Big attraction:

\[ x_{k+1} = x_k - \alpha_k H_K \nabla f_k \]

\[ H_{k+1} = V_K H_K V_K + p_K S_K S_K^T \]

\[ p_K = \frac{1}{y_k^T S_K} \quad V_K = (I - p_K y_k S_K^T) \]

Now, \( H_k \) is big so we don't want to store it.

**IDEA:**
- store the last \( m \) \((s_i, y)\) pairs
- throw away oldest
- have an initial \( H_0 \) for each step.
Parsing.

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

  "The velocity of the seismic waves rises to..."

- Hard to capture with HMM

```
S -> NP VP
  |    |    |
  NP  PP VP
     /   |
    |     |
  DT NN IN NP
    |     |
  The Velocity of the seismic waves
```
How can we extract this structure?
- complex qin with many partial answers
- one answer: Probabilistic Context Free Grammar

PCFG consists of:
- a set of terminals \( \mathcal{W}^k \), \( k = 1 \ldots N \)
- a set of nonterminals \( \mathcal{N}^i \), \( i = 1 \ldots n \)
- a start symbol \( \mathcal{N}^1 \)
- a set of rules \( \mathcal{N}^i \rightarrow \mathcal{S}^j \)
  \( \mathcal{S}^j \) is a sequence of terminals and non-terminals
- a set of probabilities on rules
  \[ P(\mathcal{N}^i \rightarrow \mathcal{S}^j | \mathcal{N}^i) = P(\mathcal{N}^i \rightarrow \mathcal{S}^j) \]
  note \( \sum_j P(\mathcal{N}^i \rightarrow \mathcal{S}^j) = 1 \)
Notation:

Sentence is $w_1, \ldots, w_m$

span $\text{Wab}$ is $w_a, \ldots, w_b$

if by a set of rewrites we can go from $N^j$ to $\text{Wab}$

then $N^j$ dominates $\text{Wab}$ and $\text{Wab}$ is yield of $N^j$

$N^j_{\text{Wab}}$ means $N^j$ dominates $a \ldots b$

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

all trees that yield $w_{1m}$
Conditions:

need

Place-invariance:

\[ \forall k \quad P(N^j_{k(k+c)} \rightarrow \xi) \text{ is the same } \]

Context-free:

\[ P(N^j_{ke} \rightarrow \xi \mid \text{anything outside } k-e) = P(N^j_{ke} \rightarrow \xi) \]

Ancestor-free:

\[ P(N^j_{ke} \rightarrow \xi \mid \text{any ancestors outside } N^j_{ke}) = P(N^j_{ke} \rightarrow \xi) \]
Under these conditions, computing $P(w, t)$ is straightforward.

Properties:

- Predictive power tends to be greater for language than HMM's with the same number of parameters.

- PCFG's favour smaller trees over larger trees.

- Probability can be wasted on infinite trees.
Example:

\[ S \rightarrow \text{rhubarb} \quad P = \frac{1}{3} \]

\[ S \rightarrow SS \quad P = \frac{2}{3} \]

\[ \rho_{\omega} = \sum_{t} P(\omega, t) = \frac{1}{3} \]

\[ \text{rhubarb rhubarb rhubarb} \quad 2/3 \times \frac{1}{3} \times \frac{1}{3} = \frac{2}{27} \]

\[ \text{rhubarb rhubarb rhubarb} \quad (\frac{2}{13}) \times \left( \frac{1}{3} \right)^2 \times 2 = \frac{8}{243} \]

\[ \sum_{\omega} P(\omega) = \sum_{w,t} P(w, t) \]

\[ = \frac{1}{3} + \frac{2}{27} + \frac{8}{243} + \ldots \]

\[ = \frac{1}{2} \]

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

Eval: \( P(w) \) ?

Inference: \( \text{argmax}_t P(w, t) \)

Learning: rule probs

- Consider only Chomsky Normal Form grammars

Rules are:

\[
N_i \rightarrow N N^k \\
N_i \rightarrow w^j
\]

\( \{ \) only forms available

Parameters are:

\[
P(N_i \rightarrow N^j N^k) \\
P(N_i \rightarrow w^j)
\]

\( n^3 \) table for

\( n \) nonterminals

\( nV \) table for

\( n \) nonterminals

\( V \) terminals

and

\[
\sum_{j, k} P(N_i \rightarrow N^j N^k) + \sum_j P(N_i \rightarrow w^j) = 1
\]
Example:

Probabilistic regular grammar
(which is rather like an HMM)

\[ \begin{align*}
N_i \to & \quad w^j N^k \\
N_i \to & \quad w^j
\end{align*} \]

\[ \begin{align*}
NP \to & \quad N \to \quad N' \to \quad N' \to \quad \text{sink}
\end{align*} \]

Now in an HMM, we worked with

\[ P(w_{t-1}, X_t = c) \]

\[ P(w_{t:T} | X_t = i) \]
All this suggests an approach to PCFG's

Outside prob

\[ \alpha_j(p, q) \overset{\text{forward}}{=} P(w_{pq} | N_j^{pq}) \]

Inside prob

\[ \beta_j(p, q) = P(w_{pq} | N_j^{pq}) \]
This is cutting up string

Notia

\[ \alpha_j(p,q) \cdot \beta_j(p,q) \]

\[ = P(w_1 \ldots m, \text{tree with } N^j \text{ yielding } w_{pq} ) \]

\[ \sum_j \alpha_j(p,q) \beta_j(p,q) \]

\[ = P(w_1 \ldots m, \text{tree with some } N \text{ yielding } w_{pq} ) \]
How to calculate probabilities:

Inside probs:

$$\beta_j(K, K) = P(N^j \rightarrow w_K)$$

Now consider $$\beta_j(p, q)$$

This is a picture of all possible cases.

So

$$\beta_j(p, q) = \sum_{r, s} \left[ \sum_{d=p+1}^{q-1} P(N^j \rightarrow NN^s) \cdot \beta_r(p, d) \cdot \beta_s(d+1) \right]$$
Outside probs:

\[ \alpha (1, m) = 1 \]

\[ = P(\text{tree yielding } w_1 \cdots w_m, \text{ rooted at } N_1) \]

\[ \alpha_j (1, m) = 0 \]

Interior strings

\[ w_1, \ldots, w_p, w_p, w_q, w_q+1, \ldots, w_m \]

We could have

\[ w_1, \ldots, w_{q-1}, w_p, w_q, w_{q+1}, \ldots, w_m \]

\[ N_p q \]

\[ N_q + \]

\[ N_p e \]
So we could have

\[ w_1 \ldots w_{p-1} w_p \ldots w_q w_{q+1} \ldots w_m \]

\[ \begin{array}{c}
N_e \Rightarrow N_{p-1}^g \\
N_{p,q} \Rightarrow N_{q+1}^j \\
N_{q+1} \Rightarrow N_{p-1}^f \\
N \Rightarrow N^j \Rightarrow N^j \Rightarrow N^j
\end{array} \]

be careful of \( N^j \Rightarrow N \Rightarrow N\]

\[ \alpha_j (p, q) = \left[ \sum_{f_{q+1}} \sum_{e=q+1}^m \left( \alpha_f (p, e) P(N \Rightarrow N_{p-1}^j \Rightarrow N_{q+1}^g) \right) \right] \]

\[ + \left[ \sum_{f_{q+1}}^p \sum_{e=1}^q \left( \alpha_f (e, q) P(N \Rightarrow N_{p-1}^j \Rightarrow N_{q+1}^g) \right) \right] \]

\[ \times \beta_g (e, p-1) \]

\[ \times \beta_g (q+1, e) \]
So we have an algorithm for computing $P(w)$

- Compute $\beta$'s bottom up
- Then compute $\alpha$'s top down

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

Inference:

- 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

\[ S_i(p, q) = \text{highest inside prob. parse tree spanning } p, q, \text{ using } N_i \text{ at top} \]

(i.e. \( N_{pq}^i \))

1) \( S_i(p, p) = \Pr(N_i \rightarrow w_p) \)

2) We have

\[ \begin{array}{c}
\text{wp} \\
\downarrow \\
\text{N}^i_k \\
\uparrow \\
\text{N}^i_\text{not} \\
\downarrow \\
\text{wr} \\
\uparrow \\
\text{wr}_{\text{top}} \\
\uparrow \\
\text{wr}_r
\end{array} \]
so we must have

$$\delta_i(p, q) = \max_{j, k, p < r < q} \left\{ \max_{i} \left( P(N \rightarrow N_i N_j N_k) \right) \times \delta_i(p, r) \delta_k(r+1, q) \right\}$$

how good the best tree is

also, keep

$$\psi_i(p, q) = \text{arg max}$$

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

3) \(\delta_i(1, M)\) is \(\text{prob of most probable parse}\)
Example

We know

Now, assume we know

Then, we must have

And on the right

\[ \prod_{r=1}^{k} p_r \]
Training:

- Assume we know terminals, non-terminals, start, all rules.
- Must now determine rule probs from data.
- Do this with EM.

Single sentence

\[ \hat{P}(N^j \rightarrow \gamma) = \frac{\#(N^j \rightarrow \gamma)}{\sum_{\gamma} \#(N^j \rightarrow \gamma)} \]
Now
\[ \alpha_j(p,q) \beta_j(p,q) = P(N \Rightarrow \omega_{im}) \]
\[ N^j \Rightarrow \omega_{pq} \mid \text{Grammar} \]
\[ = P(N^i \Rightarrow \omega_{im} \mid \Gamma) \times \]
\[ P(N^j \Rightarrow \omega_{pq} \mid N^i \Rightarrow \omega_{im}, \Gamma) \]
\[ P(N^i \Rightarrow \omega_{im} \mid \Gamma) = \alpha_i(1,m) \beta_i(1,m) = \pi \]

So:

\[ \text{number of times } N^j \text{ is used} \]
\[ = \sum_{q=1}^{m} \sum_{q=p}^{m} P(N^j \Rightarrow \omega_{pq} \mid N^i \Rightarrow \omega_{im}, \Gamma) \]
\[ = \sum_{q=1}^{m} \sum_{p=1}^{m} \frac{\alpha_j(p,q) \beta_j(p,q)}{\pi} \]
Two kinds of rule

\[ \text{nonterms} \rightarrow \text{nonterms} \quad I \quad \text{term} \quad II \]

I: need:

\[
P(N \rightarrow N' N^S \Rightarrow w_{pq} \mid N' \Rightarrow w_{in}, G) = \sum_{d=p}^{q-1} \alpha_j(p, q) P(N \rightarrow N' N^S) \beta_r(p, d) \beta_s(d+1) \prod \]

then

number of times \( N' \rightarrow N' N^S \) given \( N^j \)

\[
= \sum_{p=1}^{M-1} \sum_{q=p+1}^{M} \left[ P(N^j \rightarrow N' N^S \Rightarrow w_{pq} \mid N^j \Rightarrow w_{ij}) \right]
\]
Which yields a re-estimation formula

$$
\sum_{j} \left( \sum_{h=1}^{m} \alpha_j(h,h) \right) \frac{P(N^j \rightarrow w^k, \omega^1, \ldots, \omega^l)}{\prod_{h=1}^{m} \beta_j(h,h)}
$$

which gives the re-estimation formula.
More than one sentence is only slightly more complex — one must count over all seeds.

Important facts:

- Slow:
  - Each iteration for 1 sentence costs $O(m^3 n^3)$.
  - Sent length $\uparrow$ number of non-term local maxima are a major problem; eg 300 trials give 300 different local maxima (hmm nick)
Some evidence that method prot needs more non-terminals than are strictly needed.

Some options:

- **Lexicalization**:
  - Current parser does not know what verb is involved in, say

We should have rules that know what 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 rule)
- try to break words into classes?

Partially unsupervised learning:

- Hard for us to build an activity treebank (don't know rules)
- But we could segment
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 and paste of parse trees
We have

\[ S \rightarrow NP \rightarrow VP \rightarrow NP \]

\[ S \rightarrow N \rightarrow VP \rightarrow NP \]

We can (a) make new sentences by cut + paste.

- This is the essence of parsing: expose what cut + pastes are syntactically acceptable.

(6) parse sentences (with a very nasty search)