Learning grammar(s) statistically

Mark Johnson

joint work with Sharon Goldwater and Tom Griffiths

Cognitive and Linguistic Sciences and Computer Science
Brown University

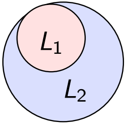
Mayfest 2006

Outline

Why statistical learning?

- Uncertainty is *pervasive* in learning
  - the input does not contain enough information to uniquely determine grammar and lexicon
  - the input is noisy (misperceived, mispronounced)
  - our scientific understanding is incomplete
- Statistical learning is compatible with linguistics
  - we can define probabilistic versions of virtually any kind of generative grammar (Abney 1997)
- *Statistical learning is much more than conditional probabilities!*

Statistical learning and implicit negative evidence



- Logical approach to acquisition
  no negative evidence ⇒ *subset problem*
  guess L_2 when true lg is L_1
- statistical learning can use *implicit negative evidence*
  - if $L_2 - L_1$ is *expected* to occur but doesn't
    ⇒ L_2 is probably wrong
  - *succeeds where logical learning fails* (e.g., PCFGs)
    - stronger input assumptions (follows distribution)
    - weaker success criteria (probabilistic)
- Both logic and statistics are kinds of inference
  - statistical inference uses more information from input
  - children seem sensitive to distributional properties
  - it would be strange if they didn't use them for learning

Probabilistic models and statistical learning

- Decompose learning problem into three components:
  1. class of *possible models*, e.g., certain type of (probabilistic) grammars, from which learner chooses
  2. *objective function* (of model and input) that learning optimizes
     - e.g., *maximum likelihood*: find model that makes input as likely as possible
  3. *search algorithm* that finds optimal model(s) for input
- Using explicit probabilistic models lets us:
  - *combine models* for subtasks *in an optimal way*
  - better *understand* our learning models
  - diagnose problems with our learning models
    - distinguish *model errors* from *search errors*

Bayesian learning

$$\underbrace{\mathrm{P(Hypothesis|Data)}}_{\text{Posterior}} \propto \underbrace{\mathrm{P(Data|Hypothesis)}}_{\text{Likelihood}} \underbrace{\mathrm{P(Hypothesis)}}_{\text{Prior}}$$

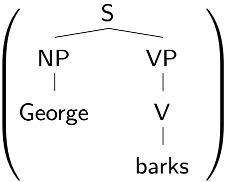
- Bayesian models integrate information from *multiple information sources*
  - *Likelihood* reflects how well grammar fits input data
  - *Prior* encodes a priori preferences for particular grammars
- Priors can prefer smaller grammars (Occam's razor, MDL)
- The *prior is as much a linguistic issue as the grammar*
  - Priors can be sensitive to linguistic structure (e.g., words should contain vowels)
  - Priors can encode *linguistic universals* and *markedness preferences*

Outline

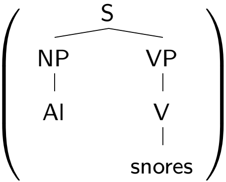
Probabilistic Context-Free Grammars

- The *probability* of a tree is the product of the probabilities of the rules used to construct it

1.0	S $\longrightarrow$ NP VP	1.0	VP $\longrightarrow$ V
0.75	NP $\longrightarrow$ George	0.25	NP $\longrightarrow$ Al
0.6	V $\longrightarrow$ barks	0.4	V $\longrightarrow$ snores

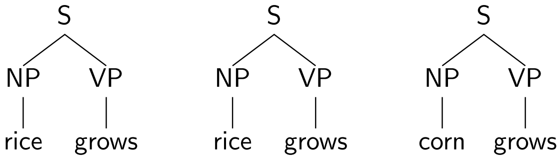
P


= 0.45

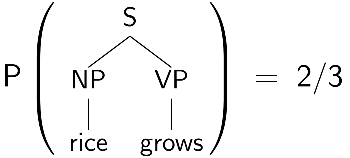
P


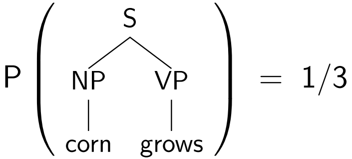
= 0.1

Learning PCFGs from trees (supervised)



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





Rel freq is *maximum likelihood estimator* (selects rule probabilities that maximize probability of trees)

Learning from words alone (unsupervised)

- Training data consists of strings of words w
- Maximum likelihood estimator (grammar that makes w as likely as possible) no longer has closed form
- *Expectation maximization* is an iterative procedure for building unsupervised learners out of supervised learners
  - parse a bunch of sentences with current guess at grammar
  - weight each parse tree by its probability under current grammar
  - estimate grammar from these weighted parse trees as before
- Each iteration is *guaranteed* not to decrease $\mathrm{P}(w)$ (but can get trapped in local minima)

Dempster, Laird and Rubin (1977) "Maximum likelihood from incomplete data via the EM algorithm"

Expectation Maximization with a toy grammar

Initial rule probs

rule	**prob**
. . .	. . .
VP → V	0.2
VP → V NP	0.2
VP → NP V	0.2
VP → V NP NP	0.2
VP → NP NP V	0.2
. . .	. . .
Det → the	0.1
N → the	0.1
V → the	0.1

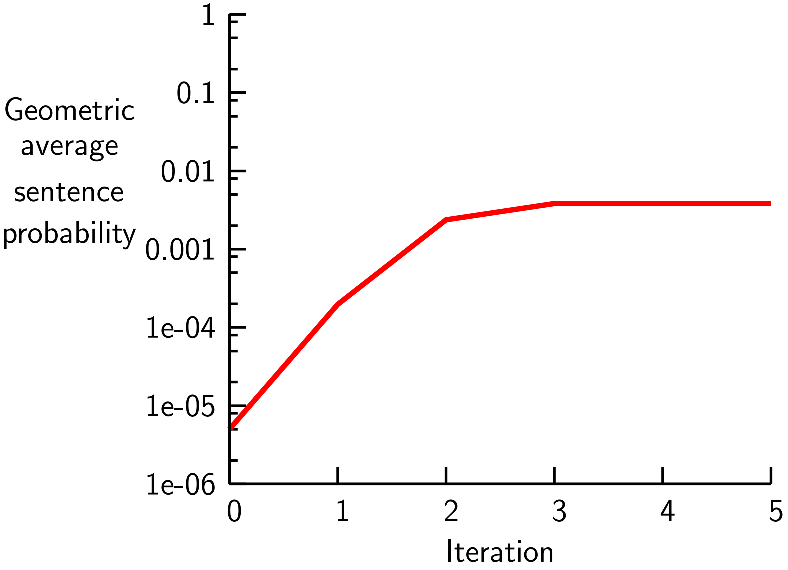
"English" input

the dog bites
the dog bites a man
a man gives the dog a bone
. . .

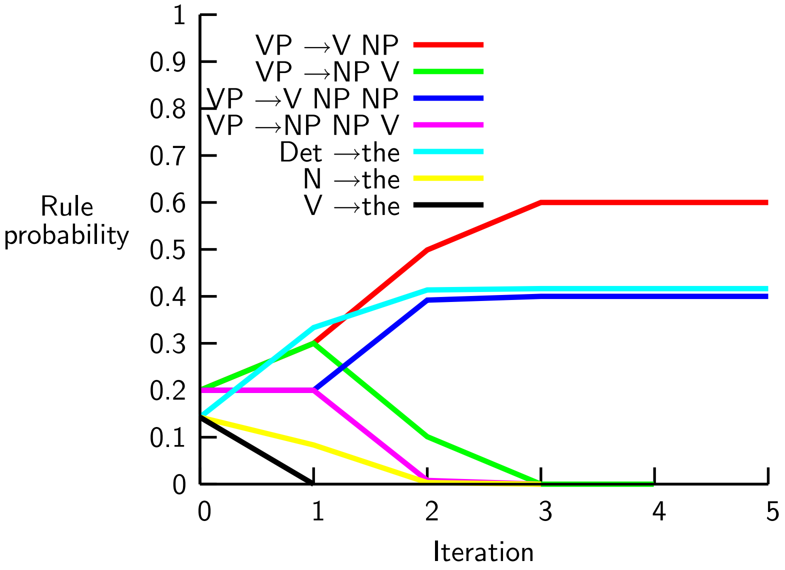
"pseudo-Japanese" input

the dog bites
the dog a man bites
a man the dog a bone gives
. . .

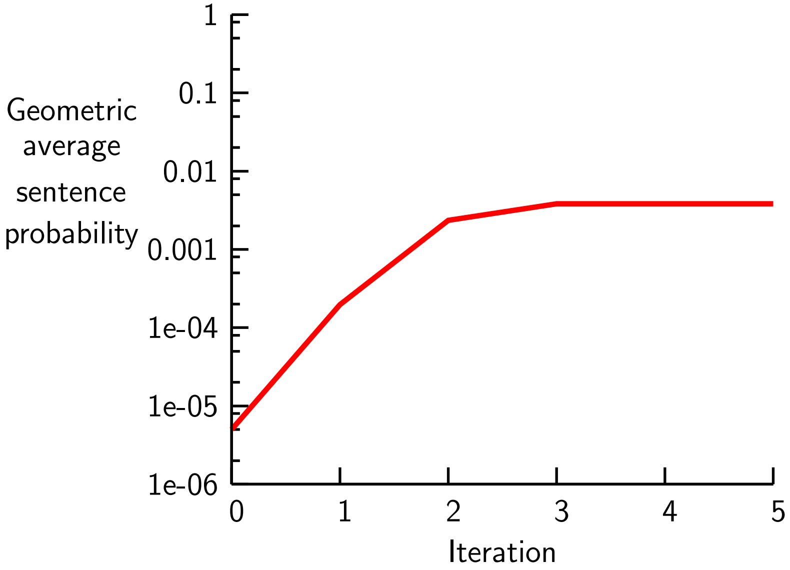
Probability of "English"



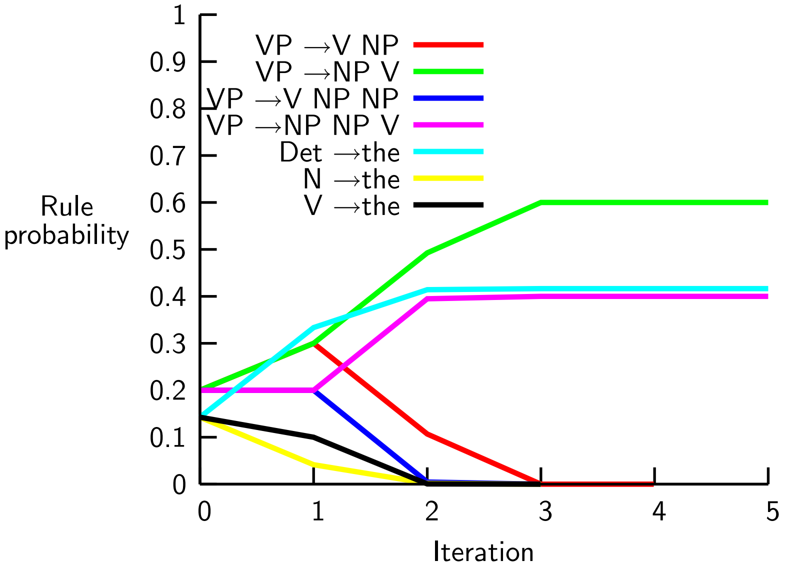
Rule probabilities from “English”



Probability of "Japanese"



Rule probabilities from “Japanese”

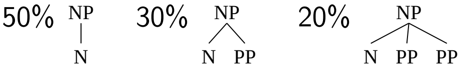


Statistical grammar learning

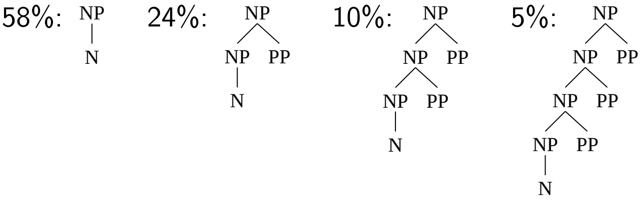
- Simple algorithm: learn from your best guesses
  - requires learner to parse the input
- "Glass box" models: learner's prior knowledge and learnt generalizations are *explicitly represented*
- Optimization of smooth function of rule weights ⇒ learning can involve small, incremental updates
- Learning structure (rules) is hard, but ...
- Parameter estimation can approximate rule learning
  - start with "superset" grammar
  - estimate rule probabilities
  - discard low probability rules

Different grammars lead to different generalizations

- In a PCFG, rules are units of generalization
  - *Training data:* 50%: N, 30%: N PP, 20%: N PP PP
  - with flat rules NP → N, NP → N PP, NP → N PP PP predicted probabilities replicate training data

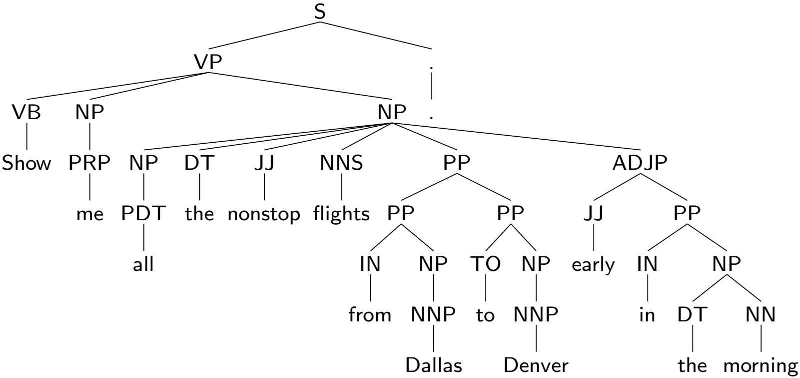


  - but with adjunction rules NP → N, NP → NP PP



PCFG learning from real language

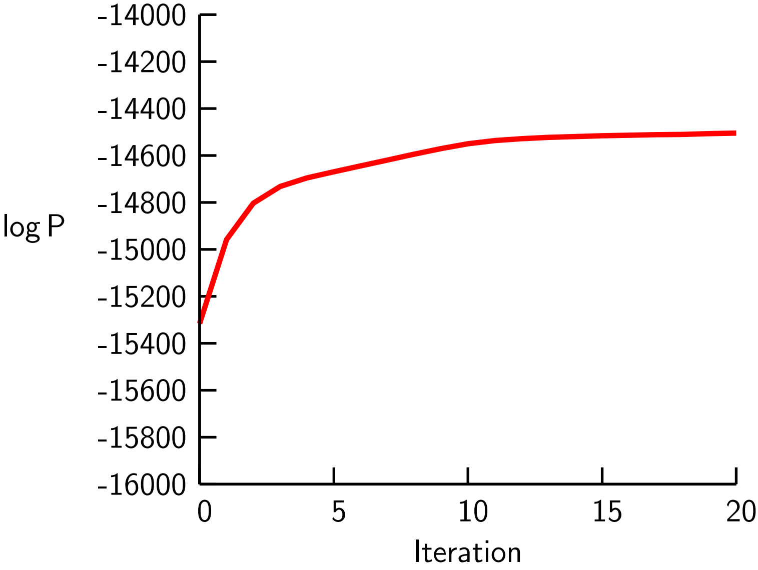
- ATIS treebank consists of 1,300 hand-constructed parse trees
- ignore the words (in this experiment)
- about 1,000 PCFG rules are needed to build these trees



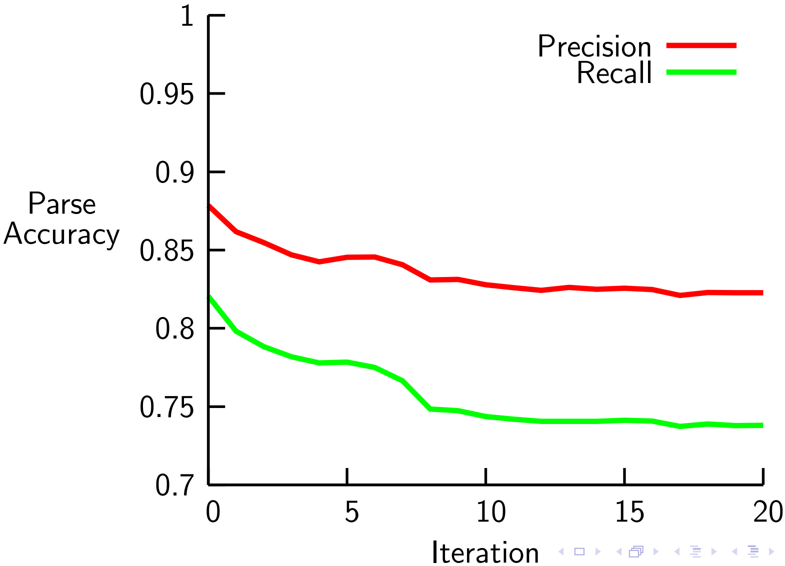
Training from real language

1. Extract productions from trees and estimate probabilities probabilities from trees to produce PCFG.
2. Initialize EM with the treebank grammar and MLE probabilities
3. Apply EM (to strings alone) to re-estimate production probabilities.
4. At each iteration:
    - Measure the likelihood of the training data and the quality of the parses produced by each grammar.
    - Test on training data (so poor performance is not due to overlearning).

Probability of training strings



Accuracy of parses produced using the learnt grammar



Why doesn't this work?

- Divergence between likelihood and parse accuracy
  $\Rightarrow$ *probabilistic model and/or objective function are wrong*
- Bayesian prior preferring smaller grammars doesn't help
- What could be wrong?
  - Wrong kind of grammar (Klein and Manning)
  - Wrong training data (Yang)
  - Predicting words is wrong objective
  - Grammar *ignores semantics* (Zettlemoyer and Collins)

de Marken (1995) "Lexical heads, phrase structure and the induction of grammar"

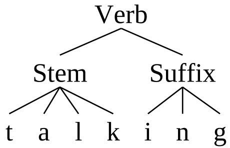
Outline

Concatenative morphology as grammar

- Too many things could be going wrong in learning syntax start with something simpler!
- Input data: regular verbs (in broad phonemic representation)
- Learning goal: segment verbs into stems and inflectional suffixes

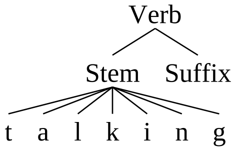
$$
\begin{aligned}
&\text{Verb} \rightarrow \text{Stem Suffix} && \\
&\text{Stem} \rightarrow w && w \in \Sigma^\star \\
&\text{Suffix} \rightarrow w && w \in \Sigma^\star
\end{aligned}
$$

Data = t a l k i n g



Maximum likelihood estimation won't work

- A *saturated model* has one parameter (i.e., rule) for each datum (word)
- The grammar that analyses *each word as a stem with a null suffix* is a saturated model
- Saturated models in general have highest likelihood
  - ⇒ saturated model *exactly replicates* training data
  - ⇒ doesn't "waste probability" on any other strings
  - ⇒ maximizes likelihood of training data



Bayesian learning

$$\underbrace{\text{P(Hypothesis}|\text{Data)}}_{\text{Posterior}} \propto \underbrace{\text{P(Data}|\text{Hypothesis)}}_{\text{Likelihood}} \underbrace{\text{P(Hypothesis)}}_{\text{Prior}}$$

- A statistical learning framework that integrates:
  - *likelihood of the data* (prediction)
  - bias or *prior knowledge* (e.g., innate constraints)
    - *markedness constraints* (e.g., syllables have onsets)
    - prefer "simple" or *sparse* grammars
    - can be over-ridden by sufficient data

The Bayesian approach to learning

$$\underbrace{\mathrm{P(Hypothesis|Data)}}_{\mathrm{Posterior}} \propto \underbrace{\mathrm{P(Data|Hypothesis)}}_{\mathrm{Likelihood}} \underbrace{\mathrm{P(Hypothesis)}}_{\mathrm{Prior}}$$

- The posterior probability quantifies how compatible a hypothesis (grammar) is with the data and the prior
- In general many grammars will have non-neglible posterior probability, especially at early stages of learning
- We lose information when we commit to a single grammar

$\Rightarrow$ Bayesians prefer to work with the full posterior distribution

Bayesian computation and Monte Carlo methods

- A grammar is a finite object, but a probability distribution over grammars need not be
- sometimes there may be an explicit formula for the posterior
- but sometimes all we can do is approximate the posterior
- One way of approximating a distribution to produce a large number of samples from it
- The more samples we collect, the closer they approximate the posterior
- *Monte Carlo methods* can be used to produce samples from a wide variety of posterior distributions

Markov Chain Monte Carlo

- Given inputs $\mathbf{w} = (w_1, \ldots, w_n)$ and (guesses for) analyses $\mathbf{t} = (t_1, \ldots, t_n)$ and grammar g, repeat:
  - Sample a new grammar g from posterior $\mathrm{P}(g|\mathbf{w}, \mathbf{t})$
  - Using new g, sample new analyses $\mathbf{t}$ from $\mathrm{P}(\mathbf{t}|g, \mathbf{w})$

$$\begin{array}{rcl}
g^{(1)} & \sim & \mathrm{P}(g|\mathbf{w}, \mathbf{t}^{(0)}) \\
\mathbf{t}^{(1)} & \sim & \mathrm{P}(\mathbf{t}|\mathbf{w}, g^{(1)}) \\
g^{(2)} & \sim & \mathrm{P}(g|\mathbf{w}, \mathbf{t}^{(1)}) \\
\mathbf{t}^{(2)} & \sim & \mathrm{P}(\mathbf{t}|\mathbf{w}, g^{(2)}) \\
 & \ldots &
\end{array}$$

- This defines a Markov Chain known as the *Gibbs sampler*
- Theorem: under a wide range of conditions, this converges to posterior distribution on g and $\mathbf{t}$

Component-wise Markov Chain Monte Carlo

- Inputs $\mathbf{w} = (w_1, \ldots, w_n)$, analyses $\mathbf{t} = (t_1, \ldots, t_n)$ and grammar g
- Sometimes it is possible to *integrate out the grammar*

$$\mathrm{P}(t_i | w_i, \mathbf{t}_{-i}) = \int \mathrm{P}(t_i | w_i, g) \mathrm{P}(g | \mathbf{w}_{-i}, \mathbf{t}_{-i}) \, dg$$

  where $\mathbf{t}_{-i}$ is the set of analyses for all inputs except w_i
- If you can integrate out the grammar, you can define a *component-wise Gibbs sampler* by repeating the following:
  - Pick an input w_i at random
  - Sample t_i from $\mathrm{P}(t | w_i, \mathbf{t}_{-i})$
- Remarkably similar to attractor networks, but has a *a sound probabilistic interpretation*

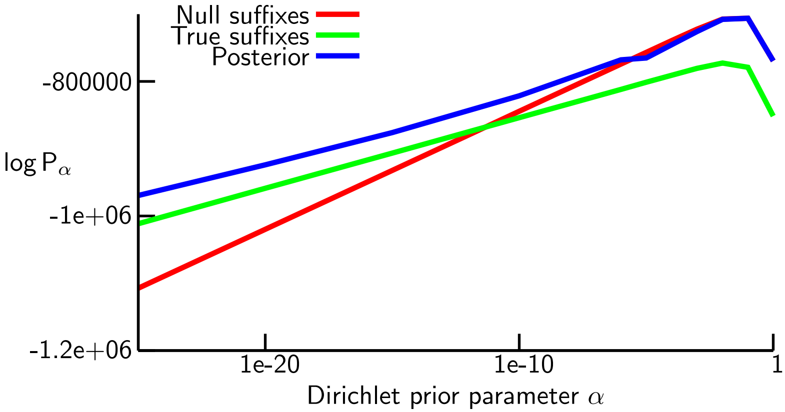
Morphological segmentation experiment

- Bayesian estimator with *Dirichlet prior* with parameter α
  - prefers sparser solutions (i.e., fewer stems and suffixes) as $\alpha \to 0$
- Component-wise Gibbs sampler samples from posterior distribution of parses
  - reanalyses each word based on parses of the other words
- Trained on orthographic verbs from U Penn. Wall Street Journal treebank
  - behaves similarly with broad phonemic child-directed input

Posterior samples from WSJ verb tokens

$\alpha = 0.1$		$\alpha = 10^{-5}$		$\alpha = 10^{-10}$		$\alpha = 10^{-15}$	
expect		expect		expect		expect	
expects		expects		expects		expects	
expected		expected		expected		expected	
expecting		expect	ing	expect	ing	expect	ing
include		include		include		include	
includes		includes		includ	es	includ	es
included		included		includ	ed	includ	ed
including		including		including		including	
add		add		add		add	
adds		adds		adds		add	s
added		added		add	ed	added	
adding		adding		add	ing	add	ing
continue		continue		continue		continue	
continues		continues		continue	s	continue	s
continued		continued		continu	ed	continu	ed
continuing		continuing		continu	ing	continu	ing
report		report		report		report	

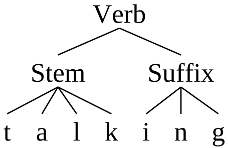
Log posterior of models on token data



- Correct solution is nowhere near as likely as posterior

$\Rightarrow$ no point trying to fix algorithm because *model is wrong!*

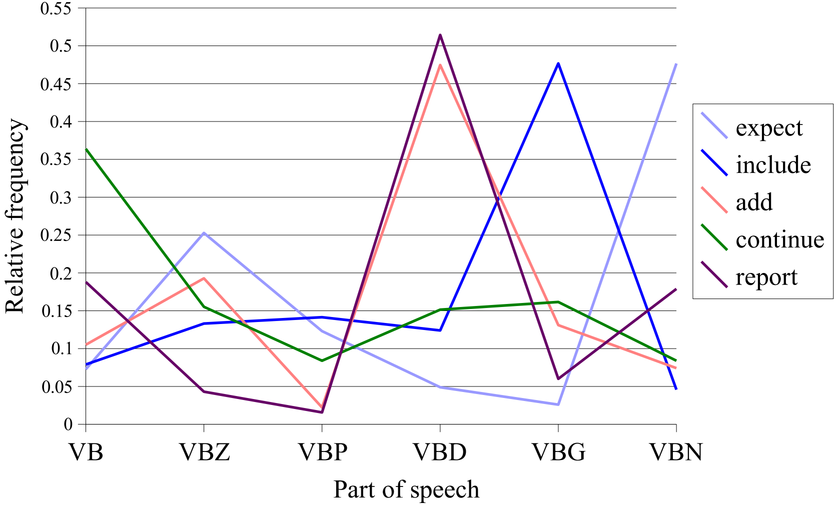
Independence assumptions in PCFG model



$$P(\text{Word}) = P(\text{Stem})P(\text{Suffix})$$

- Model expects relative frequency of each suffix *to be the same for all stems*

Relative frequencies of inflected verb forms



Types and tokens

- A word *type* is a distinct word shape
- A word *token* is an occurrence of a word

Data	=	"the cat chased the other cat"
Tokens	=	"the" 2, "cat" 2, "chased" 1, "other" 1
Types	=	"the" 1, "cat" 1, "chased" 1, "other" 1

- Using word types instead of word tokens effectively normalizes for frequency variations

Posterior samples from WSJ verb *types*

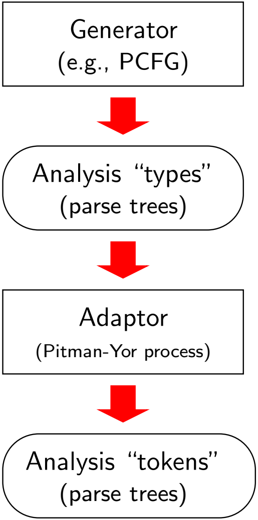
$\alpha = 0.1$		$\alpha = 10^{-5}$		$\alpha = 10^{-10}$		$\alpha = 10^{-15}$	
expect		expect		expect		exp	ect
expects		expect	s	expect	s	exp	ects
expected		expect	ed	expect	ed	exp	ected
expect	ing	expect	ing	expect	ing	exp	ecting
include		includ	e	includ	e	includ	e
include	s	includ	es	includ	es	includ	es
included		includ	ed	includ	ed	includ	ed
including		includ	ing	includ	ing	includ	ing
add		add		add		add	
adds		add	s	add	s	add	s
add	ed	add	ed	add	ed	add	ed
adding		add	ing	add	ing	add	ing
continue		continu	e	continu	e	continu	e
continue	s	continu	es	continu	es	continu	es
continu	ed	continu	ed	continu	ed	continu	ed
continuing		continu	ing	continu	ing	continu	ing
report		report		repo	rt	rep	ort

Learning from types and tokens

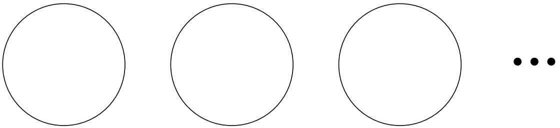
- Overdispersion in suffix distribution can be ignored by learning from types instead of tokens
- Some psycholinguistics claim that children learn morphology from types (Pierrehumbert 2003)
- To identify word types the input must be segmented into word tokens
- But the input doesn't come neatly segmented into tokens!
- We have been developing *two stage adaptor models* to deal with type-token mismatches

Two stage adaptor framework

- *Generator* produces structures
- *Adaptor* replicates them an arbitrary number of times
- Generator learns structure from "types"
- Adaptor learns (power law) frequencies from tokens



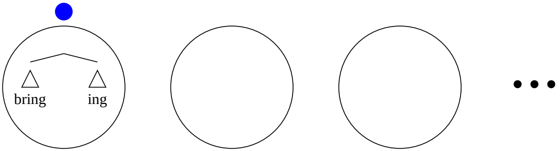
Chinese restaurant process sampler



- $\mathrm{P}(t_i | \mathbf{w}, \mathbf{t}_{-i})$ is given by a Chinese restaurant process
- The input tokens are "customers" seated at "tables"
- Each table is labeled with an analysis, which is the analysis of all of the customers at that table
- If there are currently m tables occupied, with n_k customers sitting at table k

$$\mathrm{P}(\text{next table} = k) \;\propto\; \begin{cases} n_k - a & \text{for } k \leq m \\ ma + b & \text{if } k = m + 1 \end{cases}$$

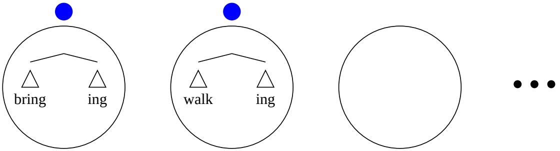
Chinese restaurant process sampler (1)



- $\mathrm{P}(t_i | \mathbf{w}, \mathbf{t}_{-i})$ is given by a Chinese restaurant process
- The input tokens are "customers" seated at "tables"
- Each table is labeled with an analysis, which is the analysis of all of the customers at that table
- If there are currently m tables occupied, with n_k customers sitting at table k

$$\mathrm{P}(\text{next table} = k) \;\; \propto \;\; \begin{cases} n_k - a & \text{for } k \leq m \\ ma + b & \text{if } k = m + 1 \end{cases}$$

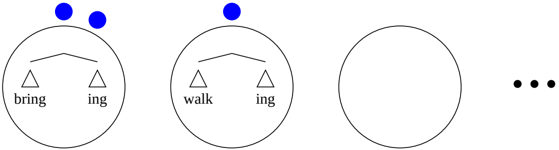
Chinese restaurant process sampler (2)



- $\mathrm{P}(t_i | \mathbf{w}, \mathbf{t}_{-i})$ is given by a Chinese restaurant process
- The input tokens are “customers” seated at “tables”
- Each table is labeled with an analysis, which is the analysis of all of the customers at that table
- If there are currently m tables occupied, with n_k customers sitting at table k

$$\mathrm{P}(\text{next table} = k) \;\propto\; \begin{cases} n_k - a & \text{for } k \leq m \\ ma + b & \text{if } k = m + 1 \end{cases}$$

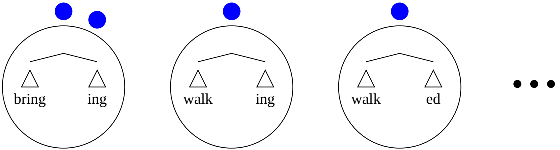
Chinese restaurant process sampler (3)



- $\mathrm{P}(t_i | \mathbf{w}, \mathbf{t}_{-i})$ is given by a Chinese restaurant process
- The input tokens are "customers" seated at "tables"
- Each table is labeled with an analysis, which is the analysis of all of the customers at that table
- If there are currently m tables occupied, with n_k customers sitting at table k

$$\mathrm{P}(\text{next table} = k) \quad \propto \quad \begin{cases} n_k - a & \text{for } k \leq m \\ ma + b & \text{if } k = m + 1 \end{cases}$$

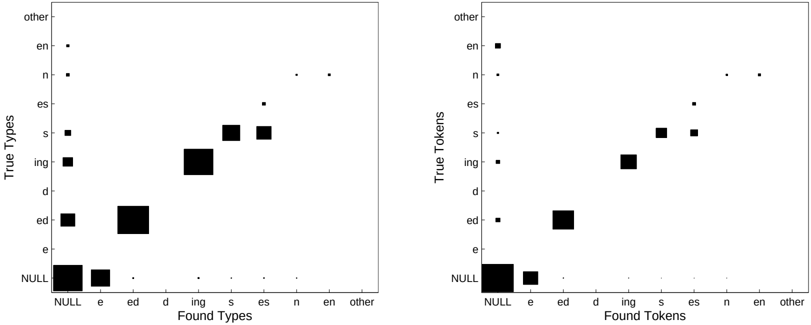
Chinese restaurant process sampler (4)



- $P(t_i | \mathbf{w}, \mathbf{t}_{-i})$ is given by a Chinese restaurant process
- The input tokens are "customers" seated at "tables"
- Each table is labeled with an analysis, which is the analysis of all of the customers at that table
- If there are currently m tables occupied, with n_k customers sitting at table k

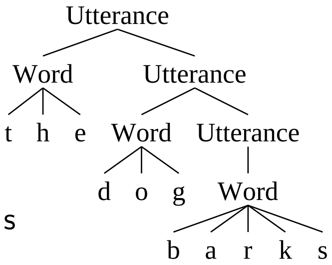
$$P(\text{next table} = k) \propto \begin{cases} n_k - a & \text{for } k \leq m \\ ma + b & \text{if } k = m + 1 \end{cases}$$

Concatenative morphology confusion matrix



Outline

Grammars for word segmentation

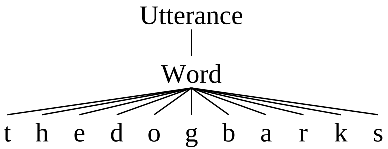


Sample input = t h e d o g b a r k s

Utterance → Word Utterance
Utterance → Word
Word → w $\quad\quad w \in \Sigma^\star$

- These are *unigram models* of sentences (each word is *conditionally independent* of its neighbours)
- This assumption is standardly made in models of word segmentation, but is it accurate?

Saturated grammar is maximum likelihood grammar



- Grammar that generates each utterance as a single word exactly matches input distribution

$\Rightarrow$ saturated grammar is maximum likelihood grammar

$\Rightarrow$ use Bayesian estimation with a sparse Dirichlet process prior

- CRP used to construct Monte Carlo Sampler

Segmentations found by unigram model

yuwant tu si D6bUk	lUk D*z 6b7 wIT hIz h&t
&nd 6dOgi	yu wanttu lUk&tDIs
lUk&tDIs	h&v6 drINk
oke nQ	WAtsDIs
WAtsD&t	WAtIzIt
lUk k&nyu tek ItQt	tek D6dOgi Qt

- Trained on Brent broad phonemic child-directed corpus
- Tends to find *multi-word expressions*, e.g, *yuwant*
- Word finding accuracy is less than Brent's accuracy
- *These solutions are more likely under Brent's model than the solutions Brent found*

⇒ Brent's search procedure is not finding optimal solution

Contextual dependencies in word segmentation

- Unigram model assumes words are independently distributed
- but words in multiword expressions are not independently distributed
  - if we train from a corpus in which the words are randomly permuted, the unigram model finds correct segmentations
- Bigram models capture word-word dependencies $P(w_{i+1}|w_i)$
- straight-forward to build a Gibbs sampler,
  even though we don't have a fixed set of words
  - Each step reanalyses a word or pair of words using the analyses of the rest of the input

Segmentations found by bigram model

yu want tu si D6 bUk	lUk D*z 6 b7 wIT hIz h&t
&nd 6 dOgi	yu want tu lUk&t DIs
lUk&t DIs	h&v 6 drINk
oke nQ	WAts DIs
WAts D&t	WAtIz It
lUk k&nyu tek It Qt	tek D6 dOgi Qt

- Bigram model segments much more accurately than unigram model and Brent's model

$\Rightarrow$ *conditional independence alone is not a good cue for word segmentation*

Outline

Conclusion

- We have mathematical and computational tools to connect learning theory and linguistic theory
- Studying learning via *explicit probabilistic models*
  - is compatible with linguistic theory
  - lets us *better understand* why a learning model succeeds or fails
- Bayesian learning lets us combine statistical learning with with prior information
  - priors can encode "Occam's razor" preferences for sparse grammars, and
  - *universal grammar* and *markedness preferences*
  - evaluate usefulness of different types of linguistic universals are for language acquisition

Future work

- Integrate the morphology and word segmentation systems
  - Are their *synergistic interactions* between these components?
- Include other linguistic phenomena
  - Would a *phonological component* improve lexical and morphological acquisition?
- Develop more *realistic training data corpora*
  - Use *forced alignment* to identify pronunciation variants and prosodic properties of words in child-directed speech
- Develop priors that encode *linguistic universals* and *markedness preferences*
  - quantitatively evaluate their usefulness for acquisition