Context-Free Parsing

He He

New York University

October 27, 2021

Logistics

- Homework 3 released
- Project proposal due next week (one page)
    - What problem are you tackling and why is it important?
    - What's your approach?
    - How do you plan to evaluate it?

Table of Contents

Context-free language

Probabilistic context-free grammars

Discriminative parsing

Langauge is a set of strings

Formal language:

- A set of **strings** consisting of **words** from an **alphabet**
- *Well-formed* according to a set of rules
- Studies the *syntactical* aspects of a language

Examples:

- Formulas (logic): $(p_1 \wedge p_2) \vee (\urcorner p_3)$
- Programming languages: `int a, b = 0;`
- Sequences from the alphabet $\{a, b\}$ that ends with two a's

Questions:

- Formal language theory: How to describe languages (expressive power, recognizability etc.)
- Linguistics: Can we design formal languages that capture syntactic properties of natural language?

Natural language syntax

Construct a formal language to represent the syntax of natural language

- *Expressivity*: how many syntactic phenomena can it cover?
- *Computation*: how fast can we parse a sentence?

Context-free grammars for natural language

- Captures nested structures which are common in natural language

  [I told Mary that [John told Jane that [Ted told Tom a secret]]].

- Captures long-range dependencies

  the burnt and badly-ground Italian *coffee*
  these burnt and badly-ground Italian *coffees*

- Strikes a good balance between expressivity and computation

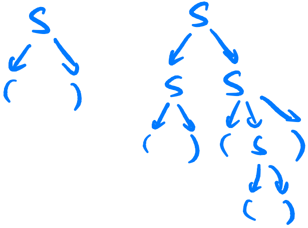
Context-free language

Context-free languages (CFL) are generated by a **context-free grammar** $G = (\Sigma, N, R, S)$:

- a finite alphabet Σ of **terminals** (words)
- a finite set of **non-terminals** N disjoint from Σ (word groups)
- a set of **production rules** R of the form $A \to \beta$, where $A \in N, \beta \in (\Sigma \cup N)^*$ (how to group words)
- a start symbol $S \in N$ (root of derivation)

Example:

$$S \to SS$$
$$S \to (S)$$
$$S \to ()$$



()(())

((()

Phrase-structure grammar for English

Sentences are broken down into **constituents**.

A constituent works as a single unit in a sentence.

- Can be moved around or replaced without breaking grammaticality.

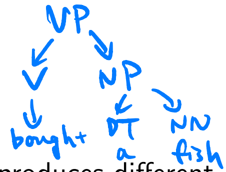
  (Abigail) and (her younger brother) (bought a fish).

Construct CFG for English

- Each word is a terminal, derived from its POS tag.

  DT → the NN → fish

- Each sentence is derived from the start symbol S.
- Each phrase type is a non-terminal.
- Each constituent is derived from a non-terminal.



Grammar design: choose the right set of non-terminals that produces different constituents.

A toy example CFG

$$N = \{\mathrm{S}, \mathrm{NP}, \mathrm{VP}, \mathrm{PP}, \mathrm{DT}, \mathrm{Vi}, \mathrm{Vt}, \mathrm{NN}, \mathrm{IN}\}$$
$$S = \mathrm{S}$$
$$\Sigma = \{\text{sleeps}, \text{saw}, \text{man}, \text{woman}, \text{dog}, \text{telescope}, \text{the}, \text{with}, \text{in}\}$$

$R =$

S	$\rightarrow$	NP	VP
VP	$\rightarrow$	Vi	
VP	$\rightarrow$	Vt	NP
VP	$\rightarrow$	VP	PP
NP	$\rightarrow$	DT	NN
NP	$\rightarrow$	NP	PP
PP	$\rightarrow$	IN	NP

grammar

Vi	$\rightarrow$	sleeps
Vt	$\rightarrow$	saw
NN	$\rightarrow$	man
NN	$\rightarrow$	woman
NN	$\rightarrow$	telescope
NN	$\rightarrow$	dog
DT	$\rightarrow$	the
IN	$\rightarrow$	with
IN	$\rightarrow$	in

lexicon

Lexicon: rules that produce the terminals

(Example from Mike Collins' notes)

Parsing

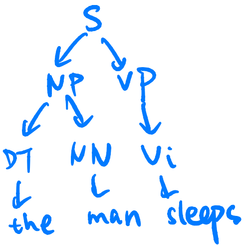
$R =$

S	$\rightarrow$	NP	VP
VP	$\rightarrow$	Vi	
VP	$\rightarrow$	Vt	NP
VP	$\rightarrow$	VP	PP
NP	$\rightarrow$	DT	NN
NP	$\rightarrow$	NP	PP
PP	$\rightarrow$	IN	NP

Vi	$\rightarrow$	sleeps
Vt	$\rightarrow$	saw
NN	$\rightarrow$	man
NN	$\rightarrow$	woman
NN	$\rightarrow$	telescope
NN	$\rightarrow$	dog
DT	$\rightarrow$	the
IN	$\rightarrow$	with
IN	$\rightarrow$	in

Can we derive the sentence "the man sleeps"?

①



② [s [NP [DT the] [NN man]] [VP [Vi sleeps]]]

③ $s_1 = S \rightarrow NP\ VP$

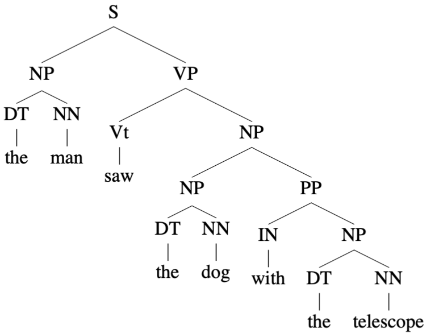
$\rightarrow DT\ NN\ VP$

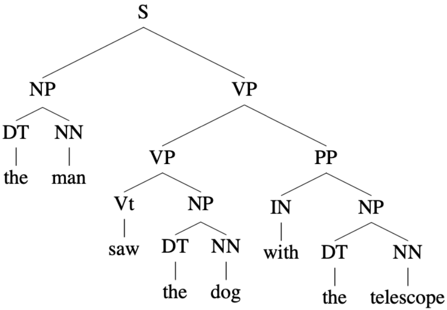
$\rightarrow The\ NN\ VP$

$\vdots$

Ambiguity

Can a sentence have multiple parse trees?





Exercise: find parse trees for
"She announced a program to promote safety in trucks and vans".

Table of Contents

Context-free language

Probabilistic context-free grammars

Discriminative parsing

PCFG

Notation: let $\mathcal{T}_G$ be the set of all possible left-most parse trees under the grammar G.

Goal: define a probability distribution $p(t)$ over parse trees $t \in \mathcal{T}_G$

Parsing: pick the most likely parse tree for a sentence s

$$\arg\max_{t \in \mathcal{T}_G(s)} p(t)$$

Three questions:

- Modeling: how to define $p(t)$ for trees?
- Learning: how to estimate parameters of the distribution $p(t)$?
- Inference: how to find the most likely tree efficiently?

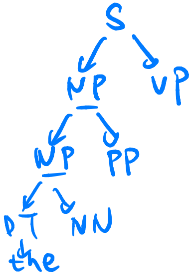
Modeling

Generate parse trees: iteratively sample a production rule to expand a non-terminal

$R =$

S	$\rightarrow$	NP	VP
VP	$\rightarrow$	Vi	
VP	$\rightarrow$	Vt	NP
VP	$\rightarrow$	VP	PP
NP	$\rightarrow$	DT	NN
NP	$\rightarrow$	NP	PP
PP	$\rightarrow$	IN	NP

Vi	$\rightarrow$	sleeps
Vt	$\rightarrow$	saw
NN	$\rightarrow$	man
NN	$\rightarrow$	woman
NN	$\rightarrow$	telescope
NN	$\rightarrow$	dog
DT	$\rightarrow$	the
IN	$\rightarrow$	with
IN	$\rightarrow$	in



PCFG

A **PCFG** consists of

- A CFG $G = (\Sigma, N, R, S)$
- Probabilities of production rules $q(\alpha \to \beta)$ for each $\alpha \to \beta \in R$ such that

$$\sum_{\beta:\, X \to \beta \in R} q(X \to \beta) = 1 \quad \forall X \in N$$

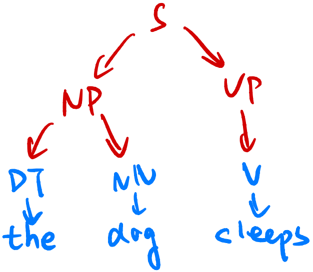
$R, q =$

S	$\to$	NP	VP	1.0
VP	$\to$	Vi		0.3
VP	$\to$	Vt	NP	0.5
VP	$\to$	VP	PP	0.2
NP	$\to$	DT	NN	0.8
NP	$\to$	NP	PP	0.2
PP	$\to$	IN	NP	1.0

Vi	$\to$	sleeps	1.0	
Vt	$\to$	saw	1.0	
NN	$\to$	man	0.1	
NN	$\to$	woman	0.1	$=1$
NN	$\to$	telescope	0.3	
NN	$\to$	dog	0.5	
DT	$\to$	the	1.0	
IN	$\to$	with	0.6	$=1$
IN	$\to$	in	0.4	

From HMM to PCFG

P (



) =

$P(DT \to \text{the}) \ldots$] terminal

$P(S \to NP\ VP)$

$P(NP \to DT\ NN)$] non-terminal

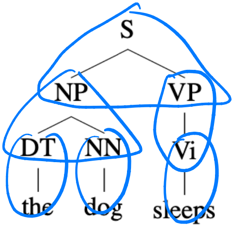
$\ldots$

Probabilities of parse trees

Given a parse tree t consisting of rules $\alpha_1 \rightarrow \beta_1, \ldots, \alpha_n \rightarrow \beta_n$, its probabilities under the PCFG is

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

Example:



Learning

Training data: treebanks

```
((S
   (NP-SBJ (DT That)
     (JJ cold) (, ,)
     (JJ empty) (NN sky) )
   (VP (VBD was)
     (ADJP-PRD (JJ full)
       (PP (IN of)
         (NP (NN fire)
           (CC and)
           (NN light) ))))
   (. .) ))
```
(a)

```
((S
   (NP-SBJ The/DT flight/NN )
   (VP should/MD
     (VP arrive/VB
       (PP-TMP at/IN
         (NP eleven/CD a.m/RB ))
       (NP-TMP tomorrow/NN )))))
```
(b)

Figure 12.7 Parsed sentences from the LDC Treebank3 version of the Brown (a) and ATIS (b) corpora.

Given a set of trees (production rules), we can estimate rule probabilities by MLE.

$$q(\alpha \to \beta) = \frac{\text{count}(\alpha \to \beta)}{\sum_{\beta' :\, \alpha \to \beta' \in R} \text{count}(\alpha \to \beta')}$$

S → NP VP

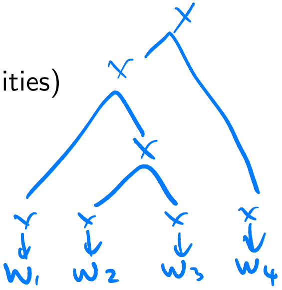
S → *

- Similar to estimate word probabilities ($\to \beta$) given the document class (α).

Parsing

Input: sentences, (P)CFG
Output: derivations / parse trees (with scores/probabilities)

Total number of parse trees for a sentence?



$(w_1 \ (w_2 \ w_3) \ w_4)$

Consider a minimal CFG:

$X \rightarrow XX$
$X \rightarrow$ aardvark|abacus|...|zyther

Given a string, # of parse trees = # of strings with balanced brackets

$((w_1 w_2)(w_3 w_4))$, $(((w_1 w_2) w_3) w_4)$, ...

# of strings with n pairs of brackets:

$$\text{Catalan number } C_n = \frac{1}{n+1}\binom{2n}{n}$$

Chomsky normal form (CNF)

A CFG is in **Chomsky normal form** if every production rule takes one of the following forms:

- Binary non-terminal production: $A \rightarrow BC$ where $A, B, C \in N$.
- Unary terminal production: $A \rightarrow a$ where $A \in N, a \in \Sigma$.

Grammars in CNF produces *binary* parse trees.

Binarize a production rule: VP $\rightarrow$ VBD NP PP

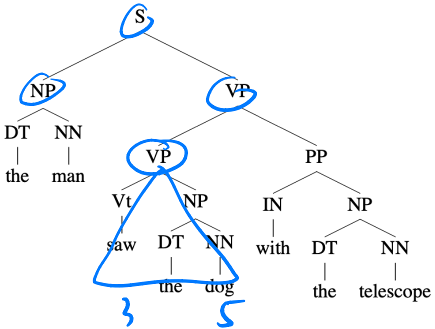
VP $\rightarrow$ VBD @VP-VBD

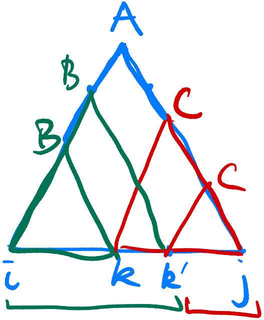
@VP-VBD $\rightarrow$ NP PP

We assume the grammar are in CNF.

Dynamic programming on the tree

$$p(t) = \underbrace{q(A \to BC)}_{\text{top rule}} \times \underbrace{p(t_B)}_{\text{left child}} \times \underbrace{p(t_C)}_{\text{right child}}$$





What are the variables when constructing a tree rooted at A spanning $x_i, \ldots, x_j$?

- The production rule $A \to BC$
- The splitting point s: B spans $x_i, \ldots, x_s$ and C spans $x_{s+1}, \ldots, x_j$

The CYK algorithm

Notation: $\mathcal{T}(i,j,X)$ is the set of trees with root node X spanning $x_i, \ldots, x_j$

Subproblem:

$$\pi(i,j,X) = \max_{t \in \mathcal{T}(i,j,X)} p(t)$$

Base case:

$$\pi(i,i,X) = \begin{cases} q(X \to x_i) & \text{if } X \to x_i \in R \\ 0 & \text{otherwise} \end{cases}$$

Recursion:

$$\pi(i,j,X) = \max_{\substack{Y,Z \in N \\ s \in \{i,\ldots,j-1\}}} q(X \to YZ) \times \pi(i,s,Y) \times \pi(s+1,j,Z)$$

Use backtracking to find the argmax tree.

Bottom-up parsing

$O(n^2 \cdot n \cdot |R|)$

X
Y Z
0 1 2
i j

$R =$

S	$\rightarrow$	NP	VP
VP	$\rightarrow$	Vi	
VP	$\rightarrow$	Vt	NP
VP	$\rightarrow$	VP	PP
NP	$\rightarrow$	DT	NN
NP	$\rightarrow$	NP	PP
PP	$\rightarrow$	IN	NP

Vi	$\rightarrow$	sleeps
Vt	$\rightarrow$	saw
NN	$\rightarrow$	man
NN	$\rightarrow$	woman
NN	$\rightarrow$	telescope
NN	$\rightarrow$	dog
DT	$\rightarrow$	the
IN	$\rightarrow$	with
IN	$\rightarrow$	in

end j

$\pi[0,0,DT]$

start i

0	the 0	man 1	saw 2	the 3	dog 4
~~1~~ 0	DT	NP	∅	∅	S
~~2~~ 1		NN	∅	∅	∅
~~3~~ 2			Vt	∅	VP
~~4~~ 3				DT	NP
4					NN

Variants of CYK

Argmax: find the most likely tree (analogous to Viterbi).

$$\pi(i,j,X) = \max_{\substack{Y,Z\in N \\ s\in\{i,\ldots,j-1\}}} q\,(X \to YZ) \times \pi(i,s,Y) \times \pi(s+1,j,Z)$$

Recognition: does the string belong to the language?

$$\pi(i,j,X) = \bigvee_{\substack{Y,Z\in N \\ s\in\{i,\ldots,j-1\}}} \mathbb{I}\,[X \to YZ \in R] \wedge \pi(i,s,Y) \wedge \pi(s+1,j,Z)$$

Marginalization: what's the probability of the string being generated from the grammar? (the **inside algorithm**) $\sum_z P(x,z)$

$$\pi(i,j,X) = \sum_{\substack{Y,Z\in N \\ s\in\{i,\ldots,j-1\}}} q\,(X \to YZ) \times \pi(i,s,Y) \times \pi(s+1,j,Z)$$

Summary

	NB	HMM	PCFG
output structure	category	sequence	tree
learning		MLE	
decoding	bruteforce	Viterbi	CKY
marginalization		$p(y_i \mid x)$, $p(y_i, y_{i-1} \mid x)$	$p(i, j, N \mid x)$
unsupervised learning		EM	

Table of Contents

Context-free language

Probabilistic context-free grammars

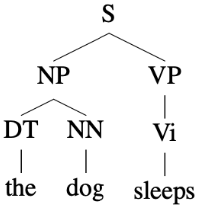
Discriminative parsing

CRF for trees

Input: sequence of words $x = (x_1, \ldots, x_n)$
Output: parse tree $y \in \mathcal{T}(x)$
Model: decompose by production rules

$$p(y \mid x; \theta) \propto \prod_{(r,s)} \psi(r, s \mid x; \theta)$$

- r: production rule
- s: start, split, end indices of the rule r



CRF parsing

Potential functions:

$$\psi(r, s \mid x; \theta) = \exp\left(\theta \cdot \phi(r, s, x)\right)$$

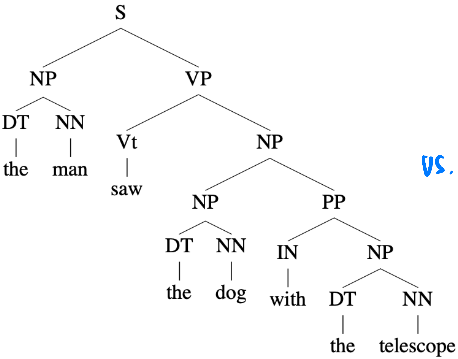
$$\prod_{(r,s)\in\mathcal{T}(x)} \psi(r, s \mid x; \theta) = \exp\left(\sum_{(r,s)\in\mathcal{T}(x)} \theta \cdot \phi(r, s, x)\right)$$

Learning: MLE

1. Compute the partition function by the inside algorithm
2. Call autograd to compute the gradient (backpropagation)

Inference: CYK

Limitations of PCFG



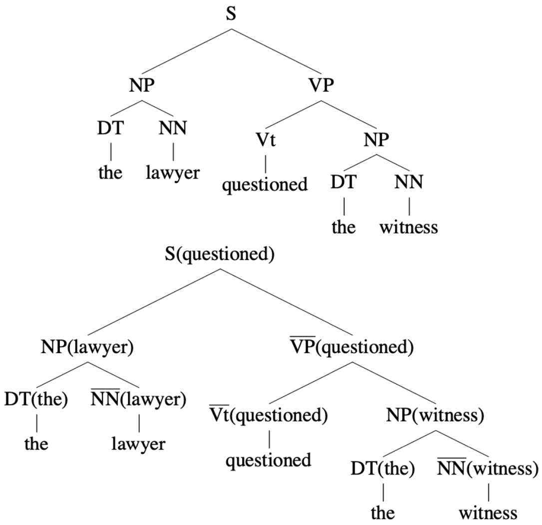
NP → NP PP

vs. VP → Vt PP

Limited lexical information

Lexicalized PCFG

Attach the “head” (most important child in a rule) of the span to each non-terminal



Features

Easy to incorporate lexical information in features!

$$\text{local score} = \theta \cdot \phi(\text{VP} \rightarrow \text{VBD NP}, (5, 6, 8), \text{...averted financial disaster...})$$

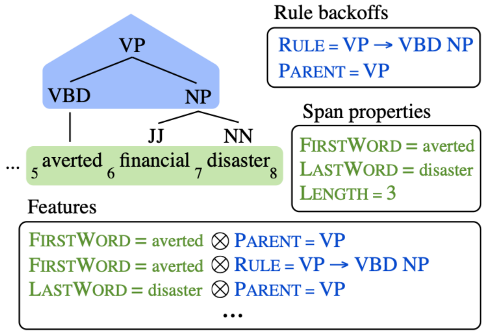


Figure: Less grammar, more features. [Hall+ 14]

Neural CRF parser

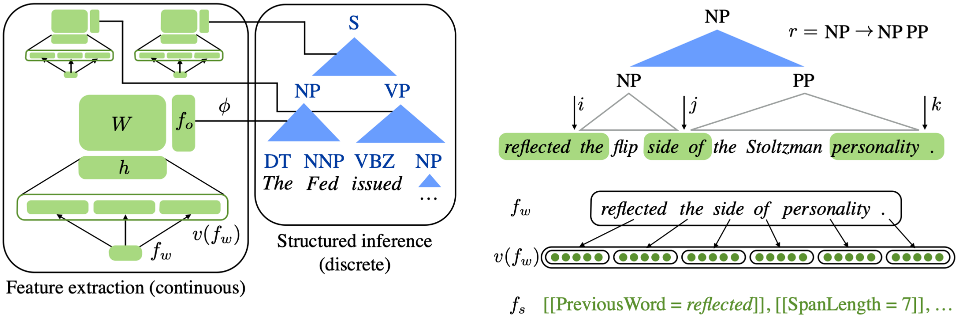


Figure: Neural CRF Parsing. [Durrett+ 15]

- f_w: lexical features
- f_o: rule features
- $h^T W f_o$: interaction between lexical and rule features

Evaluation

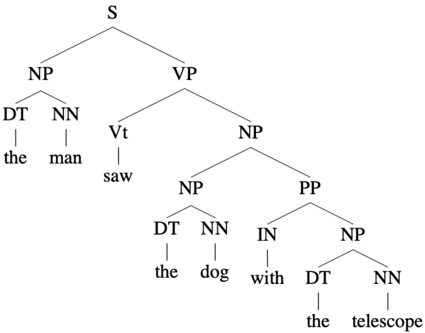
$$\text{recall} = \frac{\#\text{correct constituents}}{\#\text{total constituents in gold trees}}$$

$$\text{precision} = \frac{\#\text{correct constituents}}{\#\text{total constituents in predicted trees}}$$

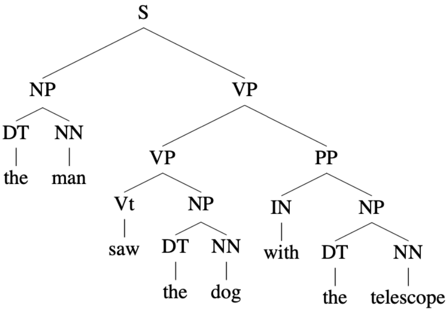
$$\text{F1} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

- Constituent: (i, j, X)
- Labeled F1: the non-terminal node label must be correct
- Unlabeled F1: just consider the tree structure

Example



(a) Gold.



(b) Predicted.