from' | 'in'
""")
```

a. What is the start state of your grammar?

[35]:
```
Grammar_all.productions()[0].lhs()
```

[35]: S

b. How many CF rules are in your grammar?

[36]:
```
len(Grammar_all.productions())
```

[36]: 71

c. How many of them are lexical?

[37]:
```
n=0
for x in Grammar_all.productions():
    if x.is_lexical():
        n = n+1
print("How many of them are lexical?  ",n)
```

```
How many of them are lexical?   45
```

d. How many VP rules are there? That is, how many rules have 'VP' on the left-hand side of the rule? That is, how many rules are of the VP -> ... form?

[38]:
```
n=0
for x in Grammar_all.productions():
```

```
    if x.lhs().symbol() == 'VP':
        n = n+1
n
```

[38]: 9

e. How many V rules are there? That is, how many rules have 'V' on the left-hand side of the fule? That is, how many rules are of the V -> ... form?

[39]:
```
n=0
for x in Grammar_all.productions():
    if x.lhs().symbol() == 'V':
        n = n+1
n
```

[39]: 8

0.2.1 4.Using grammar1, build a chart parser.

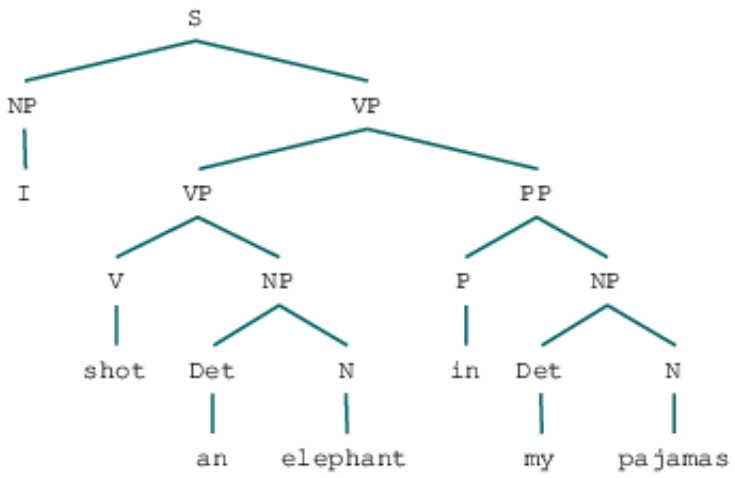
[41]:
```
sentence = word_tokenize("Lisa told her brother that she liked peanut butter␣
 ↪very much")
parser = nltk.ChartParser(Grammar_all)
for i in parser.parse(sentence):
    print(i)
```

```
(S
  (NP (N Lisa))
  (VP
    (V told)
    (NP (Det her) (N brother))
    (CP
      (COMP that)
      (S
        (NP (PRO she))
        (VP
          (VP
            (VP (V liked) (NP (N peanut) (N butter)))
            (ADVP (ADV very)))
          (ADVP (ADV much)))))))
(S
  (NP (N Lisa))
  (VP
    (V told)
    (NP
      (NP (Det her) (N brother))
      (CP
        (COMP that)
        (S
          (NP (PRO she))
```

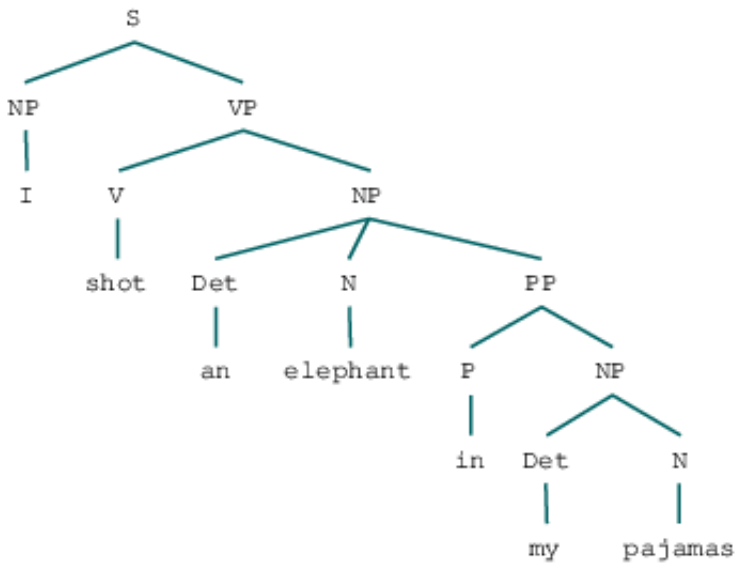
```
          (VP
            (VP
              (VP (V liked) (NP (N peanut) (N butter)))
              (ADVP (ADV very)))
            (ADVP (ADV much))))))))
(S
  (NP (N Lisa))
  (VP
    (VP
      (VP
        (V told)
        (NP (Det her) (N brother))
        (CP
          (COMP that)
          (S
            (NP (PRO she))
            (VP (V liked) (NP (N peanut) (N butter))))))
      (ADVP (ADV very)))
    (ADVP (ADV much))))
(S
  (NP (N Lisa))
  (VP
    (VP
      (VP
        (V told)
        (NP
          (NP (Det her) (N brother))
          (CP
            (COMP that)
            (S
              (NP (PRO she))
              (VP (V liked) (NP (N peanut) (N butter)))))))
      (ADVP (ADV very)))
    (ADVP (ADV much))))
(S
  (NP (N Lisa))
  (VP
    (VP
      (V told)
      (NP (Det her) (N brother))
      (CP
        (COMP that)
        (S
          (NP (PRO she))
          (VP
            (VP (V liked) (NP (N peanut) (N butter)))
            (ADVP (ADV very))))))
    (ADVP (ADV much))))
```

```
(S
  (NP (N Lisa))
  (VP
    (VP
      (V told)
      (NP
        (NP (Det her) (N brother))
        (CP
          (COMP that)
          (S
            (NP (PRO she))
            (VP
              (VP (V liked) (NP (N peanut) (N butter)))
              (ADVP (ADV very)))))))
    (ADVP (ADV much))))
```

[42]:
```
q41 =nltk.Tree.fromstring('(S (NP I) (VP (VP (V shot) (NP (Det an) (N␣
↪elephant))) (PP (P in) (NP (Det my) (N pajamas)))))')
display(q41)
```



[43]:
```
q42 =nltk.Tree.fromstring('(S (NP I) (VP (V shot) (NP (Det an) (N elephant) (PP␣
↪(P in) (NP (Det my) (N pajamas))))))')
display(q42)
```



0.2.2 5. Using the parser, parse the sentences s6 – s11. If your grammar1 is built correctly to cover all of the sentences, the parser should successfully parse all of them.

[44]:
```
!pip install simple-colors
from simple_colors import *
```

```
Collecting simple-colors
  Downloading https://files.pythonhosted.org/packages/0f/07/e6710827a51f6bb5ef67
1f84db98275b122ae3e382726041c823bead1a84/simple_colors-0.1.5-py3-none-any.whl
Installing collected packages: simple-colors
Successfully installed simple-colors-0.1.5
```

[45]:
```
print(black("(s6):the big bully punched the tiny nerdy kid after␣
 ↪school","bold"))
print("\n")
sent6 = word_tokenize("the big bully punched the tiny nerdy kid after school")
parser = nltk.ChartParser(Grammar_1)
for i in parser.parse(sent6):
    print(i)
print("--------------------------------------------------------------------------------")
print("\n")
print(black("(s7):he gave the book to his sister","bold"))
print("\n")
sent7 = word_tokenize("he gave the book to his sister")
parser = nltk.ChartParser(Grammar_1)
for i in parser.parse(sent7):
    print(i)
```

```
print("-------------------------------------------------------------------------------")
print("\n")
print(black("(s8):he gave the book that I had given him t to his␣
 ↪sister","bold"))
print("\n")
sent8 = word_tokenize("he gave the book that I had given him t to his sister")
parser = nltk.ChartParser(Grammar_1)
for i in parser.parse(sent8):
    print(i)
print("-------------------------------------------------------------------------------")
print("\n")
print(black("(s9):Homer and Marge are poor but very happy","bold"))
print("\n")
sent9 = word_tokenize("Homer and Marge are poor but very happy")
parser = nltk.ChartParser(Grammar_1)
for i in parser.parse(sent9):
    print(i)
print("-------------------------------------------------------------------------------")
print("\n")
print(black("(s10):Homer and his friends from work drank and sang in the␣
 ↪bar","bold"))
print("\n")
sent10 = word_tokenize("Homer and his friends from work drank and sang in the␣
 ↪bar")
parser = nltk.ChartParser(Grammar_1)
for i in parser.parse(sent10):
    print(i)
print("-------------------------------------------------------------------------------")
print("\n")
print(black("(s11):Lisa told her brother that she liked peanut butter very␣
 ↪much","bold"))
print("\n")
sent11 = word_tokenize("Lisa told her brother that she liked peanut butter very␣
 ↪much")
parser = nltk.ChartParser(Grammar_1)
for i in parser.parse(sent11):
    print(i)
```

(s6):the big bully punched the tiny nerdy kid after school

```
(S
  (NP (Det the) (ADJ big) (N bully))
  (VP
    (VP (V punched) (NP (Det the) (ADJ tiny) (ADJ nerdy) (N kid)))
    (PP (P after) (NP (N school)))))
(S
```

```
  (NP (Det the) (ADJ big) (N bully))
  (VP
    (V punched)
    (NP (Det the) (ADJ tiny) (ADJ nerdy) (N kid))
    (PP (P after) (NP (N school)))))
(S
  (NP (Det the) (ADJ big) (N bully))
  (VP
    (V punched)
    (NP
      (NP (Det the) (ADJ tiny) (ADJ nerdy) (N kid))
      (PP (P after) (NP (N school))))))
```

(s7):he gave the book to his sister

```
(S
  (NP (PRO he))
  (VP
    (VP (V gave) (NP (Det the) (N book)))
    (PP (P to) (NP (Det his) (N sister)))))
(S
  (NP (PRO he))
  (VP
    (V gave)
    (NP (Det the) (N book))
    (PP (P to) (NP (Det his) (N sister)))))
(S
  (NP (PRO he))
  (VP
    (V gave)
    (NP
      (NP (Det the) (N book))
      (PP (P to) (NP (Det his) (N sister))))))
```

(s8):he gave the book that I had given him t to his sister

```
(S
  (NP (PRO he))
  (VP
    (V gave)
    (NP
      (NP (Det the) (N book))
```

```
      (CP
        (COMP that)
        (S
          (NP (PRO I))
          (AUX had)
          (VP (V given) (NP (PRO him)) (NP (N t))))))
    (PP (P to) (NP (Det his) (N sister)))))
(S
  (NP (PRO he))
  (VP
    (V gave)
    (NP
      (NP
        (NP (Det the) (N book))
        (CP
          (COMP that)
          (S
            (NP (PRO I))
            (AUX had)
            (VP (V given) (NP (PRO him)) (NP (N t))))))
      (PP (P to) (NP (Det his) (N sister))))))
(S
  (NP (PRO he))
  (VP
    (V gave)
    (NP
      (NP (Det the) (N book))
      (CP
        (COMP that)
        (S
          (NP (PRO I))
          (AUX had)
          (VP
            (VP (V given) (NP (PRO him)) (NP (N t)))
            (PP (P to) (NP (Det his) (N sister)))))))))
(S
  (NP (PRO he))
  (VP
    (V gave)
    (NP
      (NP (Det the) (N book))
      (CP
        (COMP that)
        (S
          (NP (PRO I))
          (AUX had)
          (VP
            (V given)
```

```
            (NP (PRO him))
            (NP (NP (N t)) (PP (P to) (NP (Det his) (N sister))))))))))
(S
  (NP (PRO he))
  (VP
    (V gave)
    (NP (Det the) (N book))
    (CP
      (COMP that)
      (S
        (NP (PRO I))
        (AUX had)
        (VP
          (VP (V given) (NP (PRO him)) (NP (N t)))
          (PP (P to) (NP (Det his) (N sister))))))))
(S
  (NP (PRO he))
  (VP
    (V gave)
    (NP (Det the) (N book))
    (CP
      (COMP that)
      (S
        (NP (PRO I))
        (AUX had)
        (VP
          (V given)
          (NP (PRO him))
          (NP (NP (N t)) (PP (P to) (NP (Det his) (N sister))))))))))
(S
  (NP (PRO he))
  (VP
    (VP
      (V gave)
      (NP
        (NP (Det the) (N book))
        (CP
          (COMP that)
          (S (NP (PRO I)) (AUX had) (VP (V given) (NP (PRO him))))))
      (NP (N t)))
    (PP (P to) (NP (Det his) (N sister)))))
(S
  (NP (PRO he))
  (VP
    (VP
      (V gave)
      (NP (Det the) (N book))
      (CP
```

```
        (COMP that)
        (S
          (NP (PRO I))
          (AUX had)
          (VP (V given) (NP (PRO him)) (NP (N t))))))
    (PP (P to) (NP (Det his) (N sister)))))
(S
  (NP (PRO he))
  (VP
    (VP
      (V gave)
      (NP
        (NP (Det the) (N book))
        (CP
          (COMP that)
          (S
            (NP (PRO I))
            (AUX had)
            (VP (V given) (NP (PRO him)) (NP (N t)))))))
    (PP (P to) (NP (Det his) (N sister)))))
(S
  (NP (PRO he))
  (VP
    (V gave)
    (NP
      (NP (Det the) (N book))
      (CP
        (COMP that)
        (S (NP (PRO I)) (AUX had) (VP (V given) (NP (PRO him))))))
    (NP (NP (N t)) (PP (P to) (NP (Det his) (N sister))))))
```

(s9):Homer and Marge are poor but very happy

```
(S
  (NP (NP (N Homer)) (CONJ and) (NP (N Marge)))
  (VP
    (V are)
    (ADJP (ADJP (ADJ poor)) (CONJ but) (ADJP (ADV very) (ADJ happy)))))
```

(s10):Homer and his friends from work drank and sang in the bar

```
(S
```

```
  (NP
    (NP (NP (N Homer)) (CONJ and) (NP (Det his) (N friends)))
    (PP (P from) (NP (N work))))
  (VP
    (VP (VP (V drank)) (CONJ and) (VP (V sang)))
    (PP (P in) (NP (Det the) (N bar)))))
(S
  (NP
    (NP (N Homer))
    (CONJ and)
    (NP (NP (Det his) (N friends)) (PP (P from) (NP (N work)))))
  (VP
    (VP (VP (V drank)) (CONJ and) (VP (V sang)))
    (PP (P in) (NP (Det the) (N bar)))))
(S
  (NP
    (NP (NP (N Homer)) (CONJ and) (NP (Det his) (N friends)))
    (PP (P from) (NP (N work))))
  (VP
    (VP (V drank))
    (CONJ and)
    (VP (VP (V sang)) (PP (P in) (NP (Det the) (N bar))))))
(S
  (NP
    (NP (N Homer))
    (CONJ and)
    (NP (NP (Det his) (N friends)) (PP (P from) (NP (N work)))))
  (VP
    (VP (V drank))
    (CONJ and)
    (VP (VP (V sang)) (PP (P in) (NP (Det the) (N bar))))))
```

(s11):Lisa told her brother that she liked peanut butter very much

```
(S
  (NP (N Lisa))
  (VP
    (V told)
    (NP (Det her) (N brother))
    (CP
      (COMP that)
      (S
        (NP (PRO she))
        (VP
          (VP (V liked) (NP (N peanut) (N butter)))
```

```
          (ADVP (ADV very) (ADV much)))))))
(S
  (NP (N Lisa))
  (VP
    (V told)
    (NP (Det her) (N brother))
    (CP
      (COMP that)
      (S
        (NP (PRO she))
        (VP
          (VP (V liked) (NP (N peanut)) (NP (N butter)))
          (ADVP (ADV very) (ADV much)))))))
(S
  (NP (N Lisa))
  (VP
    (V told)
    (NP
      (NP (Det her) (N brother))
      (CP
        (COMP that)
        (S
          (NP (PRO she))
          (VP
            (VP (V liked) (NP (N peanut) (N butter)))
            (ADVP (ADV very) (ADV much)))))))))
(S
  (NP (N Lisa))
  (VP
    (V told)
    (NP
      (NP (Det her) (N brother))
      (CP
        (COMP that)
        (S
          (NP (PRO she))
          (VP
            (VP (V liked) (NP (N peanut)) (NP (N butter)))
            (ADVP (ADV very) (ADV much)))))))))
(S
  (NP (N Lisa))
  (VP
    (VP
      (V told)
      (NP
        (NP (Det her) (N brother))
        (CP (COMP that) (S (NP (PRO she)) (VP (V liked)))))
      (NP (N peanut) (N butter)))
```

```
    (ADVP (ADV very) (ADV much))))
(S
  (NP (N Lisa))
  (VP
    (VP
      (V told)
      (NP
        (NP (Det her) (N brother))
        (CP
          (COMP that)
          (S (NP (PRO she)) (VP (V liked) (NP (N peanut))))))
      (NP (N butter)))
    (ADVP (ADV very) (ADV much))))
(S
  (NP (N Lisa))
  (VP
    (VP
      (V told)
      (NP (Det her) (N brother))
      (CP
        (COMP that)
        (S (NP (PRO she)) (VP (V liked) (NP (N peanut) (N butter))))))
    (ADVP (ADV very) (ADV much))))
(S
  (NP (N Lisa))
  (VP
    (VP
      (V told)
      (NP (Det her) (N brother))
      (CP
        (COMP that)
        (S
          (NP (PRO she))
          (VP (V liked) (NP (N peanut)) (NP (N butter))))))
    (ADVP (ADV very) (ADV much))))
(S
  (NP (N Lisa))
  (VP
    (VP
      (V told)
      (NP
        (NP (Det her) (N brother))
        (CP
          (COMP that)
          (S
            (NP (PRO she))
            (VP (V liked) (NP (N peanut) (N butter)))))))
    (ADVP (ADV very) (ADV much))))
```

```
(S
  (NP (N Lisa))
  (VP
    (VP
      (V told)
      (NP
        (NP (Det her) (N brother))
        (CP
          (COMP that)
          (S
            (NP (PRO she))
            (VP (V liked) (NP (N peanut)) (NP (N butter)))))))
    (ADVP (ADV very) (ADV much))))
```

[ ]: