Theory and Application of Stochastic Unification-based Grammars

Mark Johnson

Brown Laboratory for Linguistic Information Processing

(BLLIP)

University of Edinburgh, January 2002

Joint work with Stuart Geman and Stefan Riezler

Supported by NSF grants LIS 9720368 and IIS0095940

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
     $\Rightarrow$ splits states $\Rightarrow$ 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

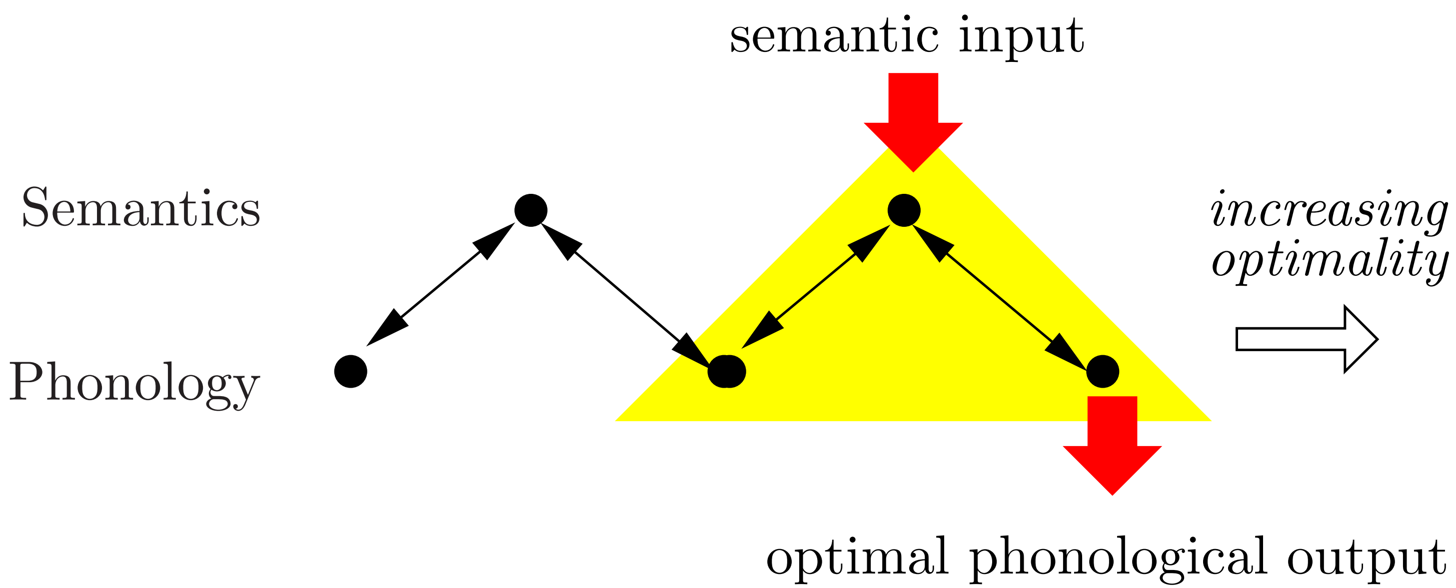
Conventional approaches to robustness

- We want to be able to analyse ill-formed input, e.g. *He walk.*
  - Ignoring agreement $\Rightarrow$ 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.

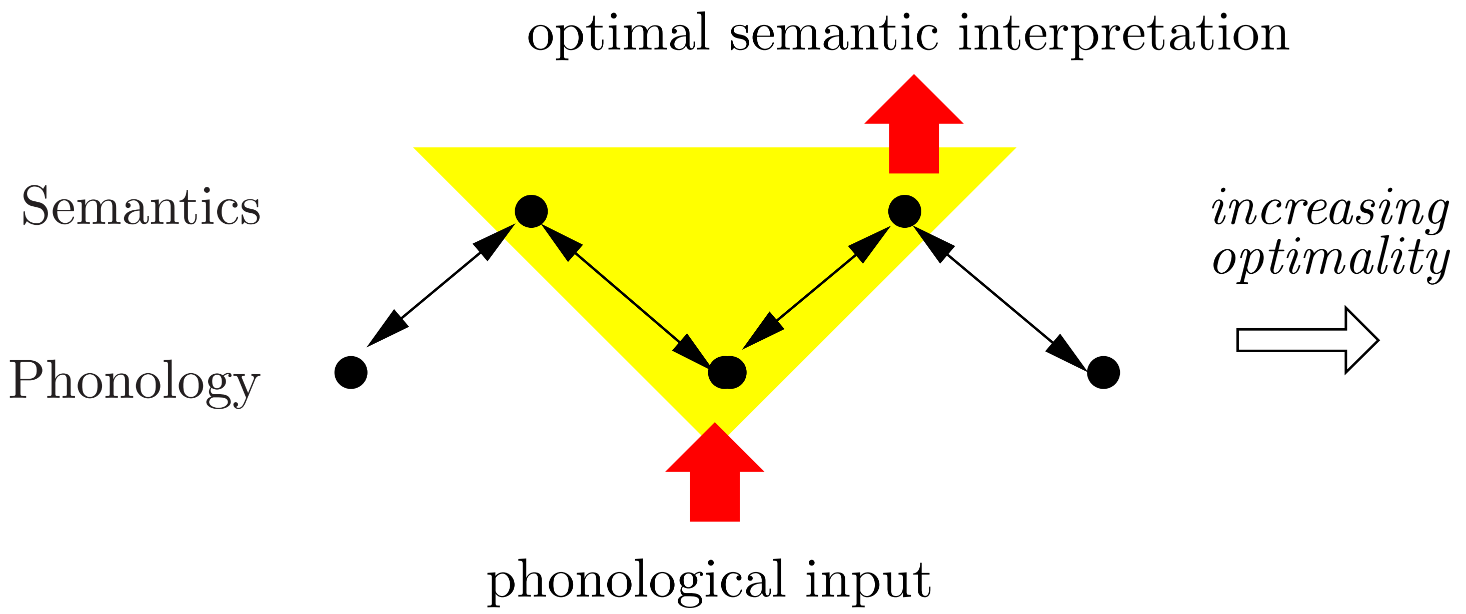
$$\text{Generate}(s) \;=\; \operatorname*{argmax}_{\omega} \mathrm{P}(\omega \mid \text{semantics}(\omega) = s)$$



Parsing with ranked analyses

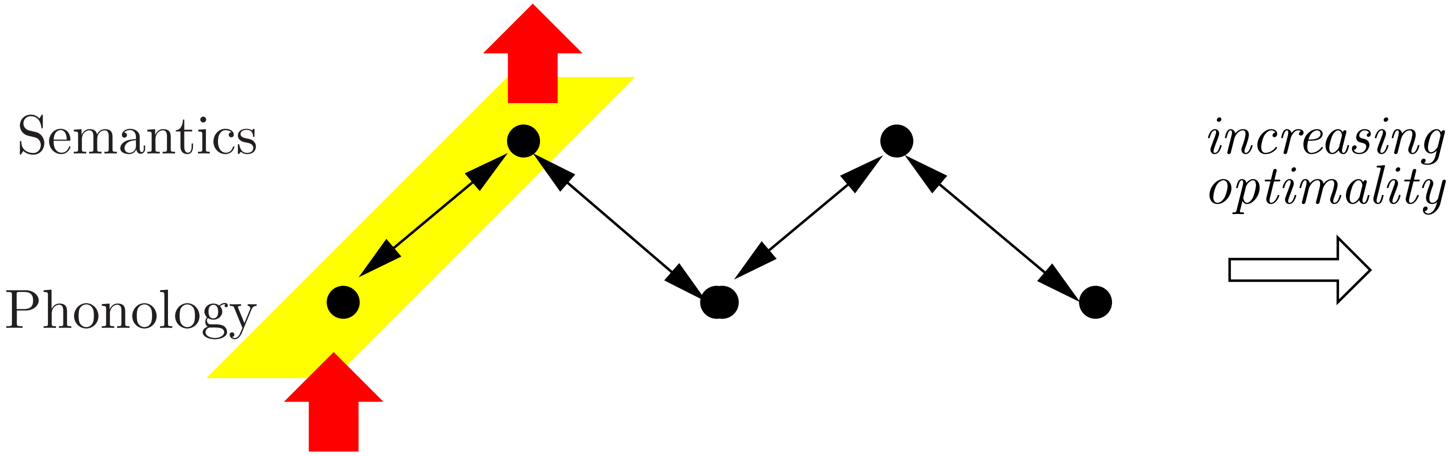
- Parsing optimizes the conditional probability of the semantics *given the phonological form* p

$$\mathrm{Parse}(p) \;=\; \operatorname*{argmax}_{\omega} \mathrm{P}(\omega \mid \mathrm{phonology}(\omega) = p)$$



Robustness and ranked interpretations

- Parsing and generation involve *different* conditional distributions!
- Grammar pairs "ungrammatical" sentences with interpretations



never generated, but interpretable!

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 suceed even if there is insufficient information for deductive approaches
- Ranked analyses provide a systematic treatment of preferences and robustness

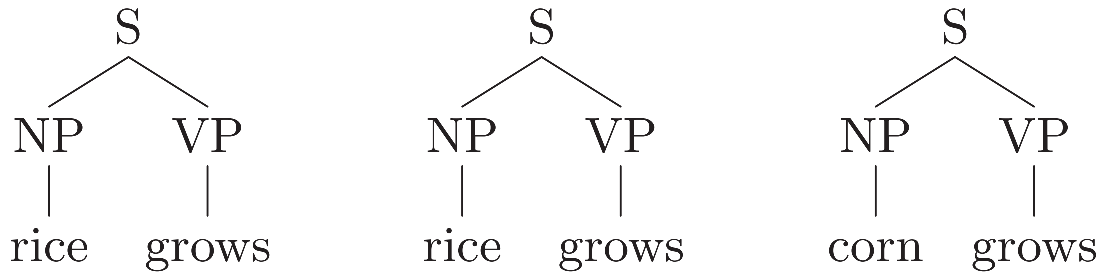
Linguistic knowledge and statistical parsing

- 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

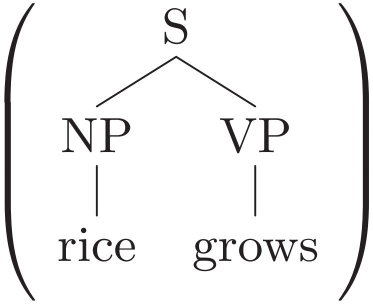
Statistical learning and parsing

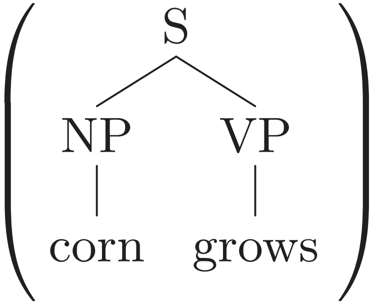
- Grammar defines (universally) possible linguistic structures Ω
- Family of probability distributions P_θ on Ω parameterized by θ
- Learning involves finding θ which makes the input most likely
- Given θ and a yield (terminal string) y, parsing involves finding most probable structure in $\{\omega |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

PCFGs and relative frequency estimator

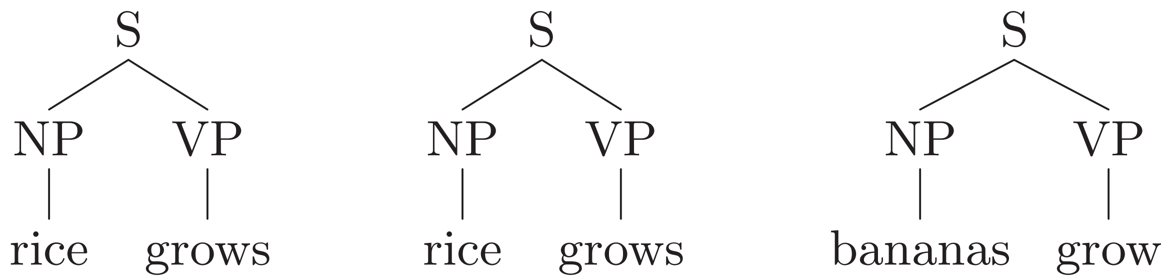


rule	count	rel freq
S $\rightarrow$ NP VP	3	1
NP $\rightarrow$ rice	2	2/3
NP $\rightarrow$ corn	1	1/3
VP $\rightarrow$ grows	3	1

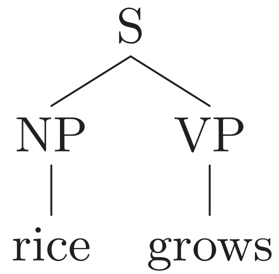
P  = 2/3

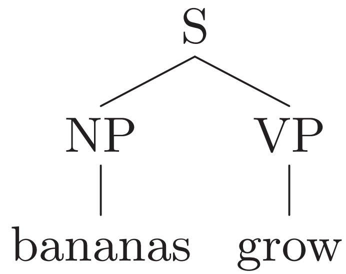
P  = 1/3

Non-local constraints



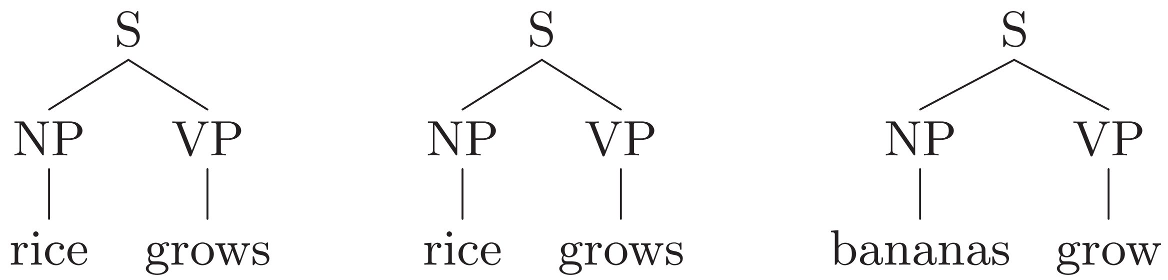
rule	count	rel freq
S → NP VP	3	1
NP → rice	2	2/3
NP → bananas	1	1/3
VP → grows	2	2/3
VP → grow	1	1/3

P () = 4/9

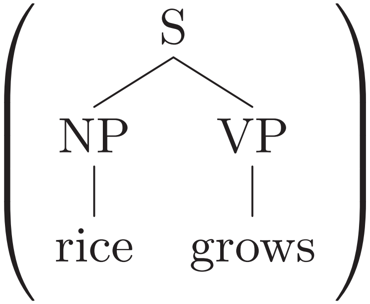
P () = 1/9

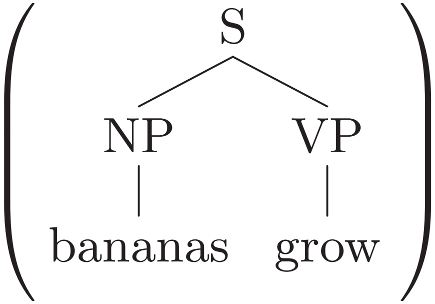
Z = 5/9

Renormalization



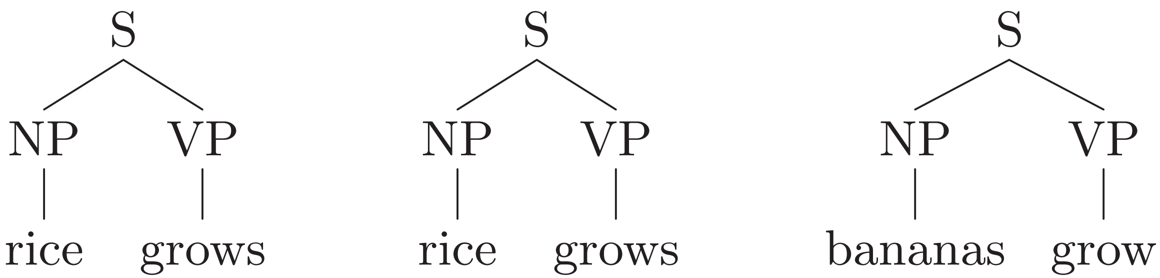
rule	**count**	**rel freq**
S → NP VP	3	1
NP → rice	2	2/3
NP → bananas	1	1/3
VP → grows	2	2/3
VP → grow	1	1/3

P  = 4/9 4/5

P  = 1/9 1/5

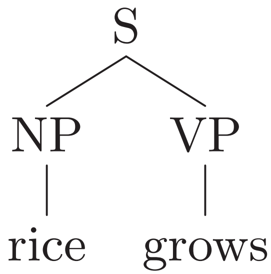
$Z = 5/9$

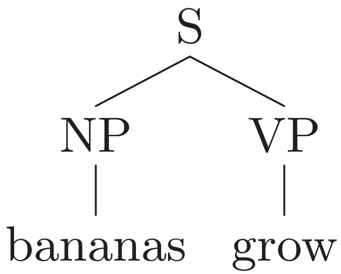
Other values do better!



rule	**count**	**rel freq**
S → NP VP	3	1
NP → rice	2	2/3
NP → bananas	1	1/3
VP → grows	2	1/2
VP → grow	1	1/2

(Abney 1997)

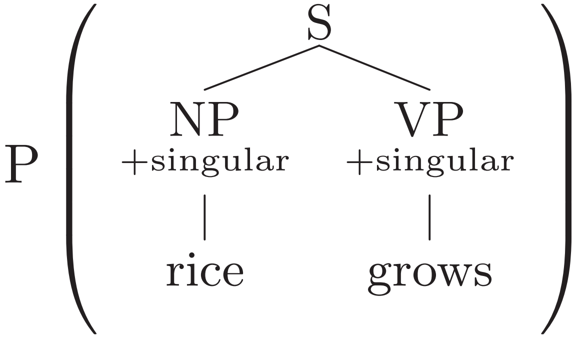
P() = 2/6 2/3

P() = 1/6 1/3

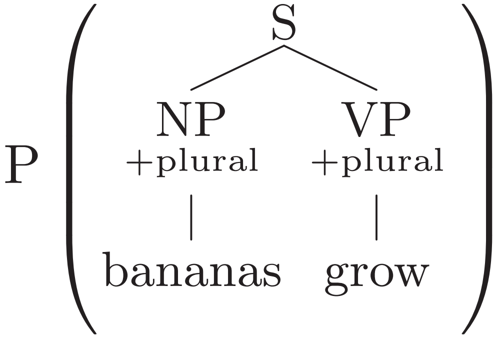
Z = 3/6

Make dependencies local – GPSG-style

rule	**count**	**rel freq**
$S \rightarrow \underset{+\text{singular}}{NP} \; \underset{+\text{singular}}{VP}$	2	2/3
$S \rightarrow \underset{+\text{plural}}{NP} \; \underset{+\text{plural}}{VP}$	1	1/3
$\underset{+\text{singular}}{NP} \rightarrow \text{rice}$	2	1
$\underset{+\text{plural}}{NP} \rightarrow \text{bananas}$	1	1
$\underset{+\text{singular}}{VP} \rightarrow \text{grows}$	2	1
$\underset{+\text{plural}}{VP} \rightarrow \text{grow}$	1	1



$= 2/3$



$= 1/3$

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

  ⇒ *no need to restrict attention to rule features*

Log linear models

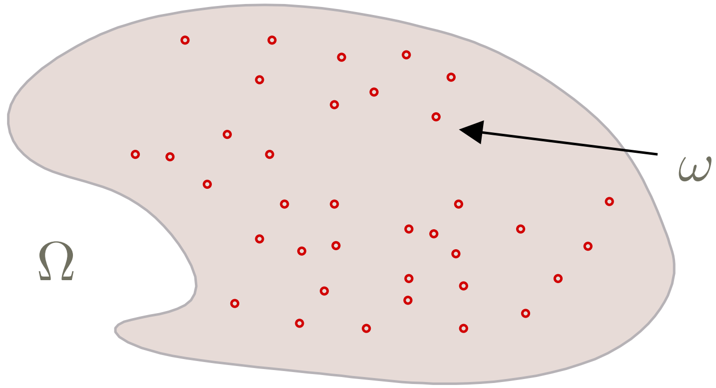
- The *log likelihood* is a *linear* function of feature values
- Ω = set of syntactic structures (not necessarily trees)
- $f_j(\omega)$ = number of occurences of jth feature in $\omega \in \Omega$ (feature $\neq$ attribute)
- λ_j are "feature weight" parameters

$$W_\lambda(\omega) = \exp(\sum_{j=1}^{m} \lambda_j f_j(\omega))$$

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

$$\mathrm{P}_\lambda(\omega) = \frac{W_\lambda(\omega)}{Z_\lambda}$$

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

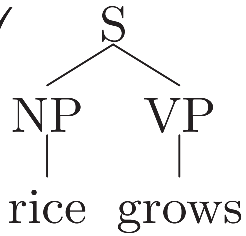


PCFGs are log-linear models

Ω = set of all trees generated by G

$f_j(\omega)$ = number of times the jth rule is used in $\omega \in \Omega$

θ_j = probability of jth rule in G $\qquad \lambda_j = \log \theta_j$



$$f\left(\text{tree}\right) = [\underbrace{1}_{\text{S}\to\text{NP VP}}, \underbrace{1}_{\text{NP}\to\text{rice}}, \underbrace{0}_{\text{NP}\to\text{bananas}}, \underbrace{1}_{\text{VP}\to\text{grows}}, \underbrace{0}_{\text{VP}\to\text{grow}}]$$

$$\mathrm{P}_\theta(\omega) = \prod_{j=1}^{m} \theta_j^{f_j(\omega)} = \exp(\sum_{j=1}^{m} \lambda_j f_j(\omega)) \quad \text{where } \lambda_j = \log \theta_j$$

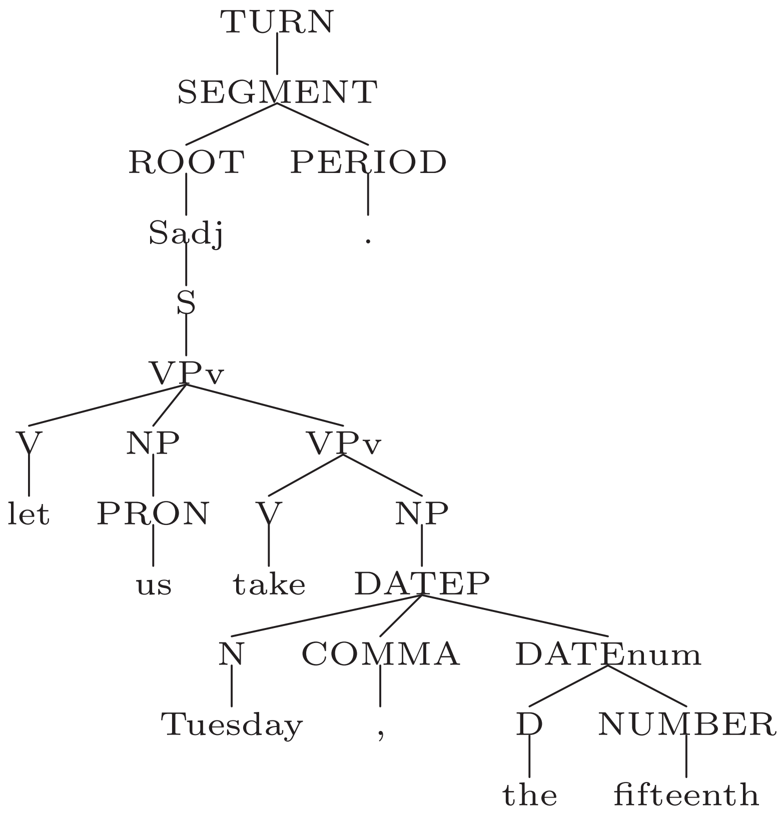
Stochastic Lexical-Functional Grammar

- Unification-based grammar (competence) defines well-formed syntactic structures Ω
  - In SLFG, these are c-structure/f-structure pairs
- Stochastic model (performance) defines a probability distribution over Ω
  - Features $f_1, \ldots, f_m$, where each f_j maps each $\omega \in \Omega$ to a feature occurence count $f_j(\omega)$
  - Probability distribution defined by log linear model

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

- Same approach applies to virtually any theory of grammar

Sample parses




```
SENTENCE_ID  BAC002_E
  OBJ         [ ANIM +
                CASE ACC
                NUM PL
                PERS 1
                PRED PRO
                PRON-FORM WE
                PRON-TYPE PERS ] 9
  PASSIVE –
  PRED  LET⟨2,10⟩9
  STMT-TYPE IMPERATIVE
  SUBJ        [ PERS 2
                PRED PRO
                PRON-TYPE NULL ] 2
  TNS-ASP [ MOOD IMPERATIVE ]
  XCOMP   OBJ [ ANIM –
                APP [ NTYPE [ NUMBER ORD
                              TIME DATE ]
                      NUM SG
                      PRED fifteen
                      SPEC [ SPEC-FORM THE
                             SPEC-TYPE DEF ] ]
                CASE ACC
                GEND NEUT
                NTYPE [ GRAIN COUNT
                        PROPER DATE
                        TIME DAY ]
                NUM SG
                PERS 3
                PRED  TUESDAY ] 13
          PASSIVE –
  22
```

Features used

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

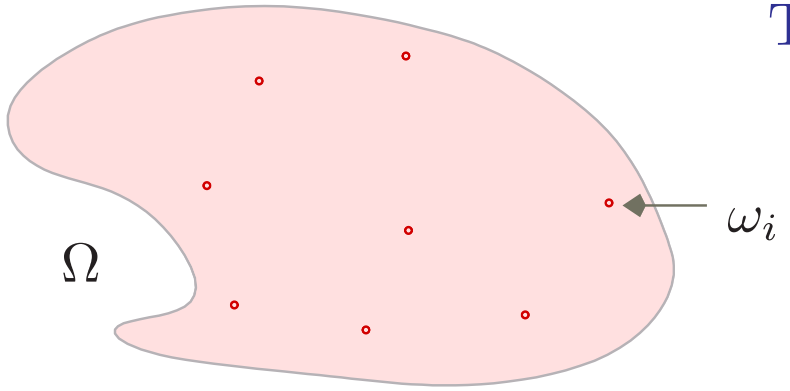
Attribute value features: For every attribute a and every atomic value v, $f_{a=v}(\omega)$ is the number of times the pair $a = v$ appears in ω

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

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 $D = \omega_1, \ldots, \omega_n$

$$\hat{\lambda} = \operatorname*{argmax}_{\lambda} L_D(\lambda)$$

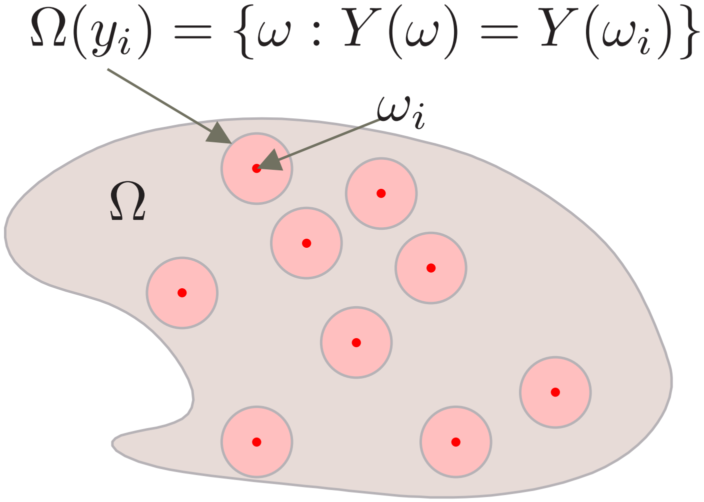
$$L_D(\lambda) = \prod_{i=1}^{n} \mathrm{P}_\lambda(\omega_i)$$

$$\mathrm{P}_\lambda(\omega) = \frac{W_\lambda(\omega)}{Z_\lambda} \quad W_\lambda(\omega) = \exp(\sum_j \lambda_j f_j(\omega)) \quad Z_\lambda = \sum_{\omega' \in \Omega} W_\lambda(\omega')$$

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

Pseudo-likelihood

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



$$\begin{aligned}
\hat{\lambda} &= \operatorname*{argmax}_{\lambda} \mathrm{PL}_D(\lambda) \\
\mathrm{PL}_D(\lambda) &= \prod_{i=1}^{n} \mathrm{P}_\lambda(\omega_i | y_i) \\
\mathrm{P}_\lambda(\omega | y) &= \frac{W_\lambda(\omega)}{Z_\lambda(y)}
\end{aligned}$$

$$W_\lambda(\omega) = \exp(\sum_j \lambda_j f_j(\omega)) \qquad Z_\lambda(y) = \sum_{\omega' \in \Omega(y)} W_\lambda(\omega')$$

Pseudo-likelihood versus likelihood

- The pseudo-partition function $Z_\lambda(y)$ is *much easier to compute* than the partition function Z_λ
  - Z_λ requires a sum over Ω
  - $Z_\lambda(y)$ requires a sum over Ω_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

	Correct parse's features	**All other parses' features**
sentence 1	$[1, 3, 2]$	$[2, 2, 3]$ $[3, 1, 5]$ $[2, 6, 3]$
sentence 2	$[7, 2, 1]$	$[2, 5, 5]$
sentence 3	$[2, 4, 2]$	$[1, 1, 7]$ $[7, 2, 1]$
...	...	...

- Training data is *fully observed* (i.e., parsed data)
- Choose λ to maximize (log) likelihood of *correct* parses relative to other parses
- Distribution of *sentences* is ignored

Pseudo-constant features are uninformative

	Correct parse's features	**All other parses' features**
sentence 1	$[1, 3, \mathit{2}]$	$[2, 2, \mathit{2}]$ $[3, 1, \mathit{2}]$ $[2, 6, \mathit{2}]$
sentence 2	$[7, 2, \mathit{5}]$	$[2, 5, \mathit{5}]$
sentence 3	$[2, 4, \mathit{4}]$	$[1, 1, \mathit{4}]$ $[7, 2, \mathit{4}]$
...	...	...

- *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 $\Rightarrow$ irrelevant

Pseudo-maximal features ⇒ unbounded $\widehat{\lambda_j}$

	Correct parse's features	**All other parses' features**
sentence 1	[1, *3*, 2]	[2, *3*, 4] [3, *1*, 1] [2, *1*, 1]
sentence 2	[2, *7*, 4]	[3, *7*, 2]
sentence 3	[2, *4*, 4]	[1, *1*, 1] [1, *2*, 4]

- A *pseudo-maximal feature* always reaches its maximum value within a parse on the correct parse
- If f_j is pseudo-maximal, $\widehat{\lambda_j} \to \infty$ (hard constraint)
- If f_j is pseudo-minimal, $\widehat{\lambda_j} \to -\infty$ (hard constraint)

Regularization

- f_j is pseudo-maximal over training data $\not\Rightarrow$ f_j is pseudo-maximal over all of Ω (sparse data)

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

$$\hat{\lambda} = \operatorname*{argmax}_{\lambda} \log \mathrm{PL}_D(\lambda) - \sum_{j=1}^{m} \frac{\lambda_j^2}{2\sigma_j^2}$$

  where σ_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

	Verbmobil corpus	Homecentre corpus
# of sentences	540	980
# of ambiguous sentences	324	424
Av. length of ambig. sentences	13.8	13.1
# of parses	3245	2865
# of features	191	227
# of rule features	59	57

Pseudo-likelihood estimator evaluation

	Verbmobil corpus 324 sentences		**Homecentre corpus** 424 sentences	
	C	$-\log \mathrm{PL}$	C	$-\log \mathrm{PL}$
Baseline estimator	88.8	533.2	136.9	590.7
Pseudo-likelihood estimator	180.0	401.3	283.25	580.6

- 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

$\Rightarrow$ Re-express hard linguistic constraints as soft constraints

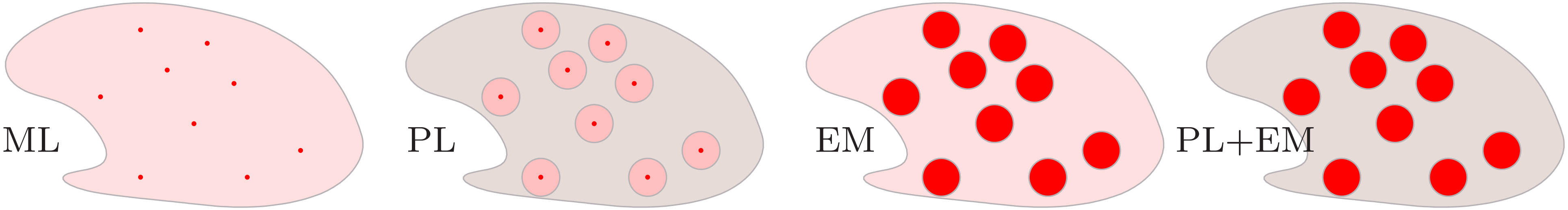
Summary

- 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



	maximizes likelihood of	**relative to**
ML	ω_i	Ω
PL	ω_i	$\Omega(y_i)$
EM	$\Omega(y_i)$	Ω
PL+EM	$\Omega(y_i)$	$\Omega(y_i)$

Psychologically-realistic conditional models

- *Joint* models $\mathrm{P}(\omega)$ predict *what is said* and *how it is said*
- *Modularity:* These two processes are very different!
- Conditional models in SUBGs: $\mathrm{P}(S|Y)$
  ($S =$ semantics, $Y =$ phonology)
- A *psycholinguistically realistic* statistical model
  – World model: $\mathrm{P}(S)$
  – Linguistic model: $\mathrm{P}(Y|S)$
- Parsing with such models:

$$\mathrm{P}(S|Y) \propto \mathrm{P}(Y|S)\mathrm{P}(S)$$

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	$\lambda_{\mathrm{VI}} > 0$	$[_\mathrm{S}$ Kim $[_\mathrm{VP}$ will love Sandy$]]$
Verb final	$\lambda_{\mathrm{VF}} > 0$	$[_\mathrm{S}$ Kim $[_\mathrm{VP}$ Sandy love will$]]$
Verb second	$\lambda_{\mathrm{V2}} > 0$	$[_\mathrm{S}$ Kim will $[_\mathrm{VP}$ 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 λ_{V2} value

    $[_{\mathrm{S}}$ Kim loves $[_{\mathrm{VP}}$ Sandy$]]$ $\Rightarrow$ $\lambda_{\mathrm{V2}} > 0$

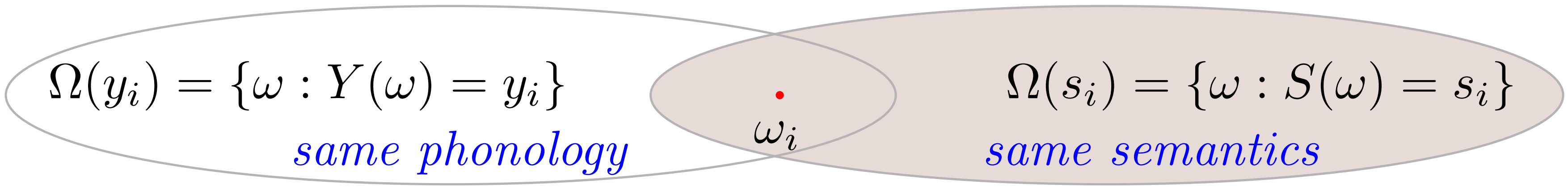
    $[_{\mathrm{S}}$ Kim $[_{\mathrm{VP}}$ loves Sandy$]]$ $\Rightarrow$ $\lambda_{\mathrm{V2}} < 0$
  - Unrealistic to assume child has access to parsed input
- PL estimator only learns from *ambiguous sentences*
  - $[_{\mathrm{S}}$ Kim $[_{\mathrm{VP}}$ Sandy love will$]]$ is uninformative to PL
- But *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-likehood

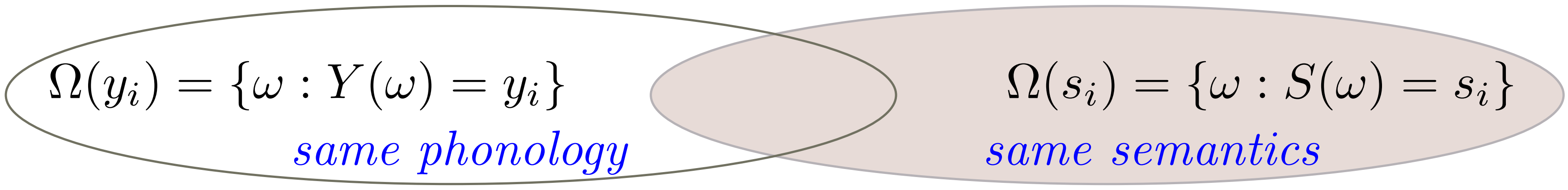


- Assume learner has access to semantics s_i and correct parse ω_i
- Treat the *semantics* s_i as visible component
- Pseudo-likelihood with *semantic comparison set*

$$\mathrm{PL}'_D(\lambda) \;=\; \prod_{i=1}^{n} \mathrm{P}_\lambda(\omega_i|s_i)$$

- Learns when a semantics can be expressed in several ways cross-linguistically
  (love(Sandy, Sasha)) $\Rightarrow^{+}$ [$_{\mathrm{S}}$ Sandy [$_{\mathrm{VP}}$ Sasha love]]) $\;\Rightarrow\;$ $\lambda_{\mathrm{VF}} > 0$

Partially observed data



- 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-likehood of the phonology

$$\mathrm{PL}_{D'}(\lambda) \;=\; \prod_{i=1}^{n} \mathrm{P}_\lambda(y_i | s_i)$$

- Learns whenever a semantics has several yields cross-linguistically
  $(\mathrm{Fut}(\mathrm{love}(\mathrm{Sandy}, \mathrm{Sasha})) \Rightarrow^{+}$ "Sandy will Sasha love"$) \;\Rightarrow\; \lambda_{\mathrm{V2}} > 0$

Learning from aligned bilingual corpora

- Adjust models λ_a, λ_b to maximize probability that *each translation pair receives same semantic interpretation*
- Training data $D = (y_{a,1}, y_{b,1}), \ldots, (y_{a,n}, y_{b,n})$

$$(\widehat{\lambda_a}, \widehat{\lambda_b}) = \operatorname*{argmax}_{\lambda_a, \lambda_b} L_D(\lambda_a, \lambda_b)$$

$$L_D(\lambda_a, \lambda_b) = \prod_{i=1}^{n} \mathrm{P}_{\lambda_a} \times \mathrm{P}_{\lambda_b}(S_a = S_b | y_{a,i}, y_{b,i})$$

$$\mathrm{P}_{\lambda_a} \times \mathrm{P}_{\lambda_b}(s_a, s_b | y_a, y_b) = \mathrm{P}_{\lambda_a}(s_a | y_a) \mathrm{P}_{\lambda_b}(s_b | y_b)$$

- More sophisticated models are possible! (c.f., co-training)

Hidden data and bidirectional optimization

- Assume that $\mathrm{P}(S|Y)$ and $\mathrm{P}(Y|S)$ are highly skewed

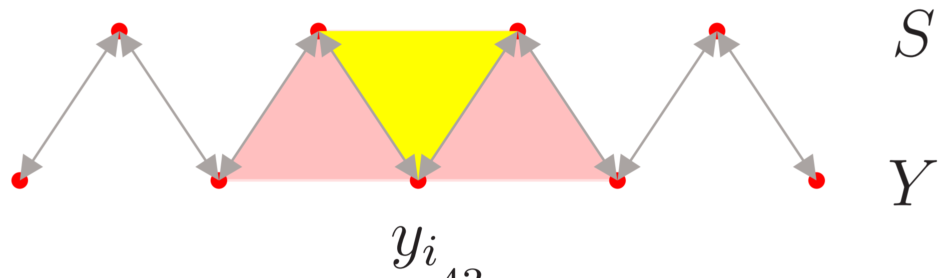
  $\Rightarrow$ Most sentences have one highly preferred interpretation

  $\Rightarrow$ Most semantics have one highly preferred sentence

- Adjust λ to maximize probability of *generating the observed string from its likely interpretations*

$$D = y_1, \ldots, y_n$$

$$PL_D(\lambda) = \prod_{i=1}^{n} \sum_{s} \mathrm{P}_\lambda(y_i|s)\mathrm{P}_\lambda(s|y_i)$$



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

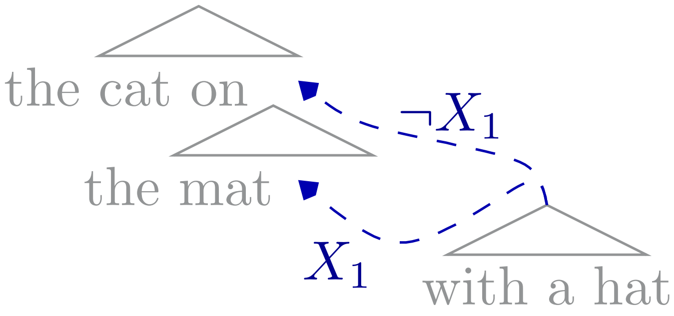
$$
\begin{aligned}
\theta_j &= \exp \lambda_j \\
W_\theta(\omega) &= \prod_{j=1}^{m} \theta_j^{f_j(\omega)} \\
\mathrm{P}_\theta(\omega|y) &= \frac{W_\theta(\omega)}{Z_\theta(y)} \\
Z_\theta(y) &= \sum_{\omega' \in \Omega(y)} W_\theta(\omega')
\end{aligned}
$$

- 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

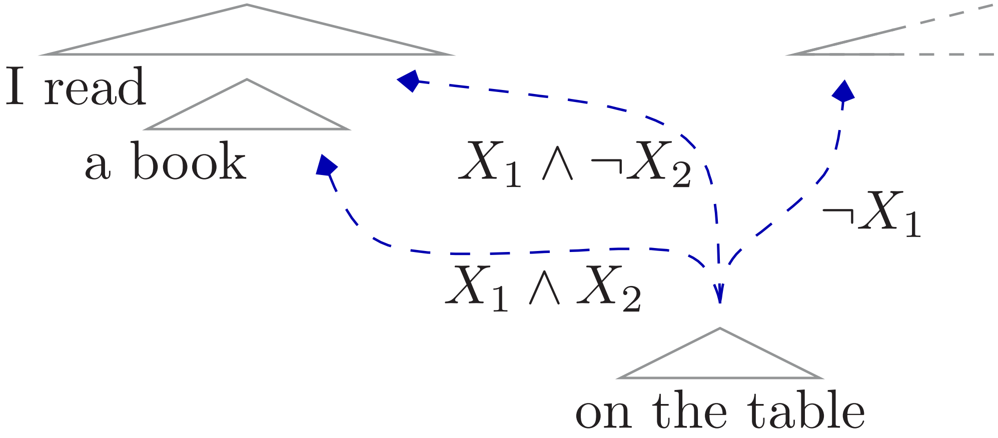
- A parse ω consists of set of fragments $\xi \in \omega$
- A fragment is in a parse when its *context function* is true
- Context functions are functions of zero or more *context variables*
- The variable assignment must satisfy "not no-good" functions
- Each parse is identified by a *unique context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

$$\begin{aligned} y &= \text{“\textit{the cat on the mat}”} \\ y_1 &= \text{“\textit{with a hat}”} \\ X_1 &\rightarrow \text{“attach } y_1 \text{ low”} \\ \neg X_1 &\rightarrow \text{“attach } y_1 \text{ high”} \end{aligned}$$



Packed parse example

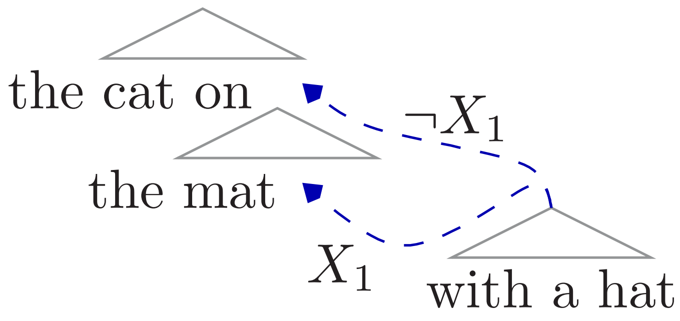
$$
\begin{aligned}
y &= \text{“}\textit{I read a book}\text{”} \\
y_1 &= \text{“}\textit{on the table}\text{”} \\
X_1 \wedge X_2 &\rightarrow \text{“attach } y_1 \text{ low”} \\
X_2 \wedge \neg X_2 &\rightarrow \text{“attach } y_1 \text{ high”} \\
\neg X_1 &\rightarrow \text{“attach } y_1 \text{ elsewhere”} \\
& X_1 \vee X_2
\end{aligned}
$$



Feature locality

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

$$
\begin{aligned}
y &= \textit{“the cat on the mat”} \\
y_1 &= \textit{“with a hat”} \\
X_1 &\rightarrow \text{“attach } y_1 \text{ low”} \;\wedge\; (y_1 \;\textsc{attach}) = \textsc{low} \\
\neg X_1 &\rightarrow \text{“attach } y_1 \text{ high”} \;\wedge\; (y_1 \;\textsc{attach}) = \textsc{high}
\end{aligned}
$$



Feature locality decomposes W_θ

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

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

$$\begin{array}{rcl}
 & & W_\theta(y \;=\; \text{``}\textit{the cat on the mat}\text{''}) \\
 & & W_\theta(y_1 \;=\; \text{``}\textit{with a hat}\text{''}) \\
X_1 & \rightarrow & W_\theta\,(\text{``attach } y_1 \text{ low''} \;\wedge\; (y_1 \;\textsc{attach}) = \textsc{low}\;) \\
\neg X_1 & \rightarrow & W_\theta\,(\text{``attach } y_1 \text{ high''} \;\wedge\; (y_1 \;\textsc{attach}) = \textsc{high}\;)
\end{array}$$

W_θ as a function of X

- Identify each parse ω by its corresponding variable assignment x
- Then $W_\theta(X) = \prod_{A \in \mathcal{A}} A(X)$,
  - Each line $\alpha(X) \to \xi$ 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

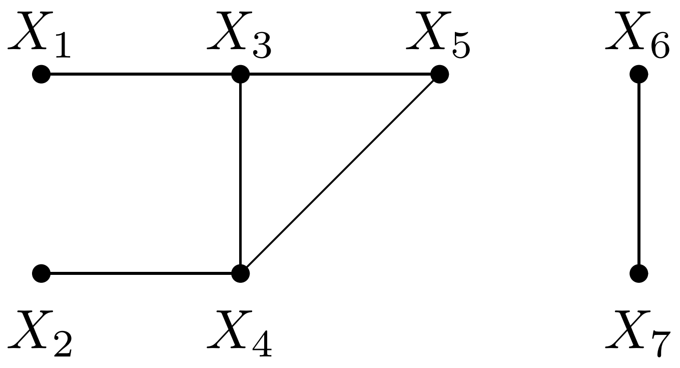
$$
\begin{array}{ccccc}
 & \vdots & & & \vdots \\
\alpha(X) & \to & \xi & \times & W_\theta(\xi)^{\alpha(X)} \\
 & \vdots & & \times & \vdots \\
 & & \eta(X) & \times & \eta(X) \\
 & \vdots & & \times & \vdots
\end{array}
$$

Dependency structure graph $\mathcal{G}_{\mathcal{A}}$

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

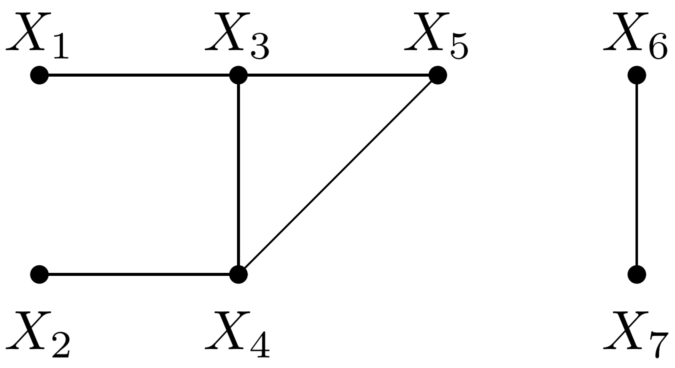
- $\mathcal{G}$ is the *dependency graph* for $\mathcal{A}$
  - context variables X are vertices of $\mathcal{G}_{\mathcal{A}}$
  - $\mathcal{G}_{\mathcal{A}}$ has an edge (X_i, X_j) if both are arguments of some $A \in \mathcal{A}$

$$A(X) = a(X_1, X_3) b(X_2, X_4) c(X_3, X_4, X_5) d(X_4, X_5) e(X_6, X_7)$$



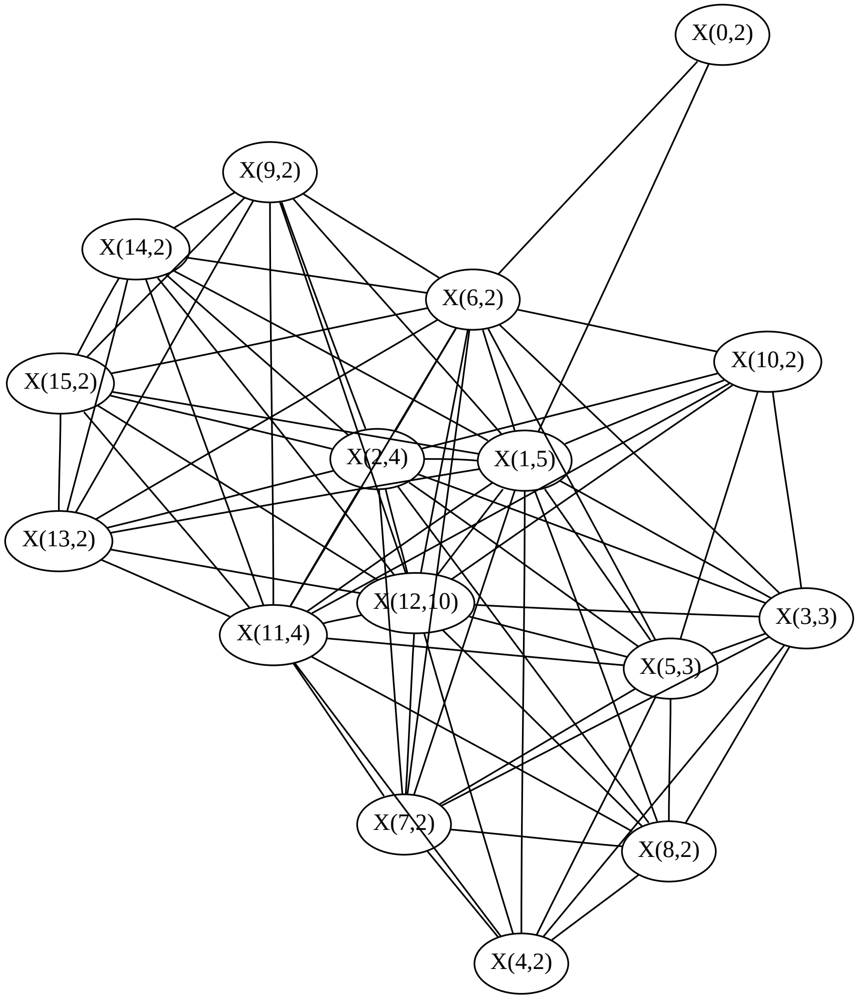
Graphical model computations

$$
\begin{aligned}
Z &= \textstyle\sum_{x \in \mathcal{X}} a(x_1, x_3) b(x_2, x_4) c(x_3, x_4, x_5) d(x_4, x_5) e(x_6, x_7) \\
F_1(X_3) &= \textstyle\sum_{x_1 \in \mathcal{X}_1} a(x_1, X_3) \\
F_2(X_4) &= \textstyle\sum_{x_2 \in \mathcal{X}_2} b(x_2, X_4) \\
F_3(X_4, X_5) &= \textstyle\sum_{x_3 \in \mathcal{X}_3} c(x_3, X_4, X_5) F_1(x_3) \\
F_4(X_5) &= \textstyle\sum_{x_4 \in \mathcal{X}_4} d(x_4, X_5) F_2(x_4) F_3(x_4, X_5) \\
F_5 &= \textstyle\sum_{x_5 \in \mathcal{X}_5} F_4(x_5) \\
F_6(X_7) &= \textstyle\sum_{x_6 \in \mathcal{X}_6} e(x_6, X_7) \\
F_7 &= \textstyle\sum_{x_7 \in \mathcal{X}_7} F_6(x_7) \\
Z &= F_5 F_7
\end{aligned}
$$



Graphical model for Homecentre example

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 = the *maximum number of conditioning variables* $\geq$ the number of variables in any function A
- m depends on the ordering of variables (and $\mathcal{G}$)
- 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?