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
How can we extract this structure?

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

PCFG consists of

- a set of terminals \( \& \sigma_k \), \( k = 1 \cdots N \)
- a set of nonterminals \( \& N^i \), \( i = 1 \cdots n \)
- a start symbol \( N^1 \)
- a set of rules \( N^i \rightarrow \sigma^j \)

\( \sigma^j \) is a sequence of terminals and non-terminals

- a set of probabilities on rules

\[ P(N^i \rightarrow \sigma^j | N^i) = P(N^i \rightarrow \sigma^j) \]

note \( \sum_j P(N^i \rightarrow \sigma^j) = 1 \)
Notation:

Sentence is \( w_1 \ldots w_m \)

Span \( \text{wab} \) is \( \text{wa} \ldots \text{wb} \)

If by a set of rewrites we can go from \( \text{N}_j \) to \( \text{wab} \)

then \( \text{N}_j \) dominates \( \text{wab} \) and \( \text{wab} \) is yield of \( \text{N}_j \)

\( \text{N}_{ab}^j \) means \( \text{N}_j \) dominates \( \text{a} \ldots \text{b} \)

\[
P(w_{1:m}) = \sum_{t} P(w_{1:m}, t)
\]

\( \Phi \) all trees that yield \( w_{1:m} \)
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:

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

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

\[ w \quad p(w) = \sum_{t} p(w, t) \]

\[ \text{rhubarb} \quad \frac{1}{3} \]

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

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

\[ \Leftrightarrow \hat{\Lambda} \]

\[ \Leftrightarrow \wedge \]

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

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

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

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

- Evaluation: $P(w)$?
- Inference: $\arg\max_t P(w, t)$
- Learning: rule probs

Consider only Chomsky Normal Form grammars.

Rules are:

\[
\begin{align*}
N^i & \rightarrow N \ N \ N^k \\
N^i & \rightarrow w^j
\end{align*}
\]

Parameters are:

\[
\begin{align*}
P(N^i \rightarrow N \ N^k) \\
P(N^i \rightarrow w^j)
\end{align*}
\]

And

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

Probabilistic regular grammar (which is rather like an HMM)

\[ N^i \rightarrow w^j N^K \]

\[ N^i \rightarrow w^j \]

\[ N^p \rightarrow N \rightarrow N' \rightarrow N' \rightarrow \text{Sink} \]

\[ \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \]

the big brown box

Now in an HMM, we worked with

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

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

Outside prob $\alpha_j(p,q)$ forward
$\quad = P(w_{pq} | N_j^+) \text{ backoff}$

Inside prob $\beta_j(p,q) = P(w_{pq} | N_j^q)$
This is setting up string

\[ N_{i} \]

outside

\[ \begin{align*}
    \text{inside} \\
    w_{i} \quad w_{p-1} \quad w_{p} \quad w_{q} \quad w_{q+1} \quad \ldots \quad w_{m}
\end{align*} \]

Noticia

\[ \alpha_{j}(p, q) \cdot \beta_{j}(p, q) \]

\[ = P(w_{i...m}, \text{tree with } N_{i}^{j} \text{ yielding } w_{p,q}) \]

\[ \sum_{j} \alpha_{j}(p, q) \beta_{j}(p, q) \]

\[ = P(w_{i...m}, \text{tree with some } N \text{ yielding } w_{p,q}) \]
How to calculate probabilities:

Inside probs:

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

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

\[ \text{This is a picture of all possible cases.} \]

So

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

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

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

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

Interior strings

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

We could have

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

\[ \downarrow \]

\[ N_{pq} \]

\[ N_{q+1} \]

\[ N_{pq+1} \]

\[ N_{p2} \]
So we could have

\[ w_1 \ldots w_e \ldots w_p \ldots w_q \ldots w_q^j \ldots w_m \]

\[ \begin{array}{c}
\Delta^g_{e_{p-1}} \\
\Delta^j_{q_{q+1}} \\
N_{e_{q}}^f \\
N^j_{q_{q}} \end{array} \]

be careful of \( N^j \rightarrow N^j N^j \)

\[
\alpha_j^j(p, q) = \left[ \sum_{f_g \neq j} \sum_{e=q+1}^{j} (\alpha_f^f(p, e) P(N^f \rightarrow N_{N^j}^j)) \times \beta_g^g(q+1, e) \right] \\
+ \left[ \sum_{f_g} \sum_{e=1}^{p-1} \alpha_f^f(e, q) P(N^f \rightarrow N^j_{N^j}^j) \times \beta_g^g(e, p-1) \right]
\]
So we have an algorithm for computing $P(w)$:

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

$$\begin{align*}
P(w) &= \alpha_1(1 \ldots m) \cdot \beta_1(1 \ldots m) \\
&= \sum_j \alpha_j(k,k) \cdot \beta_j(k,k)
\end{align*}$$

**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 a best parse tree spanning a substring rooted with a non-terminal write

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

1) \[ S_i(p, p) = P(N^i \rightarrow w_p) \]

2) we have

\[ \begin{array} {c}
\text{wp} \\
\text{wp} \\
\text{wr} \\
\text{wr}_{th1} \\
\text{wr}_{th2} \\
\text{wr} \\
\end{array} \]
so we must have

\[ \Delta_i (p, q) = \max_{j, k, \text{R} \leq r < q} \left\{ P(N_i \rightarrow N_j N_k) \times \right\} \]

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 prob of most prob parse
Example of a root is

\[ \sqrt{N_{1-m}} \]

Now assume we know

\[ S_{(1,1)} \]

is in \( Y_i(p, q) = (j, k) \)

then \( S_{(1,2)} \) or \( S_{(1,3)} \). Left, we must have

\[ \sqrt{N_{1-m}} \]

\[ N_{p, r} S_{(1,3)} \]

which is either \( S_{(1,1)} \) or \( S_{(2,3)} \) or

\[ \sqrt{N_{(r+1)}} \]

\[ S_{(1,2)} \) or \( S_{(5,3)} \).
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
\[ x_j(p, q) \beta_j(p, q) = P(N_i \Rightarrow w_{im}) \]
\[ = P(N_i \Rightarrow w_{im} | G) \times P(N_j \Rightarrow w_{pq} | N_i \Rightarrow w_{im}, G) \]
\[ = \alpha_i(1, m) \beta_i(1, m) = \Pi \]

So:
\[ \text{Number of times } N_j \text{ is used} = \sum_{q=1}^{m} \sum_{p=1}^{m} x_j(p, q) \beta_j(p, q) \]
\[ = \sum_{q=1}^{m} \sum_{p=1}^{m} x_j(p, q) \beta_j(p, q) \frac{\Pi}{\Pi} \]
Two kinds of rule

nonterms $\rightarrow$ nonterms \hspace{0.5cm} \text{I}

\hspace{1cm} \text{II}

I:

need:

\[ P(N^j \rightarrow N^i N^s \rightarrow w_{pq}, \quad |N^i =) w_{im}, G) \]

\[ = \sum_{d=p}^{q-1} \alpha_j(p, q) P(N^i \rightarrow N^i N^s) \beta_r(p, d) \beta_s(d+1) \]

\[ \Pi \]

then

number of times \( N^i \rightarrow N^i N^s \) given \( N^j \)

\[ = \sum_{p=1}^{m-1} \sum_{q=p+1}^{m} \left[ P(N^j \rightarrow N^i N^s \rightarrow w_{pq}, \quad |N^i =) w_{im} \right] \]
Which yields a re-estimation formula

\[
\begin{align*}
\text{count}(N^j \rightarrow w^k \mid N \Rightarrow \omega_{1m}, G) &= \sum_{h=1}^{m} \alpha_j(h, h) P(N^j \rightarrow \omega_h, w_h = \omega^k) \\
&= \sum_{h=1}^{m} \alpha_j(h, h) S(w_h = \omega^k) \beta_j(h, h)
\end{align*}
\]

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

Important facts:

- Slow:
  - each iteration for 1 sentence costs \(O(m^3 n^3)\)
  - sent length \(\uparrow\) \(\Rightarrow\) number of non-term

- Local maxima are a major problem; eg 300 trials give 300 different local maxima (hmm weird)
Some evidence that method `prem` needs more non-terminals than are strictly needed.

Some options:

- **lexicalization**:
  - current parser does not know what verb is involved in, say

```
YP
  /
  Y
  NP  NP
```

We should have rules that know what the verb is.
known as lexicalization

- But how much should be known?

- Some distinction 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+paste of parse trees
we have

```
S
  /\  
NP VP
  /\  /
 Sue saw Jim
```

```
S
  /\  
NP VP
  /\  /
 Kay heard Jo
```

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)