{Probabilistic|Stochastic} Context-Free Grammars (PCFGs)

FSNLP, chapter 11

Christopher Manning and Hinrich Schütze

PCFGs

A PCFG G consists of the usual parts of a CFG

- A set of terminals, $\{w^k\}, k = 1, \ldots, V$
- A set of nonterminals, $\{N^i\}, i = 1, \ldots, n$
- A designated start symbol, N^1
- A set of rules, $\{N^i \to \zeta^j\}$, (where ζ^j is a sequence of terminals and nonterminals)

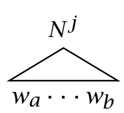
and

- A corresponding set of probabilities on rules such that:

$$\forall i \quad \sum_j P(N^i \to \zeta^j) = 1$$

PCFG notation

Sentence: sequence of words $w_1 \cdots w_m$
w_{ab}: the subsequence $w_a \cdots w_b$
N^i_{ab}: nonterminal N^i dominates $w_a \cdots w_b$



$N^i \overset{*}{\Longrightarrow} \zeta$: Repeated derivation from N^i gives ζ.

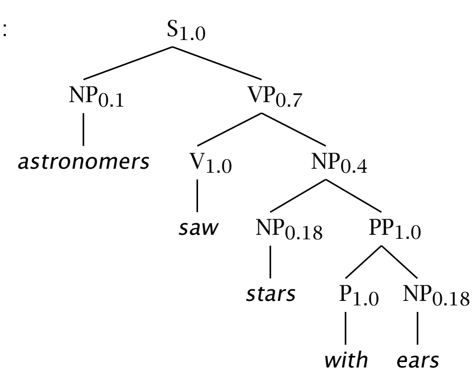
PCFG probability of a string

$$P(w_{1n}) = \sum_t P(w_{1n}, t) \quad t \text{ a parse of } w_{1n}$$
$$= \sum_{\{t:\text{yield}(t)=w_{1n}\}} P(t)$$

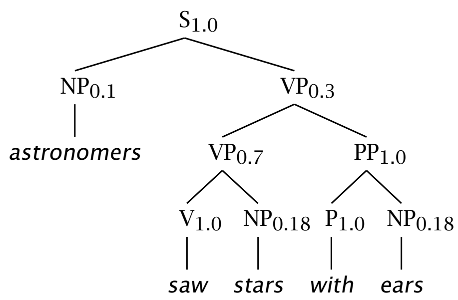
A simple PCFG (in CNF)

S → NP VP	1.0	NP → NP PP	0.4
PP → P NP	1.0	NP → *astronomers*	0.1
VP → V NP	0.7	NP → *ears*	0.18
VP → VP PP	0.3	NP → *saw*	0.04
P → *with*	1.0	NP → *stars*	0.18
V → *saw*	1.0	NP → *telescopes*	0.1

t_1:



t_2:



The two parse trees' probabilities and the sentence probability

$$
\begin{aligned}
P(t_1) &= 1.0 \times 0.1 \times 0.7 \times 1.0 \times 0.4 \\
&\quad \times 0.18 \times 1.0 \times 1.0 \times 0.18 \\
&= 0.0009072 \\
P(t_2) &= 1.0 \times 0.1 \times 0.3 \times 0.7 \times 1.0 \\
&\quad \times 0.18 \times 1.0 \times 1.0 \times 0.18 \\
&= 0.0006804 \\
P(w_{15}) &= P(t_1) + P(t_2) = 0.0015876
\end{aligned}
$$

Assumptions of PCFGs

1. Place invariance (like time invariance in HMM):

$$\forall k \quad P(N^j_{k(k+c)} \to \zeta) \text{is the same}$$

2. Context-free:

$$P(N^j_{kl} \to \zeta | \text{words outside } w_k \dots w_l) = P(N^j_{kl} \to \zeta)$$

3. Ancestor-free:

$$P(N^j_{kl} \to \zeta | \text{ancestor nodes of } N^j_{kl}) = P(N^j_{kl} \to \zeta)$$

The sufficient statistics of a PCFG are thus simply counts of how often different local tree configurations occurred (= counts of which grammar rules were applied).

(Probabilistic) CKY algorithm

```
function CKY(words, grammar) returns most probable parse/probability
   score = new double[#(words)+1][#(words)+1][#(nonterms)];
   back = new Pair[#(words)+1][#(words)+1][#(nonterms)];
   for i = 0; i < #(words); i++
     for A in nonterms
       if A → words[i] in grammar
         score[i][i+1][A] = P(A → words[i])
     // handle unaries
     boolean added = true
     while added
       added = false
       for A, B in nonterms
         if score[i][i+1][B] > 0 && A → B in grammar
           prob = P(A → B) × score[i][i+1][B]
           if (prob > score[i][i+1][A])
             score[i][i+1][A] = prob
             back[i][i+1][A] = B
             added = true
```


(Probabilistic) CKY algorithm [continued]

```
for span = 2 to #(words)
  for begin = 0 to #words − span
    end = begin + span
    for split = begin + 1 to end − 1
      for A, B, C in nonterms
        prob = score[begin][split][B] * score[split][end][C] * P(A → B C)
        if (prob > score[begin][end][A])
          score[begin][end][A] = prob
          back[begin][end][A] = new Triple(split,B,C)
    // handle unaries
    boolean added = true
    while added
      added = false
      for A, B in nonterms
        prob = P(A → B) × score[begin][end][B]
        if (prob > score[begin][end][A])
          score[begin][end][A] = prob
          back[begin][end][A] = B
          added = true
return buildTree(score, back)
```


Calculation of Viterbi probabilities (CKY algorithm)

	1	2	3	4	5
1	$\delta_{NP} =$ 0.1		$\delta_S =$ 0.0126		$\delta_S =$ 0.0009072
2		$\delta_{NP} =$ 0.04 $\delta_V =$ 1.0	$\delta_{VP} =$ 0.126		$\delta_{VP} =$ 0.009072
3			$\delta_{NP} =$ 0.18		$\delta_{NP} =$ 0.01296
4				$\delta_P =$ 1.0	$\delta_{PP} =$ 0.18
5					$\delta_{NP} =$ 0.18
	astronomers	*saw*	*stars*	*with*	*ears*

Modern Statistical Parsers

- A greatly increased ability to do accurate, robust, broad coverage parsing (Charniak 1997; Collins 1997; Ratnaparkhi 1997b; Charniak 2000)
- Achieved by converting parsing into a classification task and using statistical/machine learning methods
- Statistical methods (fairly) accurately resolve structural and real world ambiguities
- Much faster: rather than being cubic in the sentence length or worse, for modern statistical parsers parsing time is made linear (by using beam search)
- Provide probabilistic language models that can be integrated with speech recognition systems.

Supervised ML parsing

- Crucial resource has been treebanks of parses, especially the Penn Treebank (Marcus et al. 1993)
- From these train classifiers:
  - Mainly probabilistic models, but also:
  - Conventional decision trees
  - Decision lists/transformation-based learning
- Possible only when extensive resources exist
- Somewhat uninteresting from Cog. Sci. viewpoint – which would prefer bootstrapping from minimal supervision

A Penn Treebank tree (POS tags not shown)

```
( (S (NP-SBJ The move)
     (VP followed
         (NP (NP a round)
             (PP of
                 (NP (NP similar increases)
                     (PP by
                         (NP other lenders))
                     (PP against
                         (NP Arizona real estate loans)))))
         ,
         (S-ADV (NP-SBJ *)
                (VP reflecting
                    (NP (NP a continuing decline)
                        (PP-LOC in
                                (NP that market))))))
     .))
```


Probabilistic models for parsing

- **Conditional/Parsing model:** We estimate directly the probability of parses of a sentence

  $\hat{t} = \arg\max_t P(t|s,G)$ where $\sum_t P(t|s,G) = 1$
- We don't learn from the distribution of sentences we see (but nor do we assume some distribution for them)
  - (Magerman 1995; Collins 1996; Ratnaparkhi 1999)
- **Generative/Joint/Language model:**

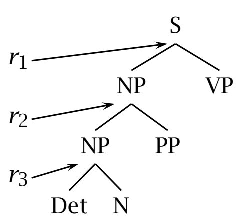
$$\sum_{\{t:\ \text{yield}(t)\in\mathcal{L}\}} P(t) = 1$$

- Most likely tree

  $\hat{t} = \arg\max_t P(t|s) = \arg\max_t \frac{P(t,s)}{P(s)} = \arg\max_t P(t,s)$
  - (Collins 1997; Charniak 1997, 2000)

Generative/Derivational model = Chain rule

$$P(t) = \sum_{\{d:\ d \text{ is a derivation of } t\}} P(d)$$



Or: $P(t) = P(d)$ where d is the canonical derivation of t

$$d = P(S \xrightarrow{r_1} \alpha_1 \xrightarrow{r_2} \ldots \xrightarrow{r_m} \alpha_m = s) = \prod_{i=1}^{m} P(r_i|r_1,\ldots r_{i-1})$$

- History-based grammars

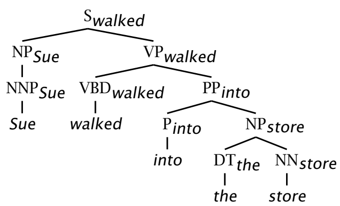
$$P(d) = \prod_{i=1}^{m} P(r_i|\pi(h_i))$$

Enriching a PCFG

- A naive PCFG with traditional nonterminals (NP, PP, etc.) works quite poorly due to the independence assumptions it embodies (Charniak 1996)
- Fix: encode more information into the nonterminal space
  - Structure sensitivity (Manning and Carpenter 1997; Johnson 1998b)
    - Expansion of nodes depends a lot on their position in the tree (independent of lexical content)
    - E.g., enrich nodes by also recording their parents: ^{S}NP is different to ^{VP}NP

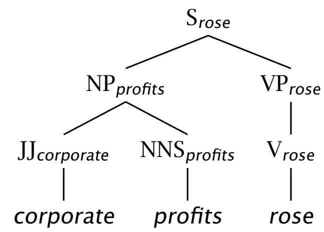
Enriching a PCFG (2)

- (Head) Lexicalization (Collins 1997; Charniak 1997)
  - The head word of a phrase gives a good representation of the phrase's structure and meaning
  - Puts the properties of words back into a PCFG

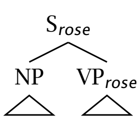


Parsing via classification decisions: Charniak (1997)

- A very simple, conservative model of lexicalized PCFG
- Probabilistic conditioning is "top-down" (but actual computation is bottom-up)



Charniak (1997) example

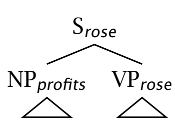


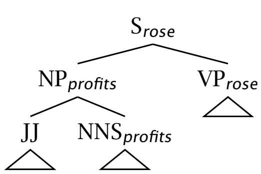
a. h = *profits*; c = NP

b. ph = *rose*; pc = S

c. $P(h|ph, c, pc)$

d. $P(r|h, c, pc)$





Charniak (1997) linear interpolation/shrinkage

$$\hat{P}(h|ph, c, pc) = \lambda_1(e)P_{\mathsf{MLE}}(h|ph, c, pc) + \lambda_2(e)P_{\mathsf{MLE}}(h|C(ph), c, pc) + \lambda_3(e)P_{\mathsf{MLE}}(h|c, pc) + \lambda_4(e)P_{\mathsf{MLE}}(h|c)$$

- $\lambda_i(e)$ is here a function of how much one would expect to see a certain occurrence, given the amount of training data, word counts, etc.
- $C(ph)$ is semantic class of parent headword
- Techniques like these for dealing with data sparseness are vital to successful model construction

Charniak (1997) shrinkage example

	$P(\text{prft}\|\text{rose}, \text{NP}, \text{S})$	$P(\text{corp}\|\text{prft}, \text{JJ}, \text{NP})$
$P(h\|ph, c, pc)$	0	0.245
$P(h\|C(ph), c, pc)$	0.00352	0.0150
$P(h\|c, pc)$	0.000627	0.00533
$P(h\|c)$	0.000557	0.00418

- Allows utilization of rich highly conditioned estimates, but smoothes when sufficient data is unavailable
- One can't just use MLEs: one commonly sees previously unseen events, which would have probability 0.

Sparseness & the Penn Treebank

- The Penn Treebank – 1 million words of parsed English *WSJ* – has been a key resource (because of the widespread reliance on supervised learning)
- But 1 million words is like nothing:
  - 965,000 constituents, but only 66 WHADJP, of which only 6 aren't *how much* or *how many*, but there is an infinite space of these (*how clever/original/incompetent* (*at risk assessment and evaluation*))
- Most of the probabilities that you would like to compute, you can't compute

Sparseness & the Penn Treebank (2)

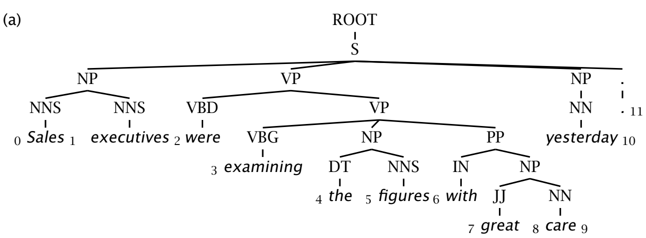
- Most intelligent processing depends on bilexical statistics: likelihoods of relationships between pairs of words.
- Extremely sparse, even on topics central to the *WSJ*:
  - stocks plummeted   2 occurrences
  - stocks stabilized   1 occurrence
  - stocks skyrocketed   0 occurrences
  - #stocks discussed   0 occurrences
- So far there has been very modest success augmenting the Penn Treebank with extra unannotated materials or using semantic classes or clusters (cf. Charniak 1997, Charniak 2000) – as soon as there are more than tiny amounts of annotated training data.

Probabilistic parsing

- Charniak (1997) expands each phrase structure tree in a single step.
- This is good for capturing dependencies between child nodes
- But it is bad because of data sparseness
- A pure dependency, one child at a time, model is worse
- But one can do better by in between models, such as generating the children as a Markov process on both sides of the head (Collins 1997; Charniak 2000)

Evaluation

(a)



(b) Brackets in gold standard tree (a.):
S-(0:11), **NP-(0:2)**, VP-(2:9), VP-(3:9), **NP-(4:6)**, PP-(6-9), NP-(7,9), *NP-(9:10)

(c) Brackets in candidate parse:
S-(0:11), **NP-(0:2)**, VP-(2:10), VP-(3:10), NP-(4:10), **NP-(4:6)**, PP-(6-10), NP-(7,10)

(d)

Precision:	3/8 = 37.5%	Crossing Brackets:	0
Recall:	3/8 = 37.5%	Crossing Accuracy:	100%
Labeled Precision:	3/8 = 37.5%	Tagging Accuracy:	10/11 = 90.9%
Labeled Recall:	3/8 = 37.5%		

Parser results

- Parsers are normally evaluated on the relation between *individual postulated nodes* and ones in the gold standard tree (Penn Treebank, section 23)
- Normally people make systems balanced for precision/recall
- Normally evaluate on sentences of 40 words or less
- Magerman (1995): about 85% labeled precision and recall
- Charniak (2000) gets 90.1% labeled precision and recall
- Good performance. Steady progress in error reduction
- At some point size of and errors in treebank must become the limiting factor
  - (Some thought that was in 1997, when several systems were getting 87.x%, but apparently not.)