Bayesian Inference of Grammars

Mark Johnson

joint work with Sharon Goldwater and Tom Griffiths

Microsoft Research / Brown University

July 2007

Outline

Research goal: language acquisition

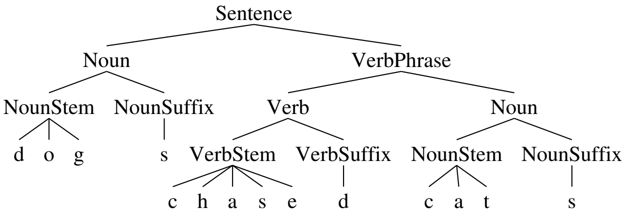
- Goal of this research (as yet unachieved):

  a grammar learning algorithm that trains from:

  Input: "d o g s c h a s e d c a t s"
  (actually broad phonemic transcription)

  and produces:

  Output:



Nonparametric Bayes: *a new lamppost*

Late one night, a drunk guy is crawling around under a lamppost. A cop comes up and asks him what he's doing.

"*I'm looking for my keys*," the drunk says. "*I lost them about three blocks away*."

"*So why aren't you looking for them where you dropped them?*" the cop asks.

The drunk looks at the cop, amazed that he'd ask so obvious a question. "*Because the light is better here.*"

Talk outline

- Maximum likelihood can be applied to grammar induction (expectation maximization) but generally produces poor results. Why?
- *Will new nonparametric Bayesian methods do better?*
  - Chinese restaurant processes and Dirichlet processes
- Strategy: develop methods that work for simple problems
  - morphological segmentation
  - word segmentation of unsegmented phonemic transcripts

Probabilistic Context-Free Grammars

- PCFGs are perhaps the simplest models of hierarchical structure
- Probability of a tree is the *product of the probabilities of the rules* used to construct it

$$\begin{array}{llll} 1.0 & \mathrm{S} \rightarrow \mathrm{NP}\ \mathrm{VP} & & \\ 0.75 & \mathrm{NP} \rightarrow \mathrm{George} & 0.25 & \mathrm{NP} \rightarrow \mathrm{Al} \\ 0.6 & \mathrm{VP} \rightarrow \mathrm{barks} & 0.4 & \mathrm{VP} \rightarrow \mathrm{snores} \end{array}$$

$$\mathrm{P}\left(\begin{array}{cc} \multicolumn{2}{c}{\mathrm{S}} \\ \mathrm{NP} & \mathrm{VP} \\ | & | \\ \mathrm{George} & \mathrm{barks} \end{array}\right) = 0.45 \qquad \mathrm{P}\left(\begin{array}{cc} \multicolumn{2}{c}{\mathrm{S}} \\ \mathrm{NP} & \mathrm{VP} \\ | & | \\ \mathrm{Al} & \mathrm{snores} \end{array}\right) = 0.1$$

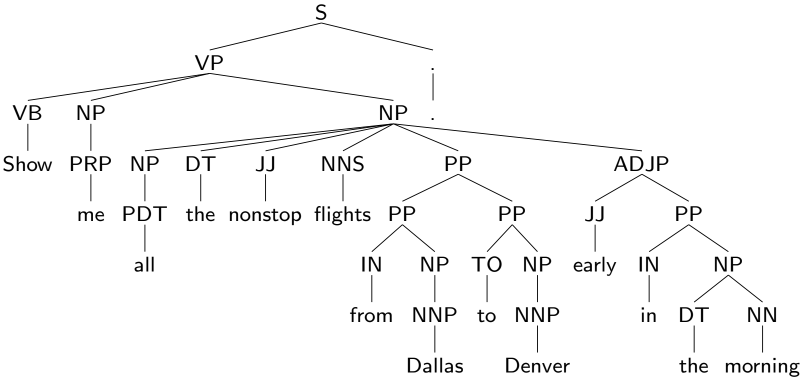
Learning the units of generalization

- Rules are units of generalization in PCFGs
- We can *estimate rule probabilities from data*
  - The Expectation Maximization algorithm finds rule probabilities that locally maximize likelihood of a set of strings
- Find rules using “generate and prune” approach:
  - generate a large number of possible rules
  - estimate the probability of each rule
  - prune low probability rules
- Unsupervised estimation of PCFGs generally produces *poor results*
- Nonparametric Bayesian techniques (CRPs, DPs)
  - *integrate production search with parameter search*
  - *let us construct more elaborate (realistic?) models*

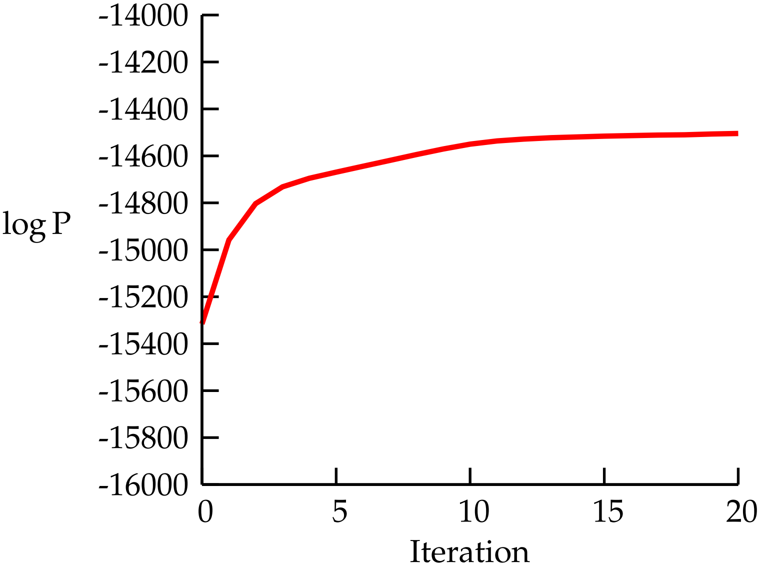
Outline

Learning PCFGs using Expectation Maximization

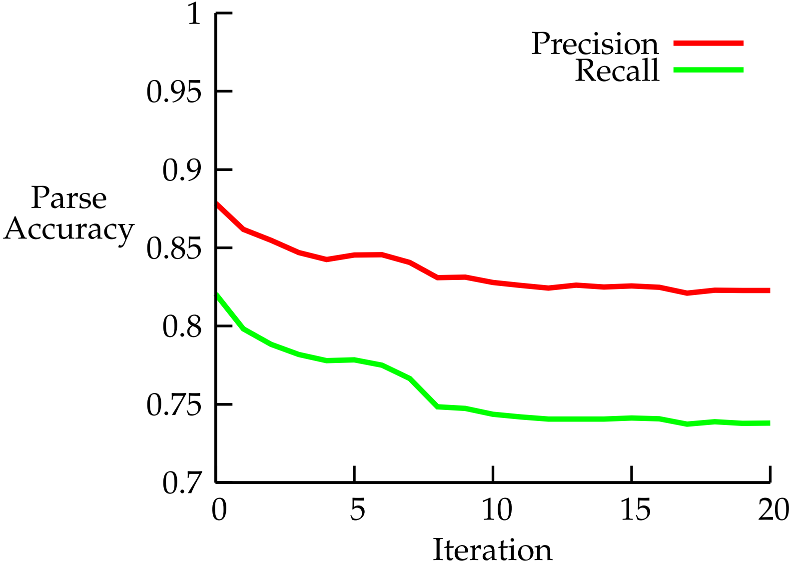
- ATIS treebank consists of 1,300 hand-constructed parse trees
- input consists of POS tags rather than words
- about 1,000 PCFG rules are needed to build these trees



Probability of training strings



Accuracy of parses



The PCFG model is wrong

- Parse accuracy drops as likelihood increases
  - higher likelihood $\not\Rightarrow$ better parses
  - *the statistical model is wrong*
- *Initialized EM with correct parse trees*
  - started with true rules and their probabilities

  $\Rightarrow$ poor performance not due to search error
- Evaluated on training data
  - poor performance not due to over-learning

Why didn't it learn the right grammar?

- higher likelihood $\not\Rightarrow$ parse accuracy

$\Rightarrow$ *probabilistic model and/or estimation procedure are wrong*

- Bayesian prior preferring smaller grammars doesn't help
- What could be wrong?
  - Wrong model of grammar (Klein and Manning)
  - Wrong estimation procedure (Smith and Eisner)
  - Wrong training data (Yang)
  - Predicting word strings is wrong objective
  - Grammar *ignores semantics* (Zettlemoyer and Collins)

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

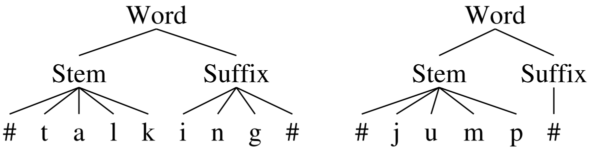
Outline

Learning English verbal morphology

Training data is a sequence of verbs, e.g.

$$\mathcal{D} = (\#\, \mathrm{t\,a\,l\,k\,i\,n\,g}\, \#,\ \#\, \mathrm{j\,u\,m\,p}\, \#, \ldots)$$

Goal: infer trees such as:

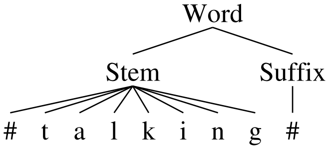


$$\begin{array}{rcl} \text{Word} & \rightarrow & \text{Stem Suffix} \\ \text{Stem} & \rightarrow & \textit{all possible stems} \\ \text{Suffix} & \rightarrow & \textit{all possible suffixes} \end{array}$$

Maximum likelihood always chooses no suffixes

- Maximum likelihood selects production probabilities that minimize KL-divergence between model and data distributions
- *Saturated model* with $\mathrm{P}(\text{Suffix} \rightarrow \#) = 1$ generates training data $\mathcal{D}$ exactly

$\Rightarrow$ saturated model is maximum likelihood estimate



Bayesian estimation

- Bayesian estimates incorporate *prior* $\mathrm{P}(\mathcal{H})$ over hypotheses $\mathcal{H}$ (grammars) as well as *likelihood* $\mathrm{P}(\mathcal{D}|\mathcal{H})$ of data $\mathcal{D}$

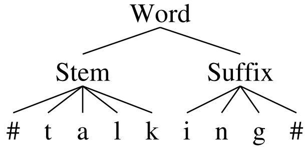
$$\underbrace{\mathrm{P}(\mathcal{H}|\mathcal{D})}_{\text{Posterior}} \propto \underbrace{\mathrm{P}(\mathcal{D}|\mathcal{H})}_{\text{Likelihood}} \; \underbrace{\mathrm{P}(\mathcal{H})}_{\text{Prior}}$$

- Priors can be sensitive to linguistic structure (e.g., a word should contain a vowel)
- Priors can encode linguistic universals and markedness preferences (e.g., complex clusters appear at word onsets)
- Priors can prefer *sparse solutions*
- The choice of the prior is as much a linguistic issue as the design of the grammar!

Morphological segmentation experiment

- Dirichlet prior prefers grammars with *fewer stems and suffixes*
  - grammars are sparser as Dirichlet parameter $\alpha \to 0$

$$\begin{array}{rcl} \text{Word} & \to & \text{Stem Suffix} \\ \text{Stem} & \to & \textit{all possible stems} \\ \text{Suffix} & \to & \textit{all possible suffixes} \end{array}$$

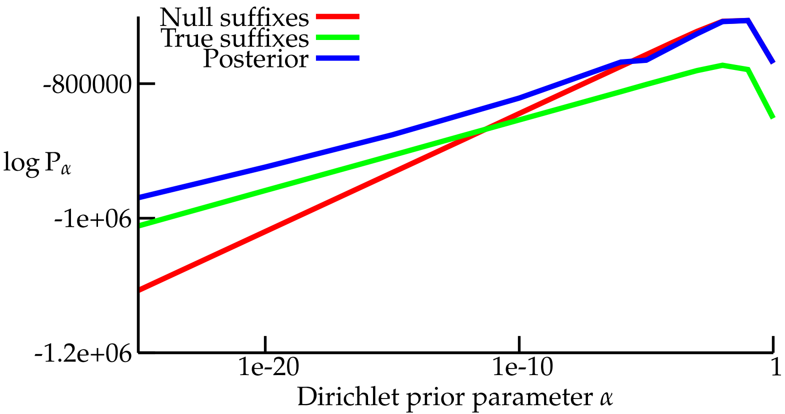


- Trained on orthographic verbs from PTB Wall Street Journal corpus
- Gibbs sampler samples from posterior distribution over parses
  - reanalyses each word using grammar estimated from parses of other words

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	

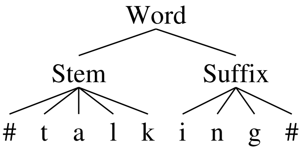
Log posterior of models on token data



- Correct solution is nowhere near as likely as posterior

$\Rightarrow$ model is wrong!

Independence assumption in PCFG model

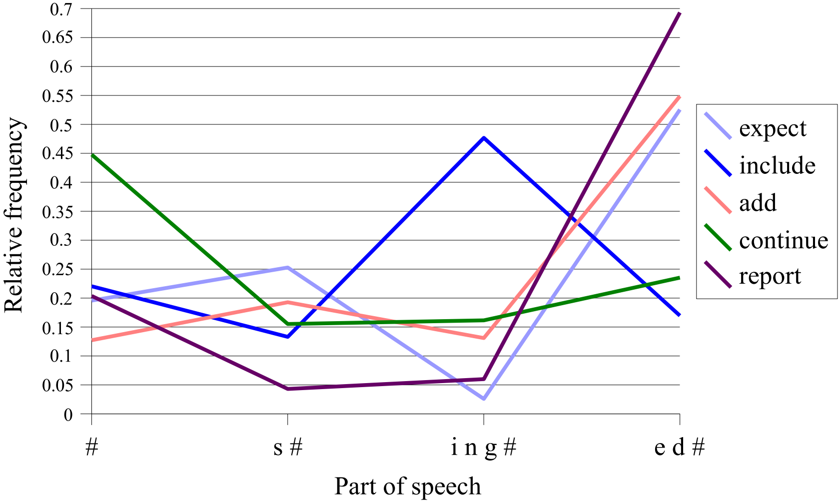
P (


)

$$= \mathrm{P}(\text{Word} \rightarrow \text{Stem Suffix})\ \mathrm{P}(\text{Stem} \rightarrow \#\,\mathrm{t\,a\,l\,k})\ \mathrm{P}(\text{Suffix} \rightarrow \mathrm{i\,n\,g}\,\#)$$

- Model assumes relative frequency of each suffix *to be the same for all stems*
- This turns out not to be true

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

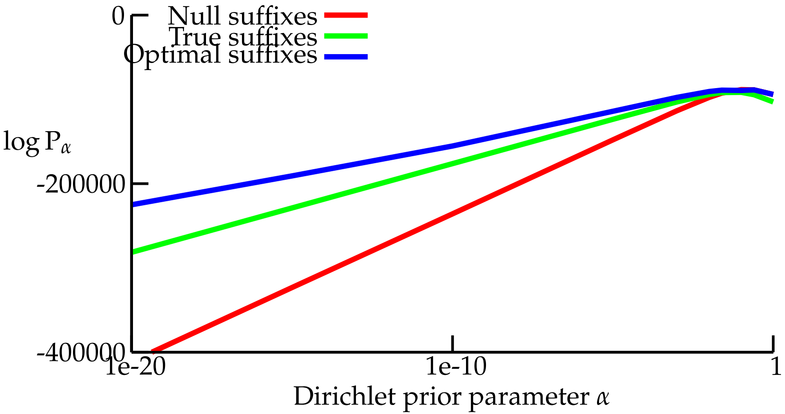
$$
\begin{aligned}
\mathcal{D} &= \text{"the cat chased the other cat"} \\
\mathrm{Tokens}(\mathcal{D}) &= \text{"the"}, \text{"cat"}, \text{"chased"}, \text{"the"}, \text{"other"}, \text{"cat"} \\
\mathrm{Types}(\mathcal{D}) &= \text{"the"}, \text{"cat"}, \text{"chased"}, \text{"other"}
\end{aligned}
$$

- Estimating production probabilities from *word types* rather than word tokens eliminates (most) frequency variation
  - 4 common verb suffixes, so when estimating from verb types $\mathrm{P}(\mathrm{Suffix} \to \mathrm{i\,n\,g\,\#}) \approx 0.25$
- Some psycholinguistics claim that children learn morphology from types (Bybee, Pierrehumbert)

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

Log posterior of models on type data



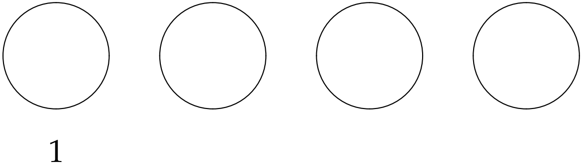
- Correct solution is close to optimal at $\alpha = 10^{-3}$

Outline

Generating tokens but learning from types

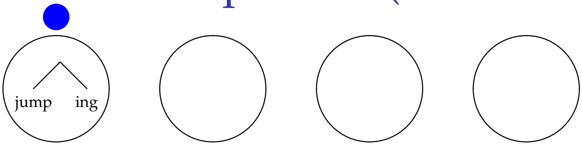
- Over-dispersion ⇒ estimate from types rather than tokens
- But in many cases the types are not available (e.g., speech does not come segmented into words ("s e e t h e d o g g i e"))
- Labeled Chinese restaurant process models
  - *Labeling distribution* P_G generates types
  - *Chinese restaurant process* replicates types to produce tokens
  - labeling distribution can be estimated from the CRP
- Adaptor grammars use CFGs to produce the labels for a hierarchy of CRPs

Chinese restaurant process (1st enters)



- Tokens $\sim$ customers, types $\sim$ tables, analyses $\sim$ labels
- Each "table" can seat an infinite number of "customers"
- Each "table" has a "dish" (label) shared by all "customers"
- Labels are generated by labeling distribution P_G
- Customer $n+1$ walks into restaurant with n_k customers sitting at table $k \in \{1, \ldots, m\}$
  - sits at old table $k \leq m$ with probability $n_k/(n+\alpha)$ and emits table's label
  - sits at new table $k = m+1$ with probability $\alpha/(n+\alpha)$ and generates new label y for table with probability $\mathrm{P}_G(y)$

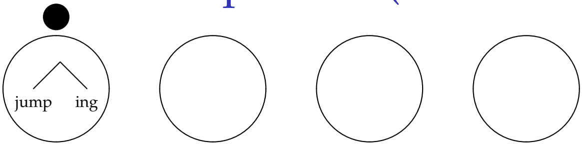
Chinese restaurant process (1st seated)



$\mathrm{P}_G(\text{jump} + \text{ing})$

- Tokens $\sim$ customers, types $\sim$ tables, analyses $\sim$ labels
- Each "table" can seat an infinite number of "customers"
- Each "table" has a "dish" (label) shared by all "customers"
- Labels are generated by labeling distribution P_G
- Customer $n + 1$ walks into restaurant with n_k customers sitting at table $k \in \{1, \ldots, m\}$
  - sits at old table $k \leq m$ with probability $n_k/(n + \alpha)$ and emits table's label
  - sits at new table $k = m + 1$ with probability $\alpha/(n + \alpha)$ and generates new label y for table with probability $\mathrm{P}_G(y)$

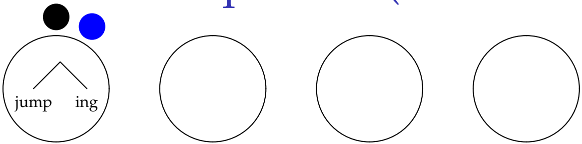
Chinese restaurant process (2nd enters)



$\frac{1}{1+\alpha}$ $\frac{\alpha}{1+\alpha}$

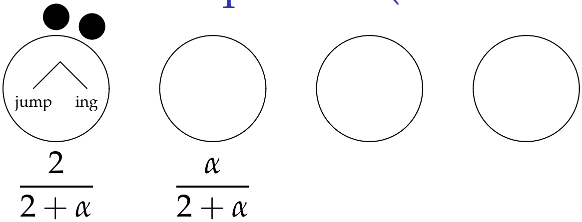
- Tokens $\sim$ customers, types $\sim$ tables, analyses $\sim$ labels
- Each "table" can seat an infinite number of "customers"
- Each "table" has a "dish" (label) shared by all "customers"
- Labels are generated by labeling distribution P_G
- Customer $n+1$ walks into restaurant with n_k customers sitting at table $k \in \{1, \ldots, m\}$
  - sits at old table $k \leq m$ with probability $n_k/(n+\alpha)$ and emits table's label
  - sits at new table $k = m+1$ with probability $\alpha/(n+\alpha)$ and generates new label y for table with probability $\mathrm{P}_G(y)$

Chinese restaurant process (2nd seated)



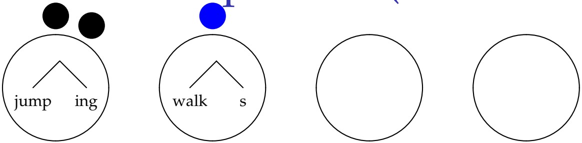
- Tokens $\sim$ customers, types $\sim$ tables, analyses $\sim$ labels
- Each "table" can seat an infinite number of "customers"
- Each "table" has a "dish" (label) shared by all "customers"
- Labels are generated by labeling distribution P_G
- Customer $n+1$ walks into restaurant with n_k customers sitting at table $k \in \{1, \ldots, m\}$
  - sits at old table $k \leq m$ with probability $n_k/(n+\alpha)$ and emits table's label
  - sits at new table $k = m+1$ with probability $\alpha/(n+\alpha)$ and generates new label y for table with probability $P_G(y)$

Chinese restaurant process (3rd enters)



- Tokens $\sim$ customers, types $\sim$ tables, analyses $\sim$ labels
- Each "table" can seat an infinite number of "customers"
- Each "table" has a "dish" (label) shared by all "customers"
- Labels are generated by labeling distribution P_G
- Customer $n+1$ walks into restaurant with n_k customers sitting at table $k \in \{1, \ldots, m\}$
  - sits at old table $k \leq m$ with probability $n_k/(n+\alpha)$ and emits table's label
  - sits at new table $k = m+1$ with probability $\alpha/(n+\alpha)$ and generates new label y for table with probability $\mathrm{P}_G(y)$

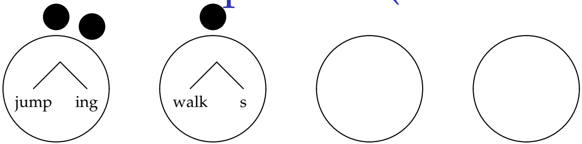
Chinese restaurant process (3rd seated)



$\mathrm{P}_G(\mathrm{walk} + \mathrm{s})$

- Tokens $\sim$ customers, types $\sim$ tables, analyses $\sim$ labels
- Each "table" can seat an infinite number of "customers"
- Each "table" has a "dish" (label) shared by all "customers"
- Labels are generated by labeling distribution P_G
- Customer $n + 1$ walks into restaurant with n_k customers sitting at table $k \in \{1, \ldots, m\}$
  - sits at old table $k \leq m$ with probability $n_k / (n + \alpha)$ and emits table's label
  - sits at new table $k = m + 1$ with probability $\alpha / (n + \alpha)$ and generates new label y for table with probability $\mathrm{P}_G(y)$

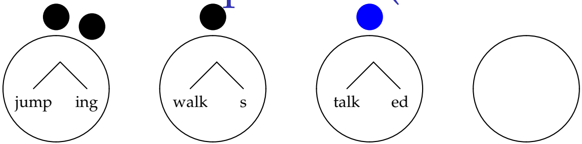
Chinese restaurant process (4th enters)



$$\frac{2}{3+\alpha} \qquad \frac{1}{3+\alpha} \qquad \frac{\alpha}{3+\alpha}$$

- Tokens $\sim$ customers, types $\sim$ tables, analyses $\sim$ labels
- Each "table" can seat an infinite number of "customers"
- Each "table" has a "dish" (label) shared by all "customers"
- Labels are generated by labeling distribution P_G
- Customer $n+1$ walks into restaurant with n_k customers sitting at table $k \in \{1, \ldots, m\}$
  - sits at old table $k \leq m$ with probability $n_k/(n+\alpha)$ and emits table's label
  - sits at new table $k = m+1$ with probability $\alpha/(n+\alpha)$ and generates new label y for table with probability $P_G(y)$

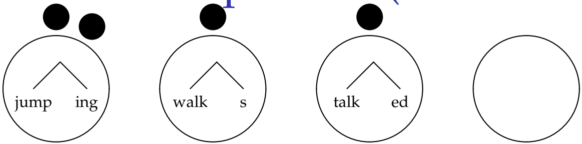
Chinese restaurant process (4th seated)



$P_G(\text{talk} + \text{ed})$

- Tokens ∼ customers, types ∼ tables, analyses ∼ labels
- Each "table" can seat an infinite number of "customers"
- Each "table" has a "dish" (label) shared by all "customers"
- Labels are generated by labeling distribution P_G
- Customer $n + 1$ walks into restaurant with n_k customers sitting at table $k \in \{1, \ldots, m\}$
  - sits at old table $k \leq m$ with probability $n_k/(n + \alpha)$ and emits table's label
  - sits at new table $k = m + 1$ with probability $\alpha/(n + \alpha)$ and generates new label y for table with probability $P_G(y)$

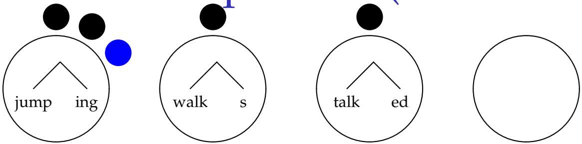
Chinese restaurant process (5th enters)



$$\frac{2}{4+\alpha} \qquad \frac{1}{4+\alpha} \qquad \frac{1}{4+\alpha} \qquad \frac{\alpha}{4+\alpha}$$

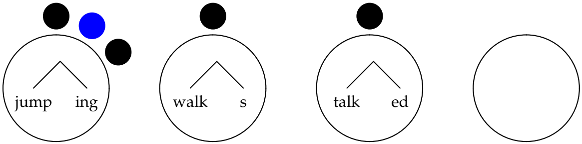
- Tokens $\sim$ customers, types $\sim$ tables, analyses $\sim$ labels
- Each "table" can seat an infinite number of "customers"
- Each "table" has a "dish" (label) shared by all "customers"
- Labels are generated by labeling distribution P_G
- Customer $n+1$ walks into restaurant with n_k customers sitting at table $k \in \{1, \ldots, m\}$
  - sits at old table $k \leq m$ with probability $n_k/(n+\alpha)$ and emits table's label
  - sits at new table $k = m+1$ with probability $\alpha/(n+\alpha)$ and generates new label y for table with probability $P_G(y)$

Chinese restaurant process (5th seated)



- Tokens $\sim$ customers, types $\sim$ tables, analyses $\sim$ labels
- Each "table" can seat an infinite number of "customers"
- Each "table" has a "dish" (label) shared by all "customers"
- Labels are generated by labeling distribution P_G
- Customer $n+1$ walks into restaurant with n_k customers sitting at table $k \in \{1, \ldots, m\}$
  - sits at old table $k \leq m$ with probability $n_k/(n+\alpha)$ and emits table's label
  - sits at new table $k = m+1$ with probability $\alpha/(n+\alpha)$ and generates new label y for table with probability $\mathrm{P}_G(y)$

Estimating CRP models via Gibbs sampling



- Gibbs sampling: resample the analysis of each word token given the analyses of other tokens
- Each word token is a customer: its table's label is its analysis
- Gibbs sampling step:
  - remove token from its current table (delete empty tables)
  - choose a table for the token (possibly creating new table)
- *Labeling distribution* P_G *estimated from labels on tables*
  - P_G is estimated from types (approximately)

Outline

Introduction

Learning probabilistic context-free grammars

Morphological segmentation

Learning from types with Chinese restaurant processes

Adaptor grammars

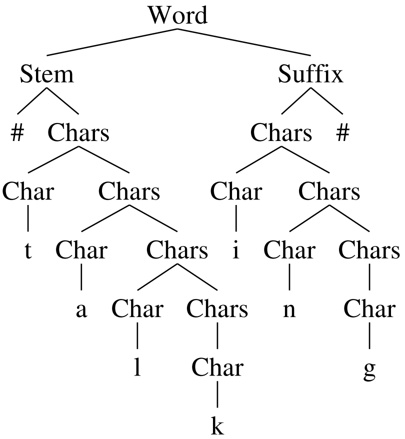
Bigram dependencies in word segmentation

Conclusion

From PCFGs to adaptor grammars

- An *adaptor grammar* is a PCFG together with a parameter α_A for each non-terminal A
- If $\alpha_A > 0$ then A is *adapted*
- Each adapted non-terminal A has its own CRP, labeled with trees generated by A
- Non-adapted non-terminals expand as in PCFG, i.e., pick production $A \to B_1 \dots B_n$ and recursively expand $B_1, \dots, B_n$
- An adapted non-terminal A expands by:
  - each expansion is a customer entering A's restaurant
  - if customer sits at new table, generate tree to label new table as if A were not adapted
  - return the tree labeling the customer's table

Adaptor grammar morphology example



$$
\begin{array}{rcl}
\text{Word} & \rightarrow & \text{Stem Suffix} \\
\text{Stem} & \rightarrow & \#\,\text{Chars} \\
\text{Suffix} & \rightarrow & \# \\
\text{Suffix} & \rightarrow & \text{Chars}\,\# \\
\text{Chars} & \rightarrow & \text{Char} \\
\text{Chars} & \rightarrow & \text{Char Chars} \\
\text{Char} & \rightarrow & \text{a} \mid \ldots \mid \text{z}
\end{array}
$$

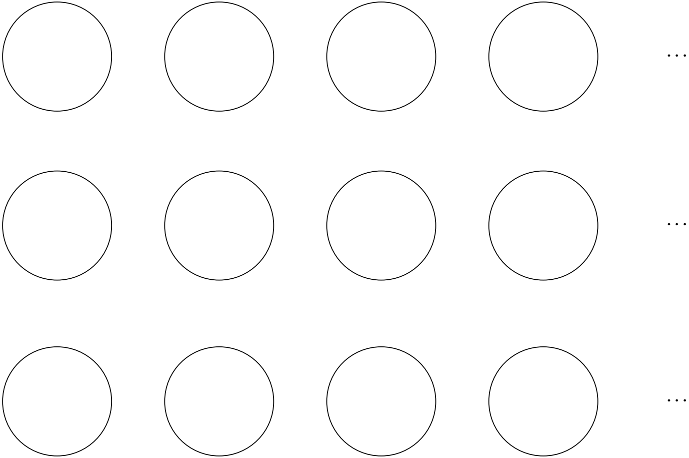
- CRPs for non-terminals Word, Stem and Suffix
  - Stem and Suffix CRPs generate all possible stems and suffixes

Morphology adaptor grammar (0)

Word restaurant
Word $\rightarrow$ Stem Suffix

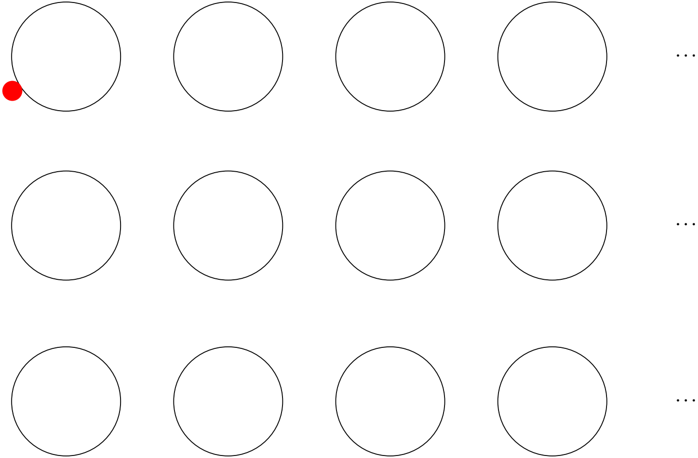
Stem restaurant
Stem $\rightarrow$ #
Stem $\rightarrow$ # Chars

Suffix restaurant
Suffix $\rightarrow$ #
Suffix $\rightarrow$ Chars #



Chars factory
Chars $\rightarrow$ Char
Chars $\rightarrow$ Char Chars
Char $\rightarrow$ a . . . z

Morphology adaptor grammar (1a)



Word restaurant
Word → Stem Suffix

Stem restaurant
Stem → #
Stem → # Chars

Suffix restaurant
Suffix → #
Suffix → Chars #

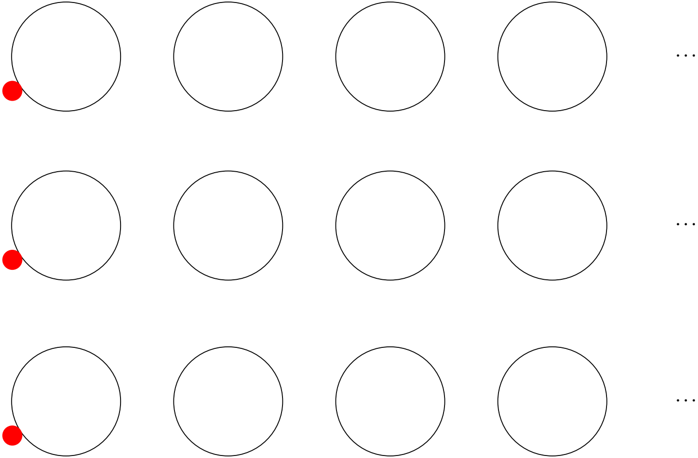
Chars factory
Chars → Char
Chars → Char Chars
Char → a . . . z

Morphology adaptor grammar (1b)

Word restaurant
Word → Stem Suffix

Stem restaurant
Stem → #
Stem → # Chars

Suffix restaurant
Suffix → #
Suffix → Chars #



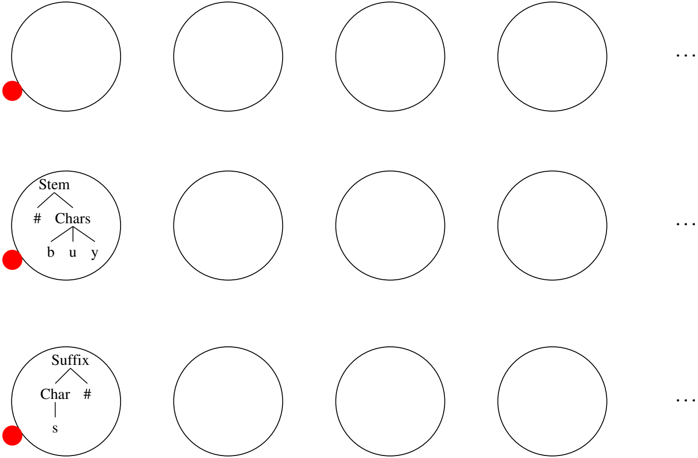
Chars factory
Chars → Char
Chars → Char Chars
Char → a . . . z

Morphology adaptor grammar (1c)

Word restaurant
Word → Stem Suffix

Stem restaurant
Stem → #
Stem → # Chars

Suffix restaurant
Suffix → #
Suffix → Chars #



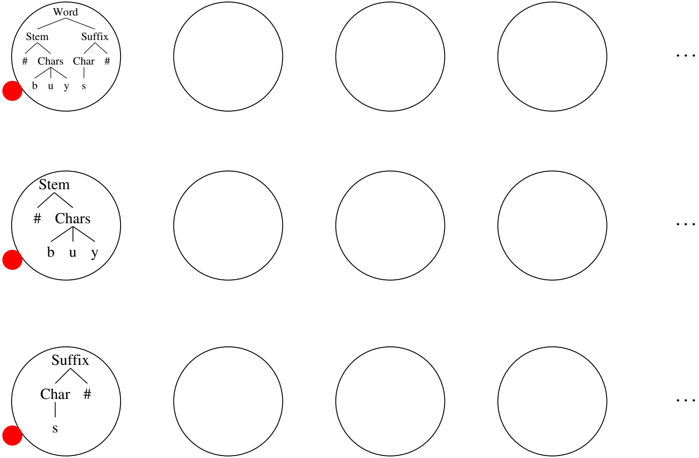
Chars factory
Chars → Char
Chars → Char Chars
Char → a . . . z

Morphology adaptor grammar (1d)

Word restaurant
Word → Stem Suffix

Stem restaurant
Stem → #
Stem → # Chars

Suffix restaurant
Suffix → #
Suffix → Chars #



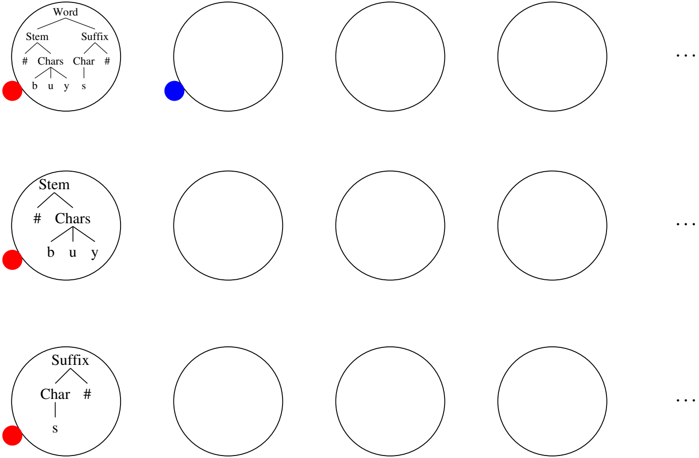
Chars factory
Chars → Char
Chars → Char Chars
Char → a . . . z

Morphology adaptor grammar (2a)

Word restaurant
Word → Stem Suffix

Stem restaurant
Stem → #
Stem → # Chars

Suffix restaurant
Suffix → #
Suffix → Chars #



Chars factory
Chars → Char
Chars → Char Chars
Char → a . . . z

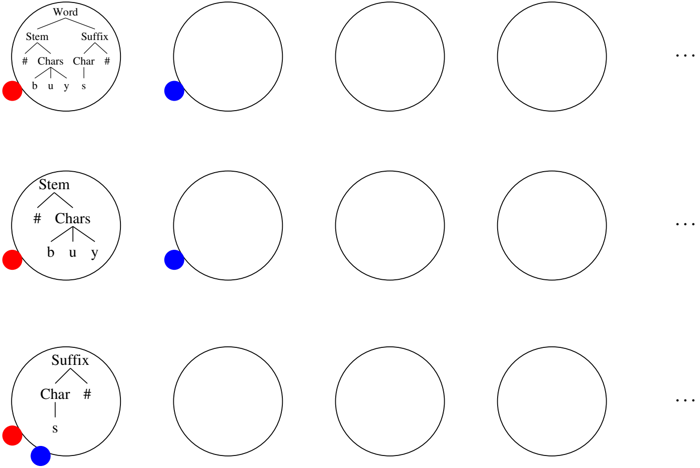
Morphology adaptor grammar (2b)

Word restaurant
Word → Stem Suffix

Stem restaurant
Stem → #
Stem → # Chars

Suffix restaurant
Suffix → #
Suffix → Chars #

Chars factory
Chars → Char
Chars → Char Chars
Char → a . . . z

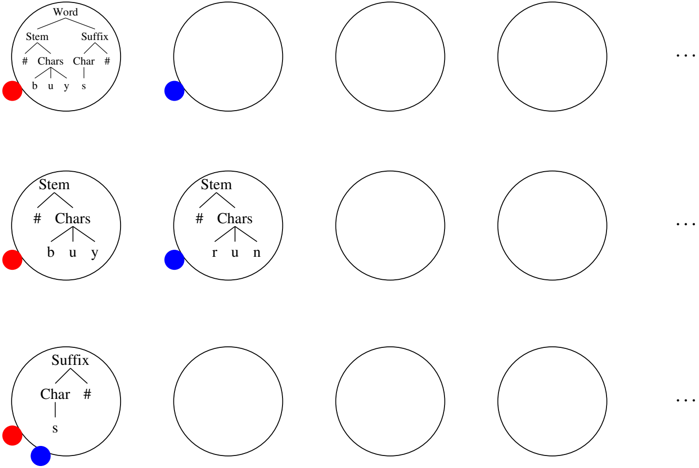


Morphology adaptor grammar (2c)

Word restaurant
Word → Stem Suffix

Stem restaurant
Stem → #
Stem → # Chars

Suffix restaurant
Suffix → #
Suffix → Chars #



Chars factory
Chars → Char
Chars → Char Chars
Char → a . . . z

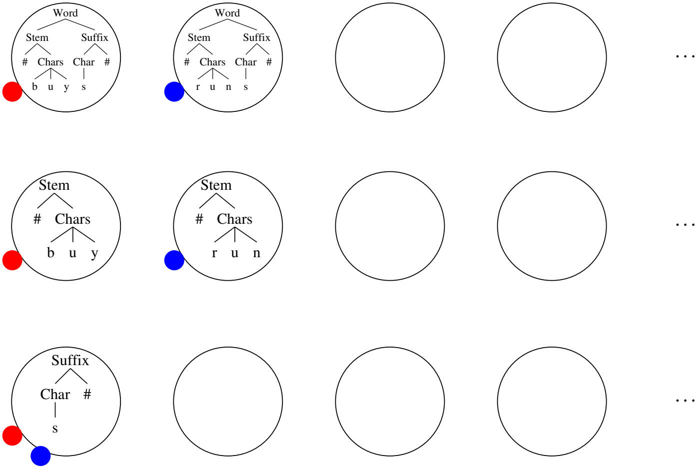
Morphology adaptor grammar (2d)

Word restaurant
Word $\rightarrow$ Stem Suffix

Stem restaurant
Stem $\rightarrow$ #
Stem $\rightarrow$ # Chars

Suffix restaurant
Suffix $\rightarrow$ #
Suffix $\rightarrow$ Chars #

Chars factory
Chars $\rightarrow$ Char
Chars $\rightarrow$ Char Chars
Char $\rightarrow$ a . . . z

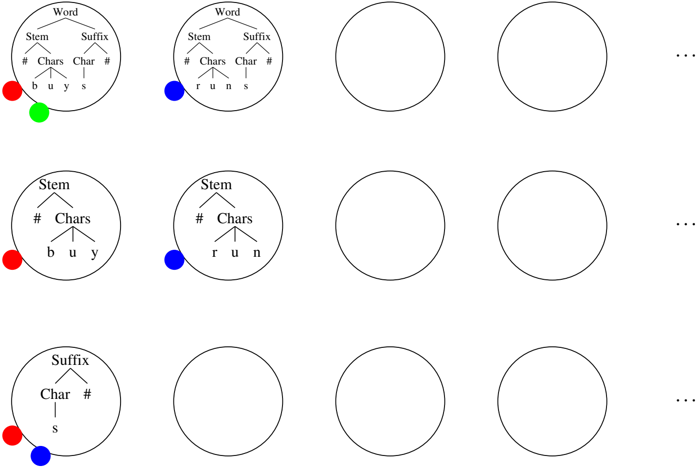


Morphology adaptor grammar (3)

Word restaurant
Word → Stem Suffix

Stem restaurant
Stem → #
Stem → # Chars

Suffix restaurant
Suffix → #
Suffix → Chars #



Chars factory
Chars → Char
Chars → Char Chars
Char → a . . . z

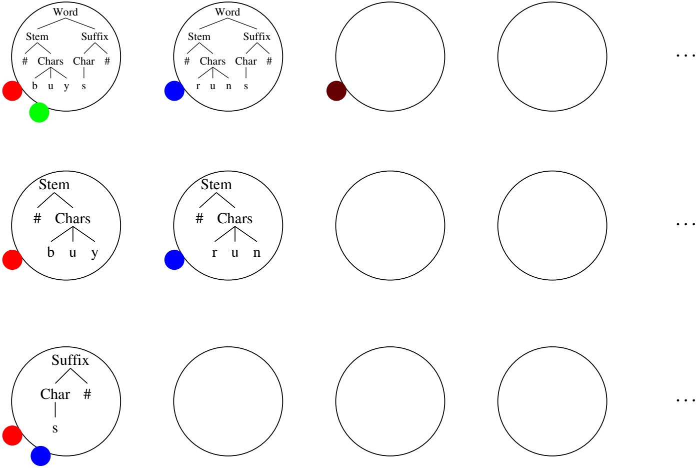
Morphology adaptor grammar (4a)

Word restaurant
Word $\longrightarrow$ Stem Suffix

Stem restaurant
Stem $\longrightarrow$ #
Stem $\longrightarrow$ # Chars

Suffix restaurant
Suffix $\longrightarrow$ #
Suffix $\longrightarrow$ Chars #

Chars factory
Chars $\longrightarrow$ Char
Chars $\longrightarrow$ Char Chars
Char $\longrightarrow$ a . . . z



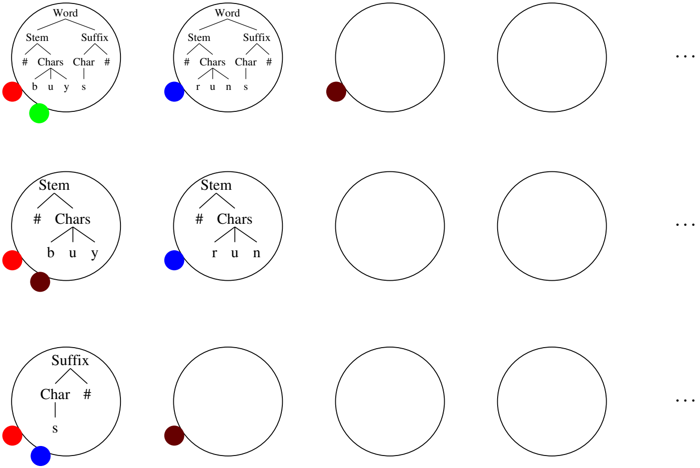
Morphology adaptor grammar (4b)

Word restaurant
Word → Stem Suffix

Stem restaurant
Stem → #
Stem → # Chars

Suffix restaurant
Suffix → #
Suffix → Chars #

Chars factory
Chars → Char
Chars → Char Chars
Char → a . . . z

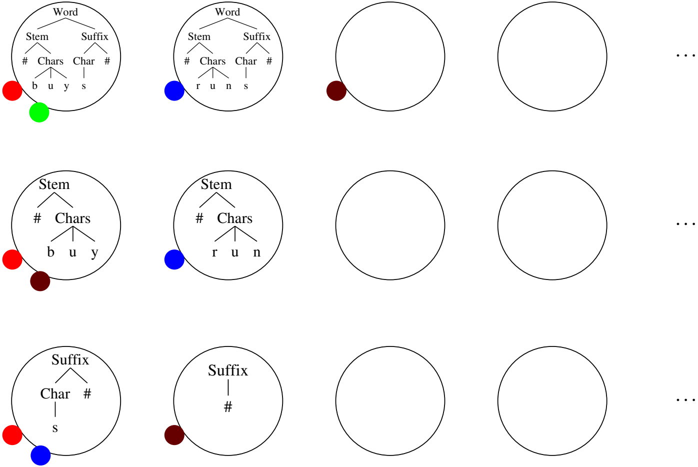


Morphology adaptor grammar (4c)

Word restaurant
Word → Stem Suffix

Stem restaurant
Stem → #
Stem → # Chars

Suffix restaurant
Suffix → #
Suffix → Chars #



Chars factory
Chars → Char
Chars → Char Chars
Char → a . . . z

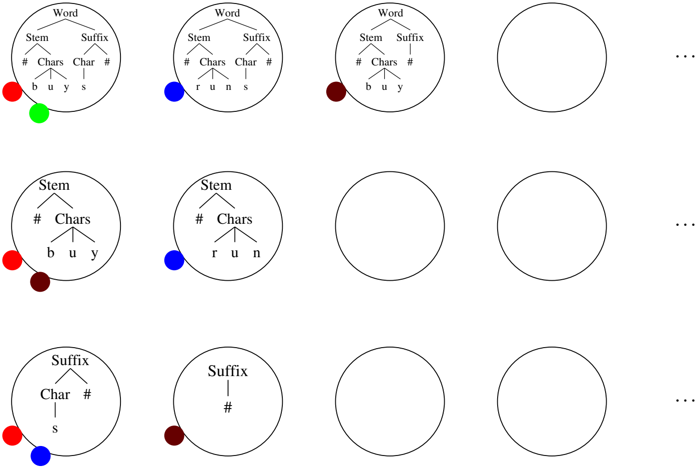
Morphology adaptor grammar (4d)

Word restaurant
Word → Stem Suffix

Stem restaurant
Stem → #
Stem → # Chars

Suffix restaurant
Suffix → #
Suffix → Chars #

Chars factory
Chars → Char
Chars → Char Chars
Char → a . . . z



From Chinese restaurants to Dirichlet processes

- Labeled Chinese restaurant processes take a base distribution P_G and return a stream of samples from a different distribution with the same support
- The Chinese restaurant process is a sequential process, generating the next item conditioned on the previous ones
- We can get a different distribution each time we run a CRP (placing customers on tables and labeling tables are random)
- Abstracting away from sequential generation, a CRP maps P_G to a *distribution over distributions* $\mathrm{DP}(\alpha, \mathrm{P}_G)$
- $\mathrm{DP}(\alpha, \mathrm{P}_G)$ is called a *Dirichlet process* with *concentration parameter* α and *base distribution* P_G
- Distributions in $\mathrm{DP}(\alpha, \mathrm{P}_G)$ are *discrete* (w.p. 1) even if the base distribution P_G is continuous

PCFGs as recursive mixtures

The distributions over strings induced by a PCFG in *Chomsky-normal form* (i.e., all productions are of the form $A \to B\,C$ or $A \to w$, where $A, B, C \in N$ and $w \in T$) is G_S where:

$$G_A = \sum_{A \to B\,C \in R_A} \theta_{A \to B\,C} G_B \bullet G_C + \sum_{A \to w \in R_A} \theta_{A \to w} \delta_w$$

$$\begin{aligned}(P \bullet Q)(z) &= \sum_{xy=z} P(x) Q(y) \\ \delta_w(x) &= 1 \text{ if } w = x \text{ and } 0 \text{ otherwise}\end{aligned}$$

In fact, $G_A(x) = \mathrm{P}(A \Rightarrow^\star x|\theta)$, the sum of the probability of all trees with root node A and yield x

Adaptor grammars

An adaptor grammar (G, θ, α) is a PCFG (G, θ) together with a parameter vector α where for each $A \in N$, α_A is the parameter of the Dirichlet process associated with A.

$$
\begin{aligned}
G_A &\sim \mathrm{DP}(\alpha_A, H_A) \text{ if } \alpha_A > 0 \\
&= H_A \text{ if } \alpha_A = 0 \\
H_A &= \sum_{A \to B\, C \in R_A} \theta_{A \to B\, C} G_B \bullet G_C + \sum_{A \to w \in R_A} \theta_{A \to w} \delta_w
\end{aligned}
$$

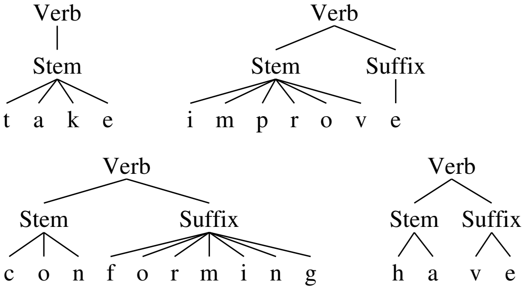
The probabilistic language defined by the grammar is G_S. There is one Dirichlet Process for each non-terminal A where $\alpha_A > 0$. Its base distribution H_A is a mixture of the language generated by the Dirichlet processes associated with other non-terminals.

Estimating adaptor grammars

- Need to estimate:
  - table labels and customer count for each table
  - (optional) probabilities of productions labeling tables
- Component-wise Metropolis-Hastings sampler
  - ith component is the parse tree for input string i
  - re-parse input i using grammar estimated from parses for other inputs
- Sampling directly from conditional distribution of parses seems intractable
  - construct PCFG approximation on the fly
  - each table label corresponds to a production in PCFG approximation

Verbal morphology

Verb → Stem
Verb → Stem Suffix
Stem → Chars
Suffix → Chars
Chars → Char
Chars → Char Chars
Char → a . . . z

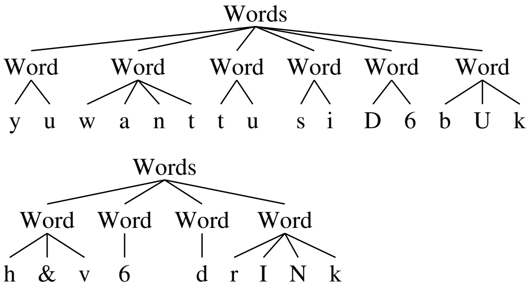


- Restaurants for Verb, Stem and Suffix
- Given orthographic verb tokens from WSJ as input, 70% tokens, 66% types correctly segmented; many errors linguistically plausible
- Extends naturally to:
  - hidden word classes
  - agglutinative languages (with little phonology)

Unigram model of word segmentation

- Unigram model: each word is generated independently
- Input is *unsegmented broad phonemic transcription* (Brent)
  Example: y u w a n t t u s i D 6 b u k
- CRP for Word non-terminal caches previously seen words

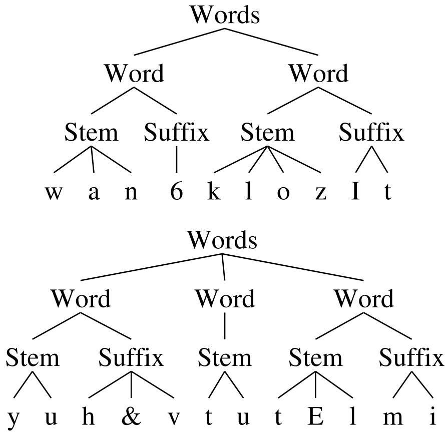
Words → Word
Words → Word Words
Word → Chars
Chars → Char
Chars → Char Chars
Char → a | … | z



- Unigram word segmentation on Brent corpus: 54% token f-score, 59% type f-score

Morphology and word segmentation

Words $\rightarrow$ Word
Words $\rightarrow$ Word Words
Word $\rightarrow$ Stem Suffix
Word $\rightarrow$ Stem
Stem $\rightarrow$ Chars
Suffix $\rightarrow$ Chars
Chars $\rightarrow$ Char
Chars $\rightarrow$ Char Chars
Char $\rightarrow$ a$|\ldots|$z

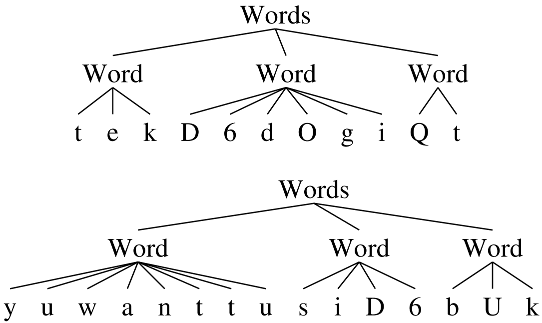


- CRPs for Word, Stem and Suffix terminals
- Doesn't do a good job of learning morphology

Outline

Unigram model often finds collocations

- Unigram word segmentation model assumes each word is generated independently
- But there are strong inter-word dependencies (collocations)
- Unigram model can only capture such dependencies by analyzing collocations as words



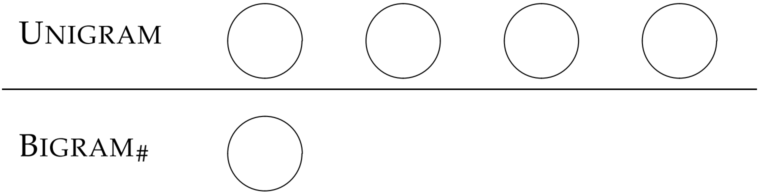
Hierarchical CRP bigram word segmentation

- Bigram model: predict next word based on preceding word
- Each word w has a CRP $\textsc{Bigram}_w$ to predict following word
  - *Set of words is unknown, so CRPs constructed on the fly*
- Each $\textsc{Bigram}_w$ CRP shares same labeling distribution $\textsc{Unigram}$
  - $\textsc{Unigram}$ is a CRP that generates a common vocabulary for all bigrams
  - $\textsc{Bigram}_w$ "backs off" to $\textsc{Unigram}$ very much like Kneser-Ney smoothing

$$
\begin{array}{rcll}
w_i & \mid & w_{i-1}, \textsc{Bigram}_{w_{i-1}} & \sim & \textsc{Bigram}_{w_{i-1}} \\
\textsc{Bigram}_{w_{i-1}} & \mid & \alpha_2, \textsc{Unigram} & \sim & \mathrm{DP}(\alpha_2, \textsc{Unigram}) \\
\textsc{Unigram} & \mid & \alpha_1, \mathrm{P}_0 & \sim & \mathrm{DP}(\alpha_1, \mathrm{P}_0)
\end{array}
$$

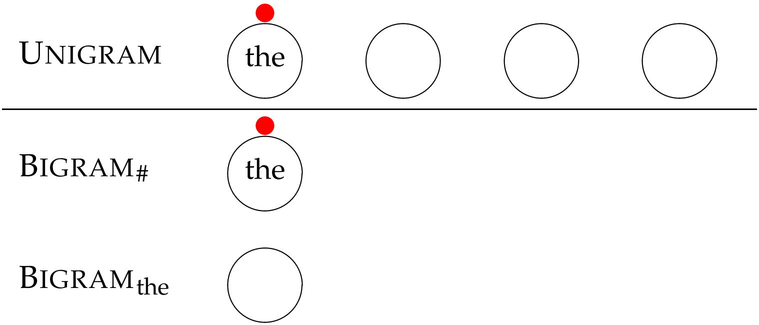
Bigram word segmentation model (0)

the book the book



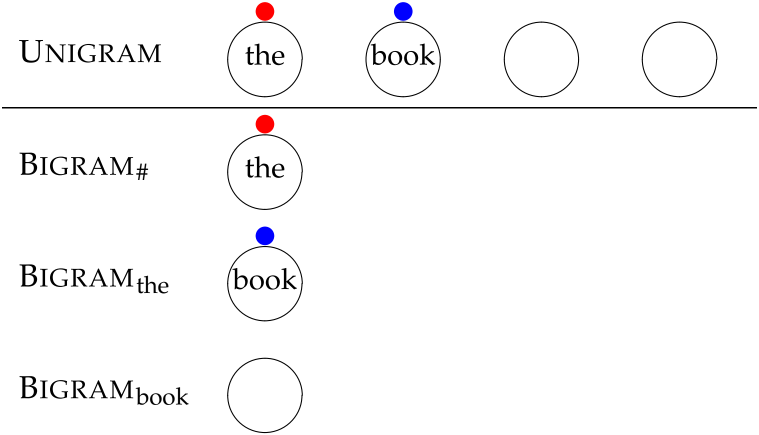
Bigram word segmentation model (1)

the book the book



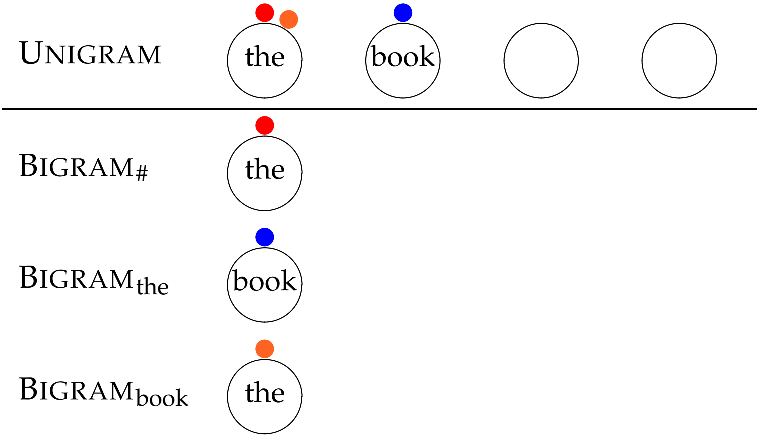
Bigram word segmentation model (2)

the book the book

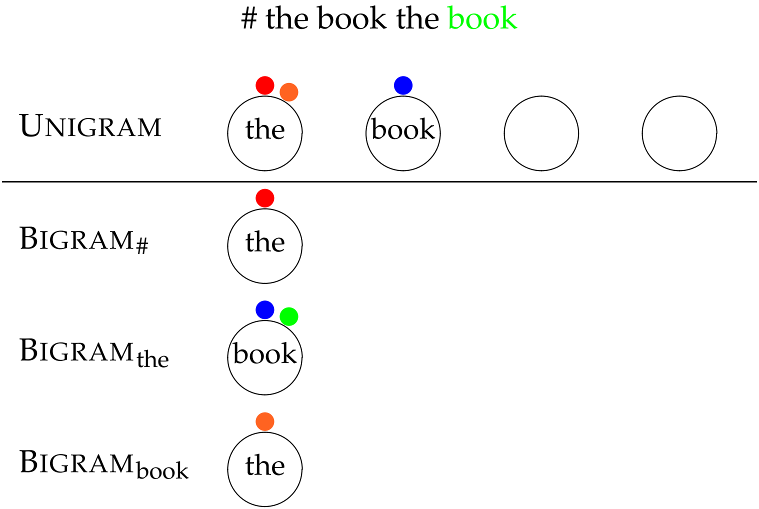


Bigram word segmentation model (3)

the book the book



Bigram word segmentation model (4)



Bigram segmentation model

- Implemented using Gibbs sampling
  - ith component is "word boundary at position i"
  - sampling amounts to possibly splitting a word at position i or joining the two words abutting at position i
- Performs significantly better than unigram model
  - bigram: 77% token f-score, 63% type f-score
  - unigram: 54% token f-score, 59% type f-score
- Number of CRPs is number of words, which is not known in advance

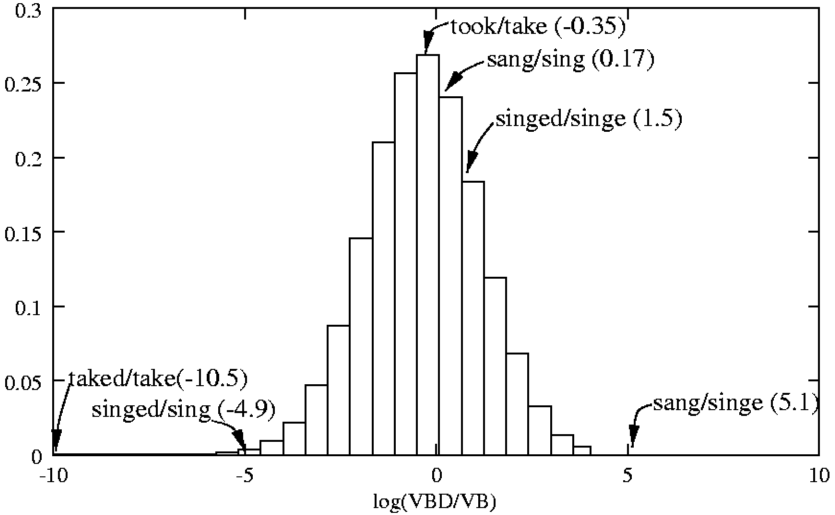
$\Rightarrow$ cannot be formulated as an adaptor grammar (which have a CRP per non-terminal)

Outline

Conclusion and future work

- Chinese restaurant process models can:
  - interpolate between types and tokens (morphology)
  - *learn the basic units of generalization* (morphology, word segmentation)
- Adaptor grammars can express many (but not all) hierarchical CRP models
  - General-purpose inference algorithm
  - Can model other tasks (e.g., hierarchical clustering)
- *Still a work in progress*
  - Is there a generalization of adaptor grammars that includes the bigram word segmentation model?
  - Difficult to select priors to get desired behavior
  - Gibbs sampler seems slow to converge with complex grammars and large data sets

Morpheme frequencies are useful



Yarowsky and Wicentowski (2000) "Minimally supervised Morphological Analysis by Multimodal Alignment"