
language
evaluation
NLP
systems
words
learning
text
data
tasks
rules
languages
often
processing
information
speech
statistical
system
translation
input
machine
algorithms
task
natural
research
models
recognition
Given
human
example
edit
based
one
especially
standard
segmentation
morphology
disambiguation
resolution
etc
gold
parsers
compared
possible
analysis
also
Natural
campaign
entities
problems
sentence
set
e.g
given
reliable
devoted
termed
probabilistic
usually
decisions
focused
knowledge
many
identify
larger
multiple
made
refer
meaning
type
Automatic
generally
tagger
people
specific
extraction
mainly
corpora
different
developed
generation
ambiguity
section
However
sources
criteria
considered
named
English
result
project
conducted
semantic
semantics
vs
problem
results
i.e
world
although
hand-written
serve
sometimes
complexity
use
chunk
determine
output
however
corresponding
grammar
automatically
understanding
related
POS
using
much
identifying
complex
Many
within
machine-learning
referred
quality
number
French
compare
procedures
including
difficult
may
door
separate
IR
process
common
names
large
intelligence
general
automatic
sentences
word
entity
make
Note
real-world
typically
written
1980s
concerned

Context Free Language

In formal language theory, a Context Free Langugae is a language generated by some Context Free Grammar.

The set of all CFL is identical to the set of languages accepted by Pushdown Automata

Context Free Grammar

Context Free Grammar is defined by 4 tuples as:

$$G = N, \sum, R, S$$

where

N = A set of nonterminals (e.g. N = {S, NP, VP, PP, Noun, Verb ...})

$\sum$ = A set of terminals (e.g. $\sum$ = {I, you, he, eat, drink, sushi, ball, ...})

R = A set of rules $R \subseteq \{A \rightarrow \beta$ with left-hand side $A \in N$ and right-hand side $\beta \in (N \cup \sum\}$

S = A start symbol $S \in N$

Example

- DT → {the, a}
- N → {ball, garden, house, sushi}
- P → {in, behind, with}
- NP → DT N
- NP → NP PP
- PP → N PP

- N: noun
- P: preposition
- NP: noun phrase
- PP: prepositional phrase

** Example** For generating a language that generates equal number of **a** and **b** in the form $a^n b^n$, The CFG will be defined as:

G = {(S,A),(a,b), (S → aAb, A → aAb|∈, S}

Example:

```
from nltk import CFG
grammar = CFG.fromstring("""
S -> NP VP
PP -> P NP
NP -> Det N | NP PP
VP -> V NP | VP PP
Det -> 'a' | 'the'
N -> 'dog' | 'cat'
V -> 'chased' | 'sat'
P -> 'on' | 'in'""")

grammar

print(grammar.start())
print(grammar.productions())
```

Output:S [S -> NP VP, PP -> P NP, NP -> Det N, NP -> NP PP, VP -> V NP, VP -> VP PP, Det -> 'a', Det -> 'the', N -> 'dog', N -> 'cat', V -> 'chased', V -> 'sat', P -> 'on', P -> 'in']

Example:

```
import nltk
from nltk import CFG
grammar = CFG.fromstring("""
S -> NP VP
PP -> P NP
NP -> Det N | NP PP | N
VP -> V NP | VP PP
Det -> 'a' | 'the'
N -> 'dog' | 'cat' | 'Mary'
V -> 'chased' | 'sat' | 'saw'
P -> 'on' | 'in'""")

sent = 'Mary saw a dog'.split()
rd_parser = nltk.ShiftReduceParser(grammar)

for tree in rd_parser.parse(sent):
    print(tree)
```

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

Parsing Algorithms

- Top-down vs. bottom-up:
  - Top-down: (goal driven): from the start symbol down.
  - Bottom-up: (data-driven): from the symbols up
- Naive vs. dynamic programming:
  - Naive: enumerate everything
  - Backtracing: try something, discard partial solutions
  - Dynamic programming: save partial solutions in a table
- Examples:
  - CKY: bottom-up dynamic programming
  - Earlyparsing: top-down dynamic programming

A simple English Grammar

S $\rightarrow$ NP VP
S $\rightarrow$ Aux NP VP
S $\rightarrow$ VP

NP $\rightarrow$ Det NOM
NP $\rightarrow$ ProperNoun

NOM $\rightarrow$ Noun
NOM $\rightarrow$ Noun NOM
NOM $\rightarrow$ NOM PP

VP $\rightarrow$ Verb
VP $\rightarrow$ Verb NP

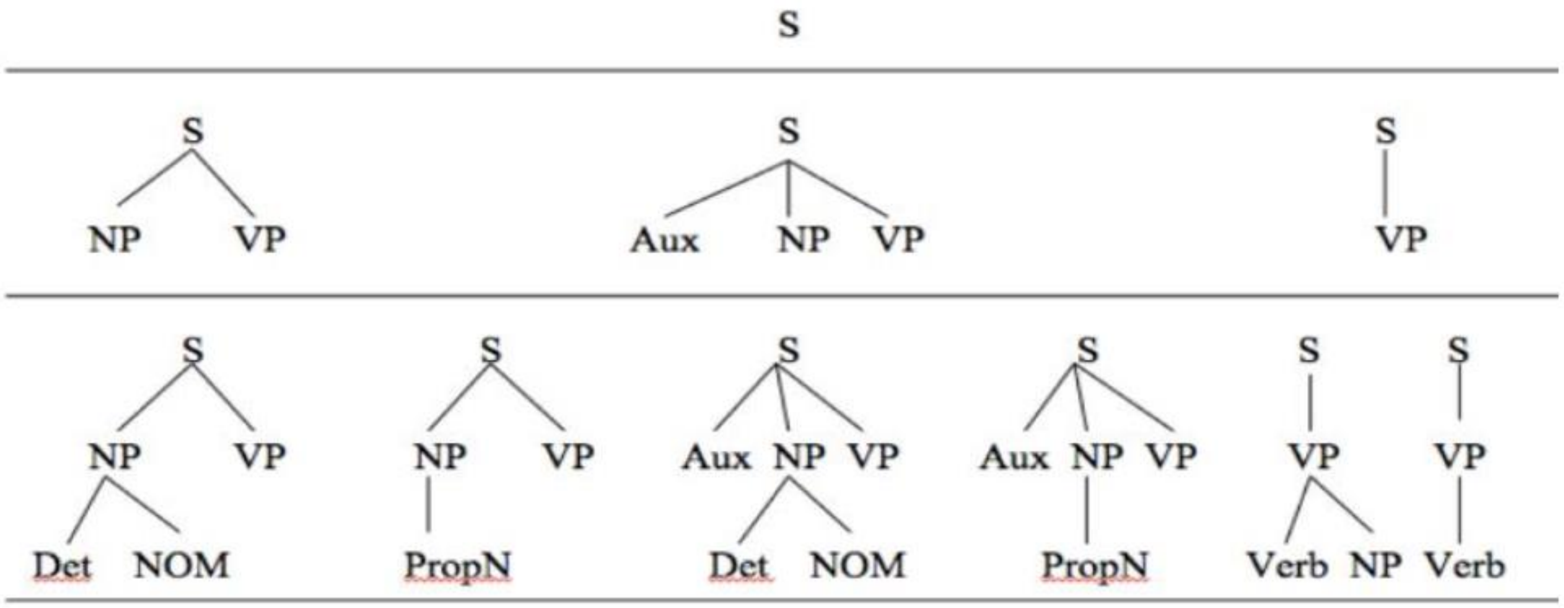
PP $\rightarrow$ Prep NOM

Det $\rightarrow$ that | this | a | the
Noun $\rightarrow$ book | flight | meal | money
Verb $\rightarrow$ book | include | prefer

Aux $\rightarrow$ does
Prep $\rightarrow$ from | to | on

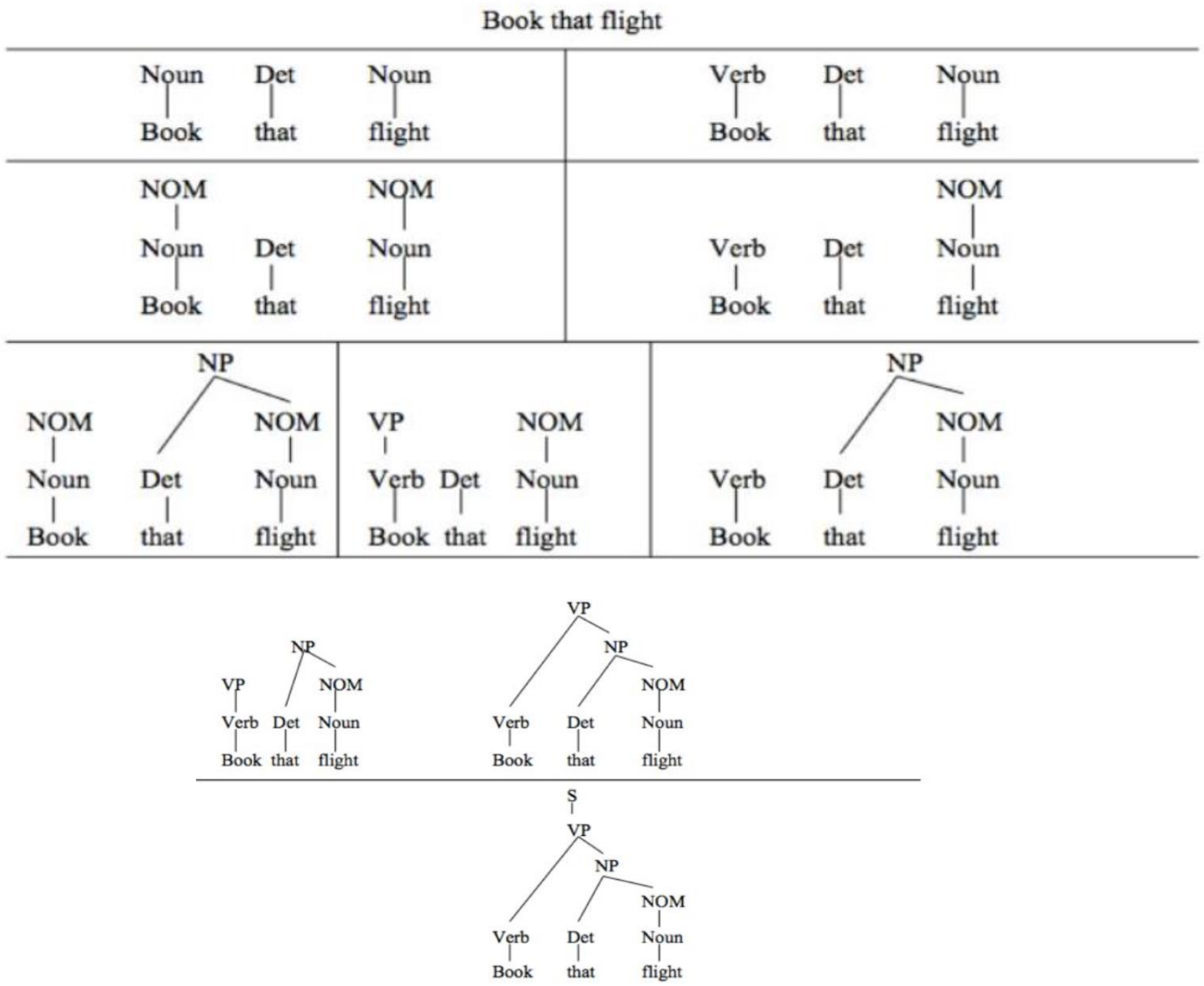
ProperNoun $\rightarrow$ Houston | Washington

A Top-Down Search Space



Input: Book that flight

A Bottom-Up Search Space



The CYK Algorithm

$X_{1,5}$				
$X_{1,4}$	$X_{2,5}$			
$X_{1,3}$	$X_{2,4}$	$X_{3,5}$		
$X_{1,2}$	$X_{2,3}$	$X_{3,4}$	$X_{4,5}$	
$X_{1,1}$	$X_{2,2}$	$X_{3,3}$	$X_{4,4}$	$X_{5,5}$
w_1	w_2	w_3	w_4	w_5

Construct a Triangular Table

Example:

He was watching TV yesterday.

He was watching TV yesterday				
He was watching TV	was watching TV yesterday			
He was watching	was watching TV	watching TV yesterday		
He was	was watching	watching TV	TV yesterday	
He	was	watching	tv	yesterday
w_1	w_2	w_3	w_4	w_5

Example:

- Fill in the CKY chart below for sentence “The rain rains down” assuming the following rules given in Table 1:

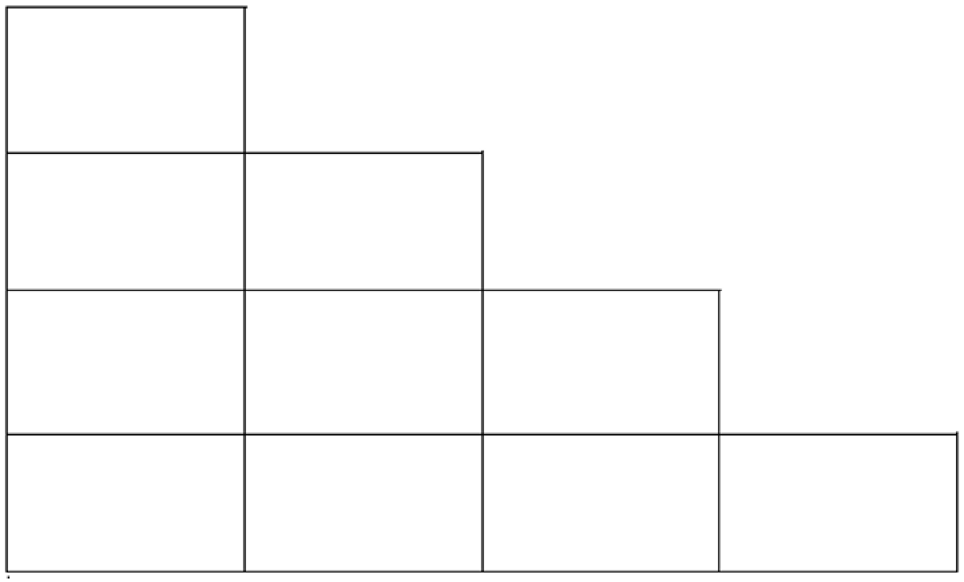


Table 1: Phrase Structure Rules

$S \rightarrow NP \quad VP$	$DT \rightarrow the$
$NP \rightarrow N$	$N \rightarrow rain$
$NP \rightarrow DT \quad N$	$N \rightarrow rains$
$VP \rightarrow V \quad ADVP$	$V \rightarrow rain$
$VP \rightarrow V$	$V \rightarrow rains$
$ADVP \rightarrow ADV$	$ADV \rightarrow down$