CS 378 Lecture 12

Today

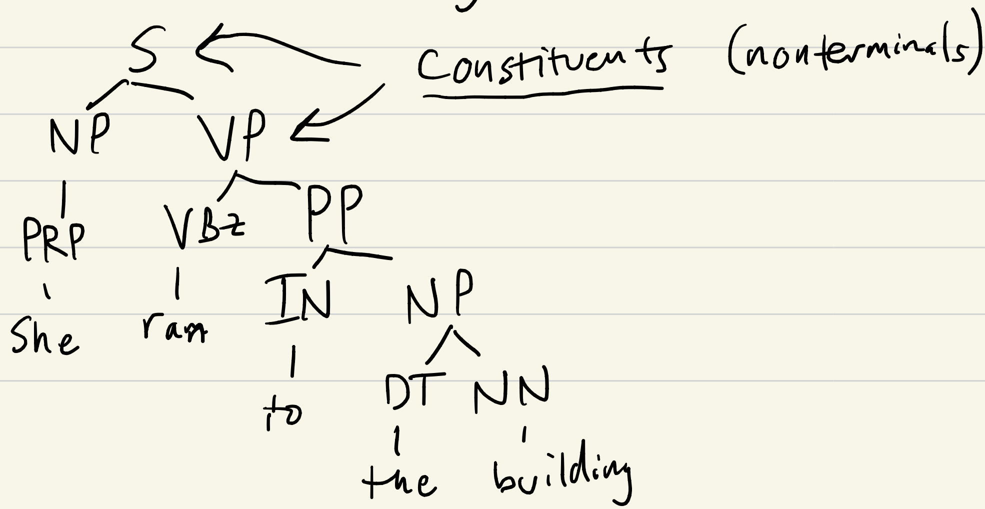
- Probabilistic context-free grammars (PCFGs)
- CKY algorithm
- Refining parsers

Announcements

- A2
- A3
- Midterm: stuff posted Thursday

Recap

Constituency grammar



Context-free Grammars

{ N, T, S, R }

- N: nonterminals — S, NP, VP, PP, DT, NN, VBD, IN, NNS (DT, NN, VBD, IN, NNS → preterminals)
- T: terminals — the, children, ate, cake, with, spoon
- S: start symbol — S
- R: rules

binary

S → NP VP

VP → VBD NP 0.5

NP → DT NN 0.5

NP → DT NNS 0.5

unary

VP → VBD 0.5

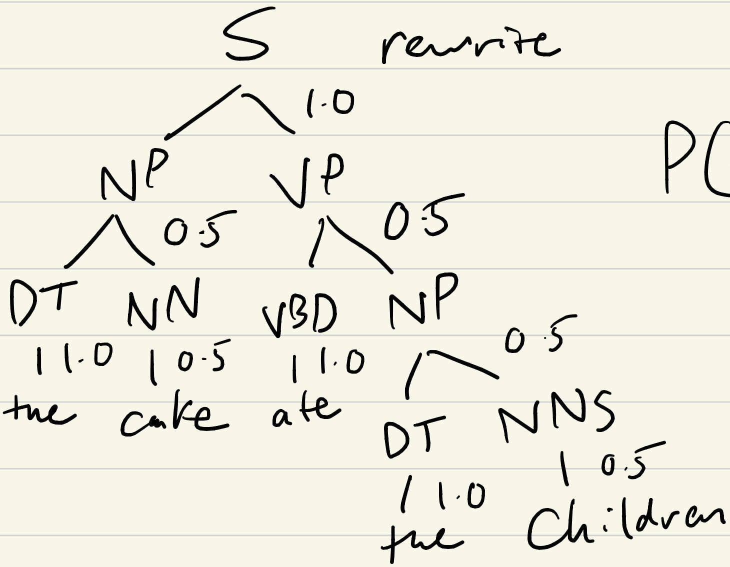
DT → the 1.0

NNS → children 1.0

NN → cake 0.5

NN → spoon 0.5

VBD → ate 1.0



$$P(\text{tree}) = \frac{1}{32}$$

Probabilistic CFGs (PCFGs)

Each rule has a probability

Probs normalize per parent

$$P(\text{rule} \mid NP) = \begin{cases} NP \rightarrow DT\ NN & 0.5 \\ NP \rightarrow DT\ NNS & 0.5 \end{cases}$$

(like transition in HMMs)

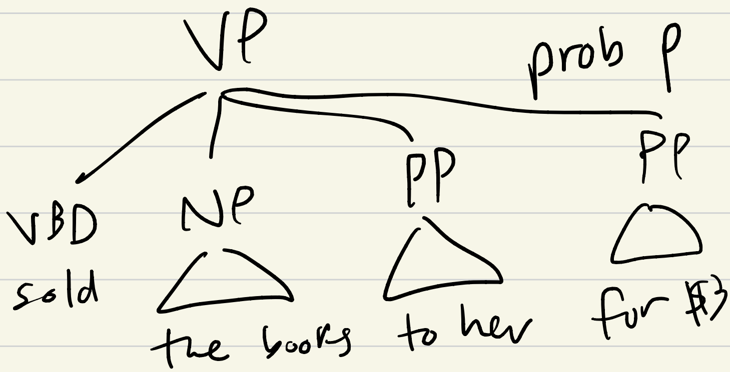
$$\underset{\text{tree}}{P(T)} = \prod_{\text{rules}} P(\text{rule} \mid \text{parent}(\text{rule}))$$

Building a parser

Input: treebank — sents labeled w/trees

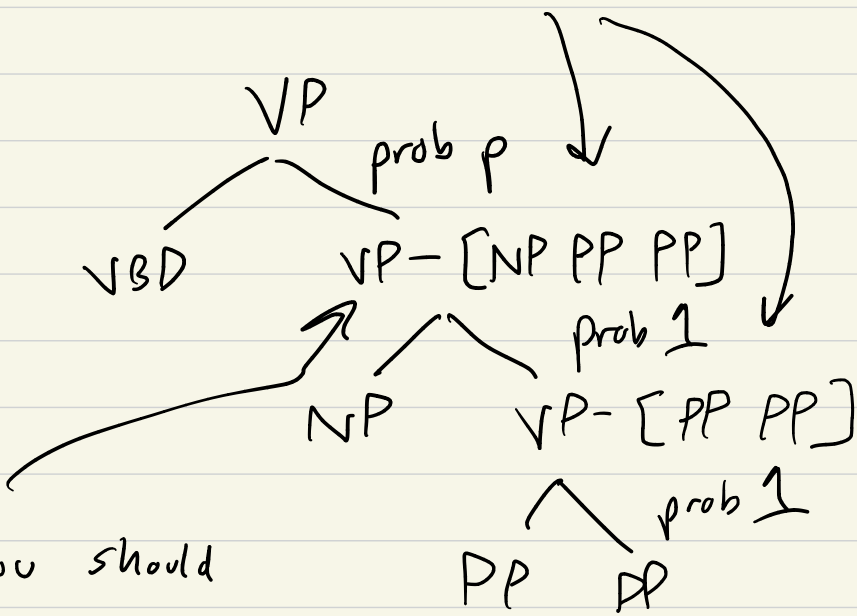
1. Grammar preprocessing
2. Read off grammar + estimate probabilities
3. Implement a parsing algorithm — CKY

① Preprocessing: binarization



Binarization: transform any arity ≥ 3 rule into binary rules (+ unary)

Solution: introduce new symbols



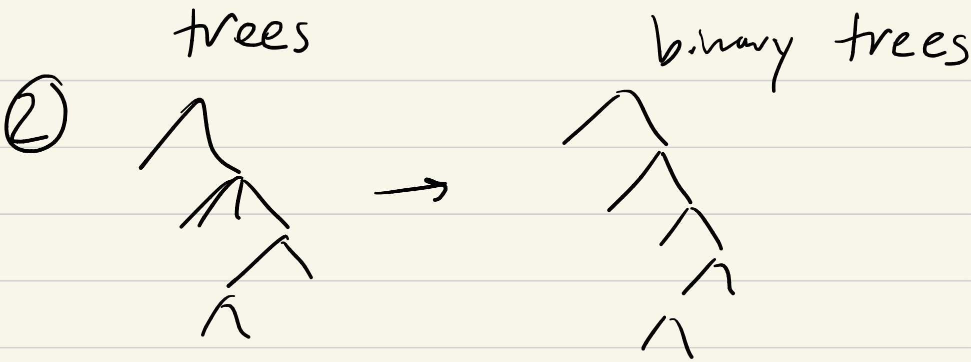
"you should generate NP PP PP as children"

In the grammar:

VP–[NP PP PP] → NP VP–[PP PP]

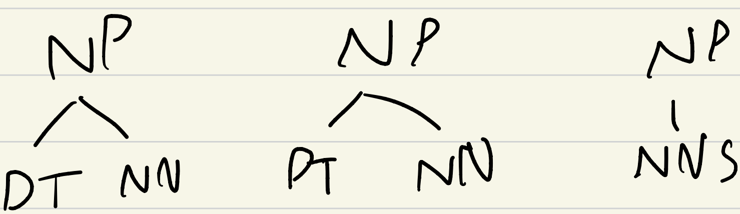
w/prob 1

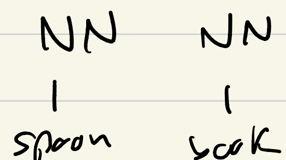
②



Now: extract our PCFG
(grammar + probabilities)

Read the grammar off of the trees
Probs: count + normalize





$P(w \mid NN) = \begin{cases} 1/2 & \text{sp} \\ 1/2 & \text{book} \end{cases}$

$$P(NP \rightarrow DT\ NN \mid NP) = 2/3$$

$$P(NP \rightarrow NNS \mid NP) = 1/3$$

This is the maximum likelihood set of parameters for this data

③ CKY algorithm

Input: PCFG
sentence $\bar{x}$

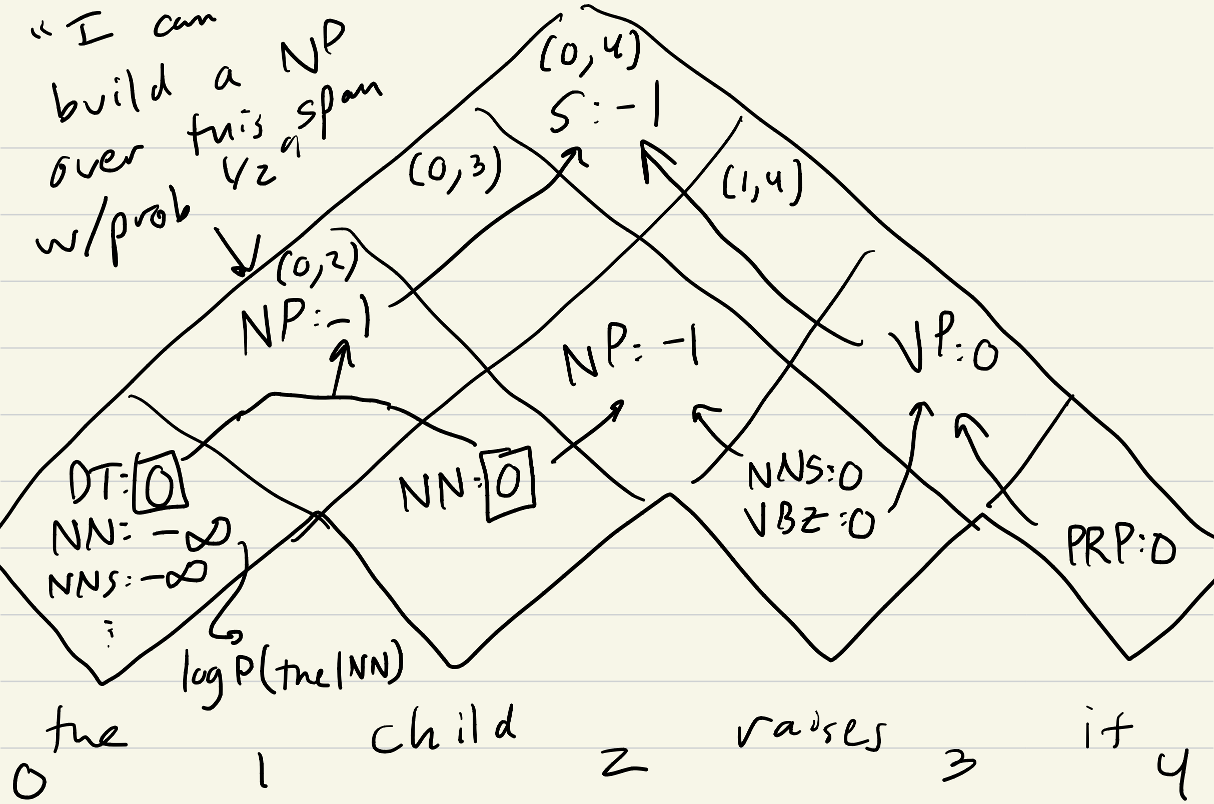
Output: most likely tree $P(T \mid \bar{x})$
for the sentence w.r.t. the PCFG

As in Viterbi:

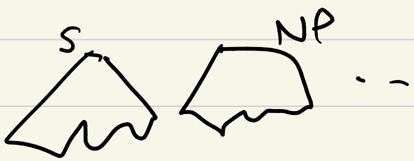
$$\underset{T}{\operatorname{argmax}}\, P(T \mid \bar{x}) = \underset{T}{\operatorname{argmax}} \frac{P(T, \bar{x})}{\cancel{P(\bar{x})}}$$

CKY: dynamic program.

Track the score of the best tree over each span of the sentence



CKY: Chart sent len × sent len × num grammar symbols



$T(i, j, X)$ = score of the best way of building constituent X over span (i, j)

CKY: base $T(i, i+1, X) = \log P(w_i \mid X)$

recursive:

$$T(i, j, X) = \max_{\substack{k: i<k<j \\ \text{split point}}} \overbrace{\max_{X \to X_1 X_2}}^{\text{"transition"}} \left[\log P(X \to X_1 X_2) + T[i, k, X_1] + T[k, j, X_2] \right]$$

Build NP(0,2):

DT(0,1) + NN(1,2) $\log P(NP \to DT\ NN)$

$\log 0.5 = -1$ $\log 1.0 = 0$

Grammar:

DT → the	1.0	1.0	S → NP VP
NN → child	1.0	0.5	NP → DT NN
NNS → raises	1.0	0.5	NP → NN NNS
VBZ → raises	1.0	1.0	VP → VBZ PRP
PRP → it	1.0		

P(the|DT) → (DT → the)

P(child|NN) → (NN → child)

S: -1

— —

NP: -1 — VP: 0

NN: 0	NNS: 0 VBZ: 0	NNS: 0 VBZ: 0	PRP: 0
child	raises	raises	it

Either:

PRP → raises ←

VP → VBZ NNS