Learning rules with Adaptor Grammars

Mark Johnson

joint work with Sharon Goldwater and Tom Griffiths

May, 2008

The drunk under the 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.*"

Ideas behind talk

- Most successful statistical learning methods are *parametric*
  - PCFGs have one probability parameter per rule
  - PCFG learning: given rules and data, learn rule probabilities
- Non-parametric learning: learn parameters (rules) as well as values
- *Adaptor grammars*:
  - are a framework for specifying hierarchical nonparametric Bayesian models
  - can express a variety of linguistically-interesting structures
  - are approximated by PCFGs, where number of rules depends on data

Outline

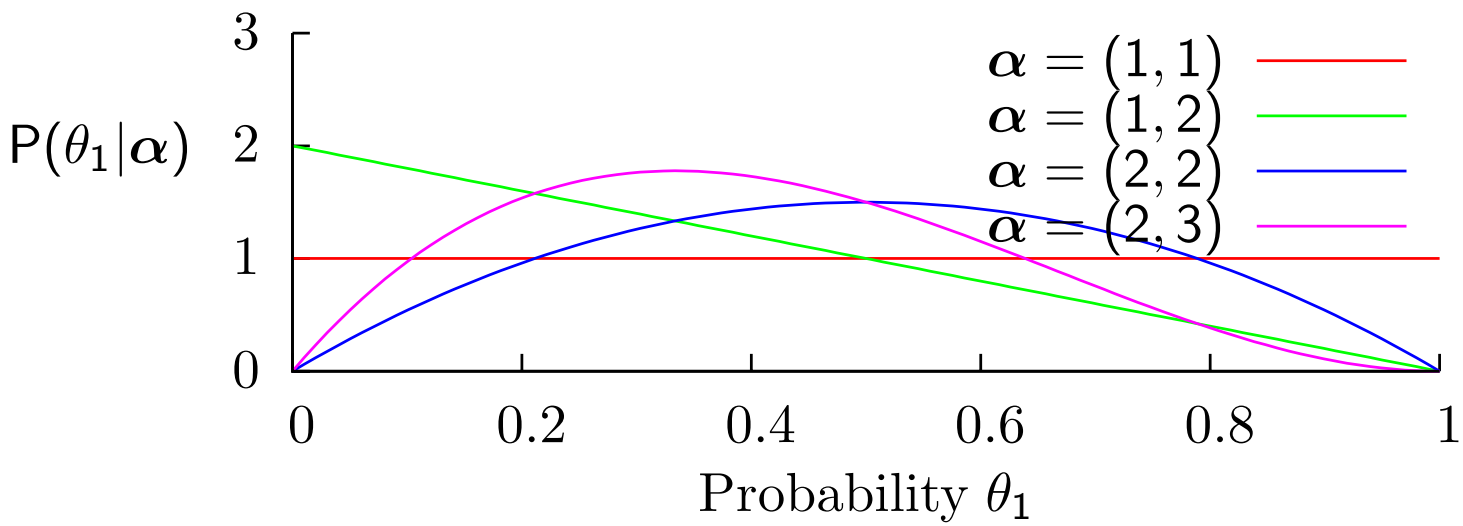
Language acquisition as Bayesian inference

$$\underbrace{\mathsf{P}(\mathsf{Grammar} \mid \mathsf{Data})}_{\text{Posterior}} \propto \underbrace{\mathsf{P}(\mathsf{Data} \mid \mathsf{Grammar})}_{\text{Likelihood}} \underbrace{\mathsf{P}(\mathsf{Grammar})}_{\text{Prior}}$$

- Likelihood measures how well grammar describes data
- Prior expresses knowledge of grammar before data is seen
  - can be very specific (e.g., Universal Grammar)
  - can be very general (e.g., prefer shorter grammars)
- Posterior is *distribution* over grammars
  - expresses uncertainty about which grammar is correct
- But: *infinitely many* grammars may be consistent with Data

Multinomial distributions and Dirichlet priors

- X follows a *multinomial distribution* if X ranges over $1, \ldots, m$ and $\mathsf{P}(X = k|\boldsymbol{\theta}) = \theta_k$
- Bayesian inference for $\boldsymbol{\theta}$: $\mathsf{P}(\boldsymbol{\theta}|X) \propto \mathsf{P}(X|\boldsymbol{\theta})\mathsf{P}(\boldsymbol{\theta})$
- A *Dirichlet distribution* $\mathsf{P}(\boldsymbol{\theta}|\boldsymbol{\alpha})$ is a probability distribution over multinomial parameters $\boldsymbol{\theta}$
  - One Dirichlet parameter α_k for each outcome k
- Dirichlets are *conjugate prior* for multinomials
  - If prior is Dirichlet $(\alpha_1, \ldots, \alpha_k, \ldots, \alpha_m)$ and we observe data $X = k$, posterior is Dirichlet $(\alpha_1, \ldots, \alpha_k + 1, \ldots, \alpha_m)$



Conditional Dirichlet-multinomial distribution

- Suppose we don't know $\boldsymbol{\theta}$, but only know Dirichlet prior $\boldsymbol{\alpha}$:

$$\mathsf{P}(X = k \mid \boldsymbol{\alpha}) \;=\; \int \mathsf{P}(X = k|\boldsymbol{\theta})\, \mathsf{P}(\boldsymbol{\theta}|\boldsymbol{\alpha})\, d\boldsymbol{\theta} \;\propto\; \alpha_k$$

- Suppose we observe $\boldsymbol{X} = (x_1, \ldots, x_n)$. What is probability of next outcome X_{n+1}?

$$\mathsf{P}(X_{n+1} = k \mid \boldsymbol{\alpha}, \boldsymbol{X}) \;\propto\; \alpha_k + n_k(\boldsymbol{X})$$

  where $n_k(\boldsymbol{X})$ is number of times k appears in $\boldsymbol{X}$
- Example: coin with sides h, t. Prior $\alpha_h = \alpha_t = 1$. $\boldsymbol{X} = (t, h, t)$. Posterior $\alpha'_h = 2, \alpha'_t = 3$, so $\mathsf{P}(X_4 = h|\boldsymbol{\alpha}, \boldsymbol{X} = (t, h, t)) = 2/5$

Predicting a sequence of outcomes

- Given Dirichlet prior $\boldsymbol{\alpha}$ and observations $\boldsymbol{X}$, what is probability of next *two* outcomes X_{n+1} and X_{n+2}?

$$\begin{aligned}
&\mathsf{P}(X_{n+1} = k, X_{n+2} = k' \mid \boldsymbol{\alpha}, \boldsymbol{X}) \\
&\quad = \quad \mathsf{P}(X_{n+1} = k \mid \boldsymbol{\alpha}, \boldsymbol{X})\, \mathsf{P}(X_{n+2} = k' \mid \boldsymbol{\alpha}, \boldsymbol{X}, X_{n+1} = k) \\
&\quad \propto \quad (\alpha_k + n_k(\boldsymbol{X}))\, (\alpha_{k'} + n_{k'}(\boldsymbol{X}) + \delta_{k,k'})
\end{aligned}$$

  i.e., the probability that $X_{n+2} = k'$ is affected by value of X_{n+1}

$\Rightarrow$ *The probability of each outcome type changes as we progress through a sequence of outcomes*

Outline

Probabilistic context-free grammars

- Rules in *Context-Free Grammars* (CFGs) expand nonterminals into sequences of terminals and nonterminals
- A *Probabilistic CFG* (PCFG) associates each nonterminal with a multinomial distribution over the rules that expand it
- Probability of a tree is the *product of the probabilities of the rules* used to construct it

Rule r	θ_r	Rule r	θ_r
S $\rightarrow$ NP VP	1.0		
NP $\rightarrow$ Sam	0.75	NP $\rightarrow$ Sandy	0.25
VP $\rightarrow$ barks	0.6	VP $\rightarrow$ snores	0.4

$$P\left(\begin{array}{ccc} & \text{S} & \\ \text{NP} & & \text{VP} \\ | & & | \\ \text{Sam} & & \text{barks} \end{array}\right) = 0.45 \qquad P\left(\begin{array}{ccc} & \text{S} & \\ \text{NP} & & \text{VP} \\ | & & | \\ \text{Sandy} & & \text{snores} \end{array}\right) = 0.1$$

Dirichlet priors on rule probabilities

- Prior $\mathsf{P}(\boldsymbol{\theta})$ specifies probability of rule probabilities $\boldsymbol{\theta}$
- Conjugate prior is a *product of Dirichlets* (one for each nonterminal)
- Prior has a parameter $\alpha_{A\rightarrow\beta}$ for each rule $A \rightarrow \beta$
- Suppose we're given prior $\boldsymbol{\alpha}$ and observe parse trees parse trees $\boldsymbol{T} = (T_1, \ldots, T_n)$. What is posterior $\mathsf{P}(\boldsymbol{\theta} \mid \boldsymbol{\alpha}, \boldsymbol{T})$?
- $\mathsf{P}(\boldsymbol{\theta}|\boldsymbol{\alpha}, \boldsymbol{T})$ is a product of Dirichlets with parameters $\boldsymbol{\alpha}'$, where:

$$\alpha'_{A\rightarrow\beta} \quad = \quad \alpha_{A\rightarrow\beta} + n_{A\rightarrow\beta}(\boldsymbol{T})$$

and $n_{A\rightarrow\beta}(\boldsymbol{T})$ is number of times $A \rightarrow \beta$ in $\boldsymbol{T}$

In summary: $\boldsymbol{\alpha}' \;=\; \boldsymbol{\alpha} + \boldsymbol{n}(\boldsymbol{T})$

$\Rightarrow$ Bayesian estimation is easy given visible data (parse trees) and Dirichlet prior

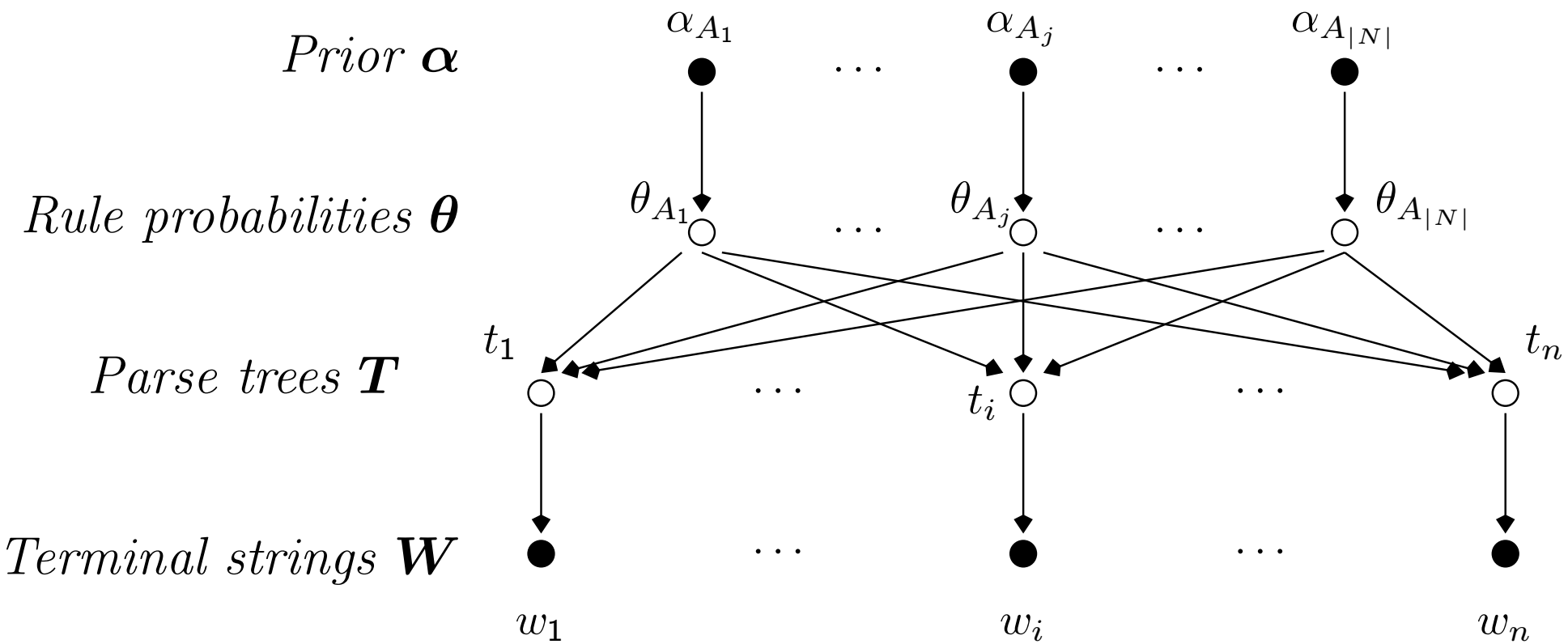
Estimating θ from strings via Gibbs sampling

- Learning from terminal strings $\boldsymbol{W} \Rightarrow$ parse trees $\boldsymbol{T}$ are *hidden*
- No known closed form for posterior $\mathsf{P}(\boldsymbol{\theta}|\boldsymbol{\alpha}, \boldsymbol{W})$, but we can approximate it by sampling
- *Gibbs sampling* is a Markov Chain Monte Carlo (MCMC) method for sampling from $\mathsf{P}(X_1, \ldots, X_n)$

  initialize $x_1, \ldots, x_n$ somehow
  repeat forever:
    for i in $1, .., n$:
      set x_i to a sample from $\mathsf{P}(X_i|\boldsymbol{X}_{-i} = \boldsymbol{x}_{-i})$

  where $\boldsymbol{X}_{-i} = (X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_n)$.
  Under very general conditions, after "burn-in" this sequence of samples has distribution $\mathsf{P}(X_1, \ldots, X_n)$
- Example: to produce samples from $\mathsf{P}(X, Y)$, guess x, y, sample $X^{(1)}$ from $\mathsf{P}(X|Y^{(0)})$, then sample $Y^{(1)}$ from $\mathsf{P}(Y|X)$, then X from $\mathsf{P}(X|Y)$, then Y from $\mathsf{P}(Y|X)$, ...

Bayes net representation of Bayesian PCFG



$$\begin{array}{rcll} \boldsymbol{\theta}_A & | & \boldsymbol{\alpha}_A & \sim \quad \mathsf{Dir}(\boldsymbol{\alpha}_A) \qquad (\boldsymbol{\theta}_A \text{ are prob. of rules expanding } A) \\ T_i & | & \boldsymbol{\theta} & \sim \quad G_S(\boldsymbol{\theta}) \qquad (T_i \text{ is parse tree for string } W_i) \\ W_i & | & T_i & = \quad \textsc{Yield}(T_i) \end{array}$$

A Gibbs sampler for trees and rule probabilities

- Generative model:

$$\mathsf{P}(\boldsymbol{W}, \boldsymbol{T}, \boldsymbol{\theta} \mid \boldsymbol{\alpha}) \;=\; \mathsf{P}(\boldsymbol{W} \mid \boldsymbol{T})\, \mathsf{P}(\boldsymbol{T} \mid \boldsymbol{\theta})\, \mathsf{P}(\boldsymbol{\theta} \mid \boldsymbol{\alpha})$$

- Dirichlet prior parameters $\boldsymbol{\alpha}$ and strings $\boldsymbol{W}$ are given
- Rule probabilities $\boldsymbol{\theta}$ and parse trees $\boldsymbol{T}$ are unknown
- Goal: estimate $\mathsf{P}(\boldsymbol{T}, \boldsymbol{\theta} \mid \boldsymbol{\alpha}, \boldsymbol{W})$
- Gibbs sampler iteration:
  - sample $\boldsymbol{\theta}$ from $\mathsf{P}(\boldsymbol{\theta} \mid \boldsymbol{T}, \boldsymbol{\alpha})$
    - i.e., sampling multinomials from Dirichlets
  - sample $\boldsymbol{T}$ from $\mathsf{P}(\boldsymbol{T} \mid \boldsymbol{W}, \boldsymbol{\theta})$
    - sampling parse tree T_i for string W_i given $\boldsymbol{\theta}$
    - can be done in parallel for each string W_i
    - $O(|W_i|^3)$ dynamic programming algorithm

Integrating out the rule probabilities

- Posterior distribution $\mathsf{P}(\boldsymbol{\theta} \mid \boldsymbol{T}, \boldsymbol{\alpha})$ is Dirichlet with parameters $\boldsymbol{\alpha}' = \boldsymbol{\alpha} + \boldsymbol{n}(\boldsymbol{T})$

$\Rightarrow$ Can calculate conditional probability of next parse tree T_n given previous parses $\boldsymbol{T}_{-n} = (T_1, \ldots, T_{n-1})$ and $\boldsymbol{\alpha}$

$$\begin{aligned} \mathsf{P}(T_n \mid \boldsymbol{T}_{-n}, \boldsymbol{\alpha}) &= \int \mathsf{P}(T_n \mid \boldsymbol{\theta})\, \mathsf{P}(\boldsymbol{\theta} \mid \boldsymbol{T}_{-n}, \boldsymbol{\alpha})\, d\boldsymbol{\theta} \\ &= \mathsf{P}(T_n \mid \boldsymbol{\alpha}') \quad \text{where } \boldsymbol{\alpha}' = \boldsymbol{\alpha} + \boldsymbol{n}(\boldsymbol{T}_{-n}) \end{aligned}$$

$\Rightarrow$ Gibbs sampler with parse trees T_i as the n components

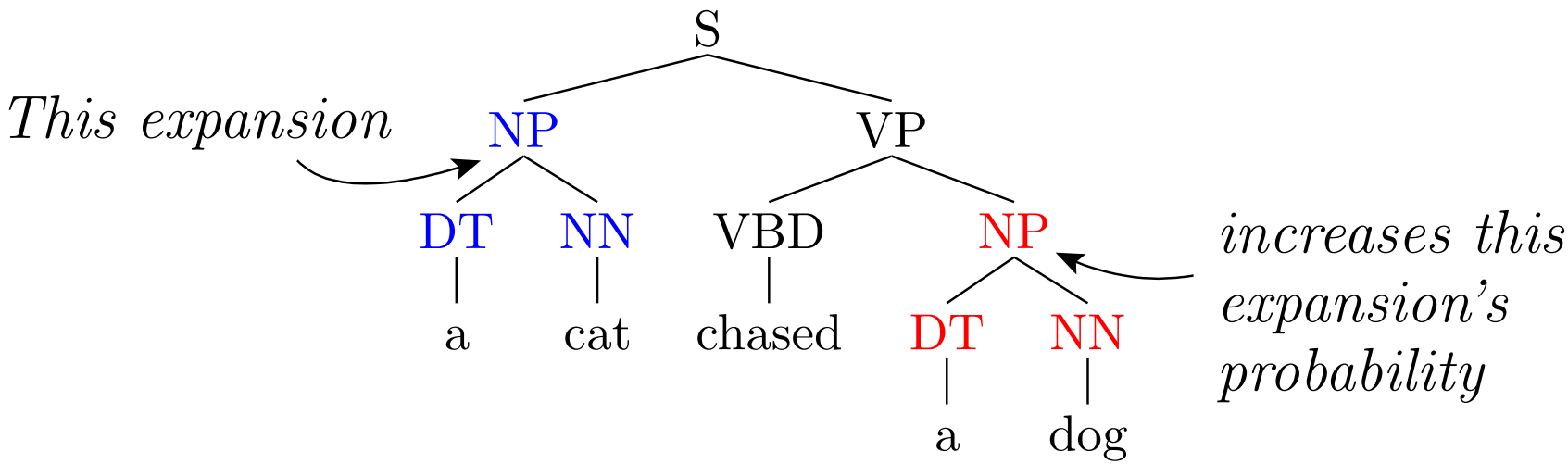
- requires sampling T_i from $\mathsf{P}(T_i \mid W_i, \boldsymbol{T}_{-i}, \boldsymbol{\alpha})$
- but calculating normalizing constant in

$$\mathsf{P}(T_i \mid W_i, \boldsymbol{T}_{-i}, \boldsymbol{\alpha}) = \frac{\mathsf{P}(T_i \mid \boldsymbol{T}_{-i}, \boldsymbol{\alpha})}{\sum_{T:\textsc{Yield}(T)=W_i} \mathsf{P}(T \mid \boldsymbol{T}_{-i}, \boldsymbol{\alpha})}$$

is hard!

Conditional parse probability in collapsed model

- Dirichlet prior on $\boldsymbol{\theta}$
  $\Rightarrow$ probability of each rule expansion $A \rightarrow \beta \propto \alpha_{A \rightarrow \beta}$
- Parse tree is a *sequence* of rule expansions
  $\Rightarrow$ Update Dirichlet prior after each rule expansion
  $\Rightarrow$ A's expansion probabilities *change* each time A expands
  $\Rightarrow$ A rule's expansion "primes" its subsequent use



- A rule's probability depends on which rules were used earlier
  $\Rightarrow$ *no dynamic programming algorithms*
  $\Rightarrow$ no efficient way of calculating normalizing constant

A Metropolis-within-Gibbs sampler for trees

- Metropolis "accept/reject" sampling procedures only require "unnormalized" probabilities
- But they require a *proposal distribution* that is
  - easy to sample from, and
  - not too different from true distribution
- Proposal distribution for $\mathsf{P}(T_i \mid \boldsymbol{\alpha}, \boldsymbol{T}_{-i}, W_i)$ is PCFG with rule probabilities

$$\theta'_{A\to\beta} \quad \propto \quad \alpha_{A\to\beta} + n_{A\to\beta}(\boldsymbol{T}_{-i})$$

- Metropolis-within-Gibbs sampling algorithm:

repeat forever:
  for i in $1, .., n$:
    compute proposal PCFG prob. $\boldsymbol{\theta}'$ from $\boldsymbol{\alpha} + n(\boldsymbol{T}_{-i})$
    sample parse tree $T^\star$ from $\mathsf{P}(T|W_i, \boldsymbol{\theta}')$
    compute "true" probability $\mathsf{P}(T^\star|\boldsymbol{\alpha}, \boldsymbol{T}_{-i})$
    Metropolis accept/reject $T^\star$ as new value for T_i

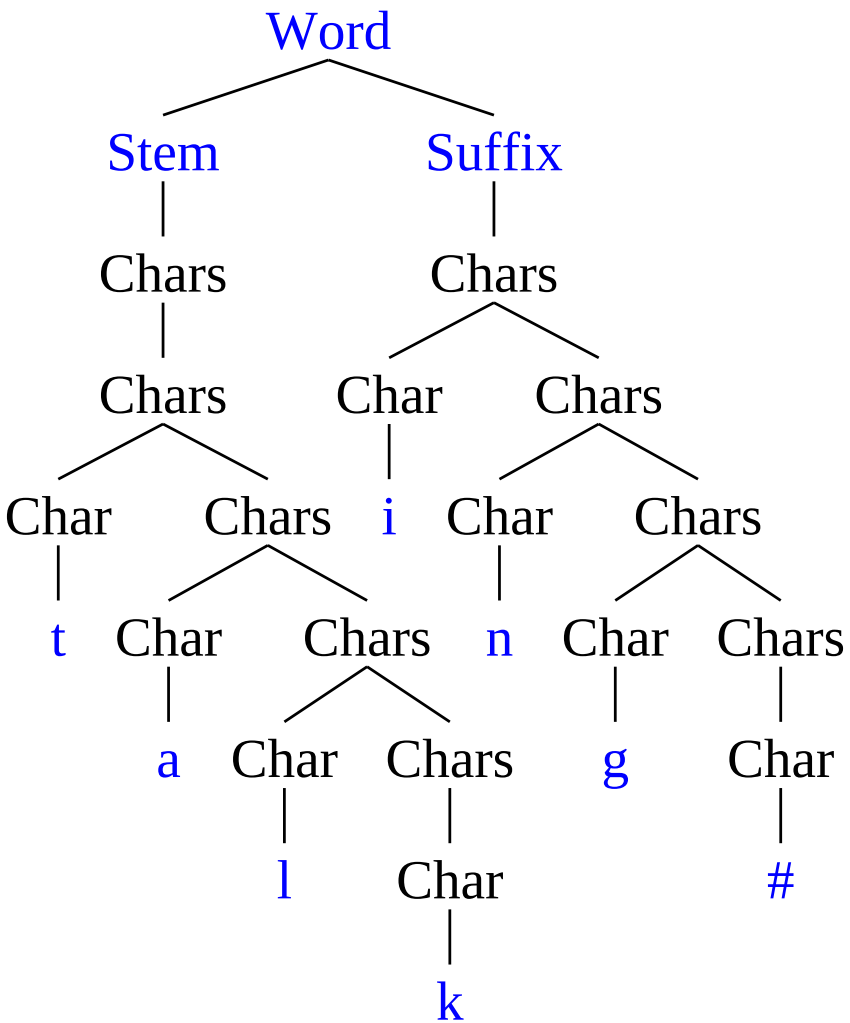
Outline

Learning syntactic structure is hard

- Bayesian PCFG estimation works well on toy data
- Results are disappointing on "real" data
  - wrong rules?
  - rules need to be given a priori; can we learn them too?
- Strategy: study simpler cases
  - Morphological segmentation (e.g., *walking* = *walk*+*ing*)
  - Word segmentation of unsegmented utterances

A CFG for stem-suffix morphology

$$
\begin{aligned}
\text{Word} &\rightarrow \text{Stem Suffix} \\
\text{Stem} &\rightarrow \text{Chars} \\
\text{Suffix} &\rightarrow \text{Chars}
\end{aligned}
\qquad
\begin{aligned}
\text{Chars} &\rightarrow \text{Char} \\
\text{Chars} &\rightarrow \text{Char Chars} \\
\text{Char} &\rightarrow \text{a} \mid \text{b} \mid \text{c} \mid \ldots
\end{aligned}
$$



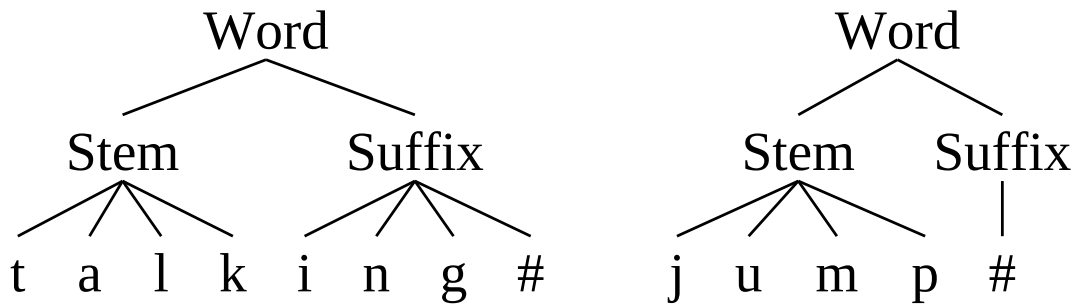
- Grammar's trees can represent any segmentation of words into stems and suffixes

$\Rightarrow$ Can represent true segmentation

- But grammar's *units of generalization (PCFG rules) are "too small" to learn morphemes*

A "CFG" with one rule per possible morpheme

$$
\begin{aligned}
\mathsf{Word} &\rightarrow \mathsf{Stem\ Suffix} \\
\mathsf{Stem} &\rightarrow \text{all possible stems} \\
\mathsf{Suffix} &\rightarrow \text{all possible suffixes}
\end{aligned}
$$



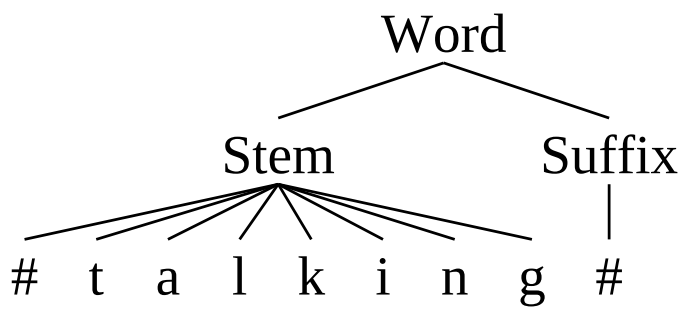
- A rule for each morpheme
  $\Rightarrow$ "PCFG" can represent probability of each morpheme
- *Unbounded number of possible rules, so this is not a PCFG*
  - not a practical problem, as only a finite set of rules could possibly be used in any particular data set

Maximum likelihood estimate for $\boldsymbol{\theta}$ is trivial

- Maximum likelihood selects $\boldsymbol{\theta}$ that minimizes KL-divergence between model and training data $\boldsymbol{W}$ distributions
- *Saturated model* in which each word is generated by its own rule replicates training data distribution $\boldsymbol{W}$ exactly

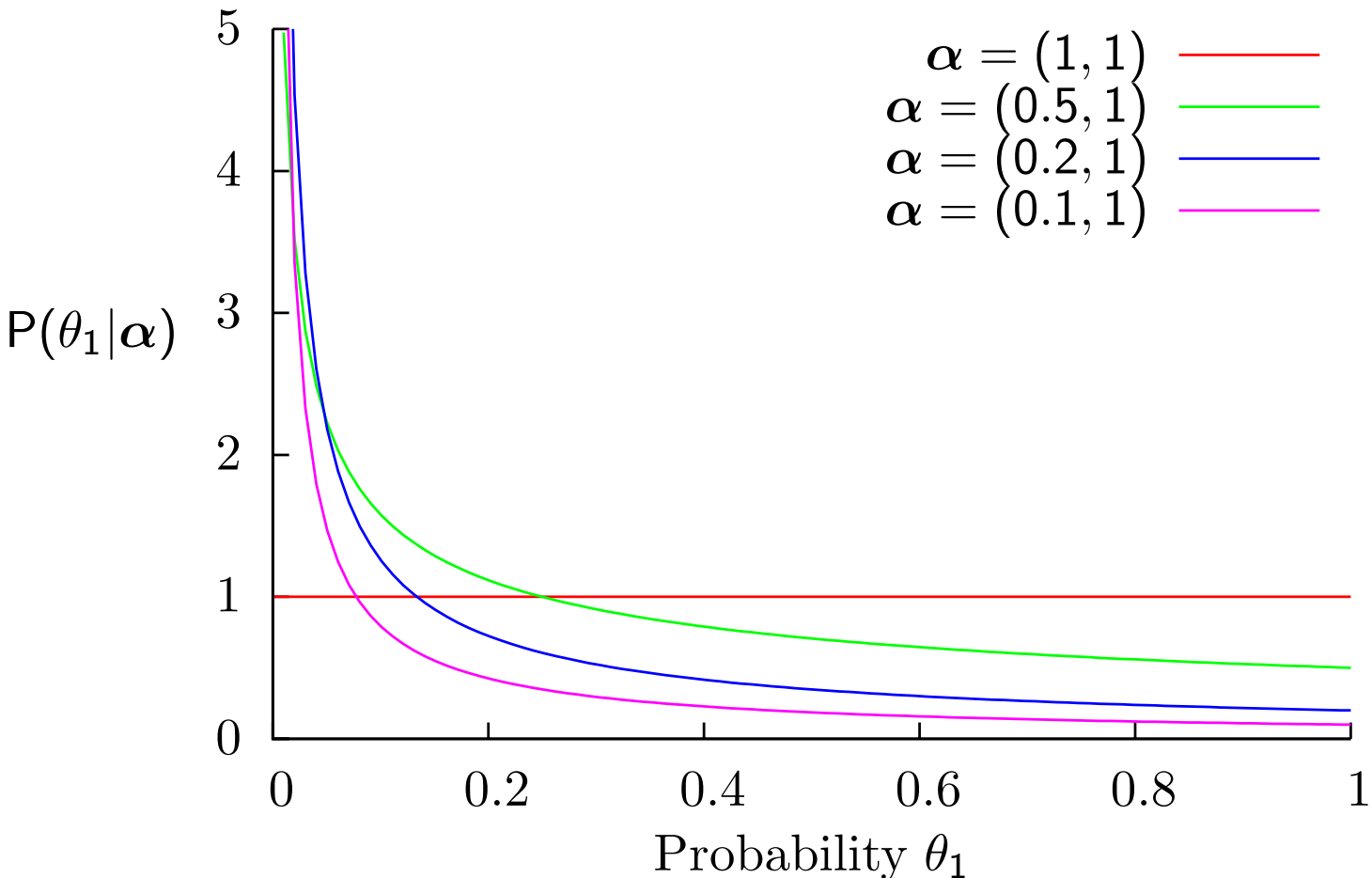
$\Rightarrow$ Saturated model is maximum likelihood estimate

- Maximum likelihood estimate does not find any suffixes



Forcing generalization via sparse Dirichlet priors

- Idea: use Bayesian prior that prefers fewer rules
- Set of rules is fixed a priori in Bayesian PCFGs, but can "turn rule off" by setting $\theta_{A\rightarrow\beta} \approx 0$
- Dirichlet prior with $\alpha_{A\rightarrow\beta} \approx 0$ prefers $\theta_{A\rightarrow\beta} \approx 0$



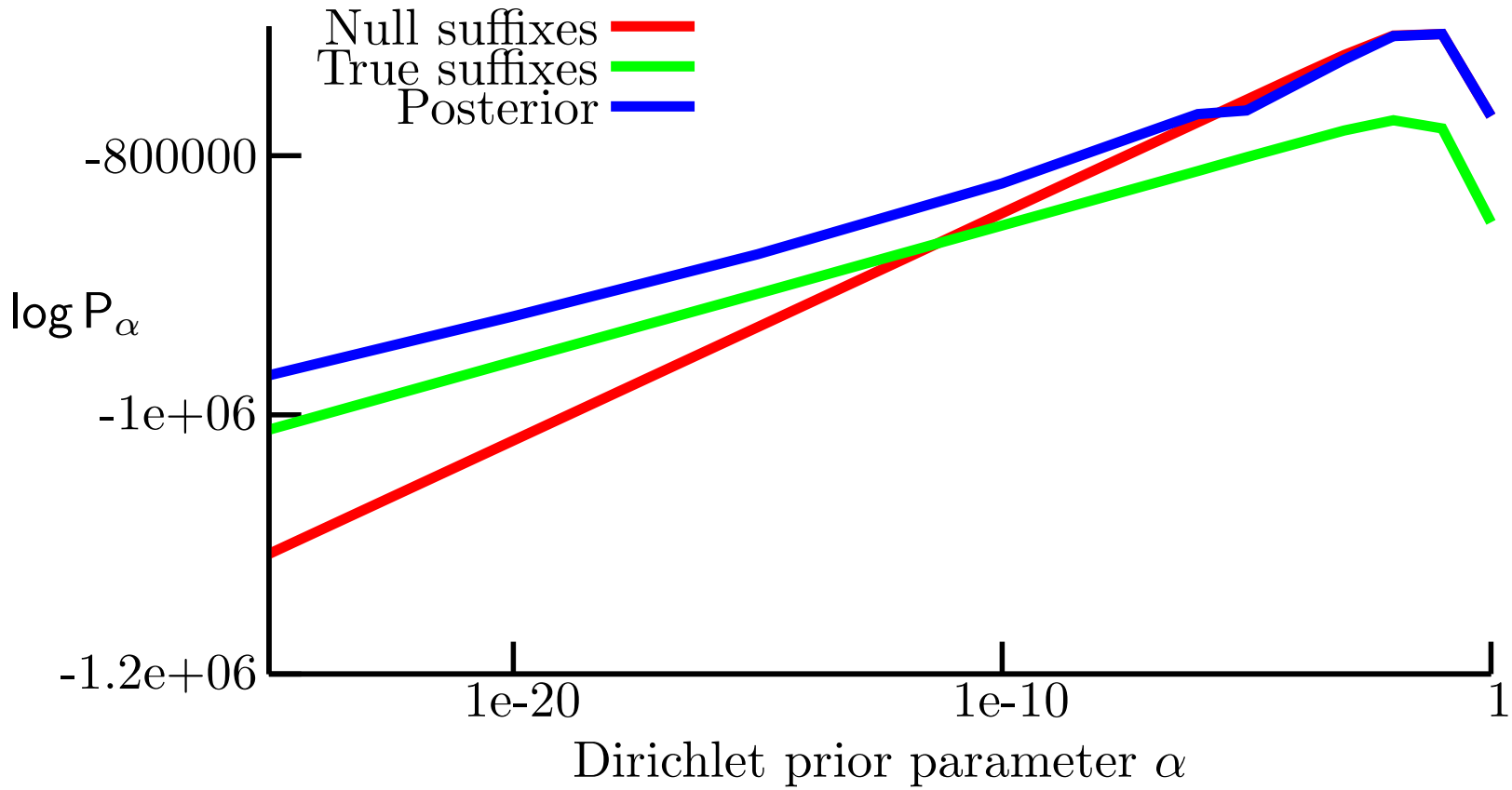
Morphological segmentation experiment

- Trained on orthographic verbs from U Penn. Wall Street Journal treebank
- Uniform Dirichlet prior prefers sparse solutions as $\alpha \rightarrow 0$
- Metropolis-within-Gibbs sampler used to sample from posterior distribution of parses
  - reanalyses each word based on a grammar estimated from the parses of the 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 assumptions in PCFGs

- Context-free grammars are "context-free" because the possible expansions of each node do not depend on expansions of other nodes
- Probabilistic CFGs extend this by requiring each node expansion to be *statistically independent* (conditioned on the node's label)
- This is a very strong assumption, which is often false!
- Morphology grammar contains rule:

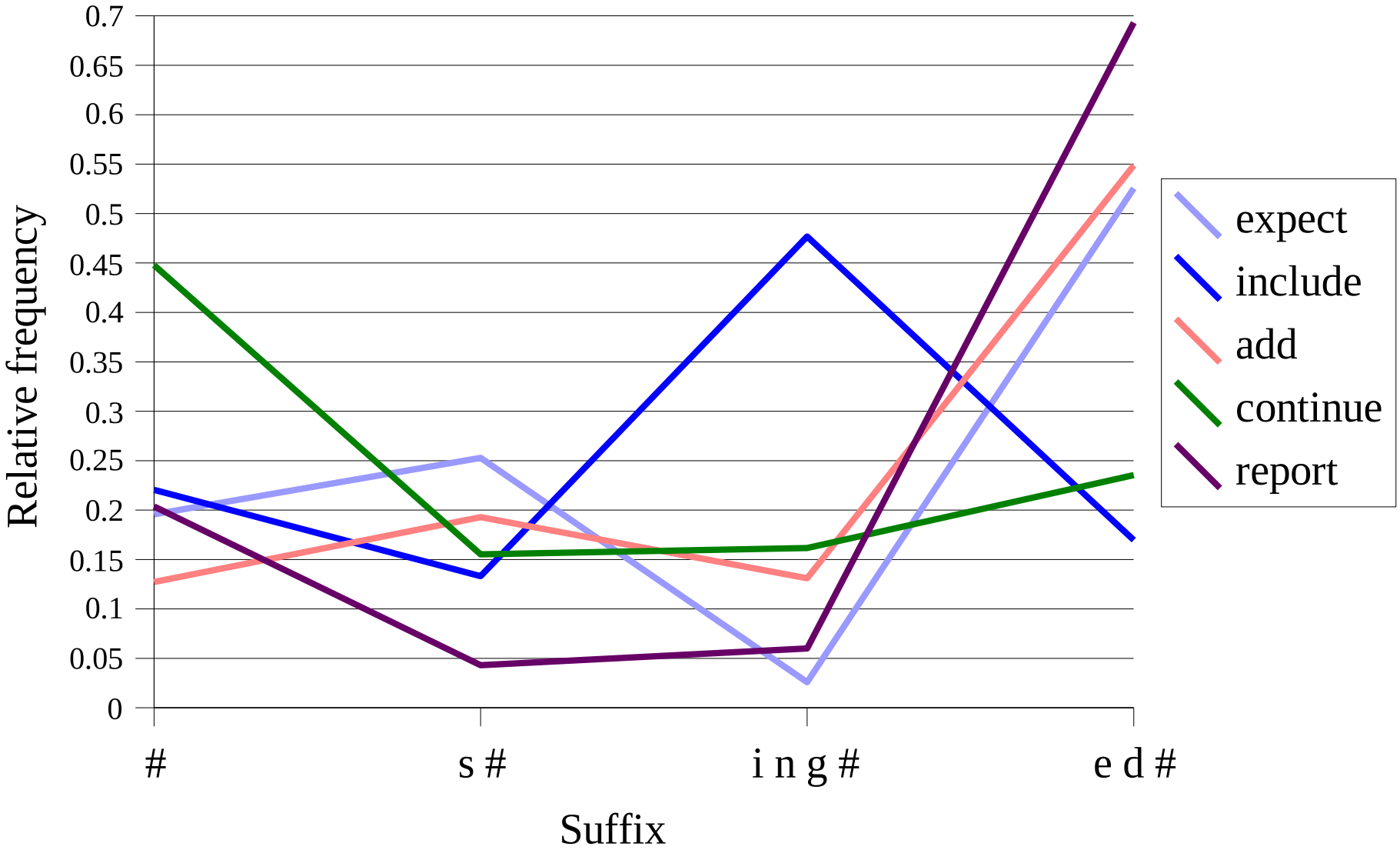
$$\mathsf{Word} \rightarrow \mathsf{Stem\ Suffix}$$

- Corresponding independence assumption:

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

causes PCFG model of morphology to fail

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

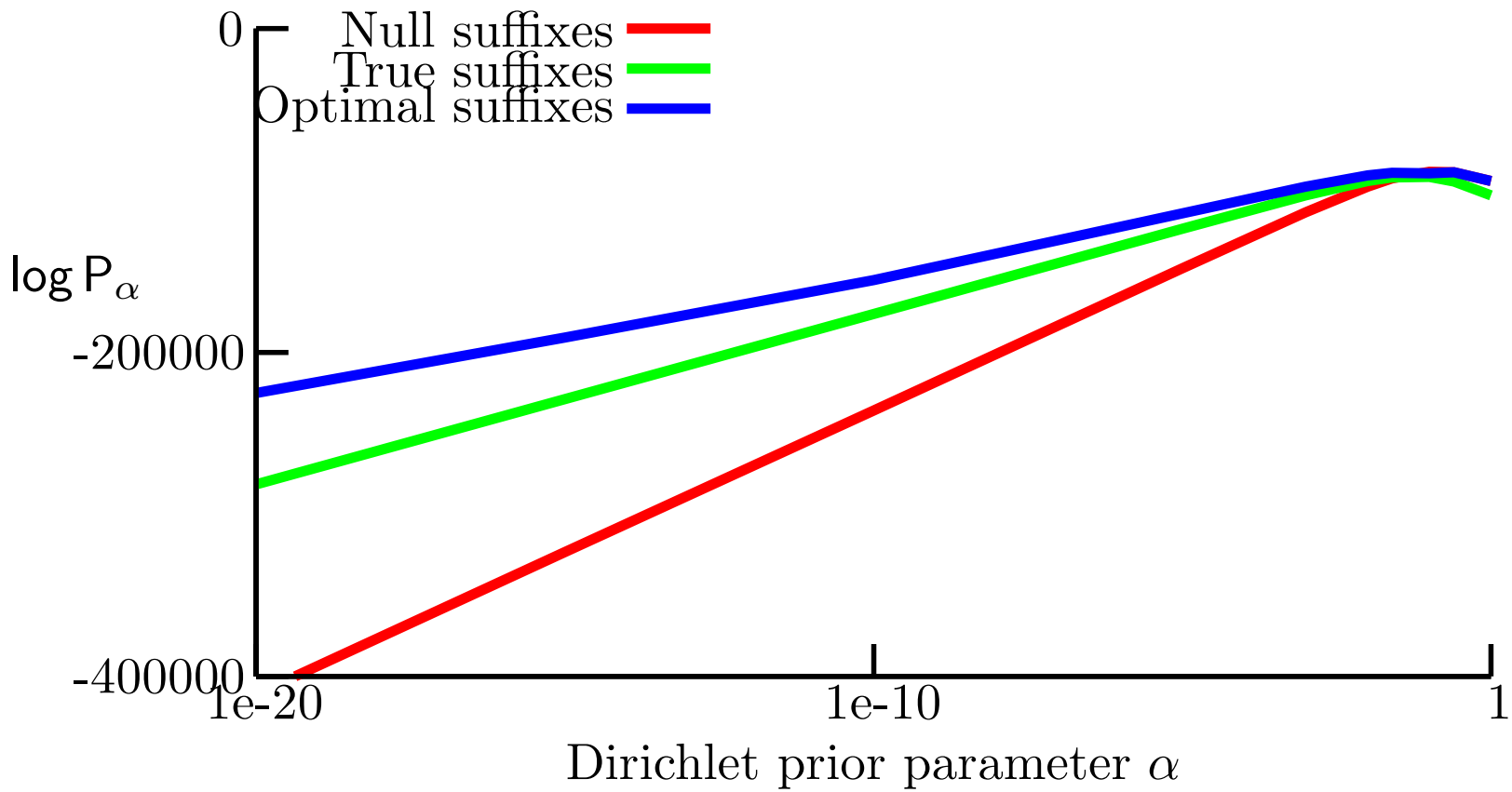
$$\begin{aligned} \mathsf{Data} &= \text{“the cat chased the other cat”} \\ \mathsf{Tokens} &= \text{“the”, “cat”, “chased”, “the”, “other”, “cat”} \\ \mathsf{Types} &= \text{“the”, “cat”, “chased”, “other”} \end{aligned}$$

- Estimating θ from *word types* rather than word tokens eliminates (most) frequency variation
  - 4 common verb suffixes, so when estimating from verb types $\theta_{\mathsf{Suffix} \rightarrow \mathsf{ing\#}} \approx 0.25$
- Several psycholinguists believe that humans learn morphology from word types
- Goldwater et al investigated a morphology-learning model that learnt from an interpolation of types and tokens

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 $\alpha = 10^{-3}$

Desiderata for an extension of PCFGs

- PCFG rules are "too small" to be effective units of generalization
  $\Rightarrow$ generalize over groups of rules
  $\Rightarrow$ units of generalization should be chosen based on data
- Type-based inference mitigates non-context-free dependencies
  $\Rightarrow$ Hierarchical Bayesian model where:
  - context-free rules generate types
  - another process replicates types to produce tokens
- *Adaptor grammars:*
  - learn probability of entire subtrees (how a nonterminal expands to terminals)
  - use grammatical hierarchy to define a Bayesian hierarchy, from which type-based inference emerges

Outline

Dirichlet-Multinomials with many outcomes

- Dirichlet prior $\boldsymbol{\alpha}$, observed data $\boldsymbol{X} = (X_1, \ldots, X_n)$

$$\mathsf{P}(X_{n+1} = k \mid \boldsymbol{X}, \boldsymbol{\alpha}) \quad \propto \quad \alpha_k + n_k(\boldsymbol{X})$$

- Consider a sequence of Dirichlet-multinomials where:
  - total Dirichlet pseudocount is fixed $\alpha_\bullet = \sum_{k=1}^{m} \alpha_k$, and
  - prior uniform over outcomes $1, \ldots, m$, so $\alpha_k = \alpha_\bullet / m$
  - number of outcomes $m \to \infty$

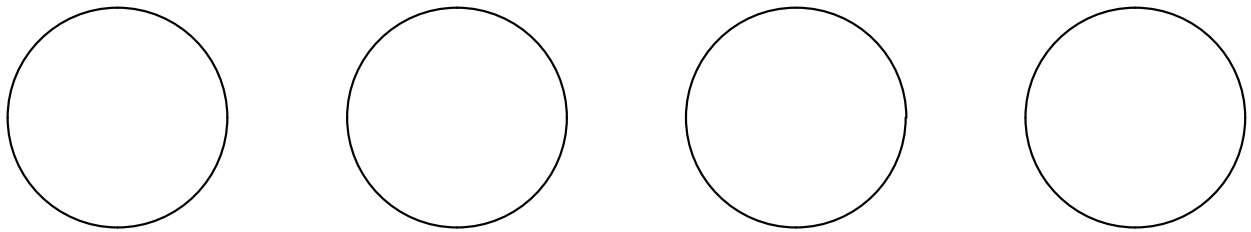
$$\mathsf{P}(X_{n+1} = k \mid \boldsymbol{X}, \alpha_\bullet) \quad \propto \quad \begin{cases} n_k(\boldsymbol{X}) & \text{if } n_k(\boldsymbol{X}) > 0 \\ \\ \alpha_\bullet / m & \text{if } n_k(\boldsymbol{X}) = 0 \end{cases}$$

But when $m \gg n$, most k are unoccupied (i.e., $n_k(\boldsymbol{X}) = 0$)

$\Rightarrow$ *Probability of a previously seen outcome* $k \quad \propto n_k(\boldsymbol{X})$

Probability of an outcome never seen before $\propto \alpha_\bullet$

The Chinese restaurant process (1)

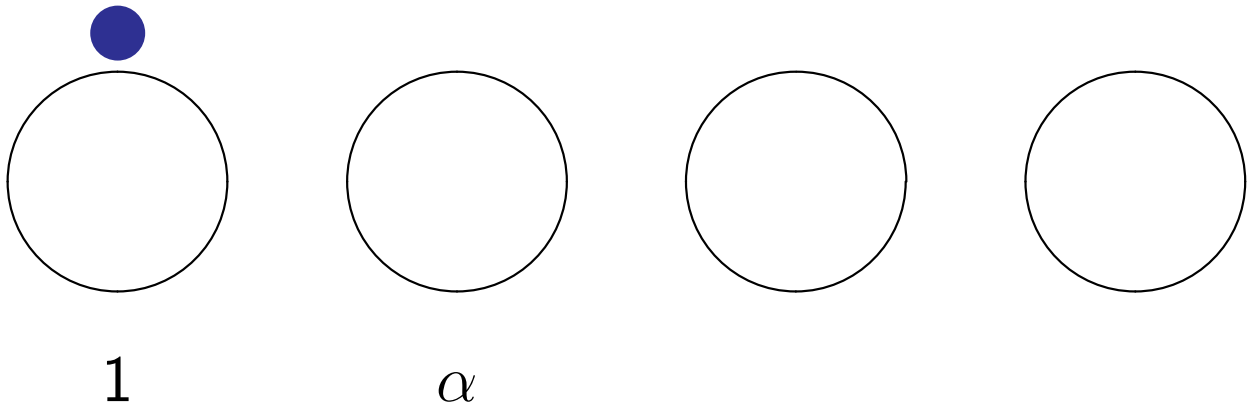


α

- Each "table" seats infinite number of "customers" (samples)
- Customer $n+1$ enters with tables $1, \ldots, m$ occupied:
  - sits at old table $k \leq m$ with probability $\propto n_k$
  - sits at new table $k = m+1$ with probability $\propto \alpha$

$\Rightarrow$ "Rich get richer" power-law dynamics

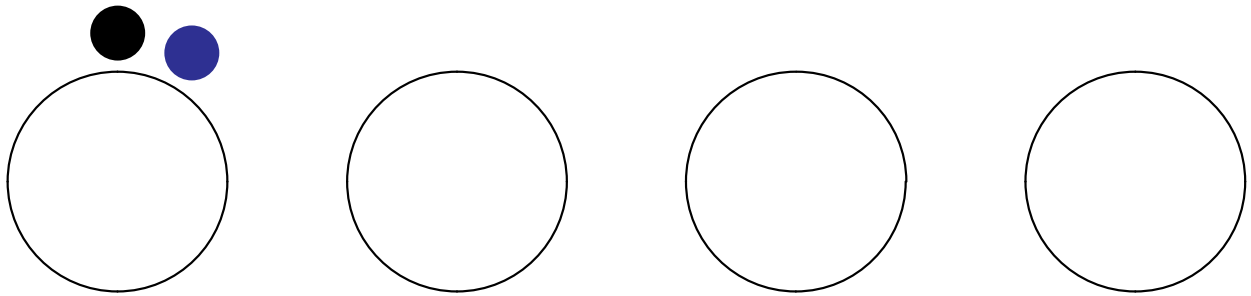
The Chinese restaurant process (2)



- Each "table" seats infinite number of "customers" (samples)
- Customer $n+1$ enters with tables $1, \ldots, m$ occupied:
  - sits at old table $k \leq m$ with probability $\propto n_k$
  - sits at new table $k = m+1$ with probability $\propto \alpha$

$\Rightarrow$ "Rich get richer" power-law dynamics

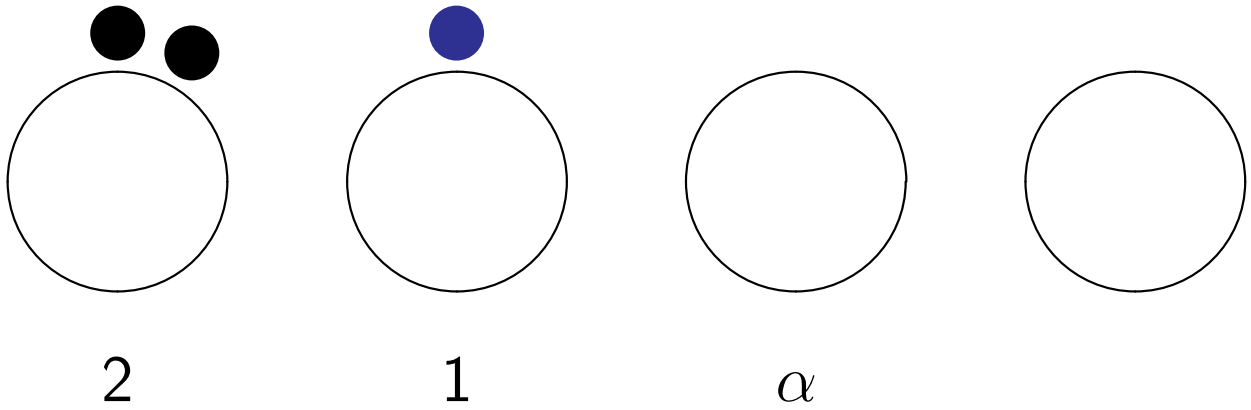
The Chinese restaurant process (3)



- Each "table" seats infinite number of "customers" (samples)
- Customer $n+1$ enters with tables $1, \ldots, m$ occupied:
  - sits at old table $k \leq m$ with probability $\propto n_k$
  - sits at new table $k = m+1$ with probability $\propto \alpha$

$\Rightarrow$ "Rich get richer" power-law dynamics

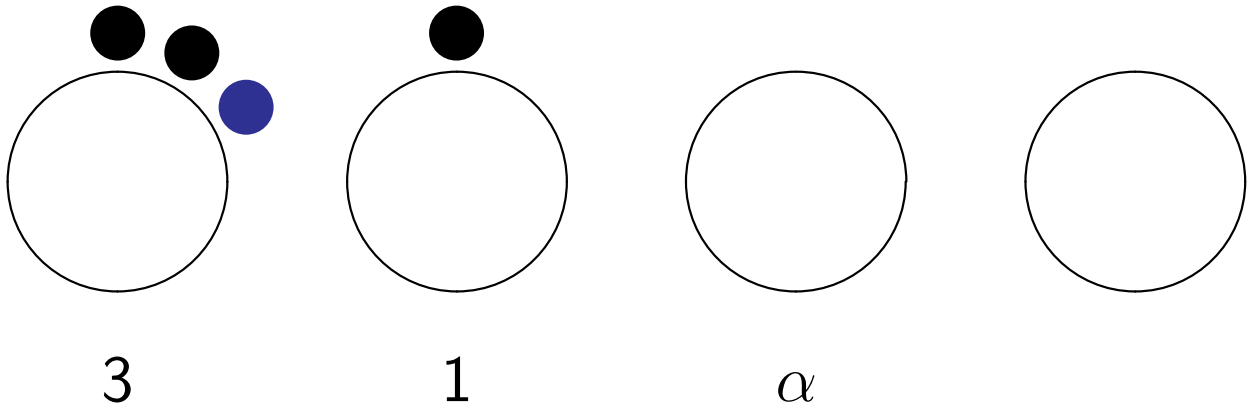
The Chinese restaurant process (4)



- Each "table" seats infinite number of "customers" (samples)
- Customer $n+1$ enters with tables $1, \ldots, m$ occupied:
  - sits at old table $k \leq m$ with probability $\propto n_k$
  - sits at new table $k = m+1$ with probability $\propto \alpha$

$\Rightarrow$ "Rich get richer" power-law dynamics

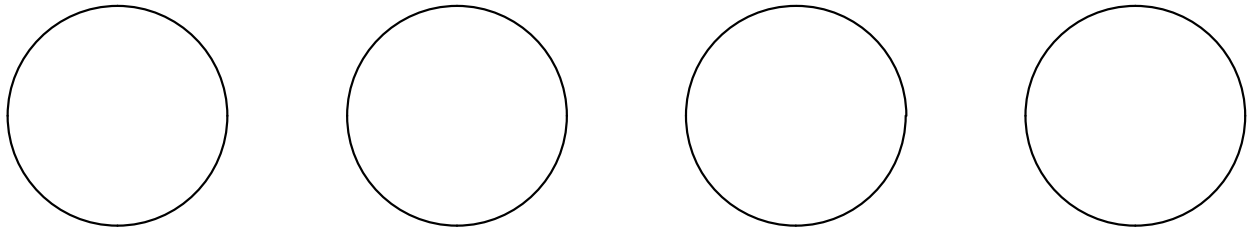
The Chinese restaurant process (5)



- Each "table" seats infinite number of "customers" (samples)
- Customer $n+1$ enters with tables $1, \ldots, m$ occupied:
  - sits at old table $k \leq m$ with probability $\propto n_k$
  - sits at new table $k = m+1$ with probability $\propto \alpha$

$\Rightarrow$ "Rich get richer" power-law dynamics

Labeled Chinese restaurant processes (1a)



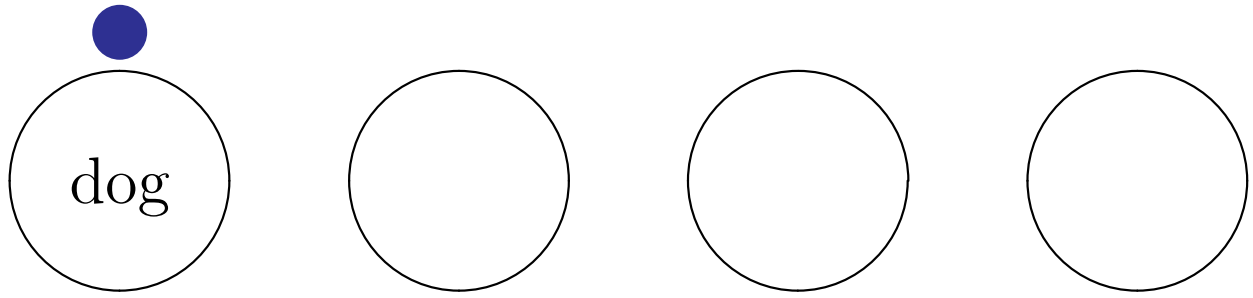
α

Generated sequence:

- Each occupied table has a label (a "dish")
- The labels Y are sampled from $\mathsf{P}_G(Y)$
- Customer $n+1$ enters with tables $1, \ldots, m$ occupied:
  - sits at old table $k \leq m$ with probability $\propto\ n_k$
  - sits at new table $k = m+1$ with probability $\propto\ \alpha$
- Emit label (dish) on table
  - if table doesn't have a label, generate one from $\mathsf{P}_G(Y)$

$\Rightarrow$ only pay probability cost for label *once per table*

Labeled Chinese restaurant processes (1b)



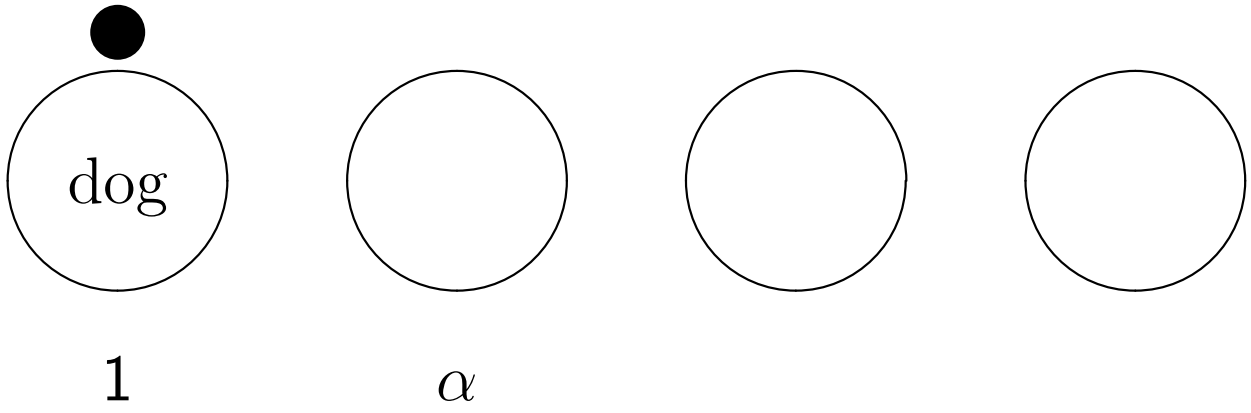
$\mathsf{P}_G(\text{dog})$

Generated sequence: dog

- Each occupied table has a label (a "dish")
- The labels Y are sampled from $\mathsf{P}_G(Y)$
- Customer $n+1$ enters with tables $1, \ldots, m$ occupied:
  - sits at old table $k \leq m$ with probability $\propto\ n_k$
  - sits at new table $k = m+1$ with probability $\propto\ \alpha$
- Emit label (dish) on table
  - if table doesn't have a label, generate one from $\mathsf{P}_G(Y)$

$\Rightarrow$ only pay probability cost for label *once per table*

Labeled Chinese restaurant processes (2a)

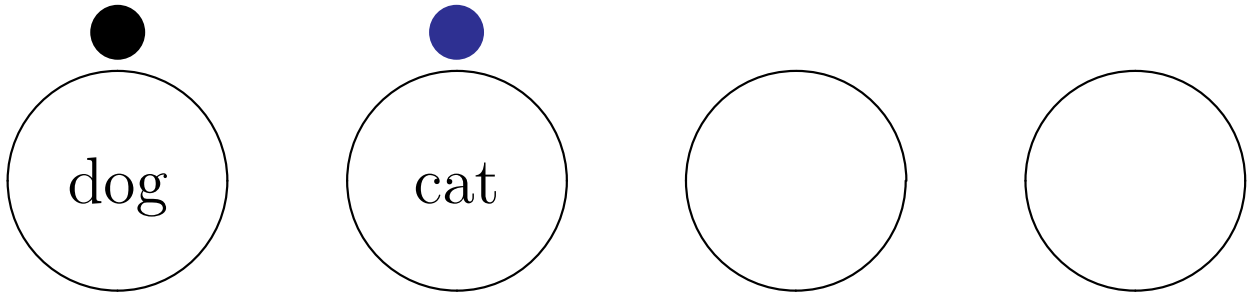


Generated sequence: dog

- Each occupied table has a label (a "dish")
- The labels Y are sampled from $\mathsf{P}_G(Y)$
- Customer $n+1$ enters with tables $1, \ldots, m$ occupied:
  - sits at old table $k \leq m$ with probability $\propto n_k$
  - sits at new table $k = m+1$ with probability $\propto \alpha$
- Emit label (dish) on table
  - if table doesn't have a label, generate one from $\mathsf{P}_G(Y)$

$\Rightarrow$ only pay probability cost for label *once per table*

Labeled Chinese restaurant processes (2b)



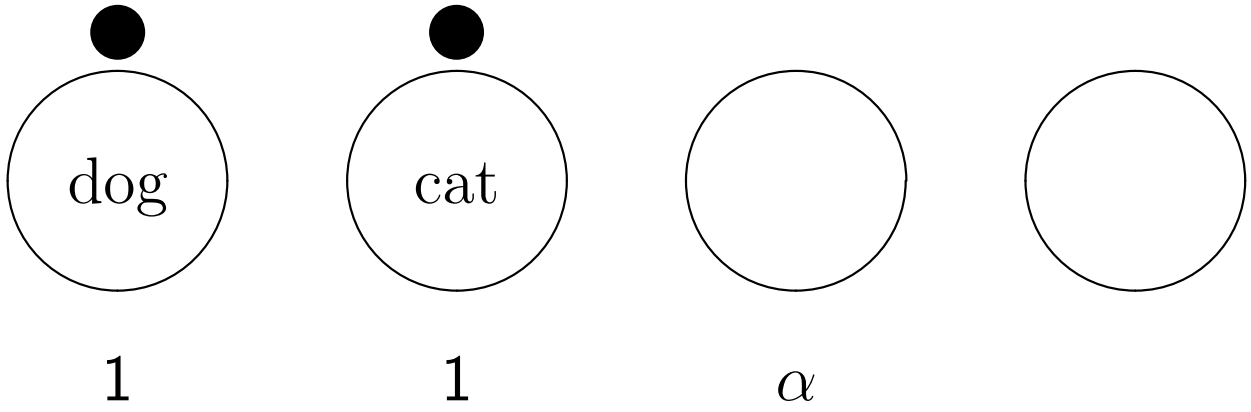
$\mathsf{P}_G(\text{cat})$

Generated sequence: dog, cat

- Each occupied table has a label (a "dish")
- The labels Y are sampled from $\mathsf{P}_G(Y)$
- Customer $n+1$ enters with tables $1, \ldots, m$ occupied:
  - sits at old table $k \leq m$ with probability $\propto\ n_k$
  - sits at new table $k = m+1$ with probability $\propto\ \alpha$
- Emit label (dish) on table
  - if table doesn't have a label, generate one from $\mathsf{P}_G(Y)$

$\Rightarrow$ only pay probability cost for label *once per table*

Labeled Chinese restaurant processes (3a)

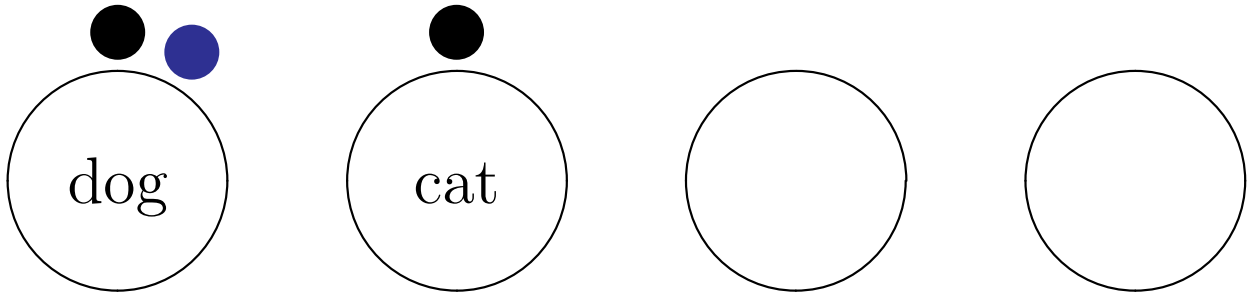


Generated sequence: dog, cat

- Each occupied table has a label (a "dish")
- The labels Y are sampled from $\mathsf{P}_G(Y)$
- Customer $n+1$ enters with tables $1, \ldots, m$ occupied:
  - sits at old table $k \leq m$ with probability $\propto\ n_k$
  - sits at new table $k = m+1$ with probability $\propto\ \alpha$
- Emit label (dish) on table
  - if table doesn't have a label, generate one from $\mathsf{P}_G(Y)$

$\Rightarrow$ only pay probability cost for label *once per table*

Labeled Chinese restaurant processes (3b)



Generated sequence: dog, cat, dog

- Each occupied table has a label (a "dish")
- The labels Y are sampled from $\mathsf{P}_G(Y)$
- Customer $n+1$ enters with tables $1, \ldots, m$ occupied:
  - sits at old table $k \leq m$ with probability $\propto\ n_k$
  - sits at new table $k = m+1$ with probability $\propto\ \alpha$
- Emit label (dish) on table
  - if table doesn't have a label, generate one from $\mathsf{P}_G(Y)$

$\Rightarrow$ only pay probability cost for label *once per table*

From Chinese restaurants to Dirichlet processes

- Chinese restaurant processes map a distribution P_G to a stream of samples from a different distribution with the same support
- CRPs specify the *conditional distribution* of the next outcome given the previous ones
- Each CRP run can produce a different distribution over labels
- It defines a mapping from α and P_G to a *distribution over distributions* $\mathsf{DP}(\alpha, \mathsf{P}_G)$
- $\mathsf{DP}(\alpha, \mathsf{P}_G)$ is called a *Dirichlet process* (DP) with *concentration parameter* α and *base distribution* P_G
- The base distribution P_G can itself be a DP $\Rightarrow$ *hierarchy* of DPs

Nonparametric extensions of PCFGs

- Chinese restaurant processes are a nonparametric extension of Dirichlet-multinomials because the number of states (occupied tables) depends on the data
- Two obvious nonparametric extensions of PCFGs:
  - let the number of nonterminals grow unboundedly
    - refine the nonterminals of an original grammar
      e.g., $\mathsf{S}_{35} \rightarrow \mathsf{NP}_{27}\ \mathsf{VP}_{17}$

    $\Rightarrow$ infinite PCFG
  - let the number of rules grow unboundedly
    - "new" rules are compositions of several rules from original grammar
    - equivalent to caching tree fragments

    $\Rightarrow$ adaptor grammars
- No reason both can't be done together ...

Outline

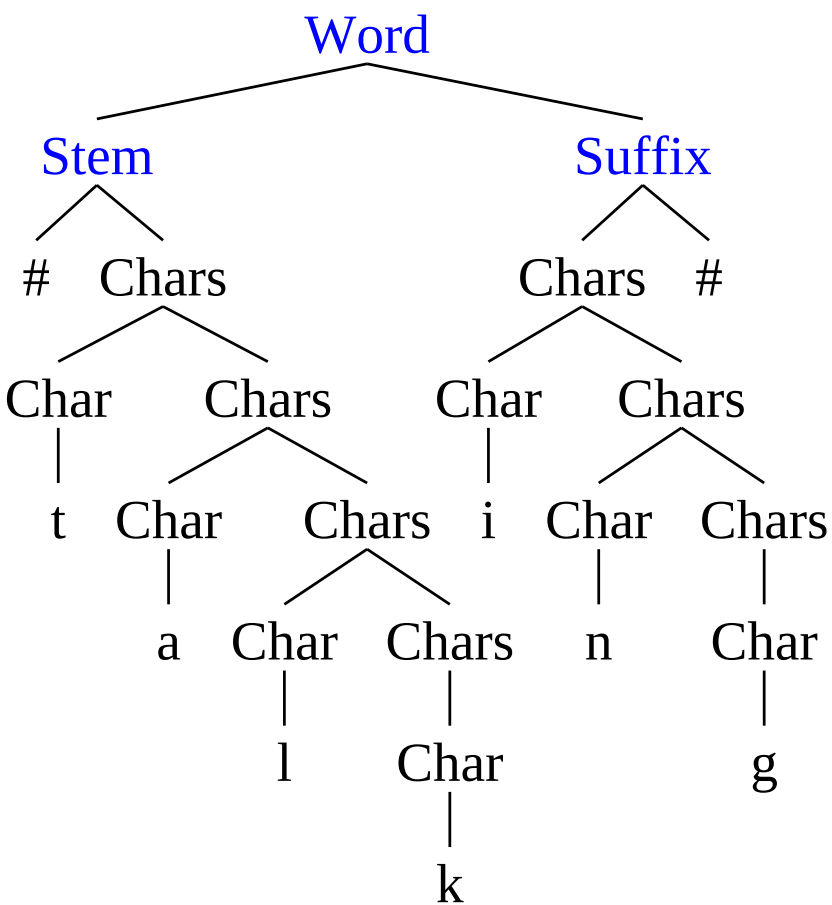
Adaptor grammars: informal description

- An adaptor grammar has a set of CFG rules
- These determine the possible structures as in a CFG
- A subset of the nonterminals are *adapted*
- *Unadapted nonterminals* expand by picking a rule and recursively expanding its children, as in a PCFG
- *Adapted nonterminals* can expand in two ways:
  - by picking a rule and recursively expanding its children, or
  - by generating a previously generated tree (with probability proportional to the number of times previously generated)
- Each adapted subtree behaves like a new rule added to the grammar
- The CFG rules of the adapted nonterminals determine the *base distribution* over these trees

Adaptor grammars as generative processes

- The sequence of trees generated by an adaptor grammar are *not* independent
  - it *learns* from the trees it generates
  - if an adapted subtree has been used frequently in the past, it's more likely to be used again
- (but the sequence of trees is *exchangable*)
- An *unadapted nonterminal* A expands using $A \rightarrow \beta$ with probability $\theta_{A \rightarrow \beta}$
- An *adapted nonterminal* A expands:
  - to a subtree τ rooted in A with probability proportional to the number of times τ was previously generated
  - using $A \rightarrow \beta$ with probability proportional to $\alpha_A \theta_{A \rightarrow \beta}$

Adaptor grammar morphology example

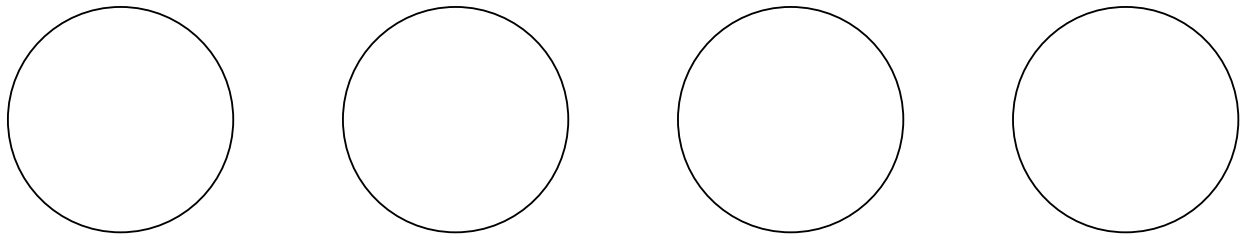


$$
\begin{aligned}
\text{Word} &\rightarrow \text{Stem Suffix} \\
\text{Stem} &\rightarrow \# \text{ Chars} \\
\text{Suffix} &\rightarrow \# \\
\text{Suffix} &\rightarrow \text{Chars } \# \\
\text{Chars} &\rightarrow \text{Char} \\
\text{Chars} &\rightarrow \text{Char Chars} \\
\text{Char} &\rightarrow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

- Stem and Suffix rules generate all possible stems and suffixes
- Adapt Word, Stem and Suffix nonterminals
- One Chinese Restaurant process per adapted nonterminal

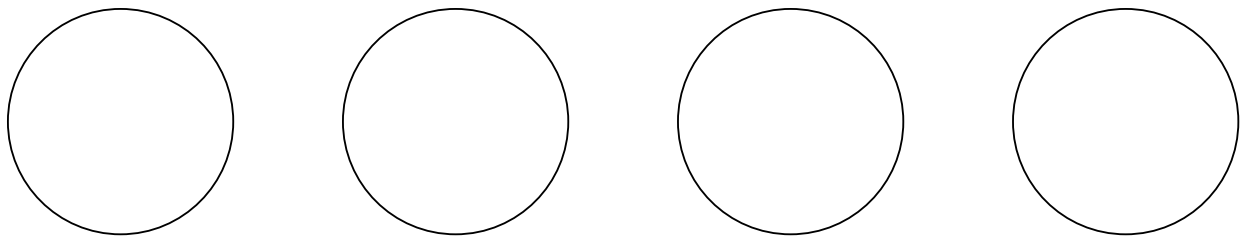
Morphology adaptor grammar (0)

Word restaurant
Word → Stem Suffix



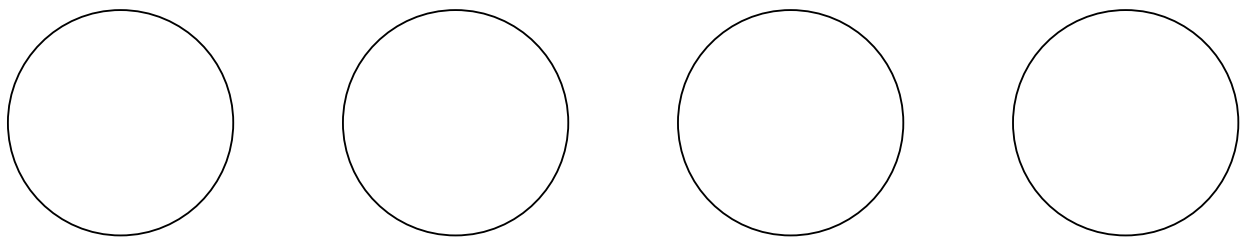
...

Stem restaurant
Stem → #
Stem → # Chars



...

Suffix restaurant
Suffix → #
Suffix → Chars #

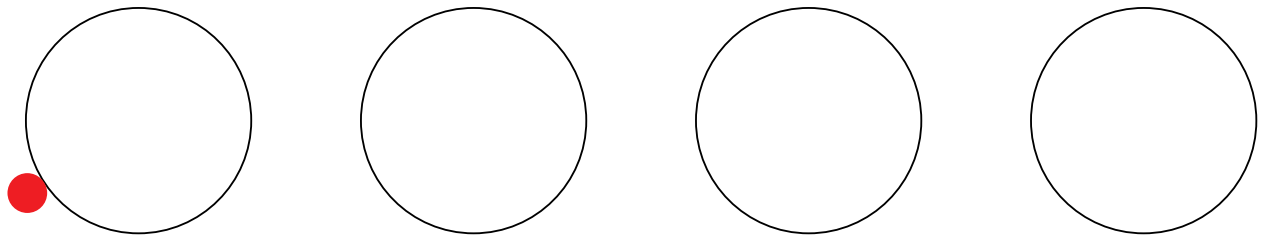


...

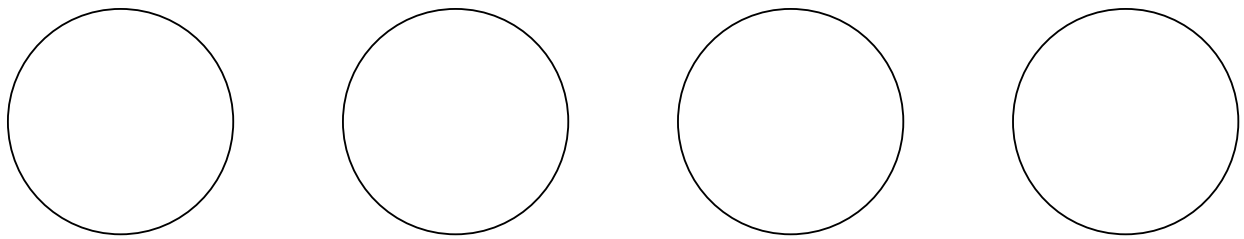
Chars factory
Chars → Char
Chars → Char Chars
Char → a . . . z

Morphology adaptor grammar (1a)

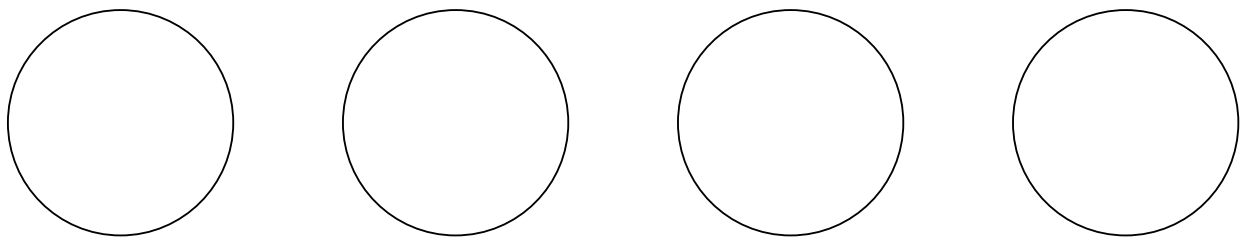
Word restaurant
Word → Stem Suffix



Stem restaurant
Stem → #
Stem → # Chars



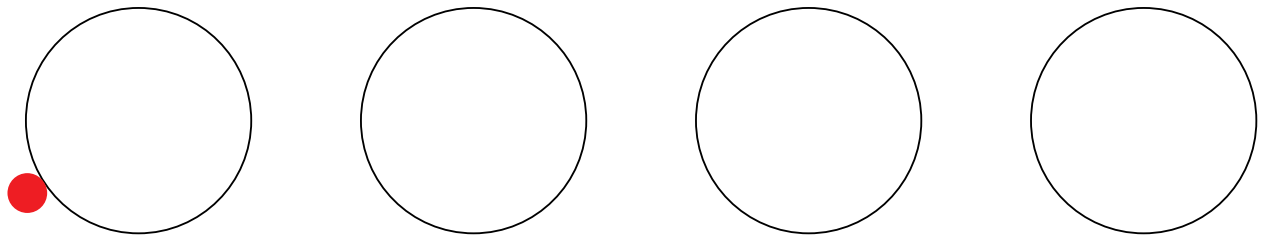
Suffix restaurant
Suffix → #
Suffix → Chars #



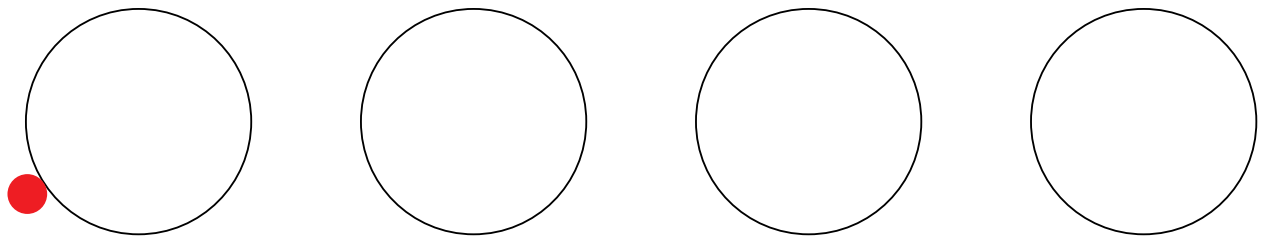
Chars factory
Chars → Char
Chars → Char Chars
Char → a . . . z

Morphology adaptor grammar (1b)

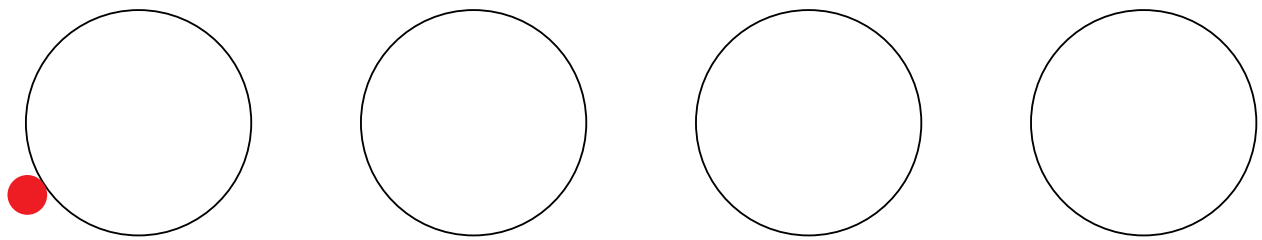
Word restaurant
Word → Stem Suffix



Stem restaurant
Stem → #
Stem → # Chars



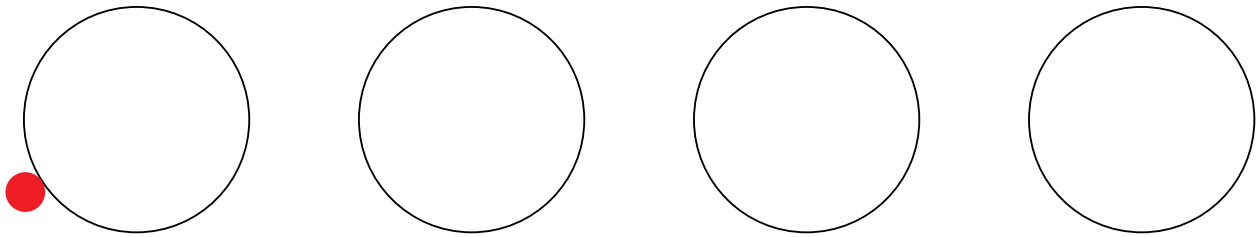
Suffix restaurant
Suffix → #
Suffix → Chars #



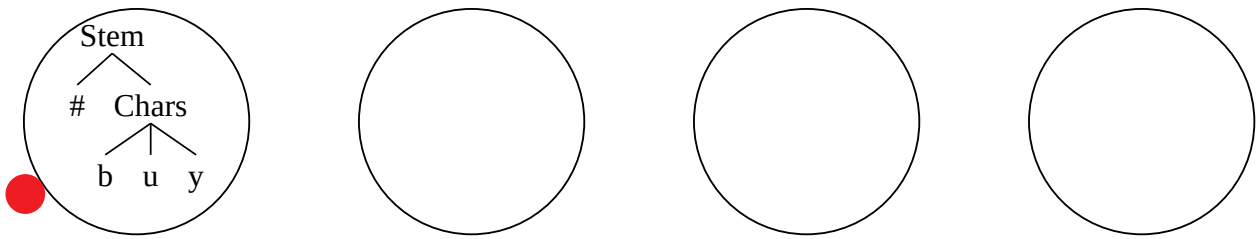
Chars factory
Chars → Char
Chars → Char Chars
Char → a . . . z

Morphology adaptor grammar (1c)

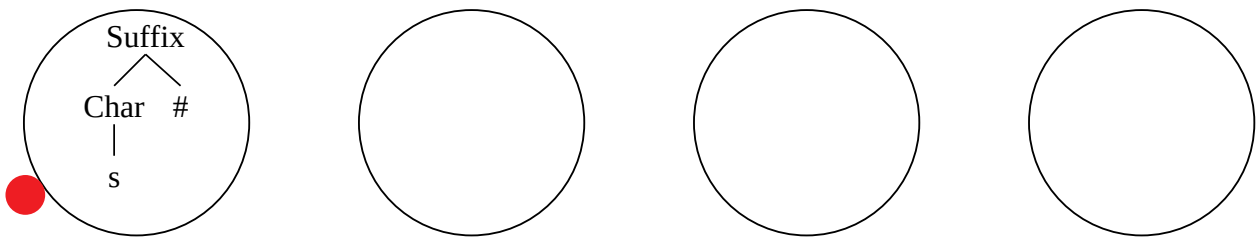
Word restaurant
Word → Stem Suffix



Stem restaurant
Stem → #
Stem → # Chars



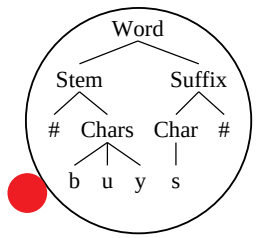
Suffix restaurant
Suffix → #
Suffix → Chars #

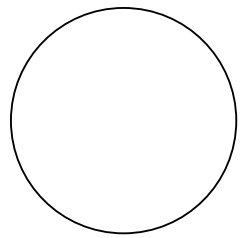


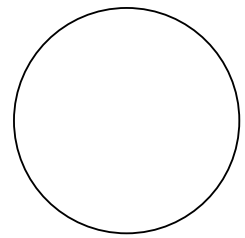
Chars factory
Chars → Char
Chars → Char Chars
Char → a . . . z

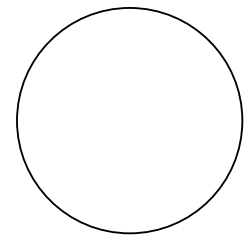
Morphology adaptor grammar (1d)

Word restaurant
Word → Stem Suffix



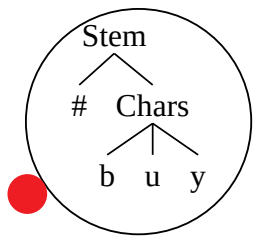


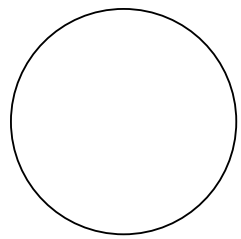


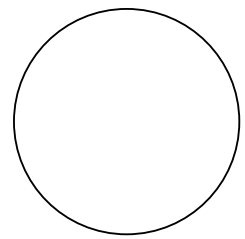


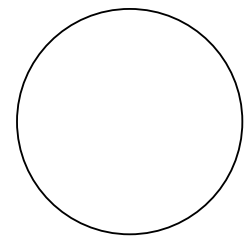
...

Stem restaurant
Stem → #
Stem → # Chars



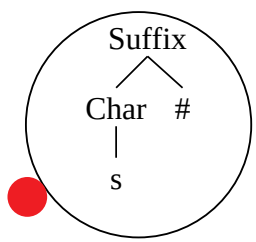


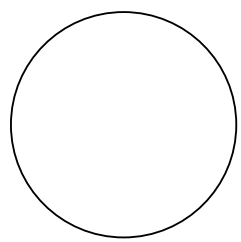


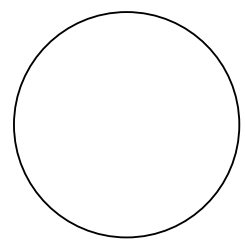


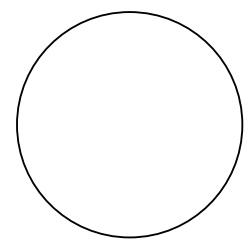
...

Suffix restaurant
Suffix → #
Suffix → Chars #







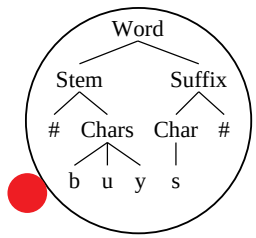


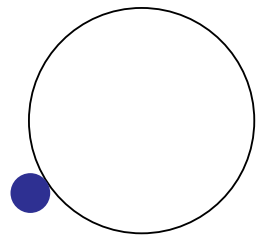
...

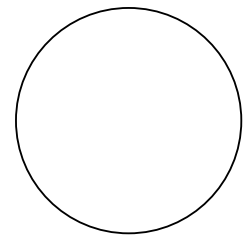
Chars factory
Chars → Char
Chars → Char Chars
Char → a . . . z

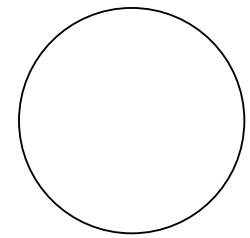
Morphology adaptor grammar (2a)

Word restaurant
Word → Stem Suffix



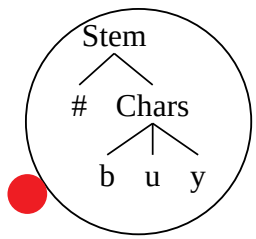


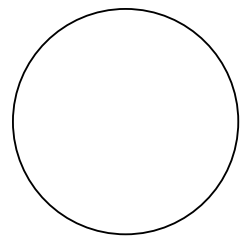


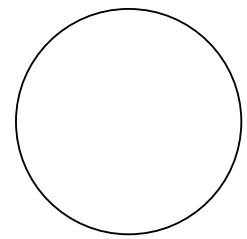


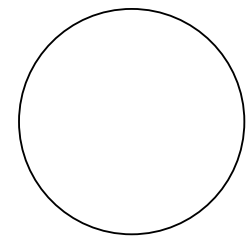
...

Stem restaurant
Stem → #
Stem → # Chars



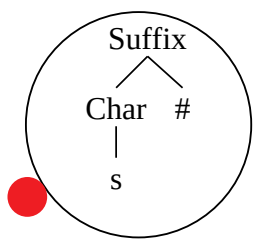


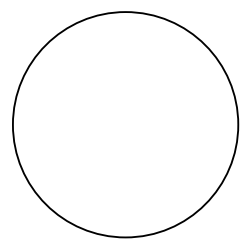


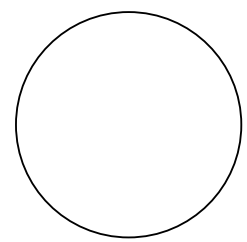


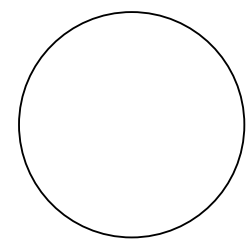
...

Suffix restaurant
Suffix → #
Suffix → Chars #







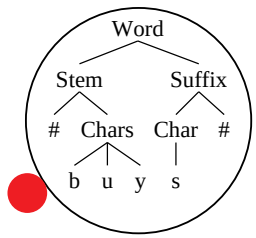


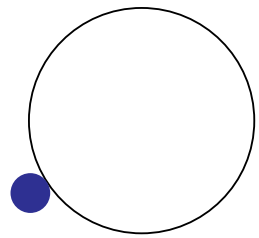
...

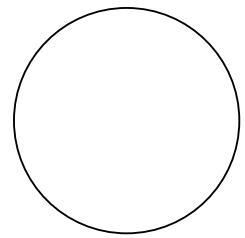
Chars factory
Chars → Char
Chars → Char Chars
Char → a . . . z

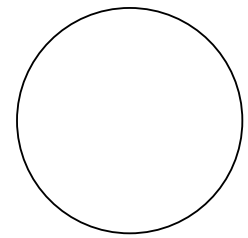
Morphology adaptor grammar (2b)

Word restaurant
Word → Stem Suffix



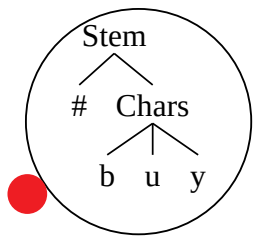


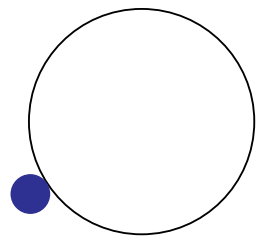


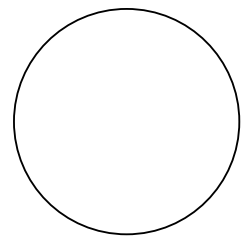


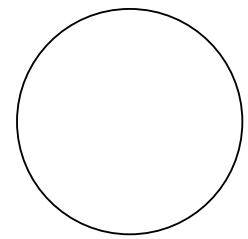
...

Stem restaurant
Stem → #
Stem → # Chars



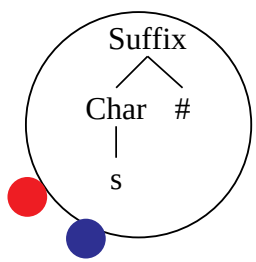


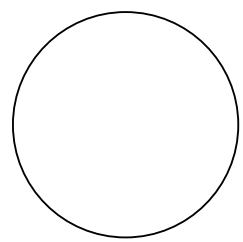


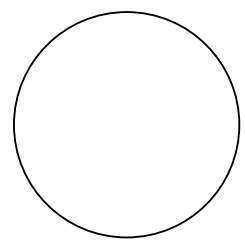


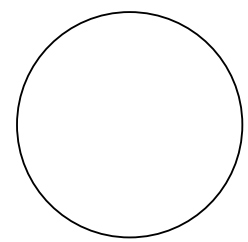
...

Suffix restaurant
Suffix → #
Suffix → Chars #







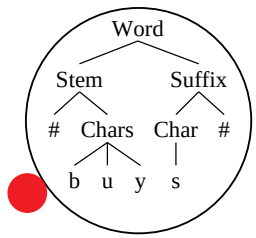


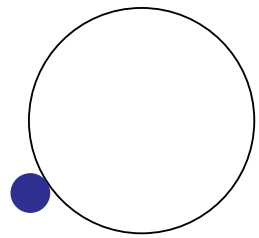
...

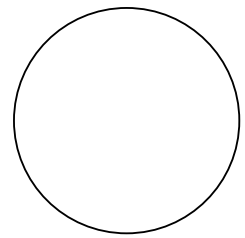
Chars factory
Chars → Char
Chars → Char Chars
Char → a . . . z

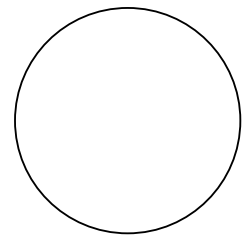
Morphology adaptor grammar (2c)

Word restaurant
Word → Stem Suffix



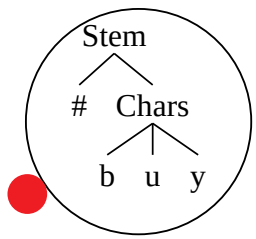


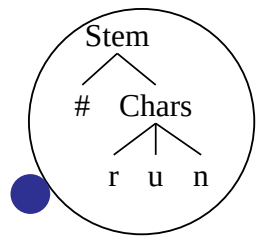


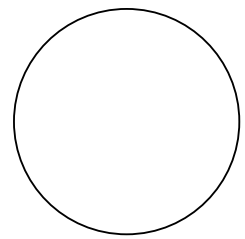


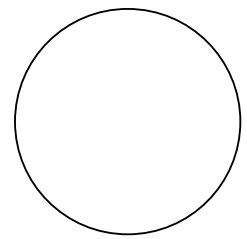
...

Stem restaurant
Stem → #
Stem → # Chars



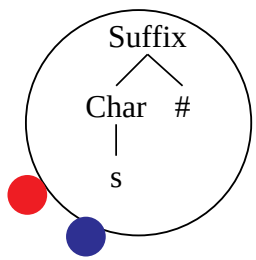


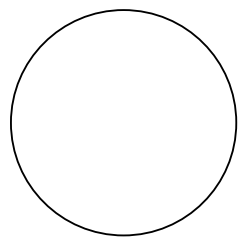


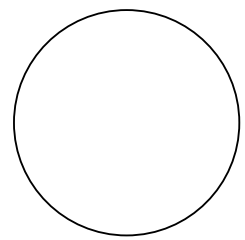


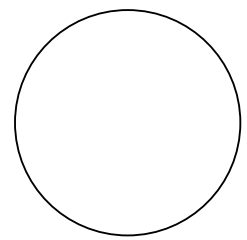
...

Suffix restaurant
Suffix → #
Suffix → Chars #







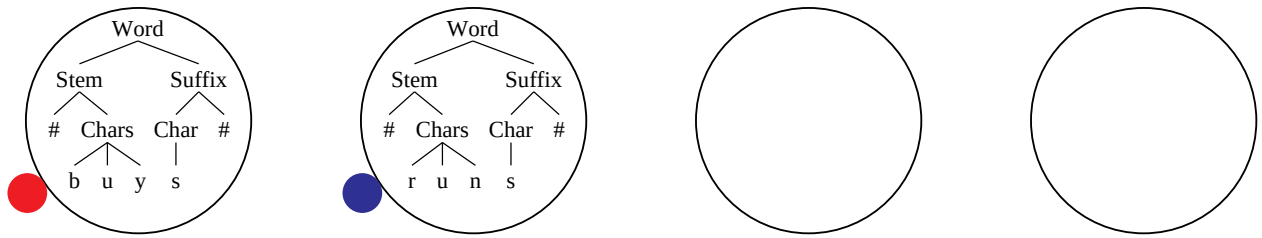


...

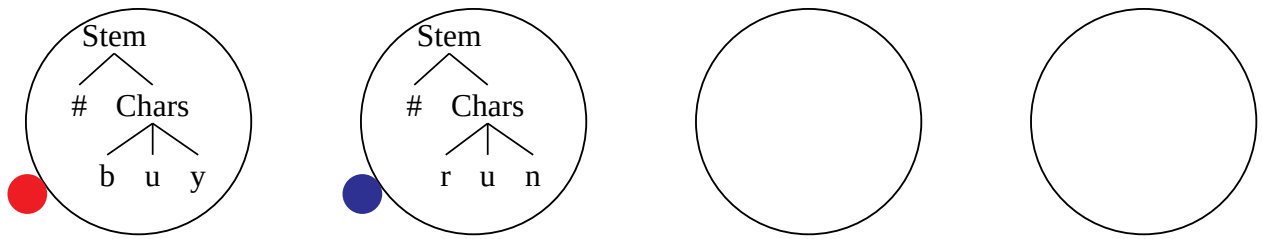
Chars factory
Chars → Char
Chars → Char Chars
Char → a . . . z

Morphology adaptor grammar (2d)

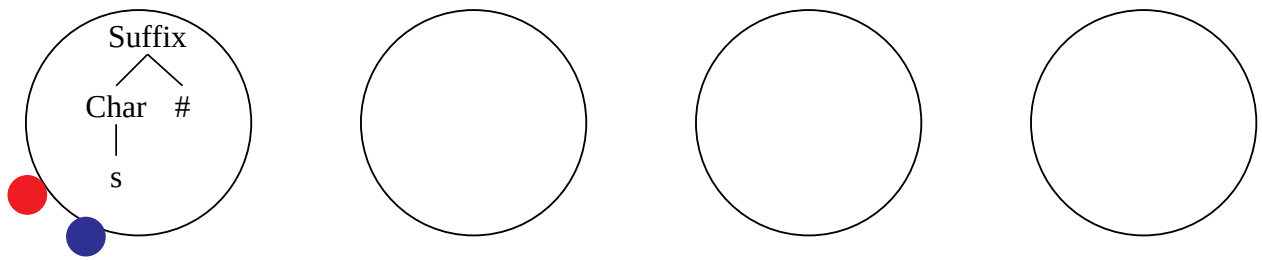
Word restaurant
Word → Stem Suffix



Stem restaurant
Stem → #
Stem → # Chars



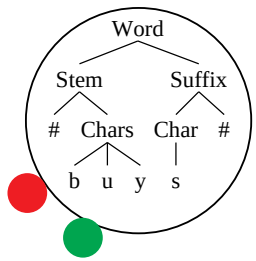
Suffix restaurant
Suffix → #
Suffix → Chars #

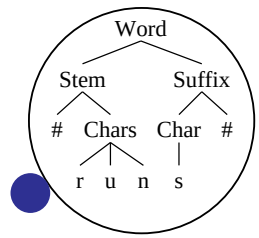


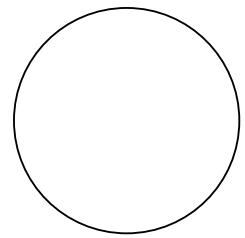
Chars factory
Chars → Char
Chars → Char Chars
Char → a . . . z

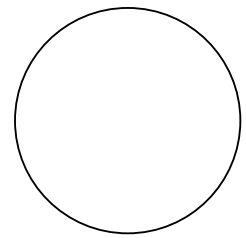
Morphology adaptor grammar (3)

Word restaurant
Word → Stem Suffix



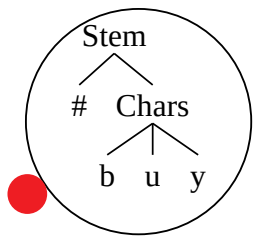


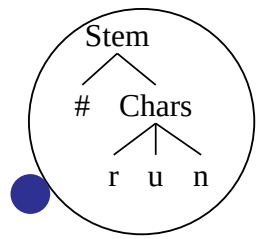


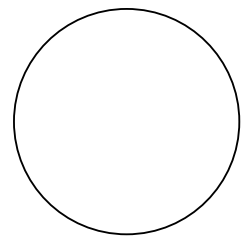


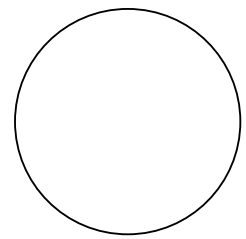
...

Stem restaurant
Stem → #
Stem → # Chars



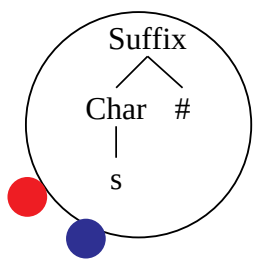


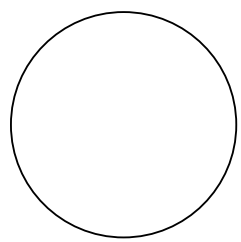


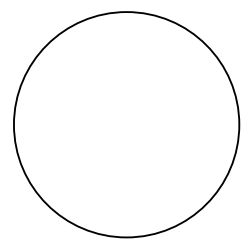


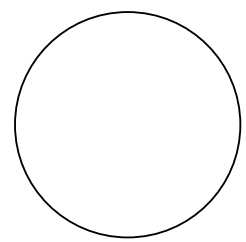
...

Suffix restaurant
Suffix → #
Suffix → Chars #







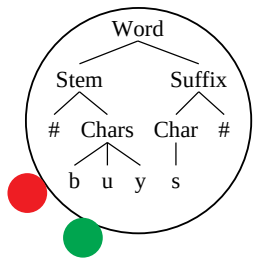


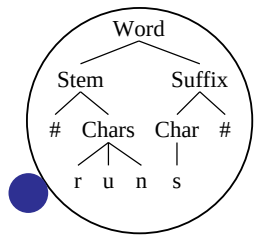
...

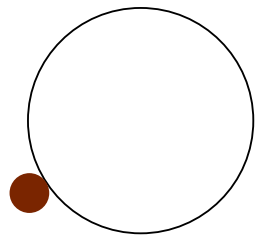
Chars factory
Chars → Char
Chars → Char Chars
Char → a . . . z

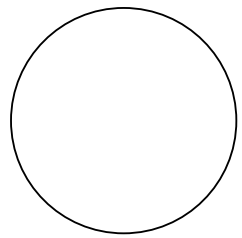
Morphology adaptor grammar (4a)

Word restaurant
Word → Stem Suffix



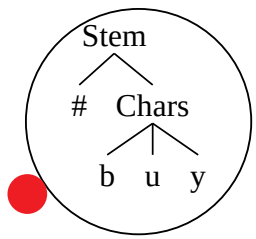


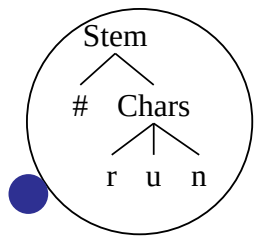


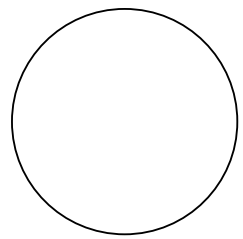


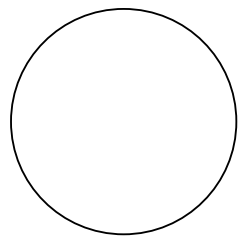
...

Stem restaurant
Stem → #
Stem → # Chars



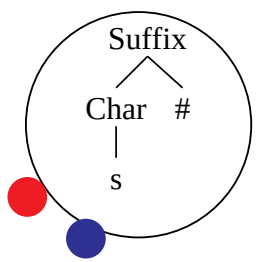


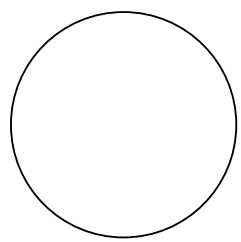


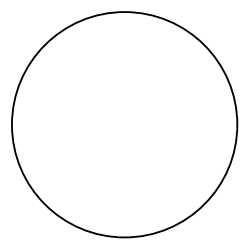


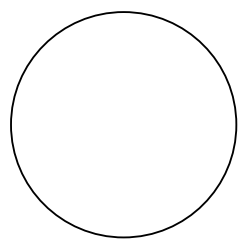
...

Suffix restaurant
Suffix → #
Suffix → Chars #







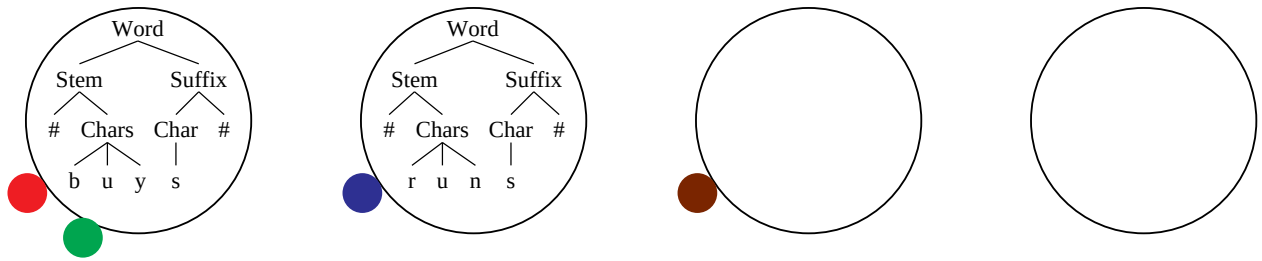


...

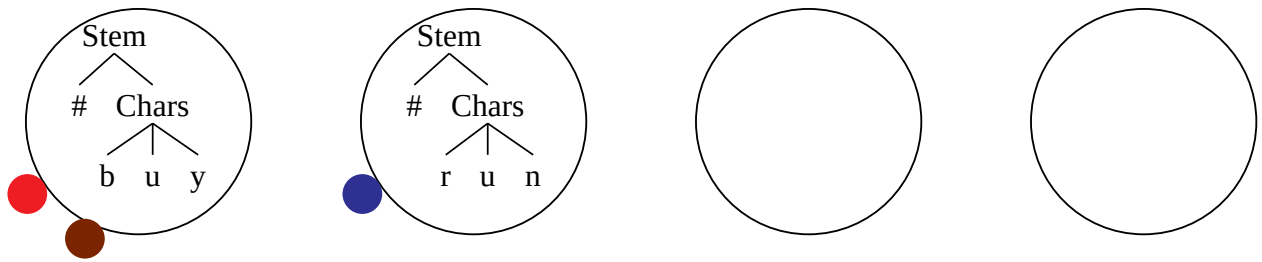
Chars factory
Chars → Char
Chars → Char Chars
Char → a . . . z

Morphology adaptor grammar (4b)

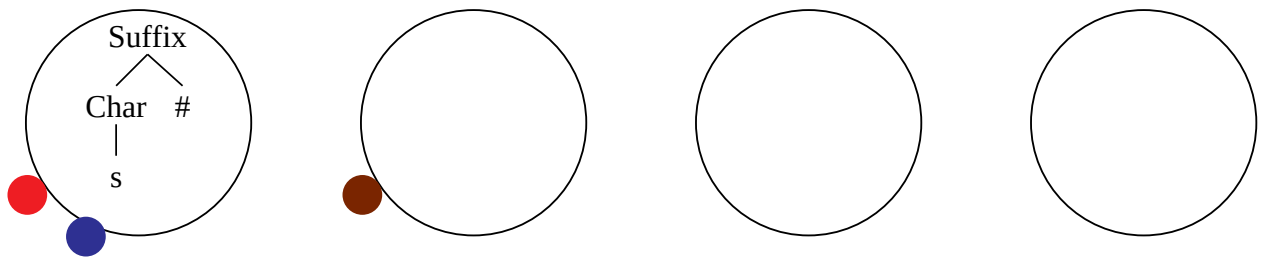
Word restaurant
Word → Stem Suffix



Stem restaurant
Stem → #
Stem → # Chars



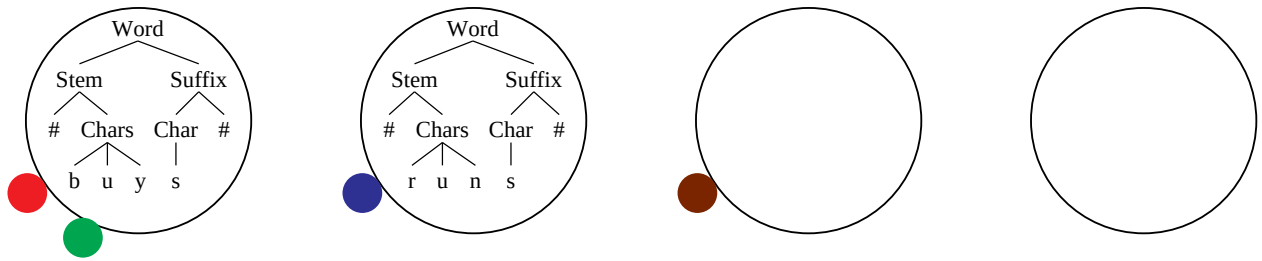
Suffix restaurant
Suffix → #
Suffix → Chars #



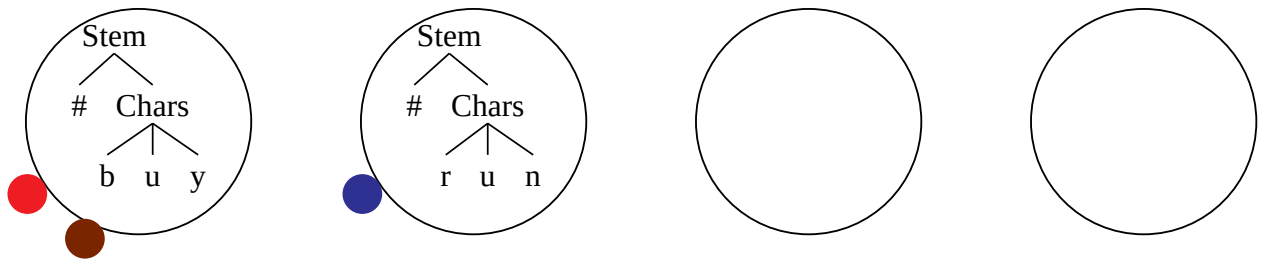
Chars factory
Chars → Char
Chars → Char Chars
Char → a . . . z

Morphology adaptor grammar (4c)

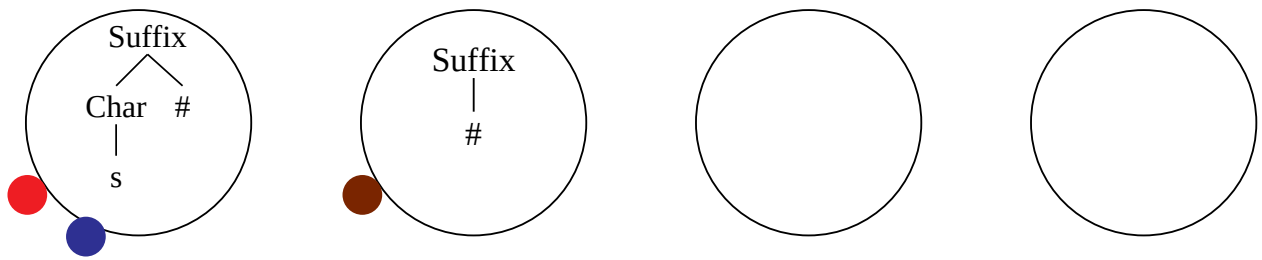
Word restaurant
Word → Stem Suffix



Stem restaurant
Stem → #
Stem → # Chars



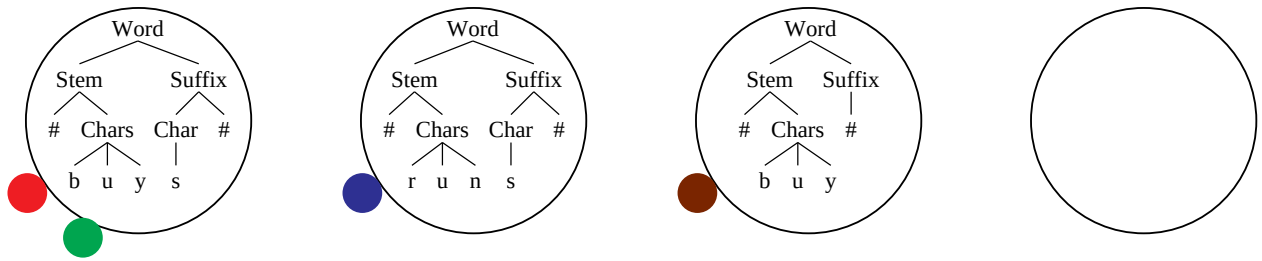
Suffix restaurant
Suffix → #
Suffix → Chars #



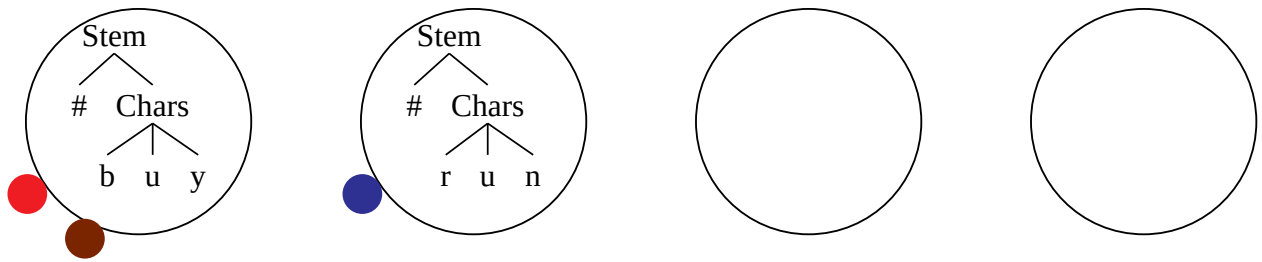
Chars factory
Chars → Char
Chars → Char Chars
Char → a . . . z

Morphology adaptor grammar (4d)

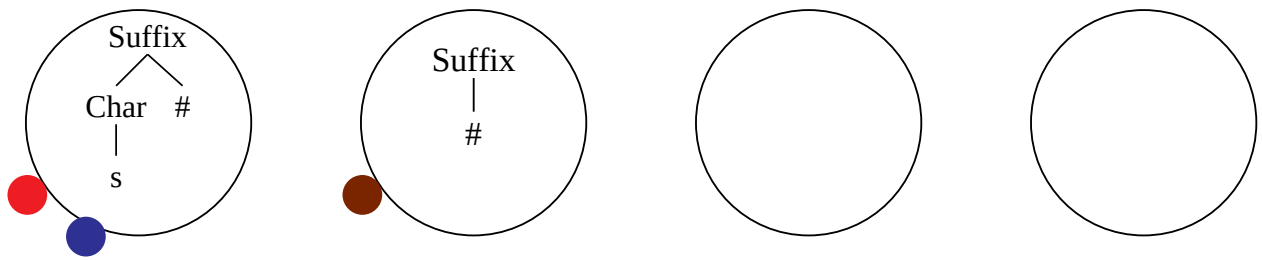
Word restaurant
Word → Stem Suffix



Stem restaurant
Stem → #
Stem → # Chars



Suffix restaurant
Suffix → #
Suffix → Chars #



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

Properties of adaptor grammars

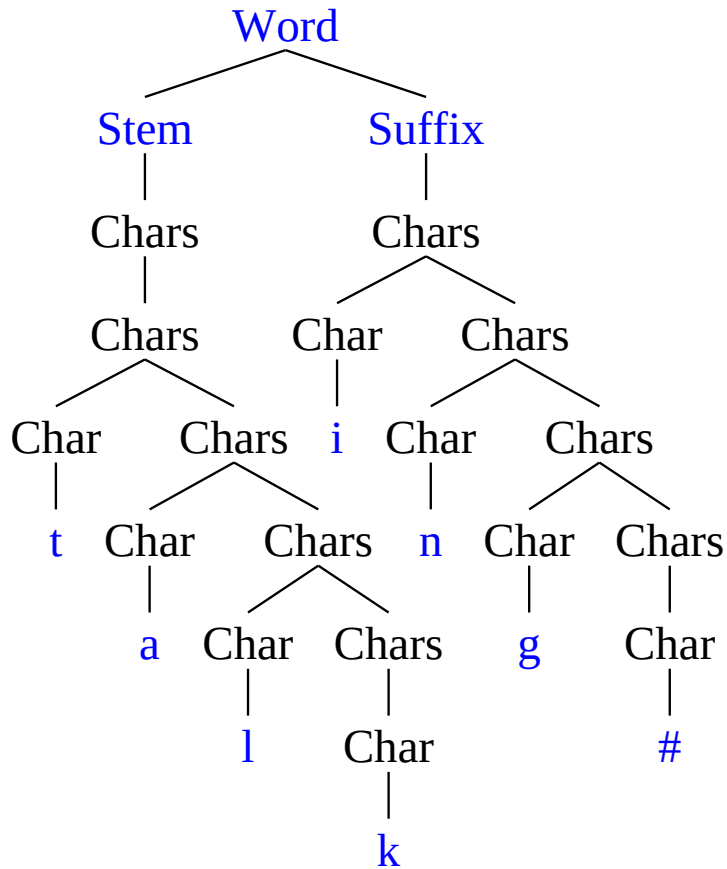
- Possible trees generated by CFG rules
  but the probability of each adapted tree is estimated separately
- Probability of a subtree τ is proportional to:
  - the number of times τ was seen before
    $\Rightarrow$ "rich get richer" dynamics (Zipf distributions)
  - plus α_A times prob. of generating it via PCFG expansion

$\Rightarrow$ Useful compound structures can be *more probable than their parts*

- PCFG rule probabilities estimated *from table labels*
  $\Rightarrow$ learns from types, not tokens
  $\Rightarrow$ dampens frequency variation

Bayesian hierarchy inverts grammatical hierarchy

- Grammatically, a Word is composed of a Stem and a Suffix, which are composed of Chars
- To generate a new Word from an adaptor grammar
  - reuse an old Word, or
  - generate a fresh one from the base distribution, i.e., generate a Stem and a Suffix
- Lower in the tree
  $\Rightarrow$ higher in Bayesian hierarchy



PCFGs as recursive mixtures

For simplicity assume all rules are of the form $A \to B\,C$ or $A \to w$, where $A, B, C \in N$ (nonterminals) and $w \in T$ (terminals).
Each nonterminal $A \in N$ generates a distribution G_A over the trees rooted in A.

$$G_A = \sum_{A \to B\,C \in R_A} \theta_{A \to B\,C} \textsc{Tree}_A(G_B, G_C) + \sum_{A \to w \in R_A} \theta_{A \to w} \textsc{Tree}_A(w)$$

where $\textsc{Tree}_A(w)$ puts all of its mass on the tree with child w and $\textsc{Tree}_A(P, Q)$ is the distribution over trees rooted in A with children distributed according to P and Q respectively.

$$\textsc{Tree}_A(P, Q)\left(\begin{array}{ccc} & A & \\ t_1 & & t_2 \end{array}\right) = P(t_1)\,Q(t_2)$$

The tree language generated by the PCFG is G_S.

Adaptor grammars as recursive mixtures

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

$$\begin{aligned} G_A &\sim \mathsf{DP}(\alpha_A, H_A) && \text{if } \alpha_A > 0 \\ &= H_A && \text{if } \alpha_A = 0 \end{aligned}$$

$$H_A = \sum_{A \to B\,C \in R_A} \theta_{A \to B\,C} \textsc{Tree}_A(G_B, G_C) + \sum_{A \to w \in R_A} \theta_{A \to w} \textsc{Tree}_A(w)$$

The grammar generates the distribution 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.

Bayesian priors on adaptor grammar parameters

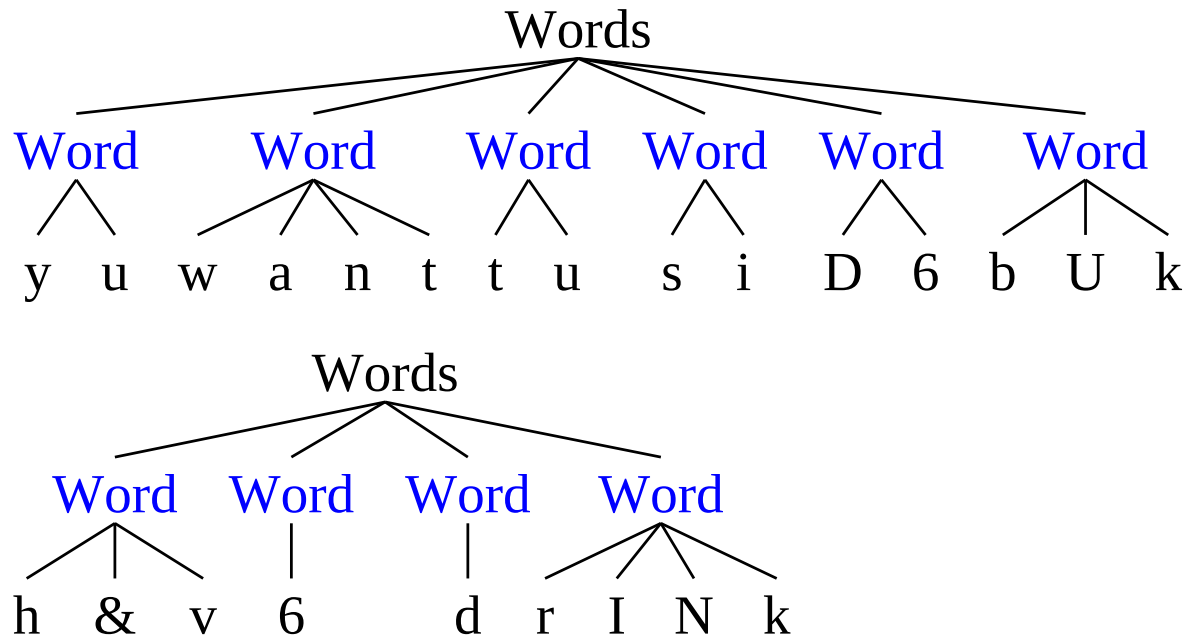
- Parameters of adaptor grammars:
  - probabilities $\theta_{A \to \beta}$ of base grammar rules $A \to \beta$
  - concentration parameters α_A of adapted nonterminals A
- Put Bayesian priors on these parameters
  - (Uniform) Dirichlet prior on base grammar rule probabilities $\boldsymbol{\theta}$
  - Vague Gamma prior on concentration parameter on α_A
- We also use a generalization of CRPs called "Pitman-Yor processes", and put a uniform Dirichlet prior on its a parameter

Outline

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
- Adaptor for Word non-terminal caches previously seen words

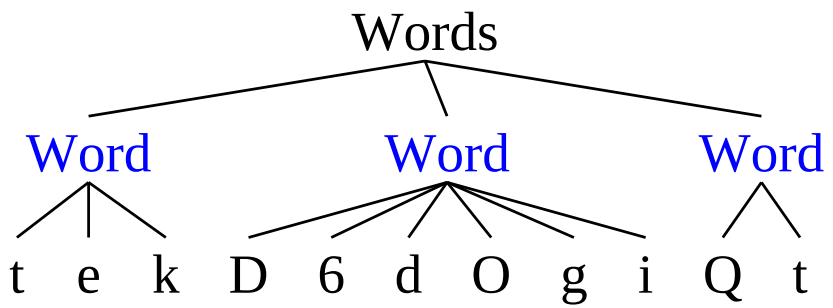
Words $\rightarrow$ Word$^+$
Word $\rightarrow$ Phoneme$^+$

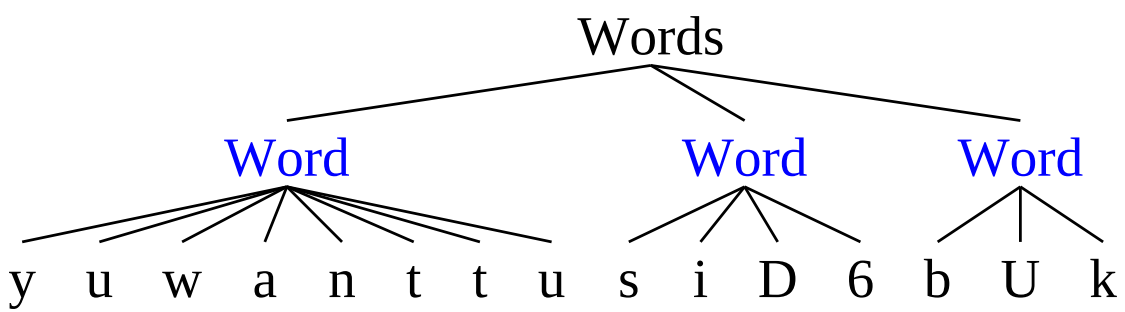


- Unigram word segmentation on Brent corpus: 55% token f-score

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





Unigram word segmentation grammar learnt

- Based on the base grammar rules

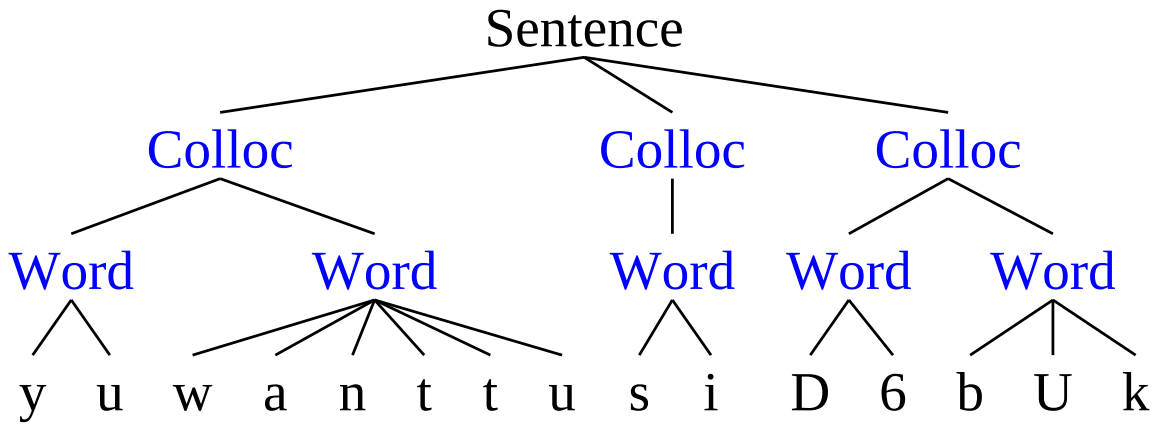
$$\text{Words} \rightarrow \text{Word}^+$$
$$\text{Word} \rightarrow \text{Phoneme}^+$$

the adapted grammar contains 1,712 rules such as:

15758	Words → Word Words
9791	Words → Word
1660	Word → Phoneme$^+$
402	Word → y u
137	Word → I n
111	Word → w I T
100	Word → D 6 d O g i
45	Word → I n D 6
20	Word → I n D 6 h Q s

Modeling collocations improves segmentation

$$\text{Sentence} \rightarrow \text{Colloc}^{+}$$
$$\text{Colloc} \rightarrow \text{Word}^{+}$$
$$\text{Word} \rightarrow \text{Phoneme}^{\star}$$

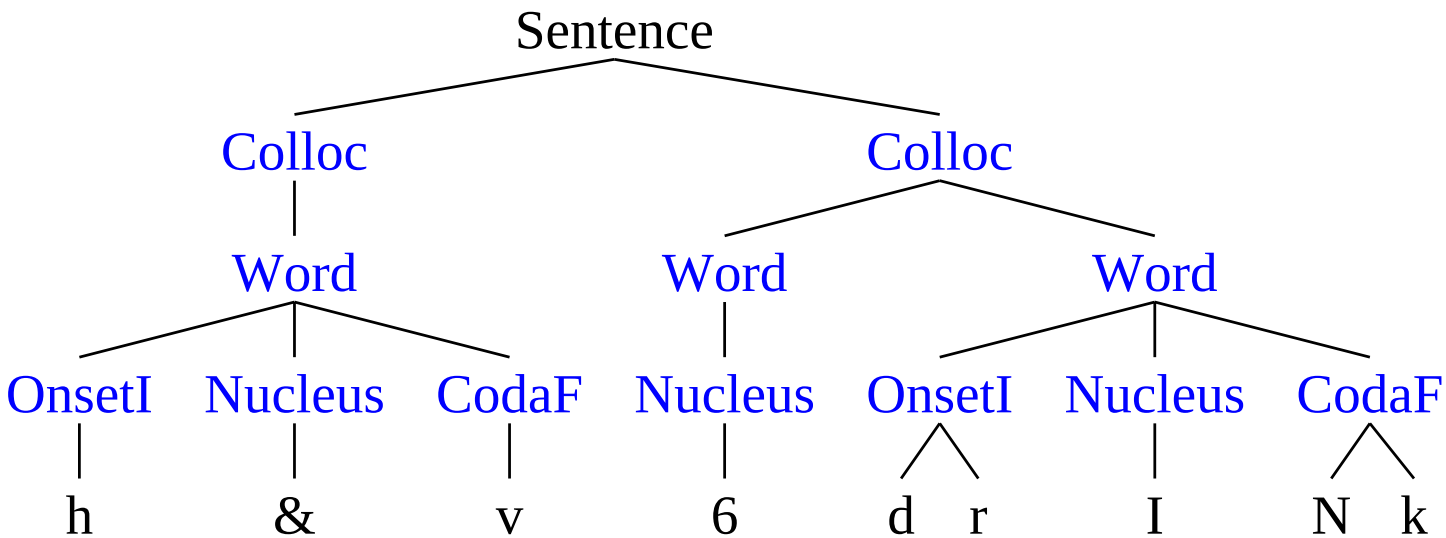


- A Colloc(ation) consists of one or more words
- Both Words and Collocs are adapted (learnt)
- Significantly improves word segmentation accuracy over unigram model (75% token f-score; same as Goldwater's bigram model)
- Two levels of Collocations improves slighly (76%)

Syllables + Collocations + Word segmentation

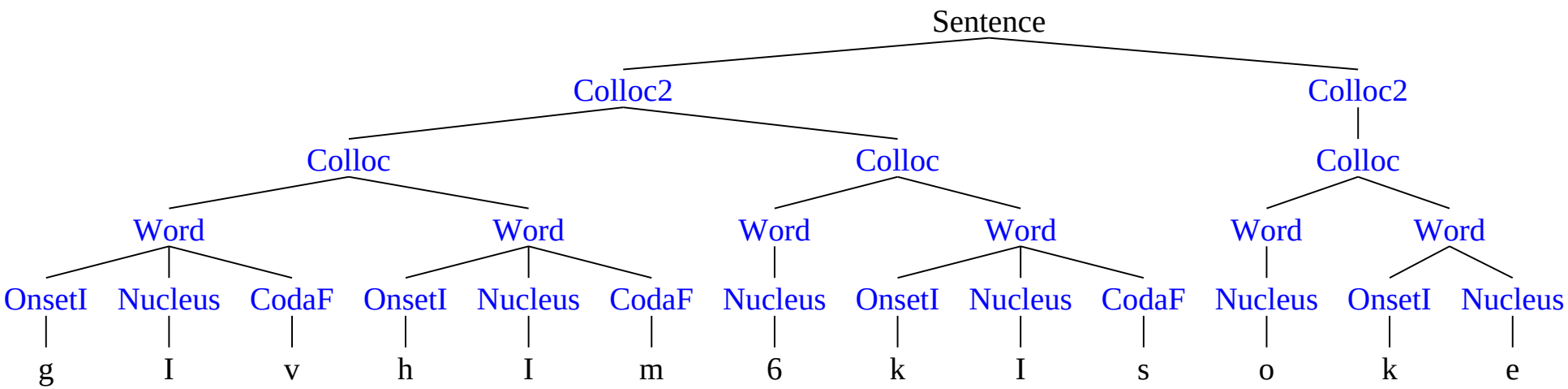
Sentence $\rightarrow$ Colloc$^+$
Word $\rightarrow$ SyllableIF
Word $\rightarrow$ SyllableI Syllable SyllableF

Colloc $\rightarrow$ Word$^+$
Word $\rightarrow$ SyllableI SyllableF
Syllable $\rightarrow$ (as before)



- Word segmentation f-score = 68%
- With 2 Collocation levels f-score = 84% (better than Colloc+morphology)
- Without distinguishing initial/final clusters f-score = 82%

Syllables + 2-level Collocations + Word segmentation



Word segmentation results summary

	Collocation levels			
	0	1	2	3
word	0.55	0.73	0.75	0.72
morphology	0.35	0.55	0.79	0.73
syllable	0.32	0.69	0.82	0.79
syllableIF	0.46	0.68	**0.84**	**0.84**

- We can learning collocations and syllable structure together with word segmentation, even though we don't know where the word boundaries are
- Learning these together improves word segmentation accuracy
  - are there other examples of *synergistic interaction* in language learning?

Outline

Estimating adaptor grammars

- Need to estimate:
  - table labels (subtrees) τ and customer count for each table
  - (optional) probabilities of base grammar productions
  - (optional) DP parameters α
- Component-wise Metropolis-within-Gibbs sampler
  - ith component is the parse tree T_i for input string W_i
  - sample parse T_i for W_i using grammar $G'(\boldsymbol{T}_{-i})$ estimated from parses $\boldsymbol{T}_{-i}$ for other inputs
- Sampling directly from conditional distribution of parses seems intractable
  - construct PCFG proposal grammar $G'(\boldsymbol{T}_{-i})$ on the fly
  - each table label τ corresponds to a production in PCFG approximation
  - Use accept/reject to convert samples from PCFG approx to samples from adaptor grammar

PCFG proposal grammar

- Recall that in a CRP,
  - picking existing table τ with prob. $\propto$ n_τ (number of customers seated at τ)
  - picking new table with prob. $\propto$ α (DP concentration parameter)
- Rules of PCFG proposal grammar G' consist of:
  - rules $A \rightarrow \beta$ from base PCFG: $\theta'_{A \rightarrow \beta} \propto \alpha_A \theta_{A \rightarrow \beta}$
  - A rule $A \rightarrow \textsc{Yield}(\tau)$ for each table τ in A's restaurant: $\theta'_{A \rightarrow \textsc{Yield}(\tau)} \propto n_\tau$, the number of customers at table τ
- Parses of G' can be mapped back to adaptor grammar parses

Implementation details

- The sampler just described can take a long time to "burn in"
- It's hard to predict when annealing will help
- Resampling the table labels (after each pass through the sentences) dramatically increases mobility
  - makes it possible to reanalyse many sentences at once

Outline

Summary and future work

- Adaptor grammars "adapt" their distribution to the strings they have generated
- They learn the probabilities of the subtrees of the adapted nonterminals
- This makes adaptor grammars *non-parametric*; the number of subtrees they track depends on the data
- A variety of different linguistic phenomena can be described with adaptor grammars
- Because they are grammars, they are easy to design and compose
- But they still have a "context-freeness" that makes it impossible to express e.g., Goldwater's bigram word segmentation model. Can we add context-sensitivity in a manageable way?
- The MCMC sampling algorithm used does not seem to scale well to large data or complicated grammars. Are there better estimators?