Lecture 10: From EM to PCFGs

Menu

- Cone-heads: the linear model
  - Or: some like it hot!
- Can we ever get stuck? Unstuck?
- Does this work for PCFGs?
- Statistical Parsing with Treebanks I
For Hidden Markov Models

**M step:** get MLE re-estimates

Figure of merit: Re-estimate so as to try to maximize: $p(o_1, o_2, \ldots, o_n)$

What's this called?

Expected values
PCFG Re-estimation

We are now done with the *expectation* step of the EM algorithm

Column K, \( p(\rightarrow H) \); cell K60 = total expected \( H \) days = 18.321

Column J, \( p(\rightarrow C) \); cell J60 = total expected \( C \) days = 14.679
Re-estimating the emission probabilities, \( b \)’s:

- \( p(\rightarrow H, 1) / p(\rightarrow H) \Rightarrow p(1 \mid H) = 1.069/18.321 = 0.0584 \)
- \( p(\rightarrow H, 2) / p(\rightarrow H) \Rightarrow p(2 \mid H) = 7.788/18.321 = 0.4251 \)
- \( p(\rightarrow H, 3) / p(\rightarrow H) \Rightarrow p(3 \mid H) = 0.5165 \)
- \( p(\rightarrow C, 1) / p(\rightarrow C) \Rightarrow p(1 \mid C) = 9.931/14.679 = 0.6765 \)
- \( p(\rightarrow C, 2) / p(\rightarrow C) \Rightarrow p(2 \mid C) = 3.212/14.679 = 0.2188 \)
- \( p(\rightarrow C, 3) / p(\rightarrow C) \Rightarrow p(3 \mid C) = 0.1047 \)

\[\begin{array}{l}
\text{Initial Guess for } b \text{ values} \\
\begin{array}{c|c|c}
p(\ldots | \ldots) & p(\ldots | H) & p(\ldots | C) \\
p(1) & 0.7 & 0.1 \\
p(2) & 0.2 & 0.2 \\
p(3) & 0.1 & 0.7 \\
\end{array}
\end{array}\]

\[\begin{array}{c|c|c}
\text{Re-estimated } b \text{ values} \\
\begin{array}{c|c|c}
p(\ldots | \ldots) & p(\ldots | H) & p(\ldots | C) \\
\text{Iteration #1} & 0.6765 & 0.0584 \\
p(1) & 0.2188 & 0.4251 \\
p(2) & 0.1047 & 0.5165 \\
p(3) & & \\
\end{array}
\end{array}\]

M-step part 2:
What about re-estimating the \( a \)’s, the transition probabilities?

These are also maximum likelihood estimates, given all the ice cream data, that, e.g., today we transitioned from state \( H \) to state \( H \) (that yesterday was hot and today also hot) [likewise for 3 remaining transitions]
Formula for re-estimating the transition probabilities

\[
= \frac{\text{probability all paths } q \to r}{\text{probability all paths}} \\
= \frac{\alpha(q) \cdot p(q \to r) \cdot \beta(r)}{\alpha(C)\beta(C) + \alpha(H)\beta(H)}
\]

The numerator for \(H-H\) on one day: what must we calculate?

We must find the total \(p(\text{paths passing through } H \text{ at the end of yesterday and } H \text{ at the end of today})\), then divide by \(p(\text{all paths thru trellis}) = p(\text{ice cream data})\)

Divide product in red by \(p(\text{ice-cream data}), p(O) = 9.13e^{-19} \text{ (cell I28)}\)
Dividing paths transitioning from $H \rightarrow H$ by all paths gives MLE

So, estimate # of $H-H$ transitions for this one day is:

$\text{Hot} \rightarrow \text{Hot}$

$$\alpha_t(H) \ast a_{H \rightarrow C} \ast b_t(o) \ast \beta_{t+1}(H)$$

$$= 0.063 \ast 0.8 \ast 0.7 \ast 2.514 \times 10^{-17}$$

$$= 0.868$$

(cell U28)

This gives us the MLE transitions day by day..., in cols R-U

Now we just have to normalize this by adding up total for each color and then dividing by the MLE total # of days
\[
p(H \mid H)_{MLE} = \frac{\text{total expected # of } H-H \text{ transitions}}{\text{total expected # of } H \text{ days}}
\]

Total expected # of \( H-H \) transitions =
\[
\sum_{t=0}^{T-1} \alpha_t(H) \cdot a_{H \rightarrow H} \cdot b_H(o_t) \cdot \beta_{t+1}(H) = 15.85 \text{ (U60)}
\]

And finally (whew!) divide by the expected # of \( H \) days (which we already know!)

\[
p(H \mid H) = \frac{\text{total expected # of } H-H \text{ transitions}}{\text{total expected # of } H \text{ days}} = \frac{15.85}{18.321} = 0.8652
\]

\[
\therefore p_{MLE}(H \mid H) = 0.8652
\]
So in all we have for the transition re-estimates:

Col. R, R60/J60  
\[ p(C \mid C) = \frac{p(C \rightarrow C)}{p(\rightarrow C)} \]  
\[ p(C \mid H) = \frac{p(H \rightarrow C)}{p(\rightarrow H)} \]

Col. S, S60/K60  
\[ p(H \mid C) = \frac{p(C \rightarrow H)}{p(\rightarrow C)} \]  
\[ p(H \mid H) = \frac{p(H \rightarrow H)}{p(\rightarrow H)} \]

\[ p(C \mid C) = 0.8957; \ p(C \mid H) = 0.0925; \]
\[ p(H \mid C) = 0.109; \ p(H \mid H) = 0.8652 \]

What about the Start state transitions? \( p(\rightarrow H) \) on day 1= cell K27=0.879

| Initial estimates: | p(...|C) | p(...|H) | p(...|START) |
|--------------------|--------|--------|--------------|
| p(1|...)       | 0.7    | 0.1    |              |
| p(2|...)       | 0.2    | 0.2    |              |
| p(3|...)       | 0.1    | 0.7    |              |
| p(C|...)      | 0.8    | 0.1    | 0.5          |
| p(H|...)      | 0.1    | 0.8    | 0.5          |
| p(STOP|...)   | 0.1    | 0.1    | 0            |

| Iteration 1 of EM: | p(...|C) | p(...|H) |
|--------------------|--------|--------|
| p(1|...)       | 0.6765 | 0.0584 |
| p(2|...)       | 0.2188 | 0.4251 |
| p(3|...)       | 0.1047 | 0.5165 |
| p(C|...)      | 0.8757 | 0.0925 |
| p(H|...)      | 0.109  | 0.8652 |
| p(STOP|...)   | 0.0153 | 0.0423 |

Hurray! We learned something... update for \( p(H|H) \)
We then start with these iteration 1 values and do the E-M steps all over again…

After iteration 2:

|        | p(…|C)     | p(…|H)    | p(…|START) |
|--------|-----------|----------|------------|
| p(1|...)  | 0.6813    | 0.0262   |            |
| p(2|...)  | 0.1567    | 0.4892   |            |
| p(3|...)  | 0.162     | 0.4846   |            |
| p(C|...)  | 0.9191    | 0.0714   | 6.7E-04    |
| p(H|...)  | 0.0808    | 0.8717   | 0.9993     |
| p(STOP|...) | 1.4E-04   | 0.0569   | 0          |

After iteration 10:

|        | p(…|C)    | p(…|H)     | p(…|START) |
|--------|---------|-----------|------------|
| p(1|...)  | 0.6407  | 1.6E-04   |            |
| p(2|...)  | 0.1481  | 0.5341    |            |
| p(3|...)  | 0.2112  | 0.4657    |            |
| p(C|...)  | 0.9337  | 0.0718    | 1.3E-13    |
| p(H|...)  | 0.0663  | 0.865     | 1.0        |
| p(STOP|...) | 2.6E-14 | 0.0632    | 0          |

6.863J/9.611J Fall 2012 Lecture 10

Weather States that Best Explain Ice Cream Consumption

10 Iterations

6.863J/9.611J Fall 2012 Lecture 10
At each iteration, we improve the language model, $p(Obs \ sequence) = p(\text{weather sequence})$

To compress the range of the graph, this is not a plot of $p(O)$, but rather perplexity per observation: $1/\sqrt[34]{p(O)} = 2^{-\text{log}_2 p(O)/34}$
Sensitivity to initial conditions: Does this always work? 3 kinds of tweaking

- What happens if the initial guesses are different?

Original

|   | p(...|C) | p(...|H) | p(...|START) |
|---|------|------|------------|
| p(C...) | 0.7  | 0.1  |            |
| p(H...) | 0.2  | 0.2  |            |
| p(3...)  | 0.1  | 0.7  |            |
| p(2...)  | 0.8  | 0.1  | 0.5        |
| p(1...)  | 0.1  | 0.8  | 0.5        |
| p(STOP|...) | 0.1  | 0.1  | 0          |

Revised—guess: no weather ‘inertia’

|   | p(...|C) | p(...|H) | p(...|START) |
|---|------|------|------------|
| p(C...) | 0.7  | 0.1  |            |
| p(H...) | 0.2  | 0.2  |            |
| p(3...)  | 0.1  | 0.7  |            |
| p(2...)  | 0.8  | 0.1  | 0.5        |
| p(1...)  | 0.1  | 0.8  | 0.5        |
| p(STOP|...) | 0.1  | 0.1  | 0          |

Lousy at first! But...

15 iters

10 Iterations
Sensitivity test #2: guess only slight ice-cream pref on hot days

- What happens if the initial guesses are different?

<table>
<thead>
<tr>
<th>Original</th>
<th>Revised—guess: <em>slight</em> ice-cream pref</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>p(1</td>
</tr>
<tr>
<td>p(2</td>
<td>…)</td>
</tr>
<tr>
<td>p(3</td>
<td>…)</td>
</tr>
<tr>
<td>p(C</td>
<td>…)</td>
</tr>
<tr>
<td>p(H</td>
<td>…)</td>
</tr>
<tr>
<td>p(STOP</td>
<td>…)</td>
</tr>
</tbody>
</table>

Lousy at first! But... if at first you don’t succeed, iter, iter again

10 iters – EM has **learned** ice-cream prefs...
Can EM ever go wrong? Sensitivity test

#3: completely symmetric initial guesses

<table>
<thead>
<tr>
<th>p(p^-1)</th>
<th>p(C^-1)</th>
<th>p(H^-1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
</tr>
<tr>
<td>0.4</td>
<td>0.4</td>
<td>0.3</td>
</tr>
<tr>
<td>0.45</td>
<td>0.45</td>
<td>0.5</td>
</tr>
<tr>
<td>0.45</td>
<td>0.45</td>
<td>0.5</td>
</tr>
<tr>
<td>0.1</td>
<td>0.1</td>
<td>0.0</td>
</tr>
</tbody>
</table>

No preference at all for more ice cream on hot days.

No weather inertia either. In short, the model is completely symmetric.

CONVERGENCE!

Weather States that Best Explain Ice Cream Consumption

Can EM ever lose completely?

NO CONVERGENCE!

(an occupational hazard)
Sensitivity Test #4: break symmetry so a slight preference for more ice-cream on cold days

|       | p(…|C) | p(…|H) | p(…|START) |
|-------|------|-------|------------|
| p(1)  | 0.3  | 0.4   | 0.3        |
| p(2)  | 0.3  | 0.3   | 0.4        |
| p(3)  | 0.4  | 0.3   | 0.4        |
| p(C)  | 0.45 | 0.45  | 0.5        |
| p(H)  | 0.45 | 0.45  | 0.5        |
| p(STOP)| 0.5  | 0.5   | 0.0        |

What happens?

It takes 35 iters, but we get this

|       | p(…|C) | p(…|H) | p(…|START) |
|-------|------|-------|------------|
| p(1)  | 0.0011 | 0.644 | 0.1476     |
| p(2)  | 0.032 | 0.1476 | 0.0496     |
| p(3)  | 0.8623 | 0.0702 | 1.0        |
| p(C)  | 0.075 | 0.9298 | 4.8E-09    |
| p(H)  | 0.0627 | 3.7E-11 | 0         |
| p(STOP)| 0.0627 | 3.7E-11 | 0         |

Is this model OK?

What is it saying?
Which language model is best? #1 or #2?
How do we determine ‘best’ Anyway?

#1

Perplexity per day of # of cones

0.5 1.5 2.5 3.5

#2

Same perplexity for both models: 2.827/day
Guarantees…

- EM, aka ‘forward-backward iteration’ converges to a local maximum
- In this space, there are 2 symmetric local maxima, both with the same perplexity of 2.87/day
- Q: How much better is 2.827 than a model with no temporal structure?
- A: if no temporal structure, like a unigram model (say, perplexity 3, because equal guess about 3 diff't observations)

Overfitting? Yes…where?

| p(…|C) | p(…|H) | p(…|START) |
|-------|-------|------------|
| p(1|…)  | 0.6407 | 1.6E-04    |
| p(2|…)  | 0.1481 | 0.5341     |
| p(3|…)  | 0.2112 | 0.4657     |
| p(C|…)  | 0.9337 | 0.0718     | 1.3E-13    |
| p(H|…)  | 0.0663 | 0.865      | 1.0        |
| p(STOP|…) | 2.6E-14 | 0.0632    | 0          |

Q: Is there a fix?
A: smooth the fractional counts!
As an initial state, consider a world where the weather is perfectly symmetric, i.e., no inertia. Is this state the best representation of the data? To explore this question, we will break the symmetry and observe the model's behavior.

### Breaking Symmetry Toward the Opposite Solution

1. **Initial State**: Perfect symmetry implies no weather bias, i.e., the model will randomly decide between hot and cold days.
2. **Symmetry Break**: To break this symmetry, we introduce a slight preference for some weather conditions. For instance, we could start with a probability of 0.7 for hot days over cold days.
3. **Model Learning**: Over several passes, the model learns to adjust its probabilities to reflect the actual weather patterns. Initially, the model might not perfectly predict ice cream consumption, but over time, it will adapt to the data.

### Observations

- **Initial Learning Phase**: The model begins to show a slight preference for ice cream consumption on hot days, even without explicit training for this behavior.
- **Symmetry Break**: By breaking the symmetry, the model can learn to distinguish between different weather conditions and adjust its predictions accordingly.
- **Long-term Adaptation**: As the model continues to learn, it adapts to the actual weather patterns, leading to more accurate predictions of ice cream consumption.

### Key Points

- **Symmetry vs. Asymmetry**: The difference between symmetric and asymmetric initial states demonstrates how the model adapts to different patterns of data.
- **Model Flexibility**: Breaking symmetry allows the model to learn more complex patterns, improving its overall predictive accuracy.
- **Real-world Application**: This method can be applied to various real-world scenarios where symmetry in data might not hold, leading to more accurate predictions.
So in fact…we *could* do better!

- There are *two* kinds of structure co-existing in this dataset:
  1. Days with a lot (or little) ice-cream tend to repeat
  2. Days with 2 ice-creams tend *not* to repeat
- The first kind of structure did better at lowering perplexity, but *both* do some work
- Q: how could we get our model to distinguish *both* kinds of structure, so lowering perplexity further?
- A: Use more states – 4 would distinguish hot/2, cold/2, hot/not2, and cold/not 2 days

Envoi…to ice-cream

- **Q:** We have now seen three locally optimal models in which the $H$ state was used for 3 different things—even though we named it H for “Hot.” What does this mean for the application of this algorithm to part-of-speech tagging?

- **A:** There is no guarantee that N and V will continue to distinguish nouns and verbs after re-estimation. They will evolve to make whatever distinctions help to predict the word sequence
- Further: we might want to have algorithm find mixtures itself (“latent” variables)
Summary so far: what you should know
• How does EM perform unsupervised learning & what makes this possible?
• EM alternates expectation & maximization steps
• EM maximizes $p(\text{observed training data})$
• EM does local gradient ascent in likelihood space, finds only a local maximum; it might not converge to a global max & is sensitive to initial conditions
• EM cannot escape zeroes or symmetries, so these should be avoided in initial conditions
• EM uses states as it sees fit, ignoring suggestive names
• EM may overfit the training data, unless smoothed

Now back to language…
• We now have a method for unsupervised learning of finite-state ie, linear structure:

![Diagram showing Forward Probability $\alpha_t(j)$ and Backward Probability $\beta_t(j)$]
What about EM for PCFGs?

- Start with a ‘pretty good’ guessed grammar (e.g., trained on treebank, or some simple model)
- Parse corpus of unparsed sentences

EM for PCFGs

- Similar, but now we consider all parses of each sentence
- Parse our corpus of unparsed sentences:
  
  - Collect counts fractionally:
    - \( c(S \rightarrow NP \ VP) += 10.8 \); \( c(S) += 2*10.8 \); …
    - \( c(S \rightarrow NP \ VP) += 1.2 \); \( c(S) += 1*1.2 \); …
  
  - But there may be exponentially many parses of a length-n sentence!
    - How can we stay fast? Similar to taggings…

# copies of this sentence in the corpus

Today stocks were up

Today stocks were up
What about EM for hierarchical structure?

- Yes! But we must now be very very clever about what α and β are…now “outside” and “inside” probs
- Consider this parse of *time flies like an arrow today* and focus on the VP *flies like an arrow*

  \[
  \alpha_{VP} = \text{“outside” the VP (from Start to ’here’= prob VP is at end of this point, words ‘outside’ it match up)}
  \]

  \[
  \alpha_{VP}(1,5) = p(\text{time [VP] today} | S)
  \]

  \[
  \beta_{VP} = \text{“inside” the VP (from ’here’ to end = prob VP derives words under it & match)}
  \]

  \[
  \beta_{VP}(1,5) = p(\text{flies like an arrow} | VP)
  \]

Inside and Outside probabilities

- **Exactly** analogous to finite-state case of ‘paths’
- What does \( \alpha \ast \beta \) mean?

  \[
  \alpha_{VP}(1,5) = p(\text{time [VP] today} | S)
  \]

  \[
  \beta_{VP}(1,5) = p(\text{flies like an arrow} | VP)
  \]

  \[
  \alpha_{VP}(1,5) \ast \beta_{VP}(1,5) = p(\text{time [VP flies like an arrow] today} | S)
  \]

  So, \( \alpha \ast \beta \) is the whole probability for one parse tree that has a VP from word 1 to 5.

  What is the analog of all paths?

  \[
  \beta_S(0,6) \text{ or } \alpha_S(0,6)
  \]
Inside and Outside probabilities

• Exactly analogous to finite-state case of ‘paths’
• Take total prob of parse paths that go thru this VP & divide by total prob of all parse paths from S, $\beta_S(0,6)$ or $\alpha_S(0,6)$, so: $\alpha(1,5) * \beta(1,5) / \beta(0,6)$

$$p(\text{time flies like an arrow today} & \text{ VP(1,5) | S})$$

$$p(\text{time flies like an arrow today} | S)$$

How can we compute $\alpha$ and $\beta$ efficiently?

---

Recall CKY parsing

```plaintext
function PROBABILISTIC-CKY(words, grammar) returns most probable parse and its probability
for $j$ ← from 1 to LENGTH(words) do
    for all { $A \mid A \rightarrow \text{words}[j] \in \text{grammar}$ }
        $table[j-1, j, A] ← P(A \rightarrow \text{words}[j])$
    for $i$ ← from $j-2$ downto 0 do
        for $k ← i+1$ to $j-1$ do
            for all { $A \mid A \rightarrow BC \in \text{grammar}$, and $table[i, j, B] > 0$ and $table[k, j, C] > 0$ }
                if $table[i, j, A] < P(A \rightarrow BC) \times table[i, k, B] \times table[k, j, C]$ then
                    $table[i, j, A] ← P(A \rightarrow BC) \times table[i, k, B] \times table[k, j, C]$
                    $back[i, j, A] ← \{k, B, C\}$
            return BUILD-TREE($back[1, \text{LENGTH(words)}, S]$, $table[1, \text{LENGTH(words)}, S]$)
```
### Compute $\beta$ bottom-up via CKY

#### Time 0

<table>
<thead>
<tr>
<th>NP</th>
<th>Vst</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>13</td>
</tr>
</tbody>
</table>

#### Time 1

<table>
<thead>
<tr>
<th>NP</th>
<th>VP</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

#### Time 2

<table>
<thead>
<tr>
<th>P</th>
<th>V</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>5</td>
</tr>
</tbody>
</table>

#### Time 3

<table>
<thead>
<tr>
<th>Det</th>
<th>NP</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
</tr>
</tbody>
</table>

#### Time 4

<table>
<thead>
<tr>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
</tr>
</tbody>
</table>

---

### Compute $\beta$ bottom-up via CKY

#### Time 0

<table>
<thead>
<tr>
<th>NP</th>
<th>Vst</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-3</td>
<td>2-3</td>
<td>2-8</td>
</tr>
<tr>
<td>2-3</td>
<td>2-8</td>
<td>2-13</td>
</tr>
</tbody>
</table>

#### Time 1

<table>
<thead>
<tr>
<th>NP</th>
<th>VP</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-4</td>
<td>2-4</td>
</tr>
</tbody>
</table>

#### Time 2

<table>
<thead>
<tr>
<th>P</th>
<th>V</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-2</td>
<td>2-5</td>
</tr>
</tbody>
</table>

#### Time 3

<table>
<thead>
<tr>
<th>Det</th>
<th>NP</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-1</td>
<td>10</td>
</tr>
</tbody>
</table>

#### Time 4

<table>
<thead>
<tr>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-8</td>
</tr>
</tbody>
</table>
The Efficient version: add as you go

<table>
<thead>
<tr>
<th>time</th>
<th>flies</th>
<th>like</th>
<th>an</th>
<th>arrow</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>NP 2(^{-3})</td>
<td>Vst 2(^{-3})</td>
<td>NP 2(^{-10})</td>
<td>NP 2(^{-24})</td>
</tr>
<tr>
<td></td>
<td>S 2(^{-8})</td>
<td>S 2(^{-13})</td>
<td>S 2(^{-8})</td>
<td>S 2(^{-22})</td>
</tr>
<tr>
<td></td>
<td>S 2(^{-13})</td>
<td>S 2(^{-22})</td>
<td>S 2(^{-27})</td>
<td>S 2(^{-27})</td>
</tr>
</tbody>
</table>

2\(^{1}\) S → NP VP
2\(^{6}\) S → Vst NP
2\(^{2}\) S → S PP
2\(^{1}\) VP → V NP
2\(^{2}\) VP → VP PP
2\(^{1}\) NP → Det N
2\(^{2}\) NP → NP PP
2\(^{3}\) NP → NP NP
2\(^{0}\) PP → P NP

Beta Function:

\(\beta_{S}(0,2)\)

\(\beta_{S}(0,2)\) * \(\beta_{pp}(2,5)\) * \(p(S \rightarrow S PP | S)\)

\(\beta_{S}(0,5)\)
Compute $\beta$ bottom-up using CKY

need some initialization up here for the width-1 case
for width := 2 to n
  (* build smallest first *)
for i := 0 to n-width
  (* start *)
  let k := i + width
  (* end *)
for j := i+1 to k-1
  (* middle *)
  for all grammar rules $X \rightarrow Y Z$
    \[ \beta_X(i,k) = \sum p(X \rightarrow Y Z | X) \ast \beta_Y(i,j) \ast \beta_Z(j,k) \]

Proof of the pudding..

Simple grammar:

```
1.0  S1 → S
1.0  S → NP VP
1.0  NP → Det N
1.0  VP → V
1.0  VP → V NP
1.0  VP → NP V
1.0  VP → V NP NP
1.0  VP → NP NP V
```

```
1.0  Det → the
1.0  N → the
1.0  V → the
1.0  Det → a
1.0  N → a
1.0  V → a
1.0  Det → dog
1.0  N → dog
1.0  V → dog
1.0  Det → man
1.0  N → man
1.0  V → man
```
Test sentences from 2 ‘languages’

- **Language 1:**
  - the dog bites a man
  - the man bites a dog
  - a man gives the dog a bone
  - the dog gives a man the bone
  - a dog bites a bone

- **Language 2:**
  - the dog a man bites
  - the man a dog bites
  - a man the dog a bone gives
  - a man the dog a bone gives
  - a dog a bone bites

Run on Language 1 results

1   S1 → S
1   S → NP VP
1   NP → Det N
0.6 VP → V NP
0.4 VP → V NP NP
0.416667 Det → the
0.583333 Det → a
0.416667 N → dog
0.333333 N → man
0.25 N → bone
0.6 V → bites
0.4 V → gives
Scaling up: Learning PCFGs by EM

- ATIS treebank consists of 1,300 hand-constructed parse trees input consists of POS tags rather than words
- about 1,000 PCFG rules are needed to build these trees

Probability of Training Strings

$\log P$

Iteration
Evaluating parser accuracy

- Most sentences are not given a completely correct parse by any currently existing parsers: total % correct, is just 70% or so
- Standardly then, evaluation is done in terms of the percentage of correct phrases (labeled spans) [label, start, finish] (e.g., [S, 0, 6])
- So, parser must try to get as many of these correct as possible, in a given sentence (why not bad: you don’t mess up whole parse, but other large parts still useful in the application)
- Labeled precision, labeled recall of constituents
Labeled Precision and Recall

GOLD

PARSE

Measure constituent match

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>27</td>
<td>0</td>
<td>89.47</td>
<td>100.00</td>
<td>17</td>
<td>19</td>
<td>17</td>
</tr>
</tbody>
</table>

**GOLD:** [S, 0, 27], [CC, 0, 1], [NP, 1, 3], [VP, 3, 16], [NP, 4, 7], [QP, 4, 6].

**PARSE:** [S, 0, 27], [CC, 0, 1], [NP, 1, 3], [VP, 3, 16], [NP, 4, 7], [PP, 6, 10].

17/19 = 89.47%
How good are PCFGs?

- Robust: parse everything, but with low probability
- Partial solution to grammar ambiguity: some idea of plausibility of a sentence
- But not so good (as we have seen and will see) because independence assumptions are too strong
- Yields a probabilistic language model – but in simple case it performs worse than a trigram model!
- WSJ parsing accuracy: 73% precision/recall
- Why not so good?
- A: lack word-based probabilities of trigram

PCFGs and Words

- A PCFG uses the actual words only to determine the probability of parts of speech
- We need to know about these words to choose a parse
- Head word gives good information
  Attachment ambiguities:
  *the astronomer saw the moon with the telescope*
- Coordination: *the dogs in the house and the cats*
- Verb ‘frames’: *put* vs. *like*:
  *Put [book] [on table] vs. likes [book on table]"
PCFGs & Independence – Accurate unlexicalized parsing

- The symbols in a PCFG define independence assumptions:
  $$S \rightarrow NP \ VP$$
  $$NP \rightarrow DT \ NN$$

Independence Assumptions Too Strong redux

- Expansion of an NP is highly dependent on the parent of the NP (subject vs. object)

<table>
<thead>
<tr>
<th>All NPs</th>
<th>NPs under S’s (subjects)</th>
<th>NPs under VPs (objects)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NP PP</td>
<td>DT NN PRP</td>
<td>NP PP DT NN PRP</td>
</tr>
<tr>
<td>11%</td>
<td>9% 6%</td>
<td>27% 7% 4%</td>
</tr>
</tbody>
</table>
Lack of transmission of information

Regular PCFG

S

NP

VP

DT

NN

V

NP

the

lawyer

questioned

the

witness

Lexicalized PCFG

S

NP(lawyer)

VP(questioned)

DT

NN

V

NP(witness)

the

lawyer

questioned

the

witness

What me worry?

• Who cares?
  • N-grams, HMMs, all make false assumptions
  • For generation/LMs, these consequences will be obvious (as you have seen in CGW!)
  • So, for parsing, do they impact accuracy?
• Symptoms of overly strong assumptions
  • CFG rewrite rules used where they don’t belong
  • Rules used too often or too rarely

Correct:

NP

NNP

NNP

JJ

NN

NP

big

board

corporate

trading

What you get:

NP

JNJ

NN

NP

NP

NP

corporate

board

big

trading

Why?
Breaking up the symbols

- Relax independence assumptions by encoding dependencies into the PCFG rules
- Mark possessive NPs as a new nonterminal, break down independence barrier

- What are the most useful features to encode?

Part of a General Strategy: Horizontal Markovization (what do vanilla PCFGs have?)

- This merges states

Reduces grammar size, first-order improves scores by 1%
Horizontal Markovization

- Ordinary PCFG: first-order vertical (depends on lefthand side nonterminal)
- Parent annotation: 2\textsuperscript{nd} order; improves from 73 to 77.8\%
- Could do more: grandparent, etc, yields 78\% but # of symbols starts getting large

Results

<table>
<thead>
<tr>
<th>Vertical Order</th>
<th>Horizontal Markov Order</th>
<th>$h = 0$</th>
<th>$h = 1$</th>
<th>$h \leq 2$</th>
<th>$h = 2$</th>
<th>$h = \infty$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v = 1$ No annotation</td>
<td></td>
<td>71.27</td>
<td>72.5</td>
<td>73.46</td>
<td>72.96</td>
<td>72.62</td>
</tr>
<tr>
<td>$v \leq 2$ Sel. Parents</td>
<td></td>
<td>(854)</td>
<td>(3119)</td>
<td>(3863)</td>
<td>(6207)</td>
<td>(9657)</td>
</tr>
<tr>
<td>$v = 2$ All Parents</td>
<td></td>
<td>74.75</td>
<td>77.42</td>
<td>77.77</td>
<td>77.50</td>
<td>76.91</td>
</tr>
<tr>
<td>$v \leq 3$ Sel. GParents</td>
<td></td>
<td>(2285)</td>
<td>(6564)</td>
<td>(7619)</td>
<td>(11398)</td>
<td>(14247)</td>
</tr>
<tr>
<td>$v = 3$ All GParents</td>
<td></td>
<td>76.50</td>
<td>78.59</td>
<td>79.07</td>
<td>78.97</td>
<td>78.54</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(4943)</td>
<td>(12374)</td>
<td>(13627)</td>
<td>(19545)</td>
<td>(20123)</td>
</tr>
</tbody>
</table>

6.863J/9.611J Fall 2012 Lecture 10
Why only children can go wrong: unary rules
‘transmute’ categories (here S to VP) so that a high
prob VP rule gets used…

Solution: Mark unary rules without parent
by –U

Marking the only child

• ^U (external unary) “I am the only child.”
• -U (internal unary) “I have only one child.”
• On the preterminal level (POS \( \rightarrow \) word), external
unary mark-up helps with
  • demonstratives (that, this) vs. articles (a, the)
    — both labeled as DT in Penn TreeBank
  • adverbs (e.g., also vs. as well).
Part of speech tag splitting: POS tags too coarse!

- Parent annotation also for preterminal tags
- Splitting of IN tags into 6 linguistically motivated groups (prepositions vs. conjunctions vs. complementizers (*that*); noun-modifying vs. primarily verb-modifying prepositions (of vs. as))
- Distinction between auxiliaries have and be.
  Special conjunction class containing but/But
- &. % get their own tags

A winner with this example

(a)  
(b)
Bottom line?

• Using all these tricks, without lexicalization: F1 of 86.3%

• What does it mean to have an ‘unlexicalized’ grammar anyway?
• Can we find the right features ‘automatically’?
• Ans: Yes!
• Answer again: latent feature ‘discovery’ (to come): toss in the kitchen sink & let statistical method find some combo that works best
• Next time: statistical Treebank parsers & lexicalization