Talk outline

• Motivation for and applications of stochastic grammars

• Discriminative training of stochastic grammars
  – supervised training from parsed corpora
  – unsupervised training from sentence-aligned bitext

• Avoiding enumerating parses
  – Packed parse representations
  – Feature locality
  – Dynamic programming using graphical model techniques
Why combine grammars and statistics?

• Language is used to convey information
  – Grammars capture the form-meaning mapping

• Interpretation is dependent on many interacting factors
  – Grammar is about expressing linguistic constraints

• Ambiguity is pervasive in language
  – Statistics is the theory of inference under uncertainty

• Learning the grammar of a language is a prerequisite
  – Language learning is a statistical inference problem
What can we do with SUBGs?

• Identify most likely parses (focused information retrieval)

• Machine translation (find most likely translation)

• Language modelling for speech recognition and OCR
  – Requires joint models
Two problems of non-statistical CL

1. *Ambiguity explodes combinatorially*

   (162) *Even though it’s possible to scan using the Auto Image Enhance mode, it’s best to use the normal scan mode to scan your documents.*

   • Refining the grammar is often self-defeating
     ⇒ splits states ⇒ makes the problem worse!

   • Preference information guides parser to correct analysis

2. *Requiring linguistic well-formedness leads to non-robustness*

   • Perfectly comprehensible sentences receive no parses
Conventional approaches to robustness

- We want to be able to analyse ill-formed input, e.g. *He walk*.
  - Ignoring agreement ⇒ spurious ambiguity
    \[ I\ saw\ the\ father\ of\ the\ children\ that\ speak(s)\ French \]

- Extra-grammatical rules, repair mechanisms, …
  - How can semantic interpretation take place without a well-formed syntactic analysis?

- A preference-based approach provides a systematic treatment of robustness too!
Generation with ranked analyses

- Probability distribution over phonology/semantics pairs \( \omega \in \Omega \)
- Generation optimizes conditional probability of phonological output given the semantic input \( s \).

\[
\text{Generate}(s) = \arg \max_{\omega} P(\omega \mid \text{semantics}(\omega) = s)
\]
Parsing with ranked analyses

- Parsing optimizes the conditional probability of the semantics given the phonological form $p$

\[
\text{Parse}(p) = \arg\max_{\omega} P(\omega \mid \text{phonology}(\omega) = p)
\]

![Diagram showing ranked analyses with increasing optimality](image)
Robustness and ranked interpretations

• Parsing and generation involve *different* conditional distributions!

• Grammar pairs “ungrammatical” sentences with interpretations

Phonology

Semantics

never generated, but interpretable!

increasing optimality
Learning & comprehension involves inference

- Both language learning and language comprehension require identifying “hidden” properties of the input.
- The input is (apparently) compatible with different hidden structures.
- Statistical inference may succeed even if there is insufficient information for deductive approaches.
- Ranked analyses provide a systematic treatment of preferences and robustness.
Linguistic knowledge and statistical parsing

- Statistical parsers are *not* “linguistics-free”
  - Conditioning features
  - Syntactic annotations in training data

- What is the most effective way to import useful linguistic knowledge?
  - *manually* specify possible linguistic structures
  - manually specify statistical features
  - learn feature weights from training data
Statistical learning and parsing

- Grammar defines (universally) possible linguistic structures $\Omega$
- Family of probability distributions $P_\theta$ on $\Omega$ parameterized by $\theta$
- Learning involves finding $\theta$ which makes the input most likely
- Given $\theta$ and a yield (terminal string) $y$, parsing involves finding most probable structure in $\{\omega | Y(\omega) = y\}$
- How can we define such probability distributions?
- Computationally efficient inference?
PCFGs and relative frequency estimator

rule | count | rel freq
---|---|---
S → NP VP | 3 | 1
NP → rice | 2 | 2/3
NP → corn | 1 | 1/3
VP → grows | 3 | 1

\[
P \left( \begin{array}{c}
S \\
NP \\
rice \\
VP \\
grows \\
\end{array} \right) = \frac{2}{3}
\]

\[
P \left( \begin{array}{c}
S \\
NP \\
corn \\
VP \\
grows \\
\end{array} \right) = \frac{1}{3}
\]
Non-local constraints

\[
P\left(\begin{array}{c}
S \\
NP \\
rice \\
VP \\
grows \\
\end{array}\right) = \frac{4}{9}
\]

\[
P\left(\begin{array}{c}
S \\
NP \\
bananas \\
VP \\
grow \\
\end{array}\right) = \frac{1}{9}
\]

\[
Z = \frac{5}{9}
\]
Renormalization

\[
P \left( \begin{array}{c}
S \\
\text{NP} & \text{VP} \\
\text{rice} & \text{grows}
\end{array} \right) = \frac{4}{9} \quad \frac{4}{5}
\]

\[
P \left( \begin{array}{c}
S \\
\text{NP} & \text{VP} \\
\text{bananas} & \text{grow}
\end{array} \right) = \frac{1}{9} \quad \frac{1}{5}
\]

\[
Z = \frac{5}{9}
\]

<table>
<thead>
<tr>
<th>rule</th>
<th>count</th>
<th>rel freq</th>
</tr>
</thead>
<tbody>
<tr>
<td>S → NP VP</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>NP → rice</td>
<td>2</td>
<td>\frac{2}{3}</td>
</tr>
<tr>
<td>NP → bananas</td>
<td>1</td>
<td>\frac{1}{3}</td>
</tr>
<tr>
<td>VP → grows</td>
<td>2</td>
<td>\frac{2}{3}</td>
</tr>
<tr>
<td>VP → grow</td>
<td>1</td>
<td>\frac{1}{3}</td>
</tr>
</tbody>
</table>
Other values do better!

\[
\begin{array}{c}
\text{rule} & \text{count} & \text{rel freq} \\
S \rightarrow \text{NP VP} & 3 & 1 \\
\text{NP} \rightarrow \text{rice} & 2 & 2/3 \\
\text{NP} \rightarrow \text{bananas} & 1 & 1/3 \\
\text{VP} \rightarrow \text{grows} & 2 & 1/2 \\
\text{VP} \rightarrow \text{grow} & 1 & 1/2 \\
\end{array}
\]

\[
P \left( \begin{array}{c}
\text{S} \\
\text{NP} \\
\text{VP} \\
\text{rice} \\
grows
\end{array} \right) = \frac{2}{6} \quad \frac{2}{3}
\]

\[
P \left( \begin{array}{c}
\text{S} \\
\text{NP} \\
\text{VP} \\
\text{bananas} \\
grow
\end{array} \right) = \frac{1}{6} \quad \frac{1}{3}
\]

\[Z = \frac{3}{6}\]
Make dependencies local – GPSG-style

<table>
<thead>
<tr>
<th>rule</th>
<th>count</th>
<th>rel freq</th>
</tr>
</thead>
<tbody>
<tr>
<td>S → NP +sing +sing</td>
<td>2</td>
<td>2/3</td>
</tr>
<tr>
<td>S → NP +plu +plu</td>
<td>1</td>
<td>1/3</td>
</tr>
<tr>
<td>NP +sing → rice</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>NP +plu → bananas</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>VP +sing → grows</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>VP +plu → grow</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Summary

All dependencies are local or context-free:

- rules are "natural" features of probability distribution
- relative rule frequency is MLE

Structures exhibit non-local dependencies:

- no easy way to obtain "natural" features
- with renormalization, relative frequency estimator is not MLE
  - MLE is much more complicated
- this estimator handles non-rule and rule features
  \( \Rightarrow \text{no need to restrict attention to rule features} \)
Log linear models

- The **log likelihood** is a **linear** function of feature values
- \( \Omega = \) set of syntactic structures (not necessarily trees)
- \( f_j(\omega) = \) number of occurrences of \( j \)th feature in \( \omega \in \Omega \)
  (feature \( \neq \) attribute)
- \( \lambda_j \) are “feature weight” parameters

\[
W_\lambda(\omega) = \exp\left(\sum_{j=1}^{m} \lambda_j f_j(\omega)\right)
\]

\[
Z_\lambda = \sum_{\omega \in \Omega} W_\lambda(\omega)
\]

\[
P_\lambda(\omega) = \frac{W_\lambda(\omega)}{Z_\lambda}
\]

\[
\log P_\lambda(\omega) = \sum_{j=1}^{m} \lambda_j f_j(\omega) - \log Z_\lambda
\]
PCFGs are log-linear models

\( \Omega = \text{set of all trees generated by } G \)

\( f_j(\omega) = \text{number of times the } j\text{th rule is used in } \omega \in \Omega \)

\( \theta_j = \text{probability of } j\text{th rule in } G \quad \lambda_j = \log \theta_j \)

\[
f \left( \begin{array}{c}
S \\
NP & VP \\
rice & grows
\end{array} \right) = \begin{bmatrix}
1 \\
S \rightarrow \text{NP} \\
1 \\
\text{NP} \rightarrow \text{rice} \\
0 \\
\text{NP} \rightarrow \text{bananas} \\
1 \\
\text{VP} \rightarrow \text{grows} \\
0 \\
\text{VP} \rightarrow \text{grow}
\end{bmatrix}
\]

\[
P_\theta(\omega) = \prod_{j=1}^{m} \theta_j^{f_j(\omega)} = \exp(\sum_{j=1}^{m} \lambda_j f_j(\omega)) \quad \text{where } \lambda_j = \log \theta_j
\]
Stochastic Lexical-Functional Grammar

- Unification-based grammar (competence) defines well-formed syntactic structures $\Omega$
  - In SLFG, these are c-structure/f-structure pairs

- Stochastic model (performance) defines a probability distribution over $\Omega$
  - Features $f_1, \ldots, f_m$, where each $f_j$ maps each $\omega \in \Omega$ to a feature occurrence count $f_j(\omega)$
  - Probability distribution defined by log linear model

$$\log P_\lambda(\omega) = \sum_{j=1}^{m} \lambda_j f_j(\omega) - \log Z_\lambda$$

- Same approach applies to virtually any theory of grammar
Sample parses

TURN
SEGMENT
ROOT PERIOD
Sadj .
S
VPv
V NP VPv
let PRON V NP
us take DATEP
N COMMA DATEnum
Tuesday ,
the fifteenth

SENTENCE_ID BAC002_E

OBJ

PASSIVE-
PRED LET(2,10)9
STMT-TYPE IMPERATIVE

SUBJ

TNS-ASP [MOOD IMPERATIVE]

XCOMP

OBJ

22
Features used

**Rule features:** For every non-terminal $X$, $f_X(\omega)$ is the number of times $X$ occurs in c-structure of $\omega$

**Attribute value features:** For every attribute $a$ and every atomic value $v$, $f_{a=v}(\omega)$ is the number of times the pair $a = v$ appears in $\omega$

**Argument and adjunct features:** For every grammatical function $g$, $f_g(\omega)$ is the number of times that $g$ appears in $\omega$

**Other features:** Dates, times, locations; right branching; attachment location; parallelism in coordination; . . .

Features are *not* independent, but dependency structure is unknown.
ML estimation for log linear models

Training data \( D = \omega_1, \ldots, \omega_n \)

\[
\hat{\lambda} = \arg\max_{\lambda} L_D(\lambda)
\]

\[
L_D(\lambda) = \prod_{i=1}^{n} P_{\lambda}(\omega_i)
\]

\[
P_{\lambda}(\omega) = \frac{W_{\lambda}(\omega)}{Z_{\lambda}} \quad W_{\lambda}(\omega) = \exp(\sum_j \lambda_j f_j(\omega)) \quad Z_{\lambda} = \sum_{\omega' \in \Omega} W_{\lambda}(\omega')
\]

• For a PCFG, \( \hat{\lambda} \) is easy to calculate, but …
• in general \( \partial L_D/\partial \lambda_j \) and \( Z_{\lambda} \) are intractable analytically and numerically
• Abney (1997) suggests a Monte-Carlo calculation method
Pseudo-likelihood

The **pseudo-likelihood** of $\omega$ is the *conditional probability* of the *hidden part* (syntactic structure) $\omega$ given its *visible part* (yield or terminal string) $y = Y(\omega)$ (Besag 1974)

\[ \Omega(y_i) = \{ \omega : Y(\omega) = Y(\omega_i) \} \]

\[ \hat{\lambda} = \arg\max_{\lambda} \text{PL}_D(\lambda) \]

\[ \text{PL}_D(\lambda) = \prod_{i=1}^{n} P_{\lambda}(\omega_i | y_i) \]

\[ P_{\lambda}(\omega | y) = \frac{W_{\lambda}(\omega)}{Z_{\lambda}(y)} \]

\[ W_{\lambda}(\omega) = \exp(\sum_{j} \lambda_j f_j(\omega)) \]

\[ Z_{\lambda}(y) = \sum_{\omega' \in \Omega(y)} W_{\lambda}(\omega') \]
Pseudo-likelihood versus likelihood

- The pseudo-partition function $Z_\lambda(y)$ is *much easier to compute* than the partition function $Z_\lambda$
  - $Z_\lambda$ requires a sum over $\Omega$
  - $Z_\lambda(y)$ requires a sum over $\Omega_y$ (parses of $y$)
- Maximum *likelihood* estimates full joint distribution
  - learns distribution of both yields and parses given yields
- Maximum *pseudo-likelihood* estimates a conditional distribution
  - learns distribution of *parses given yields*, but not yields
  - conditional distribution is what you need for parsing
  - cognitively more plausible?
- Maximizing pseudo-likelihood *does not* maximize likelihood
  - PL estimator is *consistent for the conditional distribution*
Pseudo-likelihood estimation

<table>
<thead>
<tr>
<th>Correct parse’s features</th>
<th>All other parses’ features</th>
</tr>
</thead>
<tbody>
<tr>
<td>sentence 1 [1, 3, 2]</td>
<td>[2, 2, 3] [3, 1, 5] [2, 6, 3]</td>
</tr>
<tr>
<td>sentence 2 [7, 2, 1]</td>
<td>[2, 5, 5]</td>
</tr>
<tr>
<td>sentence 3 [2, 4, 2]</td>
<td>[1, 1, 7] [7, 2, 1]</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

- Training data is fully observed (i.e., parsed data)
- Choose \( \lambda \) to maximize (log) likelihood of correct parses relative to other parses
- Distribution of sentences is ignored
Pseudo-constant features are uninformative

<table>
<thead>
<tr>
<th></th>
<th>Correct parse’s features</th>
<th>All other parses’ features</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>sentence 1</strong></td>
<td>[1, 3, 2]</td>
<td>[2, 2, 2] [3, 1, 2] [2, 6, 2]</td>
</tr>
<tr>
<td><strong>sentence 2</strong></td>
<td>[7, 2, 5]</td>
<td>[2, 5, 5]</td>
</tr>
<tr>
<td><strong>sentence 3</strong></td>
<td>[2, 4, 4]</td>
<td>[1, 1, 4] [7, 2, 4]</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

- *Pseudo-constant features* are identical within every set of parses
- They contribute the same constant factor to each parses’ likelihood
- They do not distinguish parses of any sentence ⇒ irrelevant
Pseudo-maximal features ⇒ unbounded $\hat{\lambda}_j$

<table>
<thead>
<tr>
<th></th>
<th>Correct parse’s features</th>
<th>All other parses’ features</th>
</tr>
</thead>
<tbody>
<tr>
<td>sentence 1</td>
<td>[1, 3, 2]</td>
<td>[2, 3, 4] [3, 1, 1] [2, 1, 1]</td>
</tr>
<tr>
<td>sentence 2</td>
<td>[2, 7, 4]</td>
<td>[3, 7, 2]</td>
</tr>
<tr>
<td>sentence 3</td>
<td>[2, 4, 4]</td>
<td>[1, 1, 1] [1, 2, 4]</td>
</tr>
</tbody>
</table>

- A *pseudo-maximal feature* always reaches its maximum value within a parse on the correct parse

- If $f_j$ is pseudo-maximal, $\hat{\lambda}_j \to \infty$ (hard constraint)

- If $f_j$ is pseudo-minimal, $\hat{\lambda}_j \to -\infty$ (hard constraint)
Regularization

- $f_j$ is pseudo-maximal over training data $\nRightarrow f_j$ is pseudo-maximal over all of $\Omega$ (sparse data)

- Regularization: add bias term to ensure optimal $\lambda_j$ is finite
  Multiply the pseudo-likelihood by a zero-mean normal with diagonal covariance

$$\hat{\lambda} = \arg\max_{\lambda} \log P(L_D(\lambda) - \sum_{j=1}^{m} \frac{\lambda_j^2}{2\sigma_j^2}$$

where $\sigma_j$ is 7 times the maximum value of $f_j$ found in the corpus
Stochastic LFG experiment

- Two parsed LFG corpora provided by Xerox PARC
- Grammars unavailable, but corpus contains all parses and hand-identified correct parse
- Features chosen by inspecting Verbmobil corpus only

<table>
<thead>
<tr>
<th></th>
<th>Verbmobil corpus</th>
<th>Homecentre corpus</th>
</tr>
</thead>
<tbody>
<tr>
<td># of sentences</td>
<td>540</td>
<td>980</td>
</tr>
<tr>
<td># of ambiguous sentences</td>
<td>324</td>
<td>424</td>
</tr>
<tr>
<td>Av. length of ambig. sentences</td>
<td>13.8</td>
<td>13.1</td>
</tr>
<tr>
<td># of parses</td>
<td>3245</td>
<td>2865</td>
</tr>
<tr>
<td># of features</td>
<td>191</td>
<td>227</td>
</tr>
<tr>
<td># of rule features</td>
<td>59</td>
<td>57</td>
</tr>
</tbody>
</table>
# Pseudo-likelihood estimator evaluation

<table>
<thead>
<tr>
<th></th>
<th>Verbmobil corpus</th>
<th></th>
<th>Homecentre corpus</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>324 sentences</td>
<td></td>
<td>424 sentences</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$C$</td>
<td>$-\log PL$</td>
<td>$C$</td>
<td>$-\log PL$</td>
</tr>
<tr>
<td>Baseline estimator</td>
<td>88.8</td>
<td>533.2</td>
<td>136.9</td>
<td>590.7</td>
</tr>
<tr>
<td>Pseudo-likelihood estimator</td>
<td>180.0</td>
<td>401.3</td>
<td>283.25</td>
<td>580.6</td>
</tr>
</tbody>
</table>

- Test corpus only contains sentences with more than one parse
- $C$ is the number of maximum likelihood parses of held-out test corpus that were the correct parses
- 10-fold cross-validation evaluation
- Combined system performance: 75% of MAP parses are correct
What have we achieved?

+ Log linear framework applies to *any* theory of grammar
+ Pseudo-likelihood estimator is practical for grammars with thousands of analyses/sentence
+ Features can be anything “read off” a structure
+ Systematic treatment of preferences

? Where’s the linguistic structure gone? Probability distribution determined solely by feature count vector

− Parser is just as non-robust or “brittle” as before
  → Re-express hard linguistic constraints as soft constraints
Summary

- Log-linear models provide a general way of defining probability distributions in the face of context-sensitive dependencies.
- The pseudo-likelihood estimator is computationally tractable for realistic LFGs.
- Auxiliary distributions provide a principled way of incorporating other distributional information.
- The combined LFG parser + log linear model obtains the correct parse on 73% of Verbmobil and almost 80% of Homecentre corpus sentences.
PL estimation and hidden data

- PL estimation *ignores* distribution of strings

$\Rightarrow$ Cannot learn from strings alone

<table>
<thead>
<tr>
<th></th>
<th>maximizes likelihood of</th>
<th>relative to</th>
</tr>
</thead>
<tbody>
<tr>
<td>ML</td>
<td>$\omega_i$</td>
<td>$\Omega$</td>
</tr>
<tr>
<td>PL</td>
<td>$\omega_i$</td>
<td>$\Omega(y_i)$</td>
</tr>
<tr>
<td>EM</td>
<td>$\Omega(y_i)$</td>
<td>$\Omega$</td>
</tr>
<tr>
<td>PL+EM</td>
<td>$\Omega(y_i)$</td>
<td>$\Omega(y_i)$</td>
</tr>
</tbody>
</table>
Psychologically-realistic conditional models

- **Joint** models $P(\omega)$ predict *what is said* and *how it is said*

- **Modularity:** These two processes are very different!

- Conditional models in SUBGs: $P(S|Y)$
  $(S = \text{semantics}, Y = \text{phonology})$

- A *psycholinguistically realistic* statistical model
  - World model: $P(S)$
  - Linguistic model: $P(Y|S)$

- Parsing with such models:

  $$P(S|Y) \propto P(Y|S)P(S)$$
Language acquisition as parameter estimation

- Ω contains every sentence structure from every possible human language

- Each type of syntactic construction is associated with a parameter
  
  Verb initial \( \lambda_{VI} > 0 \) \([S \text{ Kim } [VP \text{ will love Sandy}]]\)
  
  Verb final \( \lambda_{VF} > 0 \) \([S \text{ Kim } [VP \text{ Sandy love will}]]\)
  
  Verb second \( \lambda_{V2} > 0 \) \([S \text{ Kim will } [VP \text{ Sandy love}]]\)

- Learning a language involves learning which constructions it possesses
PL estimation is cognitively unnatural

- PL estimation requires *parsed input*
  - Correct parse of “NP V NP” identifies $\lambda_{V2}$ value
    \[
    [S \text{ Kim loves } [VP \text{ Sandy}]] \Rightarrow \lambda_{V2} > 0 \\
    [S \text{ Kim } [VP \text{ loves Sandy}]] \Rightarrow \lambda_{V2} < 0
    \]
  - Unrealistic to assume child has access to parsed input

- PL estimator only learns from *ambiguous sentences*
  - $[S \text{ Kim } [VP \text{ Sandy love will}]]$ is uninformative to PL

- But *unambiguous sentences* are sometimes most informative!
Components of a representation

• A representation projects several components (random variables)
  – yield $Y(\omega)$, semantics $S(\omega)$

• Pseudo-likelihood can be defined with respect to each of these
  – $\Omega(y) = \{\omega | Y(\omega) = y\}$ and $\Omega(s) = \{\omega | S(\omega) = s\}$ are small and enumerable for many grammars

  $\Rightarrow$ estimation is computationally feasible

• These sets can be used to define a wide variety of estimators
Semantic pseudo-likelihood

\[ \Omega(y_i) = \{ \omega : Y(\omega) = y_i \} \]

\[ \Omega(s_i) = \{ \omega : S(\omega) = s_i \} \]

- Assume learner has access to semantics \( s_i \) and correct parse \( \omega_i \)
- Treat the semantics \( s_i \) as visible component
- Pseudo-likelihood with semantic comparison set

\[ \text{PL}'_D(\lambda) = \prod_{i=1}^{n} P_\lambda(\omega_i|s_i) \]

- Learns when a semantics can be expressed in several ways cross-linguistically

\((\text{love}(\text{Sandy}, \text{Sasha})) \Rightarrow [_{S \text{ Sandy}} [_{VP \text{ Sasha}} \text{ love}]])) \Rightarrow \lambda_{VF} > 0\)
Partially observed data

\[ \Omega(y_i) = \{ \omega : Y(\omega) = y_i \} \]
\[ \Omega(s_i) = \{ \omega : S(\omega) = s_i \} \]

- Phonology and semantics are both visible
  Training data \( D' = \langle y_1, s_1 \rangle, \ldots, \langle y_n, s_n \rangle \)
- Maximize the semantic pseudo-likelihood of the phonology
  \[ \text{PL}_{D'}(\lambda) = \prod_{i=1}^{n} P_{\lambda}(y_i | s_i) \]
- Learns whenever a semantics has several yields cross-linguistically
  \( (\text{Fut}(\text{love}(\text{Sandy, Sasha})) \Rightarrow^+ \text{“Sandy will Sasha love”}) \Rightarrow \lambda_{V2} > 0 \)
Learning from aligned bilingual corpora

- Adjust models $\lambda_a$, $\lambda_b$ to maximize probability that each translation pair receives same semantic interpretation

- Training data $D = (y_{a,1}, y_{b,1}), \ldots, (y_{a,n}, y_{b,n})$

\[
(\hat{\lambda}_a, \hat{\lambda}_b) = \arg\max_{\lambda_a, \lambda_b} L_D(\lambda_a, \lambda_b)
\]

\[
L_D(\lambda_a, \lambda_b) = \prod_{i=1}^{n} P_{\lambda_a}(S_a = S_b | y_{a,i}, y_{b,i})
\]

\[
P_{\lambda_a} \times P_{\lambda_b}(s_a, s_b | y_a, y_b) = P_{\lambda_a}(s_a | y_a)P_{\lambda_b}(s_b | y_b)
\]

- More sophisticated models are possible! (c.f., co-training)
Hidden data and bidirectional optimization

• Assume that $P(S|Y)$ and $P(Y|S)$ are highly skewed
  $\Rightarrow$ Most sentences have one highly preferred interpretation
  $\Rightarrow$ Most semantics have one highly preferred sentence

• Adjust $\lambda$ to maximize probability of *generating the observed string from its likely interpretations*

\[
D = y_1, \ldots, y_n
\]

\[
PL_D(\lambda) = \prod_{i=1}^{n} \sum_s P_\lambda(y_i|s)P_\lambda(s|y_i)
\]
Summary

- Log linear models provide a general framework for defining probability distributions over linguistic representations.
- Joint models are difficult/impossible to estimate.
- Conditional models (conditioning on the yield) are easier to estimate.
- Learning conditional models from hidden data is difficult.
- It may be useful to condition on the semantics.
- There are many other interesting conditional models to investigate!
Parsing and estimation from packed parses

- Maxwell and Kaplan packed parse representations
- Feature locality (e.g., a f-structure constant)
- Parsing/estimation statistics are sum/max of products
- Graphical representation of product expressions
- Sum/max computations over graphs
- Other applications
  - Importance sampling
  - Best-first parsing
Reparameterization of log linear models

\[ \theta_j = \exp \lambda_j \]
\[ W_\theta(\omega) = \prod_{j=1}^{m} \theta_j^{f_j(\omega)} \]
\[ P_\theta(\omega|y) = \frac{W_\theta(\omega)}{Z_\theta(y)} \]
\[ Z_\theta(y) = \sum_{\omega' \in \Omega(y)} W_\theta(\omega') \]

- Change of variables permits zero probability events
- \( Z_\theta(y) \) involves summing over all possible parses
- Same kind of technique finds most likely parse and calculates \( E_\theta[f_j|y] \)
Maxwell and Kaplan packed parses

- A parse $\omega$ consists of set of fragments $\xi \in \omega$
- A fragment is in a parse when its context function is true
- Context functions are functions of zero or more context variables
- The variable assignment must satisfy “not no-good” functions
- Each parse is identified by a unique context variable assignment

\[
y = \text{“the cat on the mat”}
\]
\[
y_1 = \text{“with a hat”}
\]
\[
X_1 \rightarrow \text{“attach } y_1 \text{ low”}
\]
\[
\neg X_1 \rightarrow \text{“attach } y_1 \text{ high”}
\]
Packed parse example

\[
y = \text{“I read a book”}
\]
\[
y_1 = \text{“on the table”}
\]
\[
X_1 \land X_2 \rightarrow \text{“attach } y_1 \text{ low”}
\]
\[
X_2 \land \neg X_2 \rightarrow \text{“attach } y_1 \text{ high”}
\]
\[
\neg X_1 \rightarrow \text{“attach } y_1 \text{ elsewhere”}
\]
\[
X_1 \lor X_2
\]
Feature locality

- Features local to fragments: $f_j(\omega) = \sum_{\xi \in \omega} f_j(\xi)$

$$y = “\text{the cat on the mat}”$$

$$y_1 = “\text{with a hat}”$$

$X_1 \rightarrow “\text{attach } y_1 \text{ low}” \land (y_1 \text{ ATTACH}) = \text{LOW}$

$\neg X_1 \rightarrow “\text{attach } y_1 \text{ high}” \land (y_1 \text{ ATTACH}) = \text{HIGH}$
Feature locality decomposes $W_\theta$

- Feature locality: the weight of a parse is the product of the weights of its fragments

$$W_\theta(\omega) = \prod_{\xi \in \omega} W_\theta(\xi)$$

- $W_\theta(y = \text{“the cat on the mat”})$
- $W_\theta(y_1 = \text{“with a hat”})$
- $X_1 \rightarrow W_\theta(\text{“attach } y_1 \text{ low” } \land (y_1 \text{ ATTACH}) = \text{LOW })$
- $\neg X_1 \rightarrow W_\theta(\text{“attach } y_1 \text{ high” } \land (y_1 \text{ ATTACH}) = \text{HIGH })$
\( W_\theta \) as a function of \( X \)

- Identify each parse \( \omega \) by its corresponding variable assignment \( x \)

- Then \( W_\theta(X) = \prod_{A \in A} A(X) \),

  - Each line \( \alpha(X) \rightarrow \xi \) introduces a term \( W_\theta(\xi)^{\alpha(X)} \)
  
  - A “not no-good” \( \eta(X) \) introduces a term \( \eta(X) \)
  
  - Each line is a function of a subset of the variables \( X \)

\[
\begin{align*}
\vdots & \quad \vdots \\
\alpha(X) & \rightarrow \xi & \times & W_\theta(\xi)^{\alpha(X)} \\
\vdots & \quad \vdots \\
\eta(X) & \times \eta(X) \\
\vdots & \quad \vdots 
\end{align*}
\]
**Dependency structure graph** $\mathcal{G}_A$

\[
Z_\theta(y) = \sum_{x \in \mathcal{X}} W_\theta(x) = \sum_{x \in \mathcal{X}} \prod_{A \in \mathcal{A}} A(x)
\]

- $\mathcal{G}$ is the *dependency graph* for $\mathcal{A}$
  - context variables $X$ are vertices of $\mathcal{G}_A$
  - $\mathcal{G}_A$ has an edge $(X_i, X_j)$ if both are arguments of some $A \in \mathcal{A}$

\[
A(X) = a(X_1, X_3)b(X_2, X_4)c(X_3, X_4, X_5)d(X_4, X_5)e(X_6, X_7)
\]
Graphical model computations

\[ Z = \sum_{x \in X} a(x_1, x_3) b(x_2, x_4) c(x_3, x_4, x_5) d(x_4, x_5) e(x_6, x_7) \]

\[ F_1(X_3) = \sum_{x_1 \in X_1} a(x_1, X_3) \]

\[ F_2(X_4) = \sum_{x_2 \in X_2} b(x_2, X_4) \]

\[ F_3(X_4, X_5) = \sum_{x_3 \in X_3} c(x_3, X_4, X_5) F_1(x_3) \]

\[ F_4(X_5) = \sum_{x_4 \in X_4} d(x_4, X_5) F_2(x_4) F_3(x_4, X_5) \]

\[ F_5 = \sum_{x_5 \in X_5} F_4(x_5) \]

\[ F_6(X_7) = \sum_{x_6 \in X_6} e(x_6, X_7) \]

\[ F_7 = \sum_{x_7 \in X_7} F_6(x_7) \]

\[ Z = F_5 F_7 \]
Use a damp, lint-free cloth to wipe the dust and dirt buildup from the scanner plastic window and rollers.
Computational complexity

- Polynomial in \( m = \text{the maximum number of conditioning variables} \geq \text{the number of variables in any function } A \)
- \( m \) depends on the ordering of variables (and \( G \))
- Finding the variable ordering that minimizes \( m \) is NP-complete, but there are good heuristics
Conclusion

- It is possible to compute the statistics needed for parsing and estimation from Maxwell and Kaplan packed parses
  - Generalizes to all Truth Maintenance Systems (not LFG specific)
- Features must be local to parse fragments
  - May require adding features to the grammar
- Computational complexity is polynomial in the number of connected variables
- Makes available techniques for graphical models to packed parse representations
  - Importance sampling
  - Best-first parsing
Future directions

• Can we build a broad-coverage SUBG?

• Reformulate “hard” UFG constraints as “soft” stochastic features
  – Underlying UBG permits all possible structural combinations
  – Grammatical constraints are expressed as stochastic features

• Is the computation tractable if we do this?

• For what tasks is the result significantly better than simpler methods?