Big attraction:

limited memory versions

BFGS:

$$x_{k+1} = x_k - \alpha_k H_k \nabla f_k$$

$$H_{k+1} = V_k^T H_k V_k + \rho_k s_k s_k^T$$

$$\rho_k = \frac{1}{y_k' s_k}, \quad V_k = (I - \rho_k y_k s_k^T)$$

Now H_K is big ~~so we don't wan~~ to store it

IDEA:
- store the last m (s, y) pa
- throw away oldest
- have an initial H_K^0 for each step.

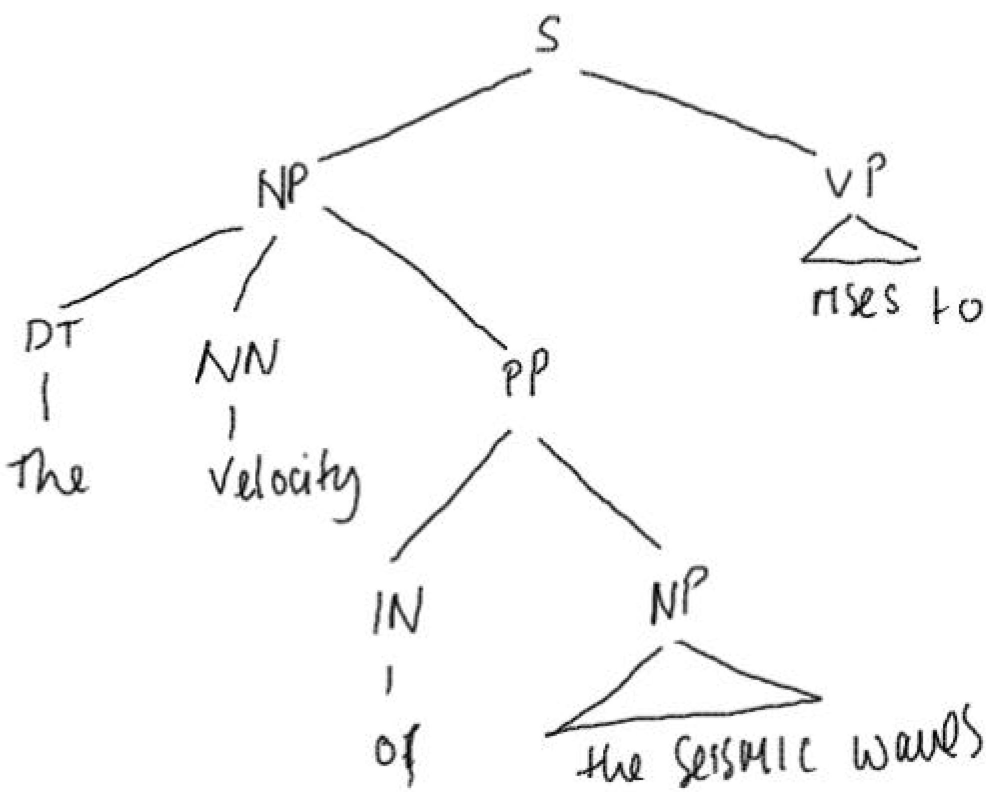
Parsing.

- We think of strings of language as having structure at long scales:

"The velocity of the seismic waves rises to.

why is this singular?

- hard to capture with HMM



How can we extract this structure?

- complex q'n with many partial answers
- One answer: Probabilistic Context Free Grammar

~~PCFG~~ consists of

- a set of terminals $\{w^k\}, k = 1 \cdots N$
- a set of nonterminals $\{N^i\}$ $i = 1 \cdots n$
- a start symbol N^1
- a set of rules $N^i \rightarrow \zeta^j$

  ζ^j is a sequence of terminals and non-terminals

- a set of probabilities on rules

$$P(N^i \rightarrow \zeta^j \mid N^i) = P(N^i \rightarrow \zeta^j)$$

note $$\sum_j P(N^i \rightarrow \zeta^j) = 1$$

Notation:

sentence is $w_1 \cdots\cdots w_m$

span w_{ab} is $w_a \cdots w_b$

if by a set of rewrites we can go from N^j to w_{ab}

then N^j dominates w_{ab} and

w_{ab} is yield of N^j

N^j_{ab} means N^j dominates $a \cdots b$

$$P(w_{1m}) = \sum_t P(w_{1m}, t)$$

all trees that yield w_{1m}

<u>Conditions</u>:

need

Place - invariance:

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

Context free:

$$P(N^j_{kl} \rightarrow \zeta \mid \text{anything outside } k-l)$$
$$= P(N^j_{kl} \rightarrow \zeta)$$

Ancestor free:

$$P(N^j_{kl} \rightarrow \zeta \mid \text{any ancestors outside } N^j_{kl})$$
$$= P(N^j_{kl} \rightarrow \zeta)$$

Under these conditions, computing $P(\omega, t)$ is straightforward

Properties

- Predictive power tends to be greater for language than HMM's w/ same # of parameters

- PCFG's favour smaller trees over larger trees.

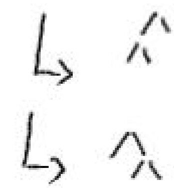
- Probability can be wasted on infinite trees

Example:

$$S \rightarrow \text{rhubarb} \qquad P = 1/3$$

$$S \rightarrow SS \qquad P = 2/3$$

w	$P(w) = \sum_t P(w,t)$
rhubarb	$1/3$
rhubarb rhubarb	$2/3 \times 1/3 \times 1/3 = 2/27$
rhubarb rhubarb rhubarb	$(2/3)^2 (1/3)^3 \times 2 = 8/243$



$$\text{So} \quad \sum_w P(w) = \sum_{w,t} P(w,t)$$

$$= 1/3 + 2/27 + 8/243 + \cdots$$

$$= 1/2 \quad !$$

(Half of the probability has gotten stuck in infinite trees)

issues: eval: $P(w)$?
inference: $\arg\max_t P(w, t)$
learning: rule probs.

- Consider only Chomsky Normal Form grammars

rules are:

$$N^i \to N^j N^k$$
$$N^i \to w^j$$

} only forms available

parameters are:

$P(N^i \to N^j N^k)$ — n^3 table for n nonterminals

$P(N^i \to w^j)$ — nV table for n nonterminals, V terminals

and

$$\sum_{j,k} P(N^i \to N^j K^k) + \sum_j P(N^i \to w^j) = 1$$

Example:

Probabilistic regular grammar
(which is rather like an HMM)

$$N^i \to w^j N^k$$
$$N^i \to w^j$$

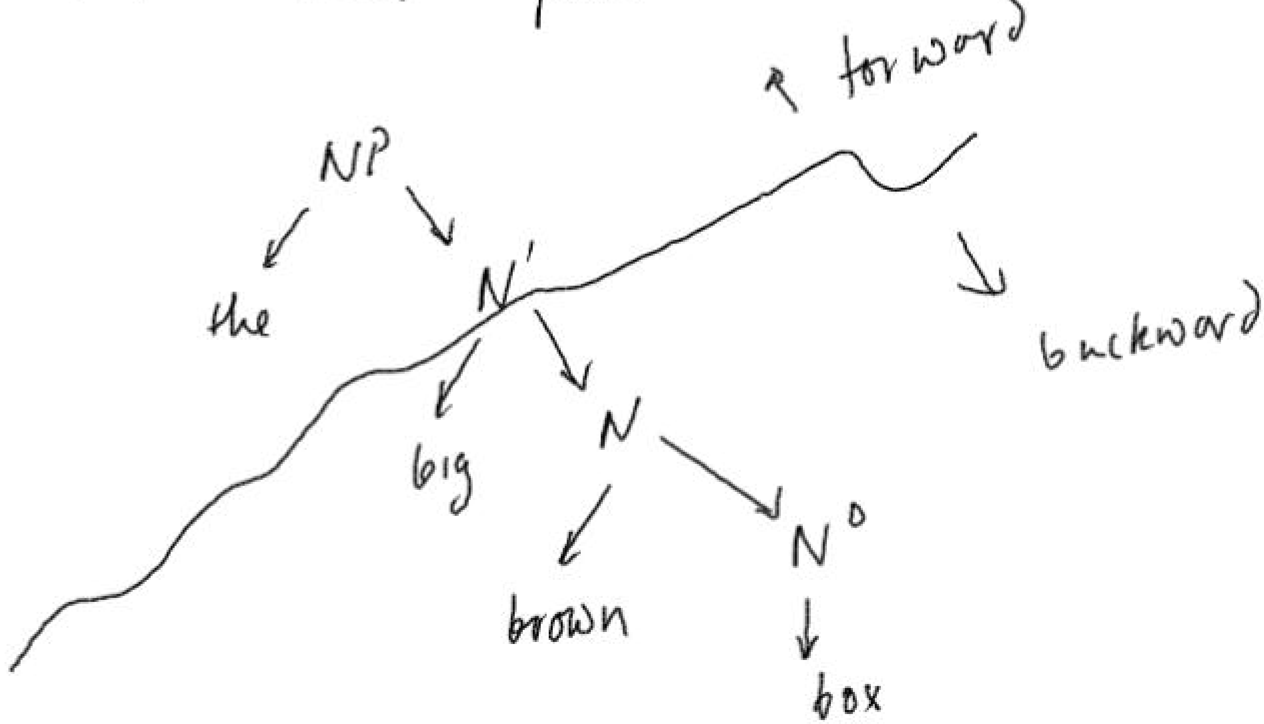
$$\begin{tikzcd}
NP \arrow[r] \arrow[d] & N \arrow[r] \arrow[d] & N' \arrow[r] \arrow[d] & N' \arrow[r] \arrow[d] & \text{sink} \\
\text{the} & \text{big} & \text{brown} & \text{box} &
\end{tikzcd}$$

if we add this, we have someth that we can impl ed with a HM

Now in an HMM, we worked with

$$P(w_{1\,t-1}, X_t = i)$$
$$P(w_{tT} \mid X_t = i)$$

but PRG parse



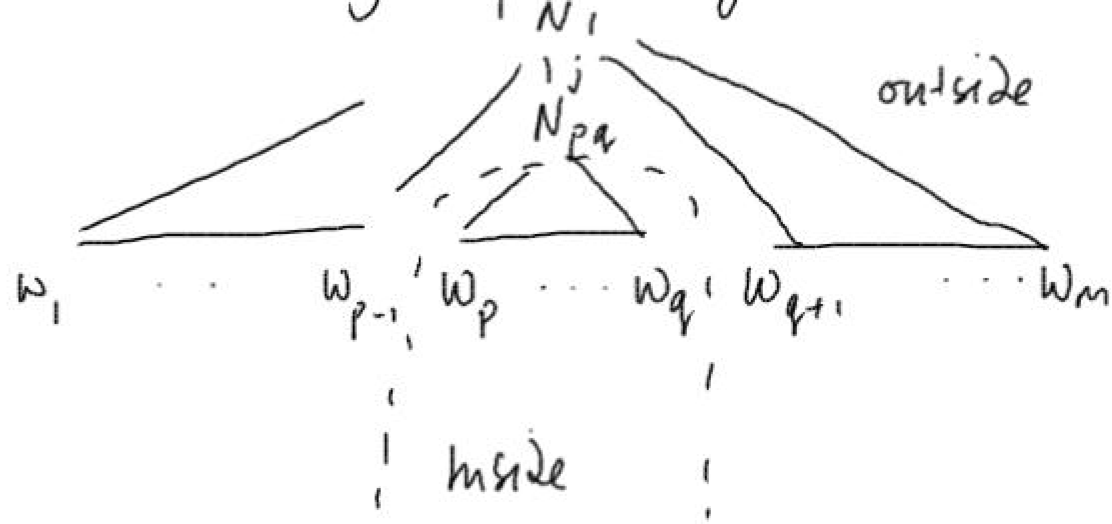
All this suggests an approach to PCFG's

Outside prob $\alpha_j(p,q)$ forward

$= P(w_{1(p-1)}, N_{pq}, w_{(q+1)m})$ backward

Inside prob $\beta_j(p,q) = P(w_{pq} \mid N^j_{pq})$

this is cutting up string



Notice

$$\alpha_j(p,q) \cdot \beta_j(p,q)$$

$$= P(w_{1 \cdots m}, \text{ tree with } N^j_{w_{pq}} \text{ yielding})$$

$$\sum_j \alpha_j(p,q)\,\beta_j(p,q)$$

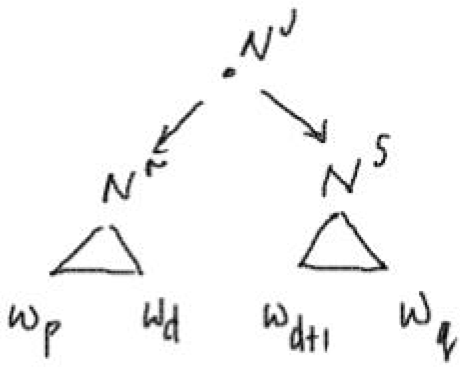
$$= P(w_{1 \cdots m}, \text{ tree with some } N \text{ yielding } w_{pq})$$

How to calculate probabilities:

Inside probs:

$$\beta_j(k,k) = P(N^j \to w_k)$$

now consider $\beta_j(p,q)$



This is a ~~pictur~~ ~~of~~ all possible cases.

so

$$\beta_j(p,q) = \sum_{r,s} \left[\sum_{d=p+1}^{q-1} \Big\{ P(N^j \to N^r N^s) \cdot \beta_r(p,d) \cdot \beta_s(d+1,$$

Outside probs:

$$\alpha_1(1,m) = 1$$

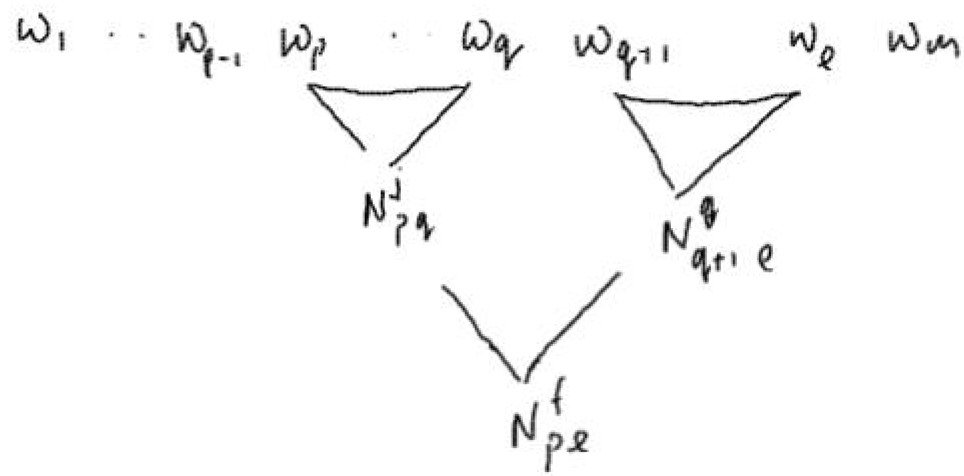
$$= P(\text{tree yielding } w_1 \cdots w_m, \text{ rooted at } N^1)$$

$$\alpha_j(1,m) = 0$$

Interior strings

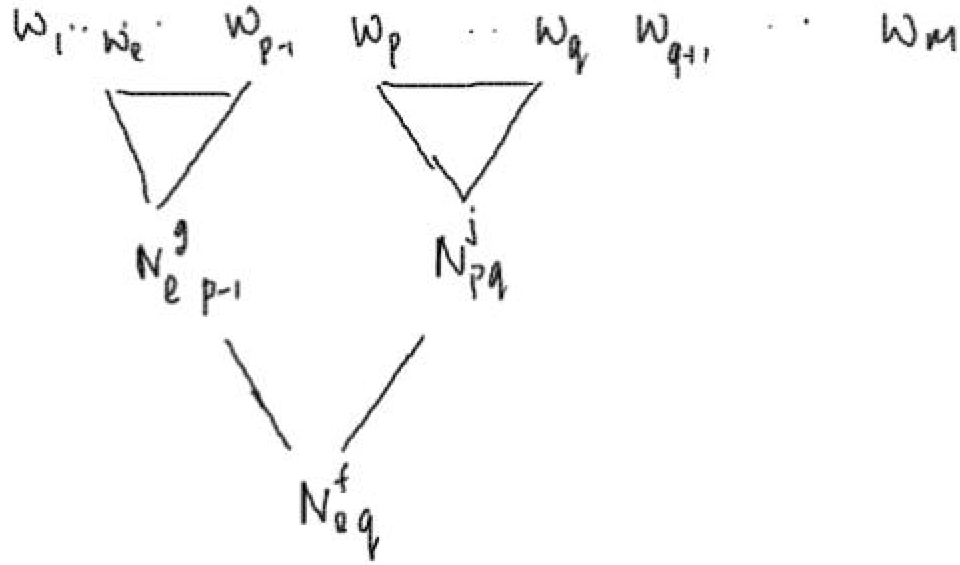
$$w_1 \cdots \cdots w_{p-1} \; w_p \qquad w_q \; w_{q+1} \cdots \cdots w_m$$

we could have



So

or we could have



be careful of $N^j \to N^j N^j$

$$\alpha_j(p,q) = \left[\sum_{f,g \neq j} \sum_{e=q+1}^{m} \Big(\alpha_f(p,e)\, P(N^f \to N^j N^g) \times \beta_g(q+1, e) \Big) \right]$$

$$+ \left[\sum_{fg} \sum_{e=1}^{p-1} \alpha_f(e,q)\, P(N^f \to N^j N^g) \times \beta_g(e, p-1) \right]$$

So we have an algorithm for computing $P(\omega)$

- compute β's bottom up
- (then compute α's top down)

$$P(\omega) = \alpha_1(1 \cdots m) . \beta_1(1 \cdots m)$$
$$= \sum_j \alpha_j(k,k) . \beta_j(k,k)$$

<u>Inference</u>:

- recall HMM
- key is to keep track of the highest probability path from state j at time t, forward
- call this an accumulator

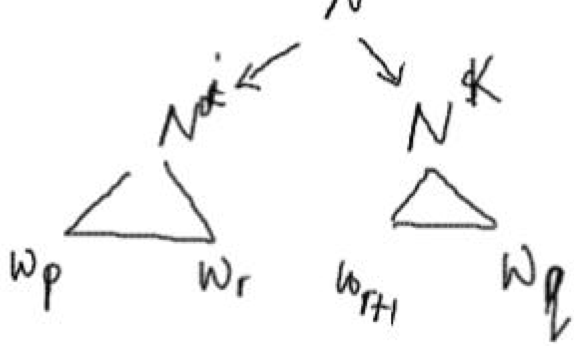
In this case, we want best parse tree spanning a substring rooted with a non-terminal

write

$$\delta_i(p,q) = \text{prob. of highest inside prob parse tree spanning } p,q \text{, using } N^i \text{ at top (i.e. } N^i_{pq})$$

1) $\delta_i(p,p) = P(N^i \to w_p)$

2) we have



so we must have

$$\delta_i(p,q) = \max_{\substack{j,k,\\ p\le r<q}} \left\{ \begin{array}{l} P(N^i \to N^j N^k) \times \\ \delta_j(p,r) \\ \delta_k(r+1,q) \end{array} \right\}$$

how good the best tree is

also, keep

$$\Psi_i(p,q) = \arg\max$$

which is ~~the tree~~ (j,k,r)

3) $\delta_1(1,m)$ is prob of most prob parse

Example

~~We know~~ root is

N^1_{1-m}

a b c

now assume $\delta_i(2,2)$ we $\delta_i(3,3)$ know

$\delta_i(1,1)$

N_{pq} is in , $\Psi_i(p,q) = (j,k,r)$

then $\delta_i(1,2)$ on $\delta_i(3,3)$ the left, we must

have

N^j_{pr} $\delta_i(1,3)$ which is either $\delta_i(1,1)\,\delta_j(2,3)$ or

and on the right $\delta_i(1,2)\,\delta_j(3,3)$

$N^k_{(r+1)\,q}$

Training:

- assume we know terminals, non-terminals, start, all rules
- must now determine rule probs from data
- Do this with EM

Single sentence

$$\hat{P}(N^j \to \zeta) = \frac{\#(N^j \to \zeta)}{\sum_{\gamma} \#(N^j \to \gamma)}$$

Now

$$\alpha_j(p,q)\,\beta_j(p,q) = P(N^1 \Rightarrow w_{1m},\; N^j \Rightarrow w_{pq} \mid \text{Grammar}$$

$$= P(N^1 \Rightarrow w_{1m} \mid G) \times P(N^j \Rightarrow w_{pq} \mid N^1 \Rightarrow w_{1m}, G$$

$$P(N^1 \Rightarrow w_{1m} \mid G) = \alpha_1(1,m)\,\beta_1(1,m) = \pi$$

so:

number ~~fraction~~ of times N^j is used

$$= \sum_{q=1}^{m} \sum_{q=p,}^{m} P(N^j \Rightarrow w_{pq} \mid N^1 \Rightarrow w_{1m}, G)$$

$$= \sum_{q=1}^{m} \sum_{p=1}^{m} \frac{\alpha_j(p,q)\,\beta_j(p,q)}{\pi}$$

Two kinds of rule

nonterms → nonterms I

term II

I: need:

$$P(N^j \to N^r N^s \Rightarrow w_{pq} \mid N^1 \Rightarrow w_{1m}, G)$$

$$= \frac{\sum_{d=p}^{q-1} \alpha_j(p,q)\, P(N^j \to N^r N^s)\, \beta_r(p,d)\, \beta_s(d+1}{\pi}$$

then

number of times $N^j \to N^r N^s$ given N^j

$$= \sum_{p=1}^{m-1} \sum_{q=p+1}^{m} \Big[P(N^j \to N^r N^s \Rightarrow w_{pq} \mid N^1 \Rightarrow w_1$$

which yields a re-estimation formula

II) $\text{count}(N^j \to w^k \mid N^1 \Rightarrow w_{1m}, G)$

$$\cancel{= \frac{\sum_{h=1}^{m} \alpha_j(h,h)\, P(N^j \to w_h, w_h = w}{\pi}}$$

$$= \frac{\sum_{h=1}^{m} \alpha_j(h,h)\, \delta(w_h = w^k) \cdot \beta_j(h,h)}{\pi}$$

which gives the re-est formula.

More than one sentence is only slightly more complex — one must count over all sents.

Important facts:

- Slow:
  - each iteration for 1 sentence costs $O(m^3 n^3)$

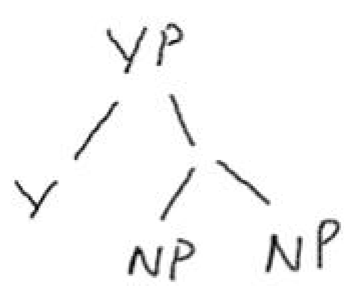
    sent length (m) — number of non-term (n)

- ~~local~~ maxima are a major problem; eg 300 trials give 300 different local maxima (Charniak)

- Some evidence that method ~~prom~~ needs more non-terminals than are strictly needed

<u>Some options</u>:

- <u>Lexicalization</u>:
  - Current parser does not know <u>what</u> verb is involved in, say



We should have rules that know <u>what</u> the word is

- known as lexicalization
- but how much should be known?
- some tension here between lexicalization (one or more rules per word?) and estimation (many instances of a ~~word~~ rule)

- try to break words into classes?

<u>Partially unsupervised learning</u>:

- Hard for us to build an activity treebank (don't know rules)
- But we could <u>segment</u>
-

- Some evidence grammar learning works better on a segmented corpus (Pereira + Schabes; Schabes et al)

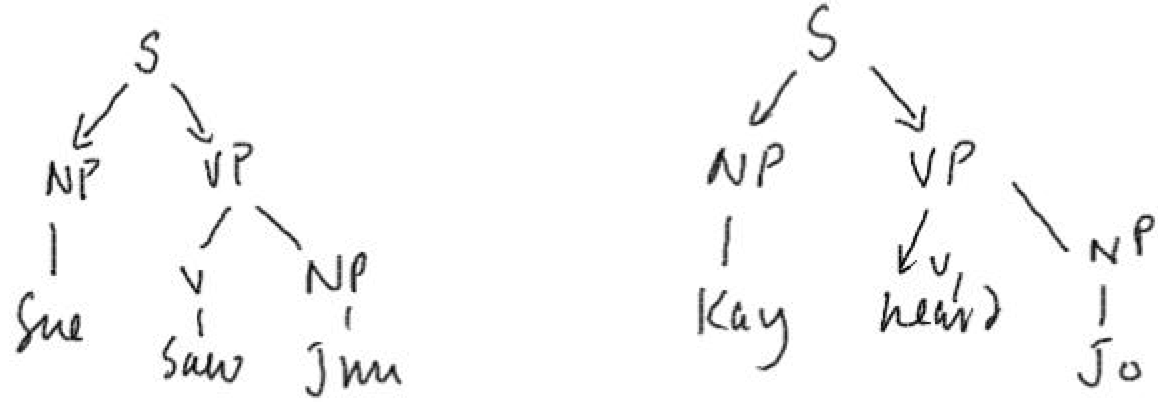
Learning Structure:

- Some evidence MDL methods apply

Data oriented parsing:

- Parse by cut + paste of parse trees

We have



we can (a) make new sents by cut + paste

— This is the essence of parsing
expose what cut + pastes are
syntactically acceptable

+ (b) parse sents (with a very nasty search)