Unit 4
Syntax

CFG, Probabilistic CFG
Word’s Constituency (Phrase level, Sentence level),
Parsing (Top-Down and Bottom-Up),
CYK Parser, Probabilistic Parsing

Natural Language Processing (NLP)
MDS 555

Objective

- CFG
- Probabilistic CFG
- Word’s Constituency (Phrase level, Sentence level)
- Parsing (Top-Down and Bottom-Up)
- CYK Parser
- Probabilistic Parsing

Grammar

- Grammar is the structure and system of a language
- It consists of
  - Syntax
  - Morphology

Constituents

- Constituents are groups of words behaving as single units and consist of phrases, words, or morphemes

Symbol	Type	Example
NP	Noun Phrases	"he," "the boy," "the man with the old black shoes"
VP	Verb Phrases	"walked," "sit down and be quiet"
PP	Prepositional Phrases	"on the floor," "with the paper," "apart from everything said before."

Context-Free Grammar (CFG)

- A context-free grammar (CFG) G is a quadruple (V, Σ, R, S) where
  - V: a set of non-terminal symbols
  - Σ: a set of terminals ($V \cap \Sigma = Ǿ$)
  - R: a set of rules ($R: V \rightarrow (V \cup \Sigma)^*$)
  - S: a start symbol.

CFG - Example

- V = {q, f,}
- Σ = {0, 1}
- R = {q $\rightarrow$ 11q, q $\rightarrow$ 00f,
- f $\rightarrow$ 11f, f $\rightarrow$ ε }
- S = q
- (R= {q $\rightarrow$ 11q | 00f, f $\rightarrow$ 11f | ε })

CFG - Rules

- If A → B, then xAy → xBy and we say that
- xAy derivates xBy.

- If s → ⋯ → t, then we write s * t.
- A string x in Σ^* is generated by $G=(V,\Sigma,R,S)$
  if S * x.
- $L(G) = \{ x \text{ in } \Sigma^* \mid S$ * $x \}$.

CFG - Example

- $G = (\{S\}, \{0,1\}. \{S \rightarrow 0S1 \mid \varepsilon \}, S)$
- ε in L(G) because

  $S \rightarrow \varepsilon$.
- 01 in L(G) because

  $S \rightarrow 0S1 \rightarrow 01$.
- 0011 in L(G) because

  $S \rightarrow 0S1 \rightarrow 00S11 \rightarrow 0011$.
- $L(G) = \{0^n 1^n \mid n \geq 0\}$

Context-free Language (CFL)

- A language L is context-free if there exists a CFG G such that L = L(G).
- A grammar **G** generates a language **L**

Example

G = (V, T, S, P)

V = {S, NP, VP, PP, Det, Noun, Verb, Aux, Pre }

T = {‘a’, ‘ate’, ‘cake’, ‘child’, ‘fork’, ‘the’, ‘with’}

S = S

P = { S → NP VP

NP → Det Noun | NP PP

PP → Pre NP

VP → Verb NP

Det → ‘a’ | ‘the’

Noun → ‘cake’ | ‘child’ | ‘fork’

Pre → ‘with’

Verb → ‘ate’}

Example

- Some notes:
  - **Note 1:** In P, pipe symbol (|) is used to combine productions into single representation for productions that have same LHS.
    - For example, Det → ‘a’ | ‘the’ derived from two rules Det → ‘a’ and Det → ‘the’. Yet it denotes two rules not one.
  - **Note 2:** The production highlighted in red are referred as grammar, and green are referred as lexicon.
  - **Note 3:**
    - NP – Noun Phrase, VP – Verb Phrase, PP – Prepositional Phrase, Det – Determiner, Aux – Auxiliary verb

Sample derivation

- S → NP VP

  → Det Noun VP

  → the Noun VP

  → the child VP

  → the child Verb NP

  → the child ate NP

  → the child ate Det Noun

  → the child ate a Noun

  → the child ate a cake

P = { S → NP VP

NP → Det Noun | NP PP

PP → Pre NP

VP → Verb NP

Det → ‘a’ | ‘the’

Noun → ‘cake’ | ‘child’ | ‘fork’

Pre → ‘with’

Verb → ‘ate’}

Probabilistic Context Free Grammar (PCFG)

- PCFG is an extension of CFG with a probability for each production rule
- Ambiguity is the reason why we are using probabilistic version of CFG
  - For instance, some sentences may have more than one underlying derivation.
  - The sentence can be parsed in more than one ways.
  - In this case, the parse of the sentence become ambiguous.
- To eliminate this ambiguity, we can use PCFG to find the probability of each parse of the given sentence

PCFG - Definition

- A probabilistic context free grammar G is a quintuple G = (N, T, S, R, P) where
  - (N, T, S, R) is a context free grammar

    where N is set of non-terminal (variable) symbols, T is set of terminal symbols, S is the start symbol and R is the set of production rules where each rule of the form A → S
  - A probability P(A → s) for each rule in R. The properties governing the probability are as follows;
    - P(A → s) is a conditional probability of choosing a rule A → s in a left-most derivation, given that A is the non-terminal that is expanded.
    - The value for each probability lies between 0 and 1.
    - The sum of all probabilities of rules with A as the left hand side non-terminal should be equal to 1.

$$\sum_{A \rightarrow s \in R:\, A=LHS} P(A \rightarrow s) = 1$$

PCFG - Example

- Probabilistic Context Free Grammar G = (N, T, S, R, P)

  N = {S, NP, VP, PP, Det, Noun, Verb, Pre}

  T = {‘a’, ‘ate’, ‘cake’, ‘child’, ‘fork’, ‘the’, ‘with’}

  S = S

  R = { S → NP VP

  NP → Det Noun | NP PP

  PP → Pre NP

  VP → Verb NP

  Det → ‘a’ | ‘the’

  Noun → ‘cake’ | ‘child’ | ‘fork’

  Pre → ‘with’

  Verb → ‘ate’ }

PCFG - Example

R = {

S → NP VP
NP → Det Noun | NP PP
PP → Pre NP
VP → Verb NP
Det → ‘a’ | ‘the’
Noun → ‘cake’ | ‘child’ | ‘fork’
Pre → ‘with’
Verb → ‘ate’ }

- P = R with associated probability as in the table below

Rule	*Probability*	*Rule*	*Probability*
S → NP VP	1.0	Det → ‘a’ Det → ‘the’	0.5 0.5
NP → NP PP NP → Det Noun	0.6 0.4	Noun → ‘cake’ Noun → ‘child’ Noun → ‘fork’	0.4 0.3 0.3
PP → Pre NP	1.0	Pre → ‘with’	1.0
VP → Verb NP	1.0	Verb → ‘ate’	1.0

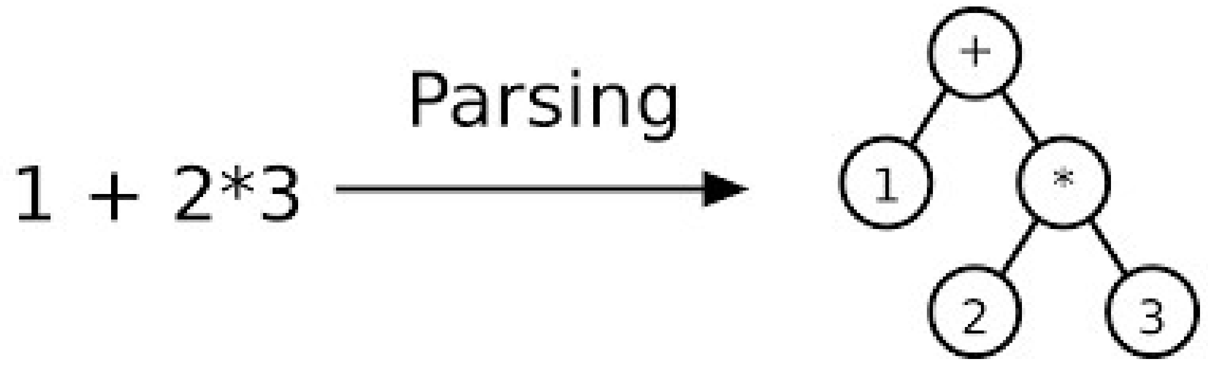
$$\sum_{A \rightarrow s \in R:\, A = NP} P(A \rightarrow s) = P(NP \rightarrow Det\,Noun) + P(NP \rightarrow NP\,PP)$$

$$= 0.4 + 0.6 = 1$$

Please observe from the table, the sum of probability values for all rules that have same left hand side is 1

Parse

- resolve (a sentence) into its component parts and describe their syntactic roles.
- On NLP – Parsing can be visualized in the tree form

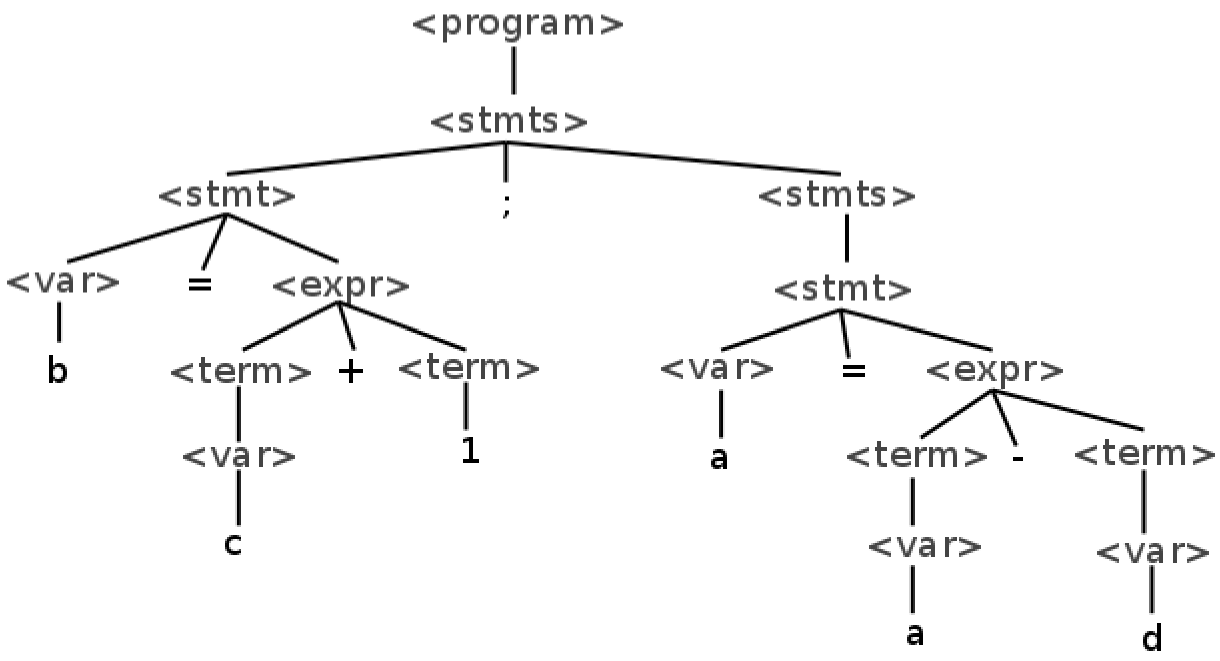


Syntax Parsing

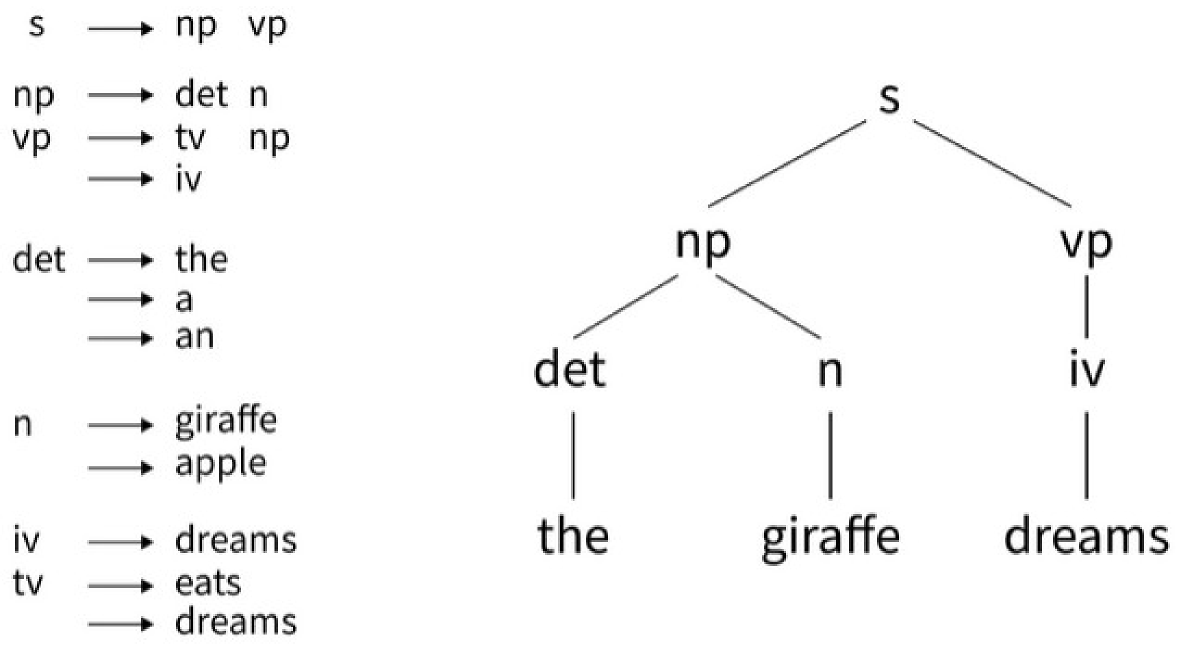
Mostly used in programming

b = c + 1;

a= a - d



- A parse of the sentence "the giraffe dreams" is:
  - s => np vp => det n vp => the n vp => the giraffe vp => the giraffe iv => the giraffe dreams



Probability of a parse tree

- Use of PCFG
- A sentence can be parsed into more than one way
- We can have more than one parse trees for the sentence as per the CFG due to ambiguity.

Probability of a parse tree

- Given a parse tree t, with the production rules α1 → β1, α2 → β2, … , αn → βn from R (ie., αi → βi ∈ R), we can find the probability of tree t using PCFG as follows;

$$P(t) = \prod_{i=1}^{n} P(\alpha_i \rightarrow \beta_i)$$

- As per the equation, the probability P(t) of parse tree is the product of probabilities of production rules in the tree t.

Probability of a parse tree

- Which is the most probable tree?
  - The probability of the parse tree t1 is greater than the probability of parse tree t2. Hence, t1 is the more probable of the two parses.

$$
\begin{aligned}
P(t_1) &= \prod_{i=1}^{n} P(\alpha_i \to \beta_i) \\
&= P(S \to \mathrm{NP\ VP}) * P(\mathrm{NP} \to \mathrm{astronomers}) * P(\mathrm{VP} \to \mathrm{V\ NP}) \\
&\quad * P(\mathrm{V} \to \mathrm{saw}) * P(\mathrm{NP} \to \mathrm{NP\ PP}) * P(\mathrm{NP} \to \mathrm{stars}) \\
&\quad * P(\mathrm{PP} \to \mathrm{P\ NP}) * P(\mathrm{P} \to \mathrm{with}) * P(\mathrm{NP} \to \mathrm{ears})
\end{aligned}
$$

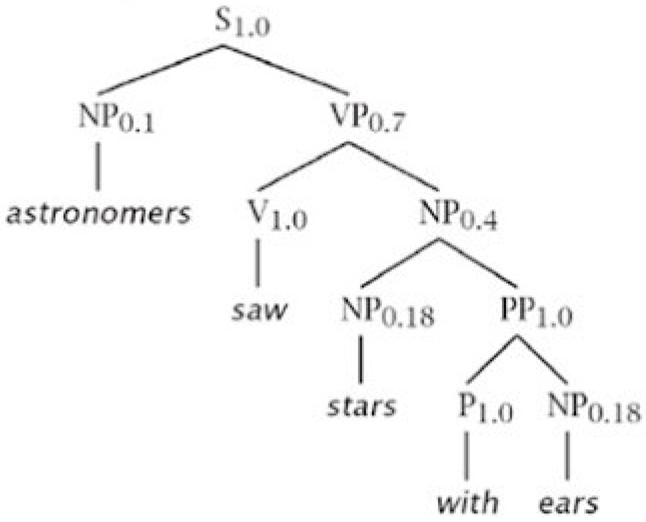
$= 1.0 * 0.1 * 0.7 * 1.0 * 0.4 * 0.18 * 1.0 * 1.0 * 0.18$

$= 0.0009072$

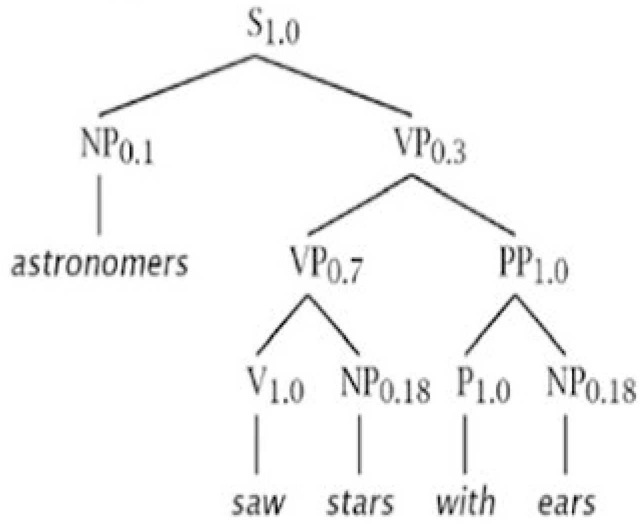
$P(t_2) = 1.0 * 0.1 * 0.3 * 0.7 * 1.0 * 0.18 * 1.0 * 1.0 * 0.18$

$= 0.0006804$

tree t_1



tree t_2



Probability of a sentence

- Probability of a sentence is the sum of probabilities of all parse trees that can be derived from the sentence under PCFG

$$\sum_{i=1}^{n} P(t_i)$$

- Probability of the sentence “astronomers saw the stars with ears”

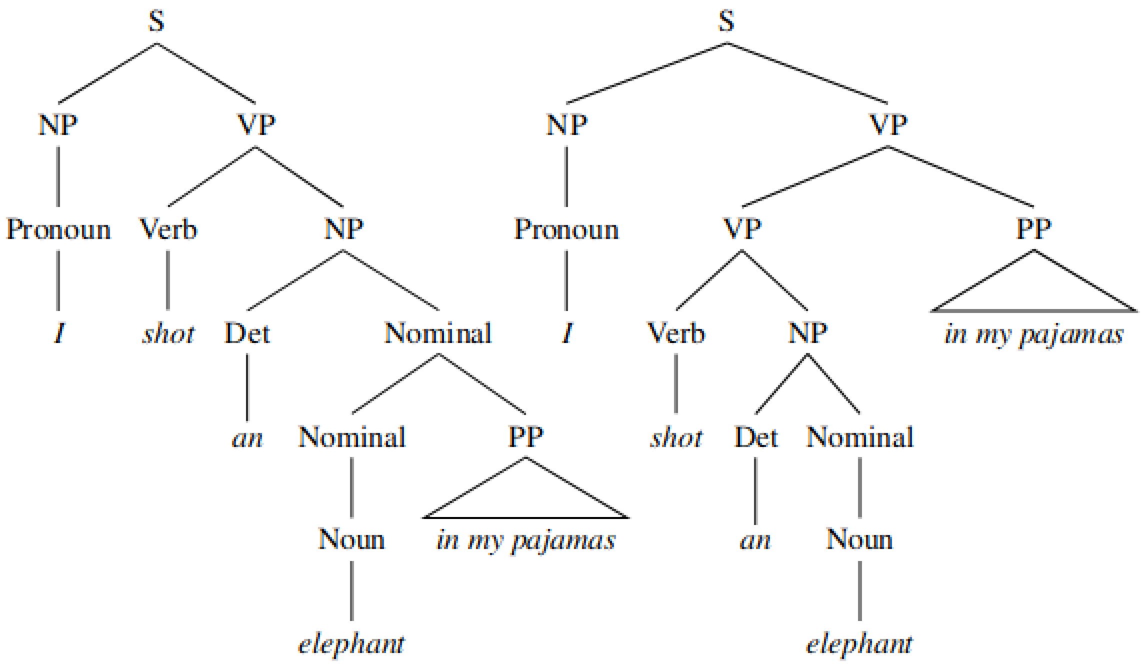
$$\sum_{i=1}^{n} P(t_i) = P(t_1) + P(t_2) = 0.0009072 + 0.0006804 = 0.001588$$

Ambiguity

- Ambiguity is the most serious problem faced by syntactic parsers
- **Structural ambiguity**
-

Ambiguity

- The phrase in my pajamas can be part of the NP headed by elephant or a part of the VP headed by shot



Ambiguity

- Two common kinds of ambiguity are
  - attachment ambiguity and
  - coordination ambiguity
- A sentence has an attachment ambiguity if a particular constituent can be attached to attachment ambiguity the parse tree at more than one place
- In coordination ambiguity phrases can be conjoined by a conjunction like and

Self Study

- Chomsky Normal Form (CNF)
- Cocke–Younger–Kasami (CYK) algorithm

Reference

- Chapter 17 - Speech and Language Processing (3rd Edition)
-

Thank you