The Big Concept

Problem: Too many rules!
- Especially with lexicalization and flattening (which help).
- So it’s hard to estimate probabilities.

Solution: Related rules tend to have related probs
- POSSIBLE relationships are given a priori
- LEARN which relationships are strong in this language

Method has connections to:
- Parameterized finite-state machines (Monday’s talk)
- Bayesian networks (inference, abduction, explaining away)
- Linguistic theory (transformations, metarules, etc.)

Want to parse (or build a syntactic language model).
Must estimate rule probabilities.

Problem: Too Many Rules

Too Many Rules ... But Luckily ...

All these rules for fund - & other, still unobserved rules - are connected by the deep structure of English.
Rules Are Related

- fund behaves like a typical singular noun ...
- ... or transitive verb ...
- ... but as noun, has an idiosyncratic fondness for purpose clauses ...
- ... and maybe other idiosyncrasies to be discovered, like unaccusativity ...

One fact! though PCFG represents it as many apparently unrelated rules.

Rules Are Related

- fund behaves like a typical singular noun ...
- ... or transitive verb ...
- ... but as noun, has an idiosyncratic fondness for purpose clauses ...

Predicts dozens of unseen rules

Rules Are Related

- fund behaves like a typical singular noun ...
- ... or transitive verb ...
- ... but as noun, has an idiosyncratic fondness for purpose clauses ...

One more fact! even if several more rules. Verb rules are RELATED. Should be able to PREDICT the ones we haven’t seen.

All This Is Quantitative!

- fund behaves like a typical singular noun ...
- ... or transitive verb ...
- ... but as noun, has an idiosyncratic fondness for purpose clauses ...
- ... and maybe other idiosyncrasies to be discovered, like unaccusativity ...

How often? and how does that tell us p(rule)?
Format of the Rules

Why use flat rules?
- Avoids silly independence assumptions: a win
  - Johnson 1998 → New experiments
  - Our method likes them
    - Traditional rules aren’t systematically related
    - But relationships exist among wide, flat rules that express different ways of filling same roles

\[ S \rightarrow \text{NP} \text{, NP} \text{ put PP} \]
The Rule Smoothing Task

Input: Rule counts (from parses or putative parses)
Output: Probability distribution over rules
Evaluation: Perplexity of held-out rule counts
That is, did we assign high probability to the rules needed to correctly parse test data?

Grid of Lexicalized Rules

S → ... encourage question fund merge repay remove
To → NP
To → NP PP
To AdvP → NP PP
To → PP
NP → NP
NP → NP PP
NP Md → NP
NP Md → PP PP
NP → SBar
(etc.)

S → To fund NP PP
(“to fund projects with ease”)
S → To merge NP PP
(“to merge projects with ease”)

Naive prob. estimates (MLE model)

<table>
<thead>
<tr>
<th>S</th>
<th>encourage</th>
<th>question</th>
<th>fund</th>
<th>merge</th>
<th>repay</th>
<th>remove</th>
</tr>
</thead>
<tbody>
<tr>
<td>To → NP</td>
<td>200</td>
<td>167</td>
<td>714</td>
<td>250</td>
<td>600</td>
<td>333</td>
</tr>
<tr>
<td>To → NP PP</td>
<td>200</td>
<td>167</td>
<td>286</td>
<td>500</td>
<td>200</td>
<td>167</td>
</tr>
<tr>
<td>To AdvP → NP PP</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>NP → NP</td>
<td>0</td>
<td>333</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>NP → NP PP</td>
<td>200</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>NP Md → NP PPTmp</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>NP Md → PP PP</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>NP → S Bar</td>
<td>200</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(other)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Estimate of p(frame | word) * 1000

Training Counts

<table>
<thead>
<tr>
<th>S</th>
<th>encourage</th>
<th>question</th>
<th>fund</th>
<th>merge</th>
<th>repay</th>
<th>remove</th>
</tr>
</thead>
<tbody>
<tr>
<td>To → NP</td>
<td>142</td>
<td>117</td>
<td>397</td>
<td>210</td>
<td>329</td>
<td>222</td>
</tr>
<tr>
<td>To → NP PP</td>
<td>77</td>
<td>64</td>
<td>120</td>
<td>181</td>
<td>329</td>
<td>80</td>
</tr>
<tr>
<td>To AdvP → NP PP</td>
<td>0.18</td>
<td>0.15</td>
<td>0.33</td>
<td>0.37</td>
<td>0.26</td>
<td>50</td>
</tr>
<tr>
<td>NP → NP</td>
<td>22</td>
<td>161</td>
<td>7.8</td>
<td>7.5</td>
<td>7.9</td>
<td>7.5</td>
</tr>
<tr>
<td>NP → NP PP</td>
<td>8.5</td>
<td>2.6</td>
<td>2.7</td>
<td>2.6</td>
<td>2.6</td>
<td></td>
</tr>
<tr>
<td>NP Md → NP</td>
<td>90</td>
<td>2.1</td>
<td>2.4</td>
<td>2.0</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>NP Md → NP PPTmp</td>
<td>1.8</td>
<td>0.16</td>
<td>0.17</td>
<td>0.16</td>
<td>0.19</td>
<td></td>
</tr>
<tr>
<td>NP Md → PP PP</td>
<td>0.1</td>
<td>0.027</td>
<td>0.027</td>
<td>0.038</td>
<td>0.078</td>
<td>59</td>
</tr>
<tr>
<td>To → PP</td>
<td>9.2</td>
<td>6.5</td>
<td>126</td>
<td>10</td>
<td>9.1</td>
<td></td>
</tr>
<tr>
<td>To → S</td>
<td>98</td>
<td>1.6</td>
<td>4.3</td>
<td>3.9</td>
<td>3.6</td>
<td>2.7</td>
</tr>
<tr>
<td>NP → S Bar</td>
<td>3.4</td>
<td>190</td>
<td>3.2</td>
<td>3.2</td>
<td>3.2</td>
<td>3.2</td>
</tr>
<tr>
<td>(other)</td>
<td>478</td>
<td>449</td>
<td>449</td>
<td>461</td>
<td>461</td>
<td>482</td>
</tr>
</tbody>
</table>

Estimate of p(frame | word) * 1000

The Rule Smoothing Task

Input: Rule counts (from parses or putative parses)
Output: Probability distribution over rules
Evaluation: Perplexity of held-out rule counts
Rule probabilities: \( p(S \rightarrow NP \text{ put } NP \text{ PP } | S, \text{ put}) \)
Infinite set of possible rules; so we will estimate
\( p(S \rightarrow NP \text{ AdvP } \text{ put } \text{ PP PP PP NP AdjP S } | S, \text{ put}) \)
= a very tiny number > 0

To —— NP
To —— NP PP
To AdvP —— NP
To AdvP —— NP PP
NP —— NP .
NP —— NP PP .
NP Md —— NP
NP Md —— NP PPTmp
NP Md —— PP PP
NP —— SBar .
(etc.)

Naive prob. estimates (MLE model)

<table>
<thead>
<tr>
<th>S</th>
<th>encourage</th>
<th>question</th>
<th>fund</th>
<th>merge</th>
<th>repay</th>
<th>remove</th>
</tr>
</thead>
<tbody>
<tr>
<td>To → NP</td>
<td>200</td>
<td>167</td>
<td>714</td>
<td>250</td>
<td>600</td>
<td>333</td>
</tr>
<tr>
<td>To → NP PP</td>
<td>200</td>
<td>167</td>
<td>286</td>
<td>500</td>
<td>200</td>
<td>167</td>
</tr>
<tr>
<td>To AdvP → NP PP</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>NP → NP</td>
<td>0</td>
<td>333</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>NP → NP PP</td>
<td>200</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>NP Md → NP PPTmp</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>NP Md → PP PP</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>NP → S Bar</td>
<td>200</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(other)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Estimate of p(frame | word) * 1000

TASK: counts → probs (“smoothing”)
**Smooth Matrix via LSA / SVD, or SBS?**

<table>
<thead>
<tr>
<th>S →</th>
<th>encourage</th>
<th>question</th>
<th>fund</th>
<th>merge</th>
<th>reply</th>
<th>remove</th>
</tr>
</thead>
<tbody>
<tr>
<td>To → NP</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>To → NP PP</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>To AdvP → NP</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>To AdvP → NP PP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NP → NP</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NP → NP PP</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NP Md → NP</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NP Md → NP PPTmp</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NP Md → PP PP</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>To → PP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>To → S</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NP → SBar</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Count of (word, frame)

**Only a Few Deep Facts**

- **fund** behaves like a transitive verb 10% of time ...
- and noun 90% of time ...
- ... takes purpose clauses 5 times as often as typical noun.

**Smoothing via a Bayesian Prior**

- Choose grammar to maximize
  \[ p(\text{observed rule counts} | \text{grammar}) \cdot p(\text{grammar}) \]

- grammar = probability distribution over rules

- **Our job:** Define \( p(\text{grammar}) \)

- **Question:** What makes a grammar likely, a priori?

- **This paper’s answer:** Systematicity. Rules are mainly derivable from other rules. Relatively few stipulations ("deep facts").

**Smoothing via a Bayesian Prior**

- Previous work (several papers in past decade):
  - Rules should be few, short, and approx. equiprobable
  - These priors try to keep rules out of grammar
  - Bad idea for lexicalized grammars ...

- This work:
  - Prior tries to get related rules into grammar
  - transitive \( \Rightarrow \) passive at \(-1/20\) the probability
  - NSF spragges the project \( \Rightarrow \) The project is spragged by NSF
  - Would be weird for the passive to be missing, and prior knows it!
  - In fact, weird if \( p(\text{passive}) \) is too far from \( 1/20 \cdot p(\text{active}) \)

- Few facts, not few rules!

**for now, stick to Simple Edit Transformations**

See paper for various evidence that these should be predictive.

\[ S \rightarrow NP \quad \text{see} \]

1. I see
2. I see you
3. I see you with my own eyes

**Smoothing via a Bayesian Prior**

- \( p(S \rightarrow NP \quad \text{see} \quad \text{SBAR} \quad \text{PP}) \)

\[ = 0.5 \cdot 0.1 \cdot 0.1 \cdot 0.4 + 0.1 \cdot 0.01 \ldots \]
Arc Probabilities: A Conditional Log-Linear Model

To make sure outgoing arcs sum to 1, introduce a normalizing factor $Z$ (at each vertex).

$$\frac{1}{Z} \exp \theta_3 + \theta_5 + \theta_8$$

Models $p(\text{arc} | \text{vertex})$

Not enough just to say “Insert PP.”

Each arc bears several features, whose weights determine its probability.

$$\frac{1}{Z} \exp \theta_3 + \theta_5 + \theta_8$$

feature weights

A feature of weight 0 has no effect.

Raising a feature’s weight strengthens all arcs with that feature.

Arc Probabilities: A Conditional Log-Linear Model

Infinitely Many Arc Probabilities: Derive From Finite Parameter Set

Could get mixture behavior by adjusting start probs.

But not quite right - can’t handle negative exceptions within a paradigm.

And what of the language’s transformation probs?

Why not just give any two PP-insertion arcs the same probability?

Arc Probabilities: A Conditional Log-Linear Model

Arc Probabilities: A Conditional Log-Linear Model

Not enough just to say “Insert PP.”

Each arc bears several features, whose weights determine its probability.

$$\frac{1}{Z} \exp \theta_3 + \theta_5 + \theta_8$$

feature weights

A feature of weight 0 has no effect.

Raising a feature’s weight strengthens all arcs with that feature.

Arc Probabilities: A Conditional Log-Linear Model

Infinitely Many Arc Probabilities: Derive From Finite Parameter Set

Could get mixture behavior by adjusting start probs.

But not quite right - can’t handle negative exceptions within a paradigm.

And what of the language’s transformation probs?

Why not just give any two PP-insertion arcs the same probability?

Arc Probabilities: A Conditional Log-Linear Model

Arc Probabilities: A Conditional Log-Linear Model

Not enough just to say “Insert PP.”

Each arc bears several features, whose weights determine its probability.

$$\frac{1}{Z} \exp \theta_3 + \theta_5 + \theta_8$$

feature weights

A feature of weight 0 has no effect.

Raising a feature’s weight strengthens all arcs with that feature.

Arc Probabilities: A Conditional Log-Linear Model

Infinitely Many Arc Probabilities: Derive From Finite Parameter Set

Could get mixture behavior by adjusting start probs.

But not quite right - can’t handle negative exceptions within a paradigm.

And what of the language’s transformation probs?

Why not just give any two PP-insertion arcs the same probability?
Arc Probabilities: A Conditional Log-Linear Model

\[ \frac{1}{Z} \exp (\theta_3 + \theta_5 + \theta_7) \]

\[ \frac{1}{Z} \exp (\theta_3 + \theta_5 + \theta_7) \]

\[ \frac{1}{Z} \exp (\theta_3 + \theta_5 + \theta_7) \]

These arcs share most features.
So their probabilities tend to rise and fall together.
To fit data, could manipulate them independently (via \(\theta_5, \theta_6\)).

Prior Distribution

- PCFG grammar is determined by \(\theta_0, \theta_1, \theta_2, \ldots\)

Universal Grammar

Grammar is determined by \(\theta_0, \theta_1, \theta_2, \ldots\)

Our prior: \(\theta_1 \sim N(0, \sigma^2)\), IID

Thus: \(-\log p(\text{grammar}) = c + (\theta_0^2 + \theta_1^2 + \theta_2^2 + \ldots)/\sigma^2\)

So good grammars have few large weights.
Prior prefers one generalization to many exceptions.

Prior Distribution
Arc Probabilities: A Conditional Log-Linear Model

\[ \frac{1}{Z} \exp \theta_3 + \theta_5 + \theta_7 \]

\[ \frac{1}{Z} \exp \theta_3 + \theta_5 + \theta_7 \]

To raise both rules’ probs, cheaper to use \( \theta_3 \) than both \( \theta_5 \) & \( \theta_7 \). This generalizes – also raises other cases of PP-insertion!

Other models of this string:
- max-likelihood
- n-gram
- Collins arg/adj hybrids

Reparameterization

- Grammar is determined by \( \theta_0, \theta_1, \theta_2, \ldots \)
- A priori, the \( \theta_i \) are normally distributed

- We’ve reparameterized!
- The parameters are feature weights \( \theta_i \), not rule probabilities
- Important tendencies captured in big weights
  - Similarly: Fourier transform – find the formants
  - Similarly: SVD – find the principal components
  - It’s on this deep level that we want to compare events, impose priors, etc.
**Simple Bigram Model (Eisner 1996)**

- A parser assumes tree is probable if its component rules are:

- Try assuming rule is probable if its component bigrams are:

  \[
  p(A | \text{start}) \times p(B | A) \\
  \times p(C | B) \times p(D | C) \\
  \times p(\text{stop} | D)
  \]

- Markov process, 1 symbol of memory, conditioned on L, w, side of ——

- One-count backoff to handle sparse data (Chen & Goodman 1996)

  \[
  p(L \rightarrow ABC—D | w) = p(L | w) \times p(A, B, C, D | L, w)
  \]

**Perplexity: Predicting test frames**

- 20% further reduction from previous lit.
- Can get big perplexity reduction just by flattening.

**Perplexity: Predicting test frames**

- Best model with transformations
- Best model without transformations

- Test rules with 0 training observations
  - Best model with transformations
  - Best model without transformations

- Test rules with 1 training observation
  - Best model with transformations
  - Best model without transformations
**Forced matching task**

- Test model’s ability to extrapolate novel frames for a word
- Randomly select **two** (word, frame) pairs from test data
  - … ensuring that neither frame was ever seen in training
- Ask model to choose a matching:
  
  \[
  \begin{align*}
  \text{word 1} & \quad \text{frame A} & \quad \text{word 1} & \quad \text{frame A} \\
  \text{word 2} & \quad \text{frame B} & \quad \text{word 2} & \quad \text{frame B}
  \end{align*}
  \]

  i.e., does frame A look more like word 1’s known frames or word 2’s?

  - 20% fewer errors than bigram model

**Summary: Reparameterize PCFG in terms of deep transformation weights, to be learned under a simple prior.**

- **Problem:** Too many rules!
  - Especially with lexicalization and flattening (which help).
  - So it’s hard to estimate probabilities.
- **Solution:** Related rules tend to have related probs
  - **POSSIBLE** relationships are given a priori
  - **LEARN** which relationships are strong in this language
    (just like feature selection)
- Method has connections to:
  - Parameterized finite-state machines (Monday’s talk)
  - Bayesian networks (inference, abduction, explaining away)
  - Linguistic theory (transformations, metarules, etc.)

**Graceful degradation**

- Twice as much data but no transformations

**FIN**