# Learning rules with Adaptor Grammars
(the Google edition)

Mark Johnson

joint work with Sharon Goldwater and Tom Griffiths

July 2009

# 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 non-parametric extension of PCFGs which *learn the rules* as well as their probabilities
- Monte Carlo Markov Chain (MCMC) *sampling methods* are natural inference procedures for non-parametric models
  - The adaptor grammar inference procedure samples rules as well their probabilities
- MCMC inference techniques that prove useful include:
  - *estimating hyper-parameters* using slice sampling
  - *modal decoding* from multiple samples from multiple runs
  - *random initialization* instead of incremental initialization
  - *table label resampling* as well as sentence resampling

  All but the last apply to a wide variety of non-parametric models

# Language acquisition as Bayesian inference

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

- 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
- Note: *infinitely many* grammars may have positive posterior probability

# 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 $r$ | $\theta_r$ |
|---|---|
| S $\rightarrow$ NP VP | 1.0 |
| NP $\rightarrow$ Sam | 0.75 |
| VP $\rightarrow$ barks | 0.6 |

| Rule $r$ | $\theta_r$ |
|---|---|
| NP $\rightarrow$ Sandy | 0.25 |
| VP $\rightarrow$ snores | 0.4 |

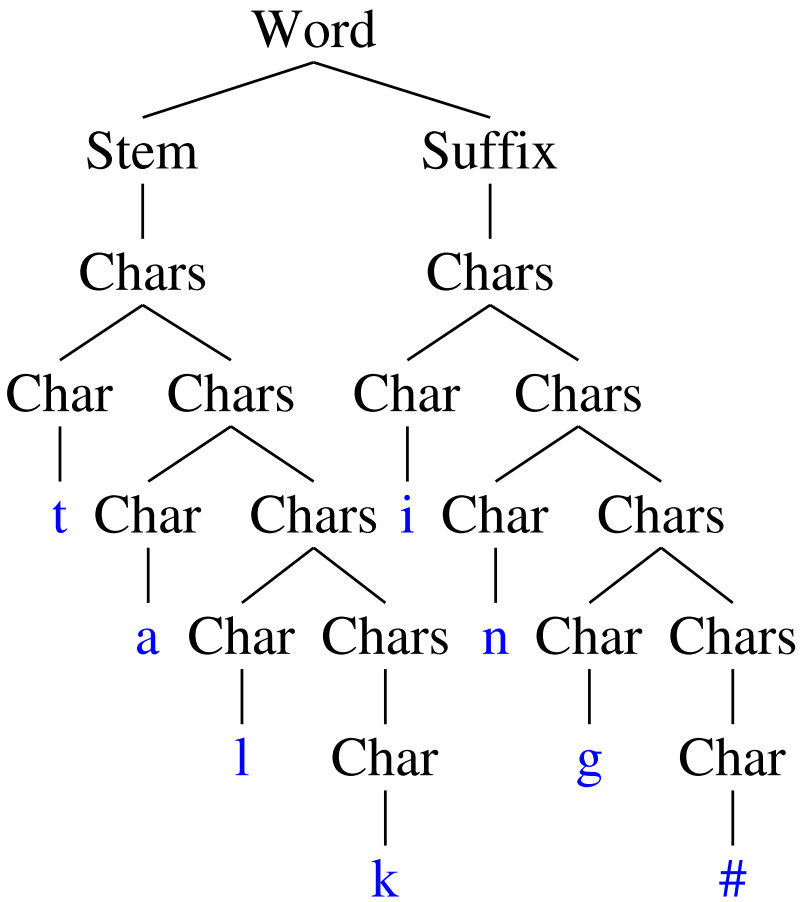
$$P\left(\begin{array}{c} \text{S} \\ \text{NP} \quad \text{VP} \\ | \quad\quad | \\ \text{Sam} \quad \text{barks} \end{array}\right) = 0.45 \qquad P\left(\begin{array}{c} \text{S} \\ \text{NP} \quad \text{VP} \\ | \quad\quad | \\ \text{Sandy} \quad \text{snores} \end{array}\right) = 0.1$$

# Learning syntactic structure is hard

- Bayesian PCFG estimation works well on toy data
- Results are disappointing on "real" data
  - wrong data?
  - wrong rules?
    (rules in PCFG are 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

| | | | | | |
|---|---|---|---|---|---|
| Word | $\rightarrow$ | Stem Suffix | Chars | $\rightarrow$ | Char |
| Stem | $\rightarrow$ | Chars | Chars | $\rightarrow$ | Char Chars |
| Suffix | $\rightarrow$ | Chars | Char | $\rightarrow$ | a \| b \| c \| ... |



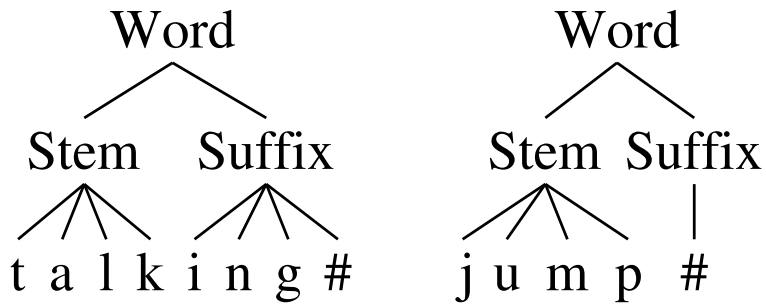

- Grammar's trees can represent any segmentation of words into stems and suffixes

$\Rightarrow$ 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

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



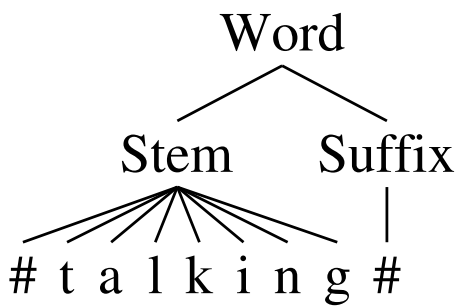

- A rule for each morpheme
  $\Rightarrow$ "PCFG" can represent probability of each morpheme
- *Unbounded number of possible rules, so this is not a PCFG*
  - not a practical problem, as only a finite set of rules could possibly be used in any particular data set

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

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

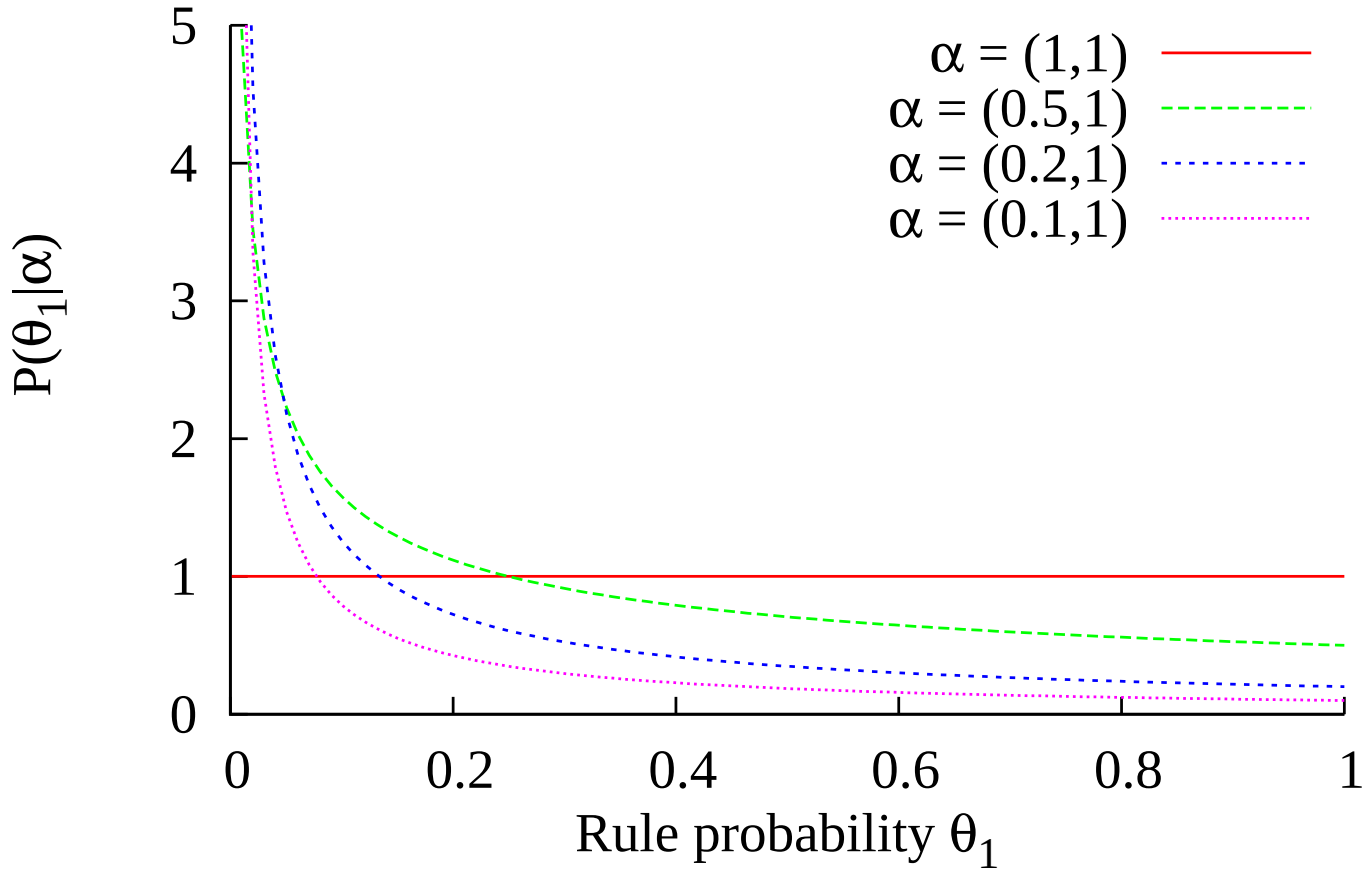
$\Rightarrow$ Saturated model is maximum likelihood estimate

- 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 in standard PCFG estimation, but can "turn rule off" by setting $\theta_{A \to \beta} \approx 0$
- Dirichlet prior with $\alpha_{A \to \beta} \approx 0$ prefers $\theta_{A \to \beta} \approx 0$



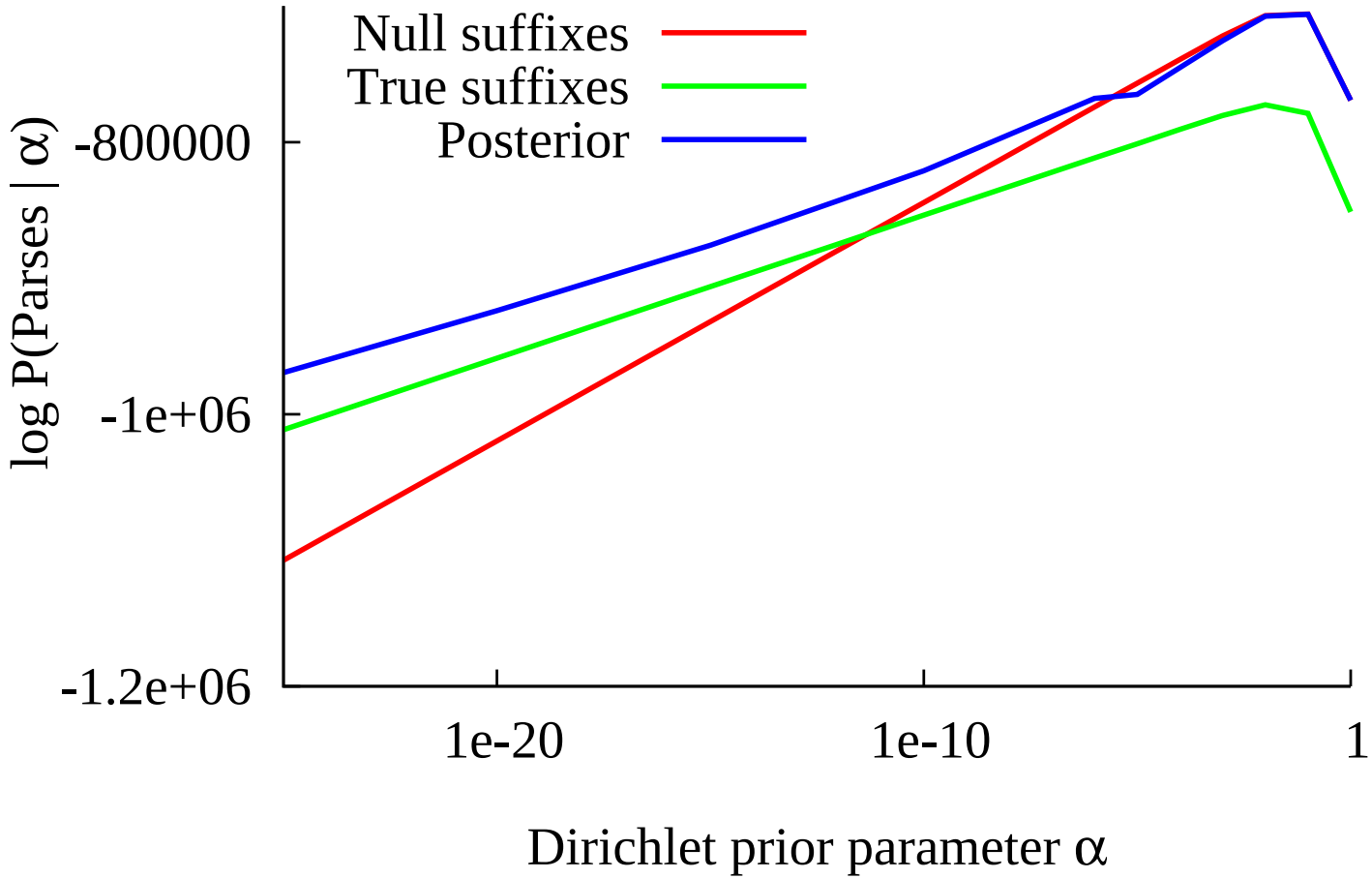
# Morphological segmentation experiment

- Trained on orthographic verbs from U Penn. Wall Street Journal treebank
- Uniform Dirichlet prior prefers sparse solutions as $\alpha \rightarrow 0$
- Gibbs sampler samples from posterior distribution of parses
  - reanalyses each word based on 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 |

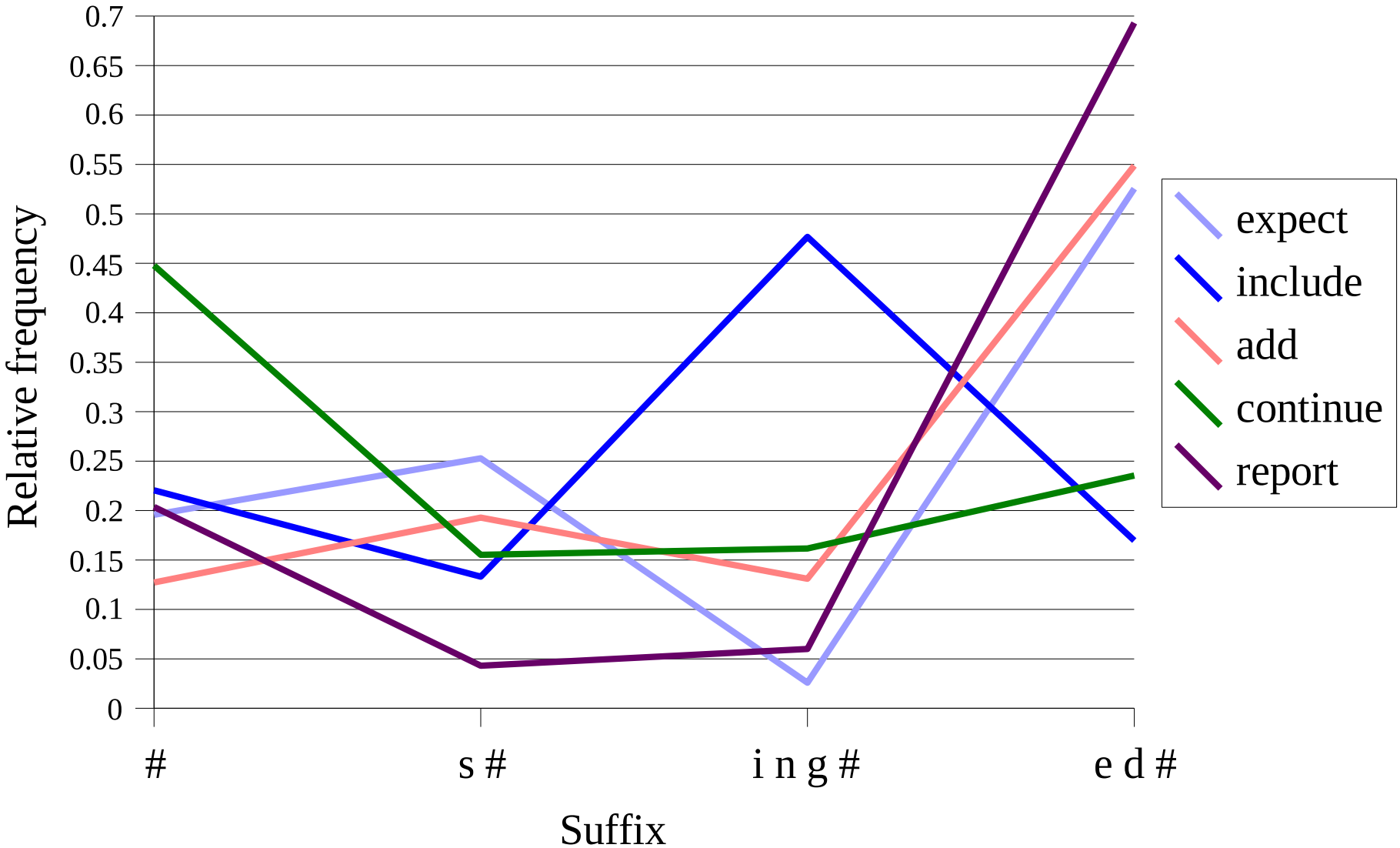
# Log posterior for models on token data




- Correct solution is nowhere near as likely as posterior

$\Rightarrow$ model is wrong!

# 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

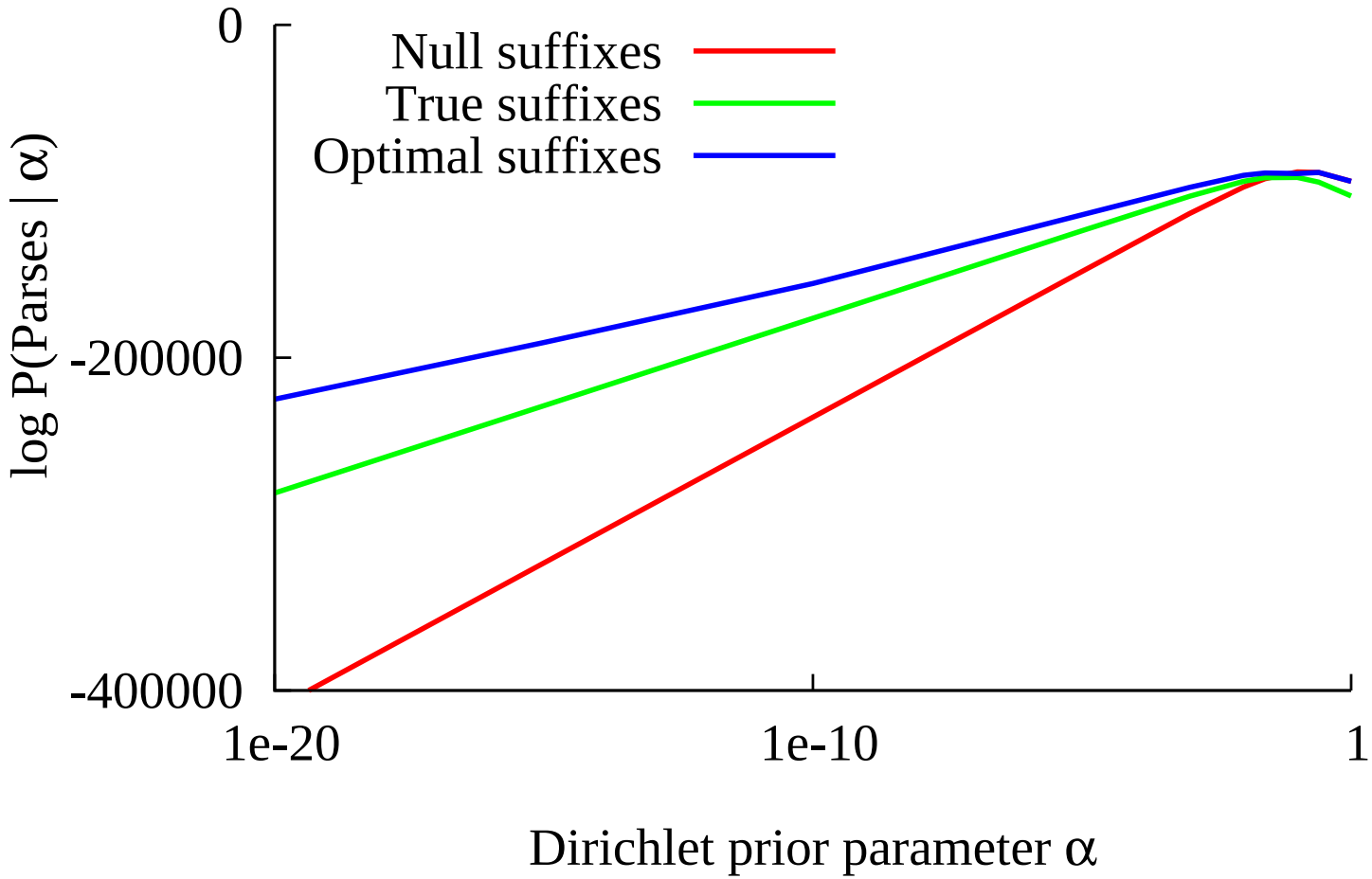
$$\begin{aligned} \text{Data} &= \text{“the cat chased the other cat”} \\ \text{Tokens} &= \text{“the”, “cat”, “chased”, “the”, “other”, “cat”} \\ \text{Types} &= \text{“the”, “cat”, “chased”, “other”} \end{aligned}$$

- Estimating $\theta$ from *word types* rather than word tokens eliminates (most) frequency variation
  - 4 common verb suffixes, so when estimating from verb types $\theta_{\text{Suffix}\rightarrow \text{i n g \#}} \approx 0.25$
- 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 over-dispersion
  $\Rightarrow$ 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
  - inspired by Goldwater's work

# Outline

# Dirichlet-Multinomials with many outcomes

- Dirichlet prior $\boldsymbol{\alpha}$, observed data $\mathbf{z} = (z_1, \ldots, z_n)$

$$\mathrm{P}(Z_{n+1} = k \mid \mathbf{z}, \boldsymbol{\alpha}) \quad \propto \quad \alpha_k + n_k(\mathbf{z})$$

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

$$\mathrm{P}(Z_{n+1} = k \mid \mathbf{z}, \alpha) \quad \propto \quad \begin{cases} n_k(\mathbf{z}) & \text{if } n_k(\mathbf{z}) > 0 \\ \alpha/m & \text{if } n_k(\mathbf{z}) = 0 \end{cases}$$

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

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

*Probability of an outcome never seen before* $\quad \propto \quad \alpha$

# From Dirichlet-multinomials to Chinese Restaurant Processes

- Observations $\mathbf{z} = (z_1, \ldots, z_n)$ ranging over outcomes $1, \ldots, m$
- Outcome $k$ observed $n_k(\mathbf{z})$ times in data $\mathbf{z}$
- *Predictive distribution* with uniform Dirichlet prior:

$$\mathrm{P}(Z_{n+1} = k \mid \mathbf{z}) \;\propto\; n_k(\mathbf{z}) + \alpha/m$$

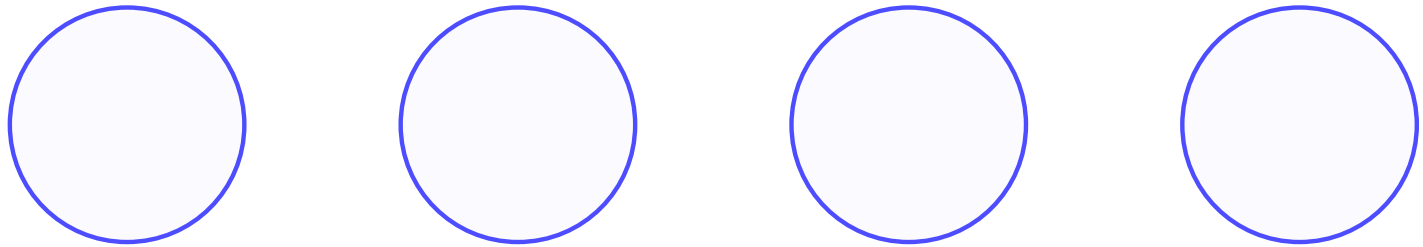
- Let $m \to \infty$

$$\begin{aligned} \mathrm{P}(Z_{n+1} = k \mid \mathbf{z}) &\propto n_k(\mathbf{z}) \text{ if } k \text{ appears in } \mathbf{z} \\ \mathrm{P}(Z_{n+1} \notin \mathbf{z} \mid \mathbf{z}) &\propto \alpha \end{aligned}$$

- If outcomes are exchangable $\Rightarrow$ number in order of occurence $\Rightarrow$ *Chinese Restaurant Process*

$$\mathrm{P}(Z_{n+1} = k \mid \mathbf{z}) \;\propto\; \begin{cases} n_k(\mathbf{z}) & \text{if } k \leq m = \max(\mathbf{z}) \\ \alpha & \text{if } k = m + 1 \end{cases}$$

# Chinese Restaurant Process (0)

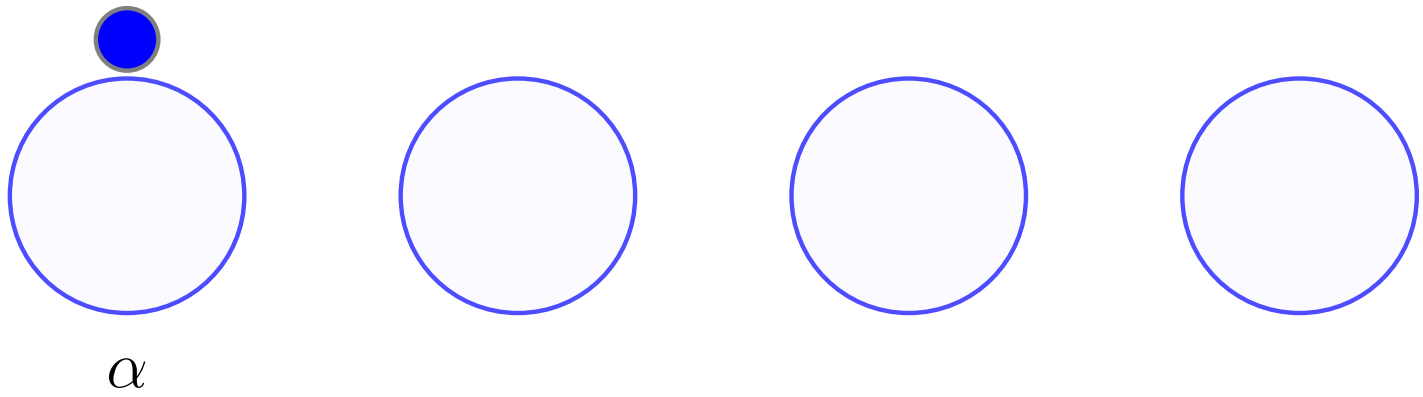


- Customer $\longrightarrow$ table mapping $\mathbf{z} =$
- $\mathrm{P}(\mathbf{z}) = 1$

- Next customer chooses a table according to:

$$\mathrm{P}(Z_{n+1} = k \mid \mathbf{z}) \quad \propto \quad \begin{cases} n_k(\mathbf{z}) & \text{if } k \leq m = \max(\mathbf{z}) \\ \alpha & \text{if } k = m + 1 \end{cases}$$

# Chinese Restaurant Process (1)

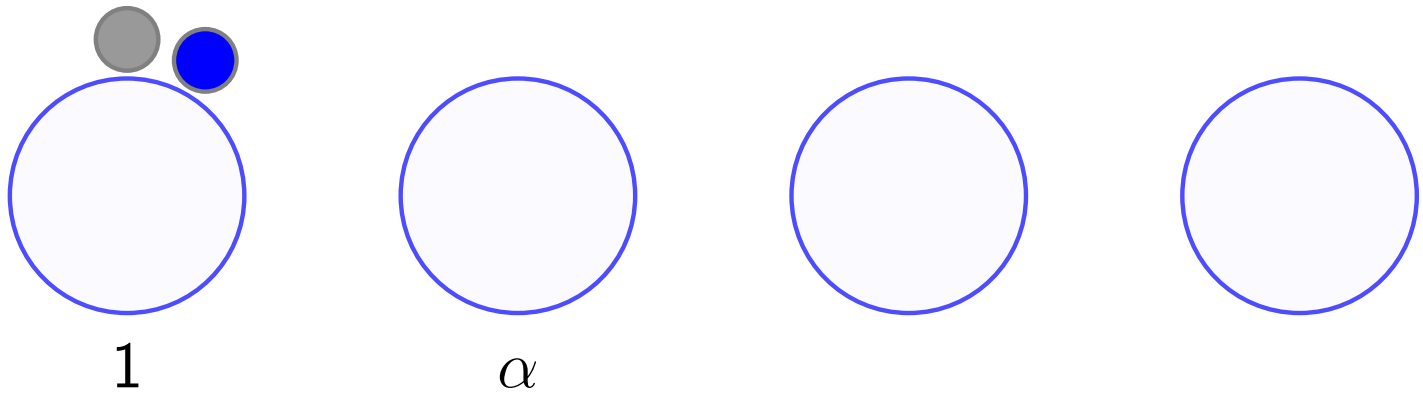



- Customer $\rightarrow$ table mapping $\mathbf{z} = 1$
- $\mathrm{P}(\mathbf{z}) = \alpha/\alpha$

- Next customer chooses a table according to:

$$\mathrm{P}(Z_{n+1} = k \mid \mathbf{z}) \quad \propto \quad \begin{cases} n_k(\mathbf{z}) & \text{if } k \leq m = \max(\mathbf{z}) \\ \alpha & \text{if } k = m + 1 \end{cases}$$

# Chinese Restaurant Process (2)

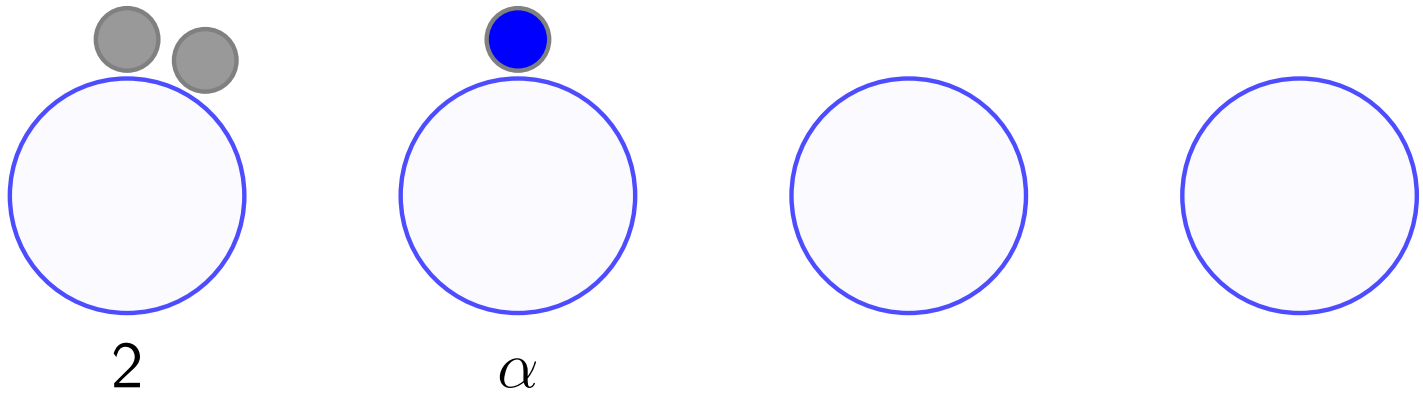



- Customer $\rightarrow$ table mapping $\mathbf{z} = 1, 1$
- $\mathrm{P}(\mathbf{z}) = \alpha/\alpha \times 1/(1+\alpha)$

- Next customer chooses a table according to:

$$\mathrm{P}(Z_{n+1} = k \mid \mathbf{z}) \quad \propto \quad \begin{cases} n_k(\mathbf{z}) & \text{if } k \leq m = \max(\mathbf{z}) \\ \alpha & \text{if } k = m+1 \end{cases}$$

# Chinese Restaurant Process (3)

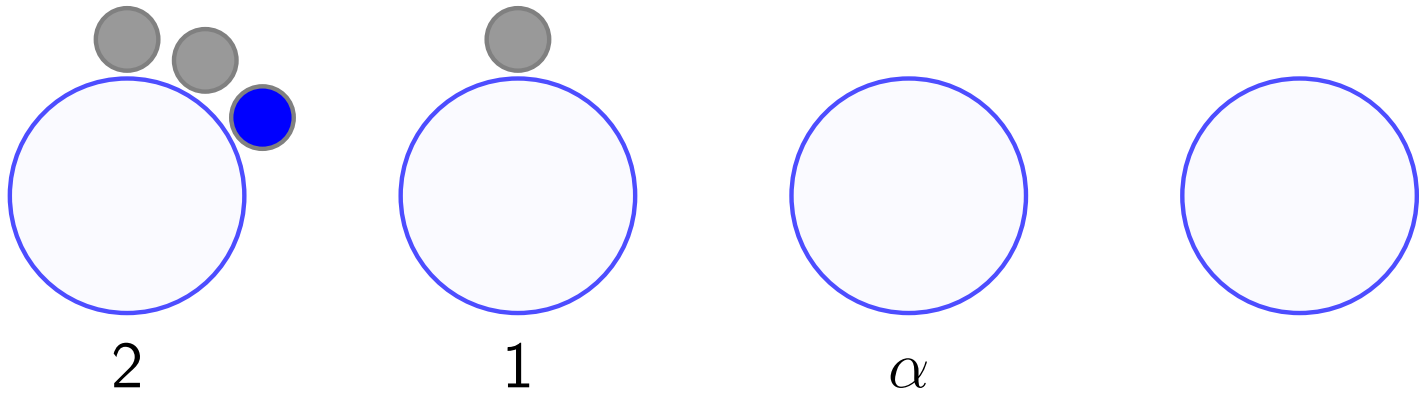



- Customer $\rightarrow$ table mapping $\mathbf{z} = 1, 1, 2$
- $\mathsf{P}(\mathbf{z}) = \alpha/\alpha \times 1/(1+\alpha) \times \alpha/(2+\alpha)$

- Next customer chooses a table according to:

$$\mathsf{P}(Z_{n+1} = k \mid \mathbf{z}) \quad \propto \quad \begin{cases} n_k(\mathbf{z}) & \text{if } k \leq m = \max(\mathbf{z}) \\ \alpha & \text{if } k = m+1 \end{cases}$$

# Chinese Restaurant Process (4)

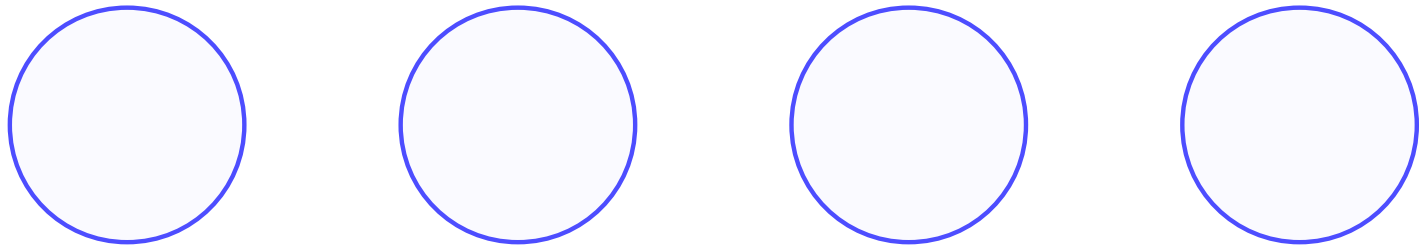



- Customer $\rightarrow$ table mapping $\mathbf{z} = 1, 1, 2, 1$
- $\mathsf{P}(\mathbf{z}) = \alpha/\alpha \times 1/(1+\alpha) \times \alpha/(2+\alpha) \times 2/(3+\alpha)$

- Next customer chooses a table according to:

$$\mathsf{P}(Z_{n+1} = k \mid \mathbf{z}) \quad \propto \quad \begin{cases} n_k(\mathbf{z}) & \text{if } k \leq m = \max(\mathbf{z}) \\ \alpha & \text{if } k = m+1 \end{cases}$$

# Pitman-Yor Process (0)

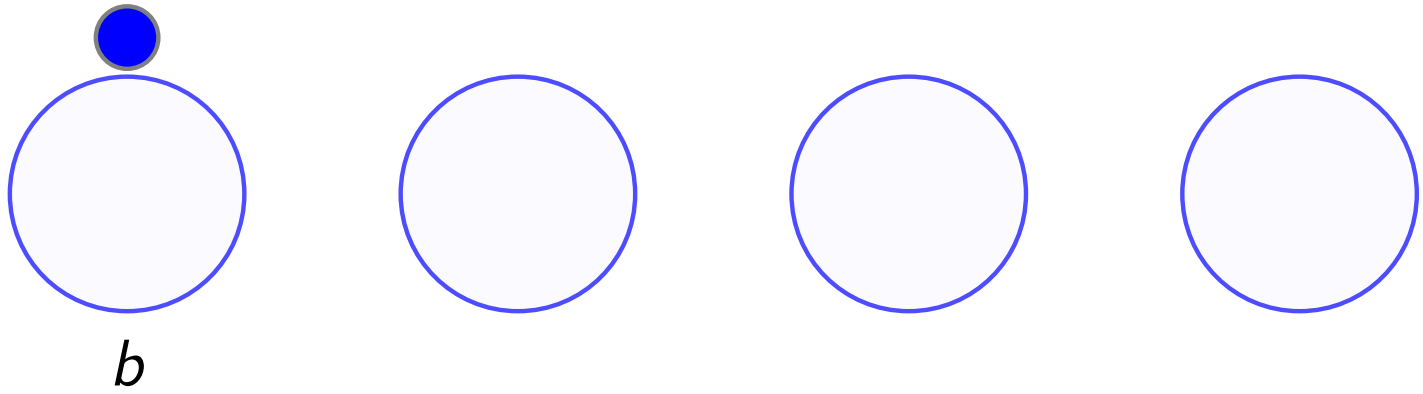


- Customer $\longrightarrow$ table mapping $\mathbf{z} =$
- $\mathrm{P}(\mathbf{z}) = 1$

- In CRPs, probability of choosing a table $\propto$ number of customers $\Rightarrow$ strong *rich get richer* effect
- Pitman-Yor processes take mass $a$ from each occupied table and give it to the new table

$$\mathrm{P}(Z_{n+1} = k \mid \mathbf{z}) \quad \propto \quad \begin{cases} n_k(\mathbf{z}) - a & \text{if } k \leq m = \max(\mathbf{z}) \\ ma + b & \text{if } k = m + 1 \end{cases}$$

# Pitman-Yor Process (1)

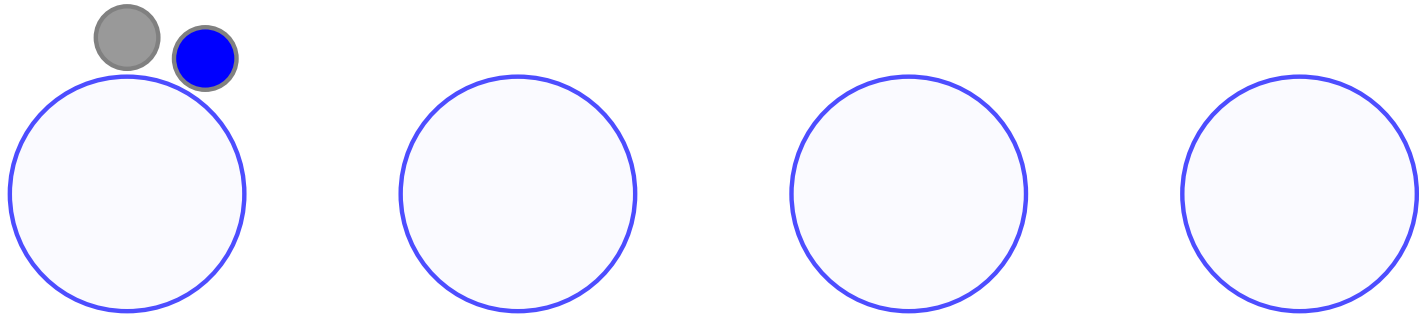



- Customer $\longrightarrow$ table mapping $\mathbf{z} = 1$
- $\mathrm{P}(\mathbf{z}) = b/b$

- In CRPs, probability of choosing a table $\propto$ number of customers $\Rightarrow$ strong *rich get richer* effect
- Pitman-Yor processes take mass $a$ from each occupied table and give it to the new table

$$\mathrm{P}(Z_{n+1} = k \mid \mathbf{z}) \quad \propto \quad \begin{cases} n_k(\mathbf{z}) - a & \text{if } k \leq m = \max(\mathbf{z}) \\ ma + b & \text{if } k = m + 1 \end{cases}$$

# Pitman-Yor Process (2)

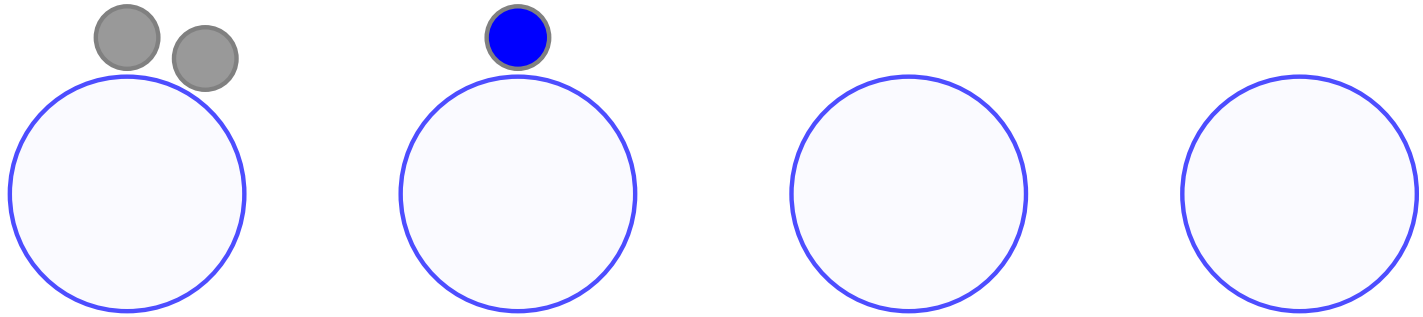



- Customer $\longrightarrow$ table mapping $\mathbf{z} = 1, 1$
- $\mathrm{P}(\mathbf{z}) = b/b \times (1-a)/(1+b)$

- In CRPs, probability of choosing a table $\propto$ number of customers $\Rightarrow$ strong *rich get richer* effect
- Pitman-Yor processes take mass $a$ from each occupied table and give it to the new table

$$\mathrm{P}(Z_{n+1} = k \mid \mathbf{z}) \quad \propto \quad \begin{cases} n_k(\mathbf{z}) - a & \text{if } k \leq m = \max(\mathbf{z}) \\ ma + b & \text{if } k = m+1 \end{cases}$$

# Pitman-Yor Process (3)

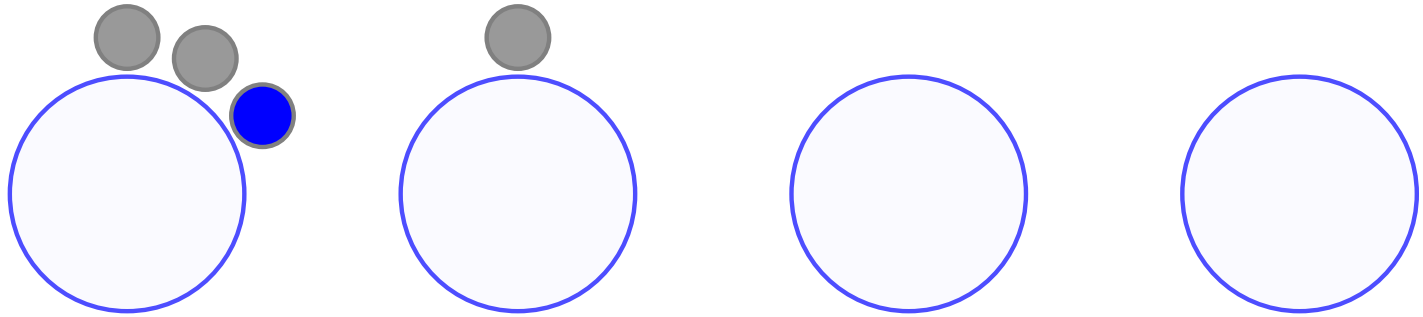



- Customer $\longrightarrow$ table mapping $\mathbf{z} = 1, 1, 2$
- $\mathrm{P}(\mathbf{z}) = b/b \times (1 - a)/(1 + b) \times (a + b)/(2 + b)$

- In CRPs, probability of choosing a table $\propto$ number of customers $\Rightarrow$ strong *rich get richer* effect
- Pitman-Yor processes take mass $a$ from each occupied table and give it to the new table

$$\mathrm{P}(Z_{n+1} = k \mid \mathbf{z}) \quad \propto \quad \begin{cases} n_k(\mathbf{z}) - a & \text{if } k \leq m = \max(\mathbf{z}) \\ ma + b & \text{if } k = m + 1 \end{cases}$$

# Pitman-Yor Process (4)

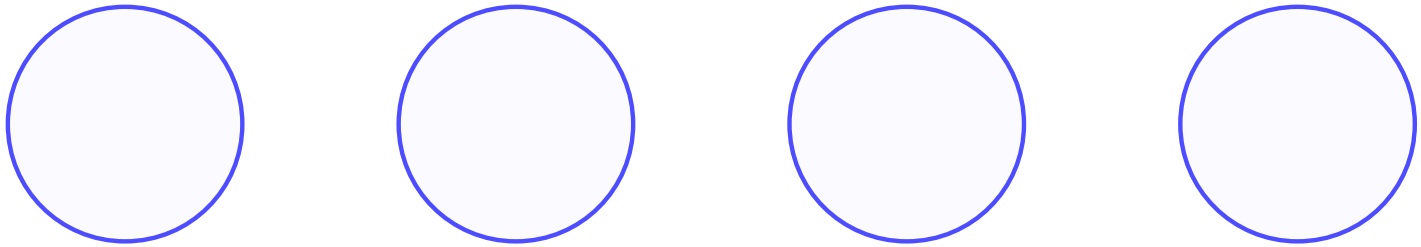



- Customer $\longrightarrow$ table mapping $\mathbf{z} = 1, 1, 2, 1$
- $\mathrm{P}(\mathbf{z}) = b/b \times (1-a)/(1+b) \times (a+b)/(2+b) \times (2-a)/(3+b)$

- In CRPs, probability of choosing a table $\propto$ number of customers $\Rightarrow$ strong *rich get richer* effect
- Pitman-Yor processes take mass $a$ from each occupied table and give it to the new table

$$\mathrm{P}(Z_{n+1} = k \mid \mathbf{z}) \quad \propto \quad \begin{cases} n_k(\mathbf{z}) - a & \text{if } k \leq m = \max(\mathbf{z}) \\ ma + b & \text{if } k = m+1 \end{cases}$$

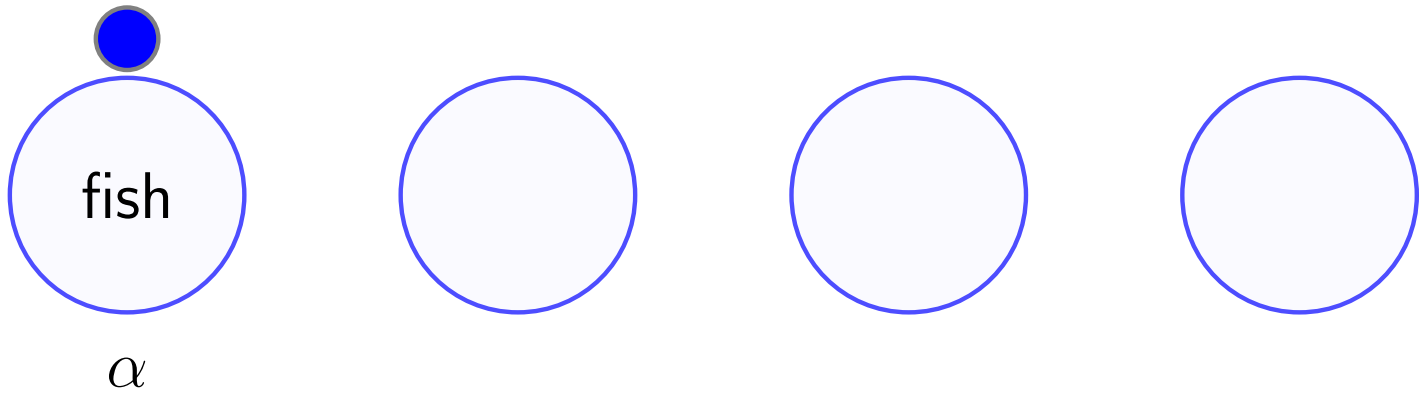
# Labeled Chinese Restaurant Process (0)



- Table $\longrightarrow$ label mapping $\mathbf{y} =$
- Customer $\longrightarrow$ table mapping $\mathbf{z} =$
- Output sequence $\mathbf{x} =$
- $\mathrm{P}(\mathbf{x}) = 1$

- *Base distribution* $\mathrm{P}_0(Y)$ generates a *label* $y_k$ for each table $k$
- All customers sitting at table $k$ (i.e., $z_i = k$) share label $y_k$
- Customer $i$ sitting at table $z_i$ has label $x_i = y_{z_i}$

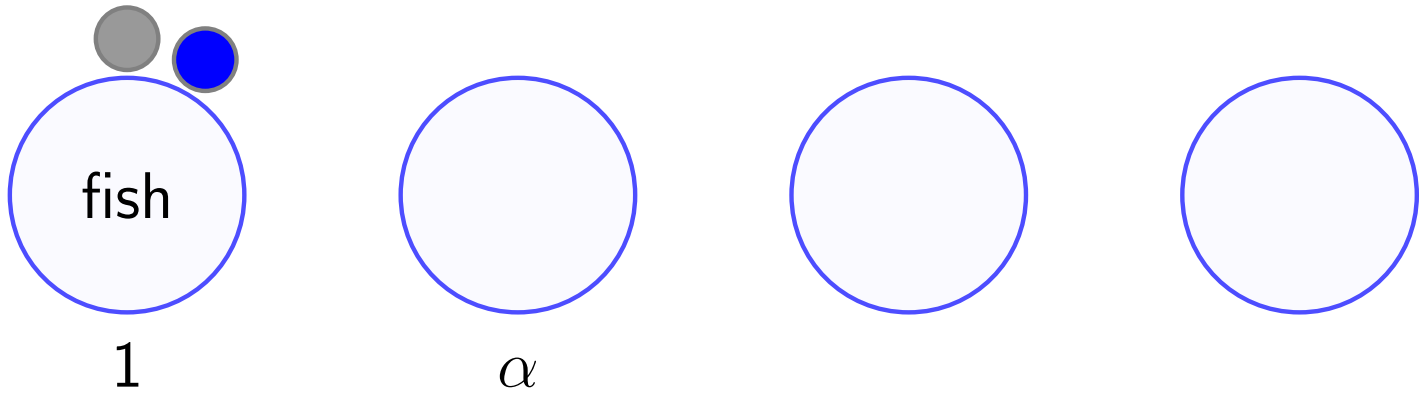
# Labeled Chinese Restaurant Process (1)




- Table $\rightarrow$ label mapping $\mathbf{y} =$ fish
- Customer $\rightarrow$ table mapping $\mathbf{z} = 1$
- Output sequence $\mathbf{x} =$ fish
- $\mathrm{P}(\mathbf{x}) = \alpha/\alpha \times \mathrm{P}_0(\text{fish})$

- *Base distribution* $\mathrm{P}_0(Y)$ generates a *label* $y_k$ for each table $k$
- All customers sitting at table $k$ (i.e., $z_i = k$) share label $y_k$
- Customer $i$ sitting at table $z_i$ has label $x_i = y_{z_i}$

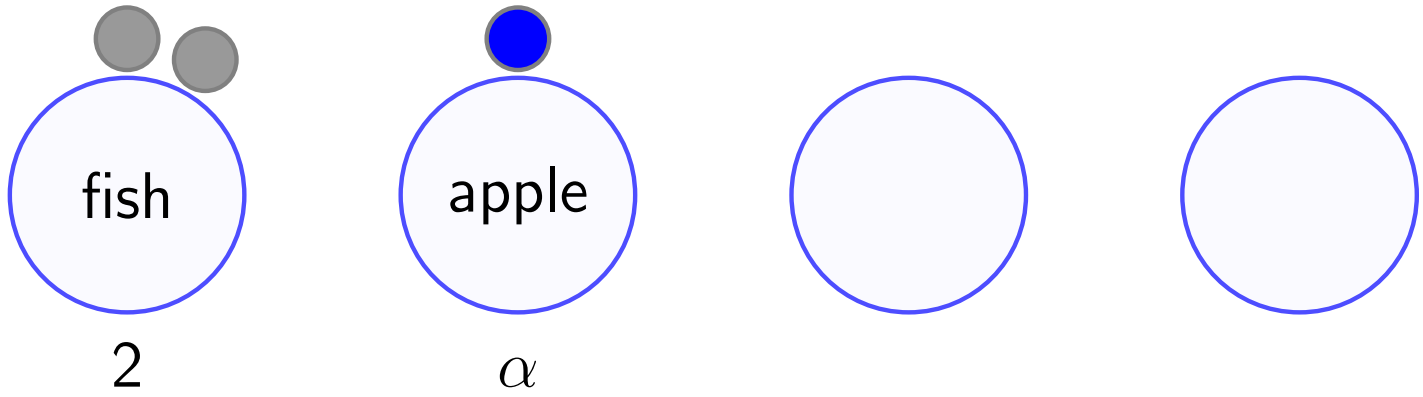
# Labeled Chinese Restaurant Process (2)




- Table $\rightarrow$ label mapping $\mathbf{y} =$ fish
- Customer $\rightarrow$ table mapping $\mathbf{z} = 1, 1$
- Output sequence $\mathbf{x} =$ fish,fish
- $\mathrm{P}(\mathbf{x}) = \mathrm{P}_0(\text{fish}) \times 1/(1+\alpha)$

- *Base distribution* $\mathrm{P}_0(Y)$ generates a *label* $y_k$ for each table $k$
- All customers sitting at table $k$ (i.e., $z_i = k$) share label $y_k$
- Customer $i$ sitting at table $z_i$ has label $x_i = y_{z_i}$

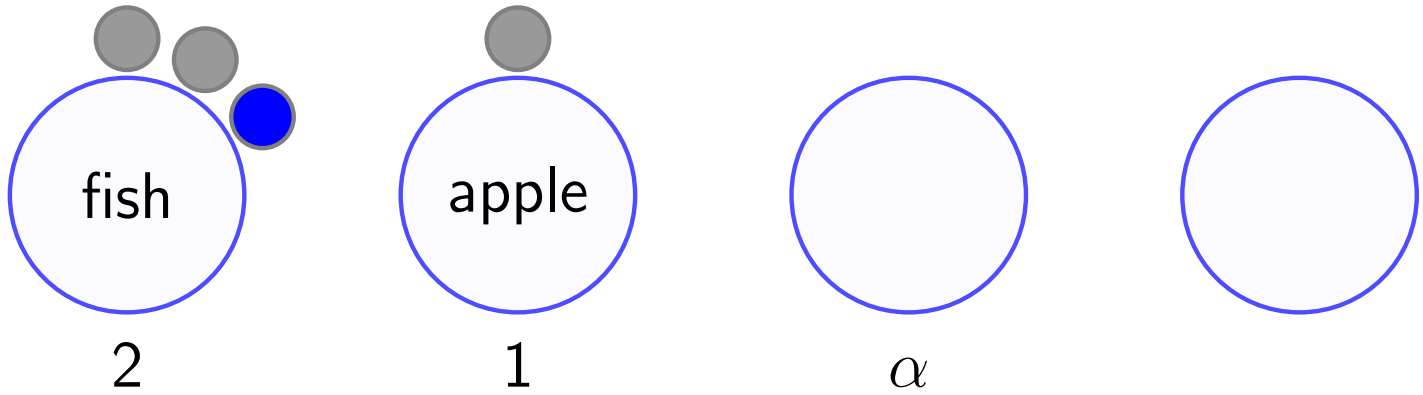
# Labeled Chinese Restaurant Process (3)




- Table $\rightarrow$ label mapping $\mathbf{y}$ = fish,apple
- Customer $\rightarrow$ table mapping $\mathbf{z} = 1, 1, 2$
- Output sequence $\mathbf{x}$ = fish,fish,apple
- $\mathrm{P}(\mathbf{x}) = \mathrm{P}_0(\text{fish}) \times 1/(1+\alpha) \times \alpha/(2+\alpha)\mathrm{P}_0(\text{apple})$

- *Base distribution* $\mathrm{P}_0(Y)$ generates a *label* $y_k$ for each table $k$
- All customers sitting at table $k$ (i.e., $z_i = k$) share label $y_k$
- Customer $i$ sitting at table $z_i$ has label $x_i = y_{z_i}$

# Labeled Chinese Restaurant Process (4)




- Table $\rightarrow$ label mapping $\mathbf{y} =$ fish,apple
- Customer $\rightarrow$ table mapping $\mathbf{z} = 1, 1, 2$
- Output sequence $\mathbf{x} =$ fish,fish,apple,fish
- $\mathrm{P}(\mathbf{x}) = \mathrm{P}_0(\text{fish}) \times 1/(1+\alpha) \times \alpha/(2+\alpha)\mathrm{P}_0(\text{apple}) \times 2/(3+\alpha)$

- *Base distribution* $\mathrm{P}_0(Y)$ generates a *label* $y_k$ for each table $k$
- All customers sitting at table $k$ (i.e., $z_i = k$) share label $y_k$
- Customer $i$ sitting at table $z_i$ has label $x_i = y_{z_i}$

# Summary: Chinese Restaurant Processes and Pitman-Yor Processes

- *Chinese Restaurant Processes* (CRPs) generalize Dirichlet-Multinomials to an *unbounded number of outcomes*
  - *concentration parameter* $\alpha$ controls how likely a new outcome is
  - CRPs exhibit a *rich get richer* power-law behaviour
- *Pitman-Yor Processes* (PYPs) generalize CRPs by adding an additional parameter (each PYP has *a* and *b* parameters)
  - PYPs can describe a wider range of distributions than CRPs
- *Labeled CRPs and PYPs* use a *base distribution* to label each table
  - base distribution can have *infinite support*
  - concentrates mass on a countable subset

# Labeled Chinese restaurants and Dirichlet processes

- A labeled Chinese restaurant processes maps a *base distribution* $\mathsf{P}_B$ 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 $\alpha$ and $\mathsf{P}_B$ to a *distribution over distributions* $\mathrm{DP}(\alpha, \mathsf{P}_B)$
- $\mathrm{DP}(\alpha, \mathsf{P}_B)$ is called a *Dirichlet process* (DP) with *concentration parameter* $\alpha$ and *base distribution* $\mathsf{P}_B$
- The base distribution $\mathsf{P}_B$ can itself be defined by 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., $S_{35} \rightarrow NP_{27}\ VP_{17}$
    - $\Rightarrow$ 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

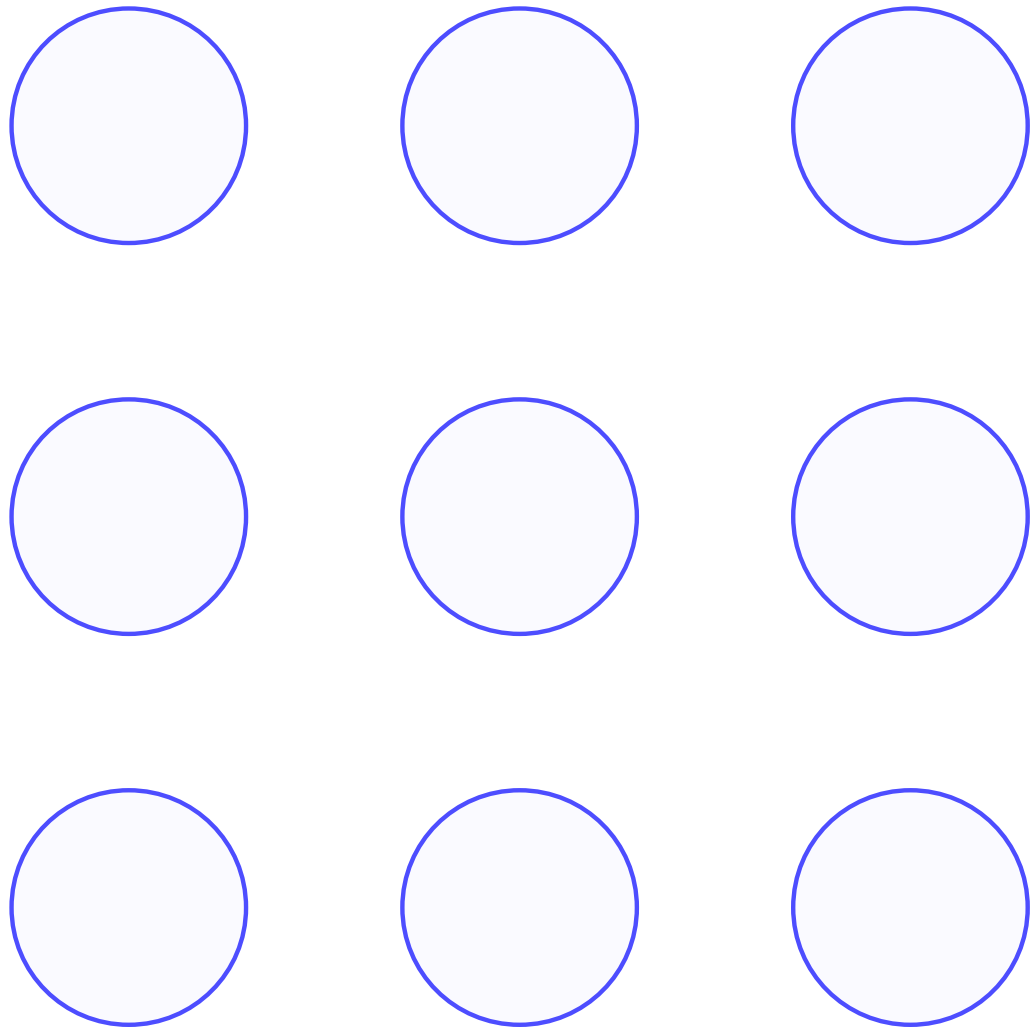
- The trees generated by an adaptor grammar are defined by CFG rules as in a CFG
- A subset of the nonterminals are *adapted*
- *Unadapted nonterminals* expand by picking a rule and recursively expanding its children, as in a PCFG
- *Adapted nonterminals* can expand in two ways:
  - by picking a rule and recursively expanding its children, or
  - by generating a previously generated tree (with probability proportional to the number of times previously generated)
- Implemented by having a CRP/PYP for each adapted nonterminal
- The CFG rules of the adapted nonterminals determine the *base distributions* of these CRPs/PYPs

# Adaptor grammar for stem-suffix morphology (0)

Word $\rightarrow$ Stem Suffix

Stem $\rightarrow$ Phoneme$^{+}$

Suffix $\rightarrow$ Phoneme$^{\star}$



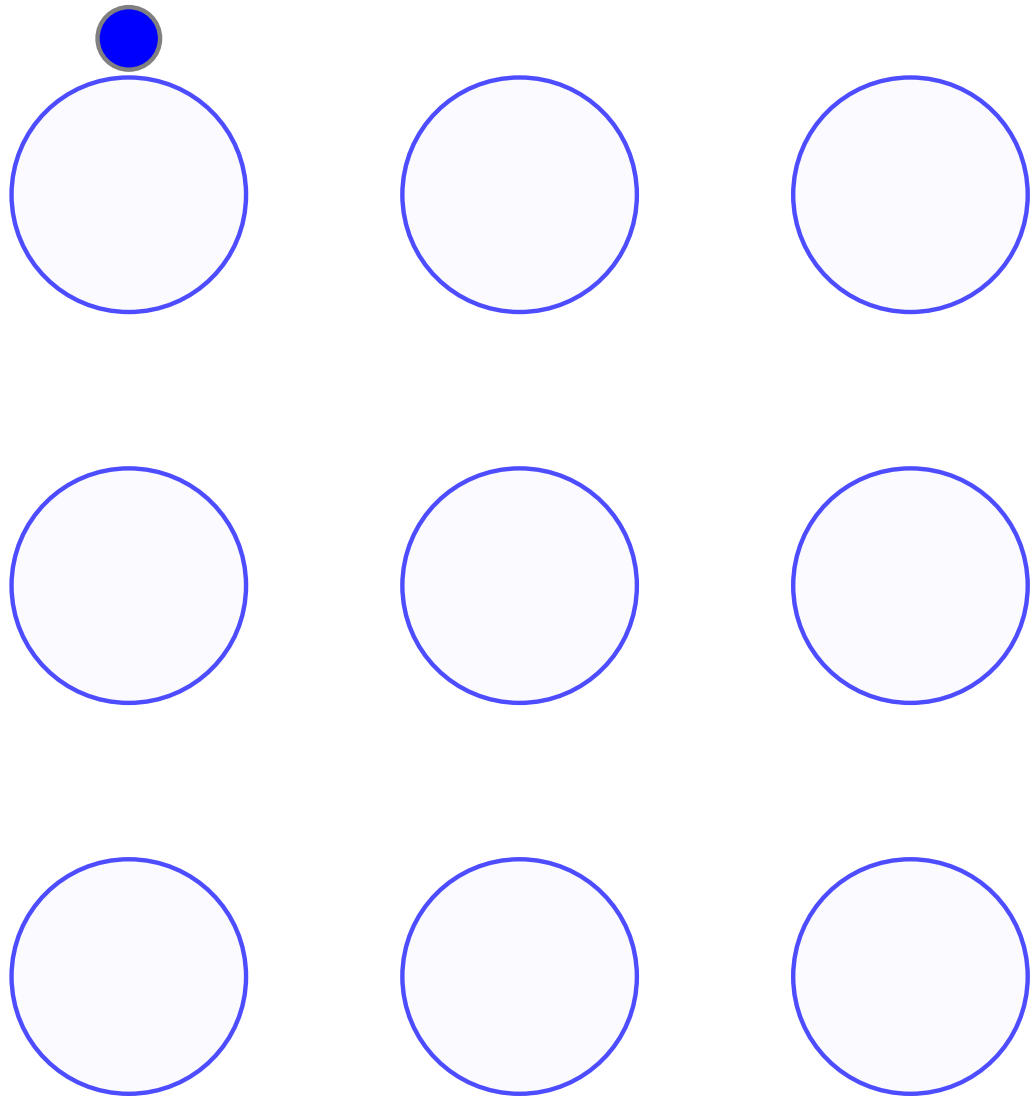
Generated words:

# Adaptor grammar for stem-suffix morphology (1a)

Word $\rightarrow$ Stem Suffix

Stem $\rightarrow$ Phoneme$^{+}$

Suffix $\rightarrow$ Phoneme$^{\star}$



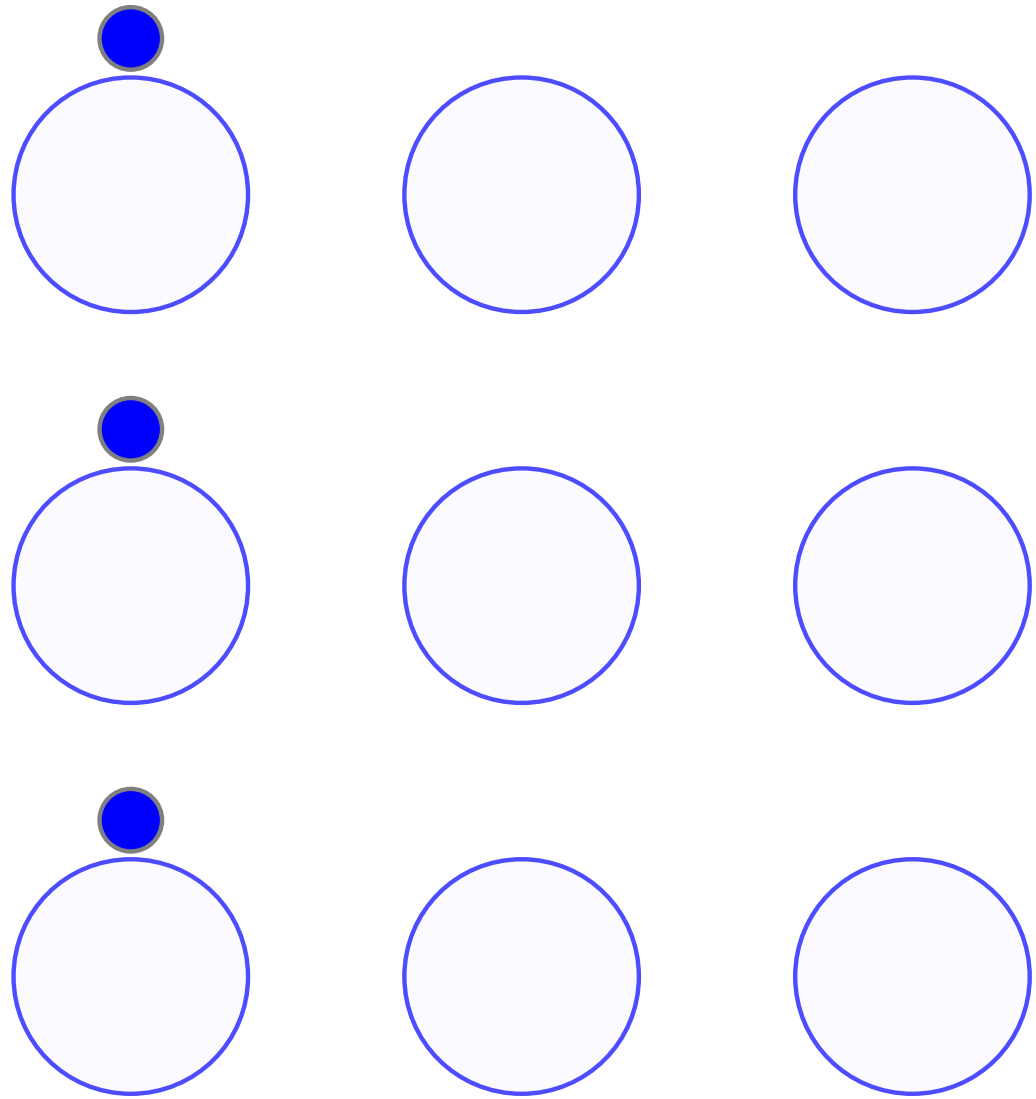
Generated words:

# Adaptor grammar for stem-suffix morphology (1b)

Word → Stem Suffix

Stem → Phoneme$^{+}$

Suffix → Phoneme$^{\star}$



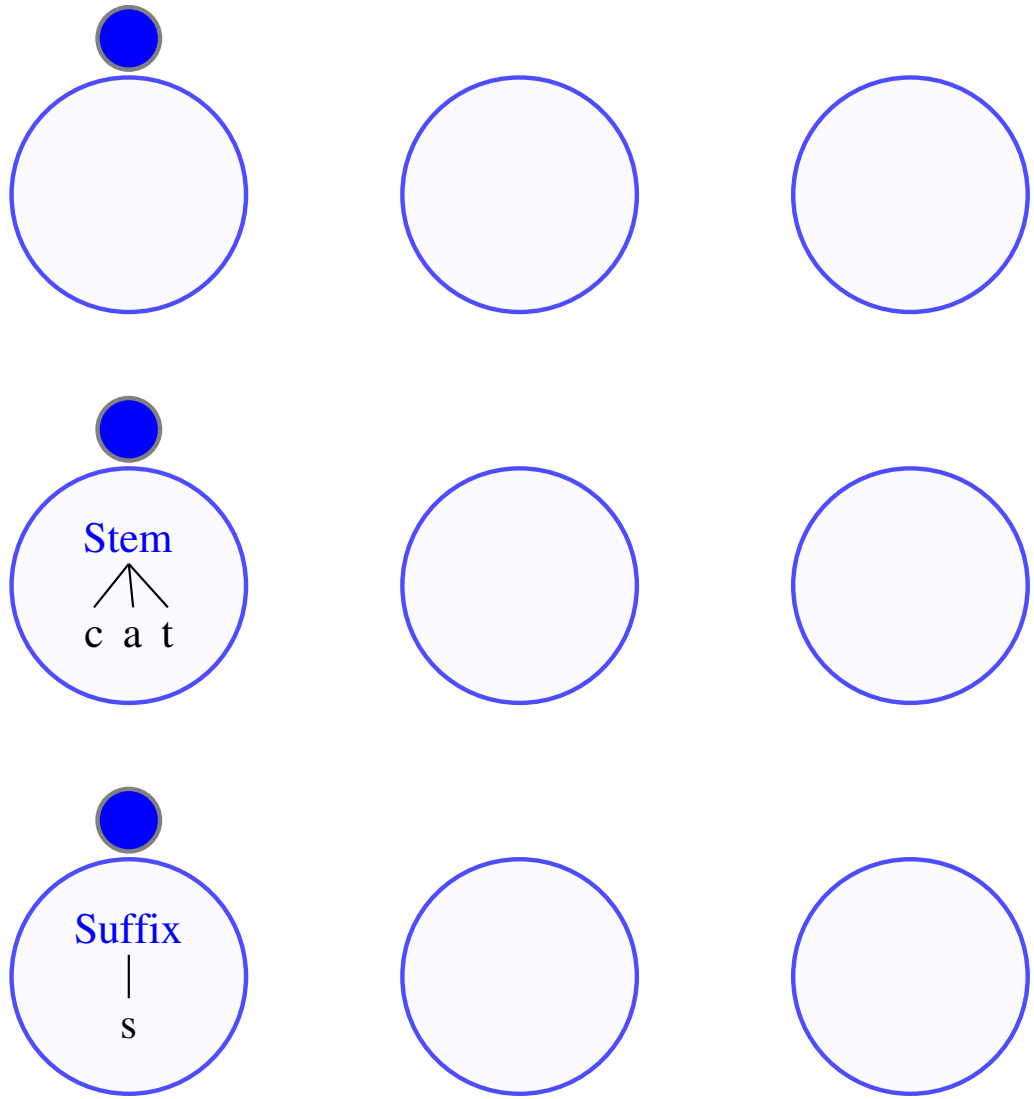
Generated words:

# Adaptor grammar for stem-suffix morphology (1c)

Word → Stem Suffix

Stem → Phoneme$^{+}$

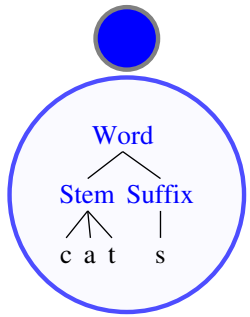
Suffix → Phoneme$^{\star}$



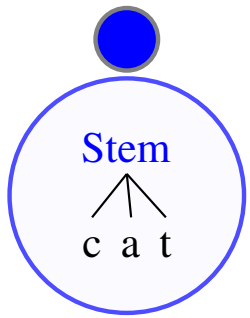

Generated words:

# Adaptor grammar for stem-suffix morphology (1d)

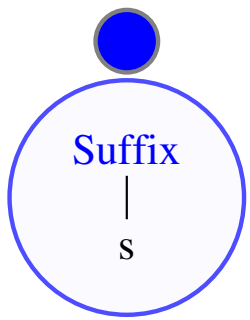
Word → Stem Suffix




Stem → Phoneme$^{+}$




Suffix → Phoneme$^{\star}$



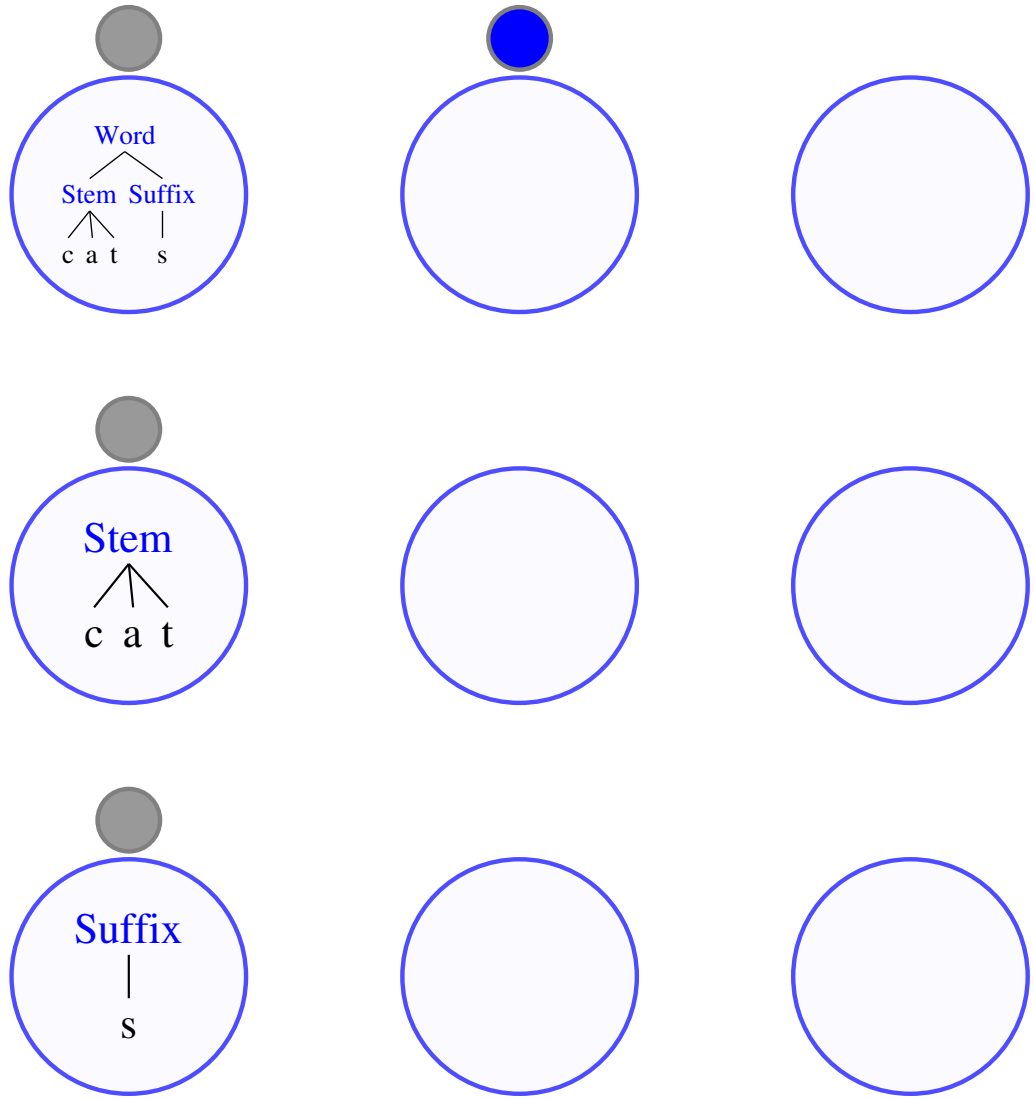

Generated words: cats

# Adaptor grammar for stem-suffix morphology (2a)

Word → Stem Suffix

Stem → Phoneme$^{+}$

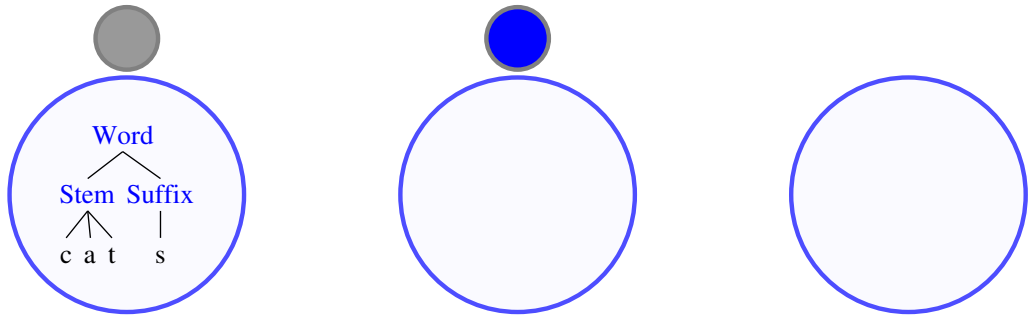
Suffix → Phoneme$^{\star}$



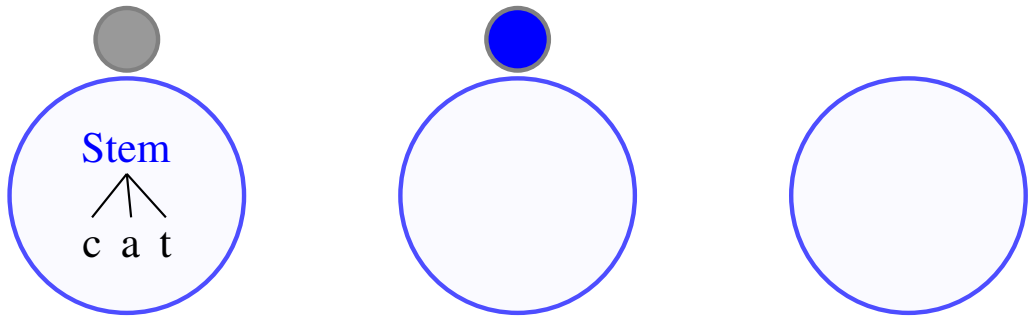

Generated words: cats

# Adaptor grammar for stem-suffix morphology (2b)

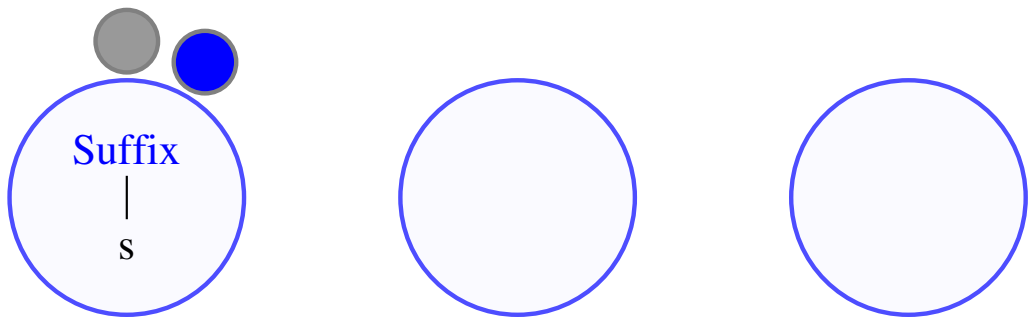
Word → Stem Suffix




Stem → Phoneme$^{+}$




Suffix → Phoneme$^{\star}$



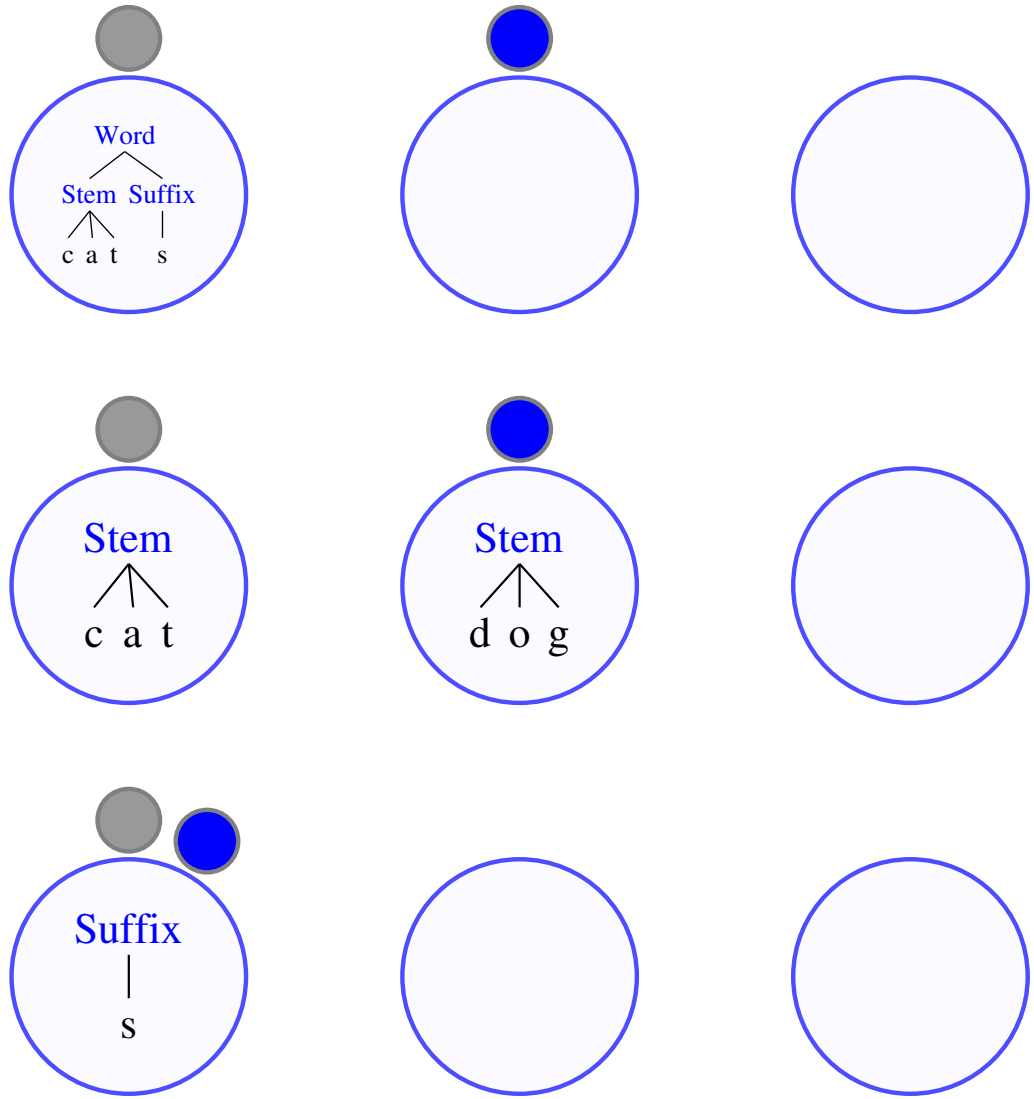

Generated words: cats

# Adaptor grammar for stem-suffix morphology (2c)

Word → Stem Suffix

Stem → Phoneme$^{+}$

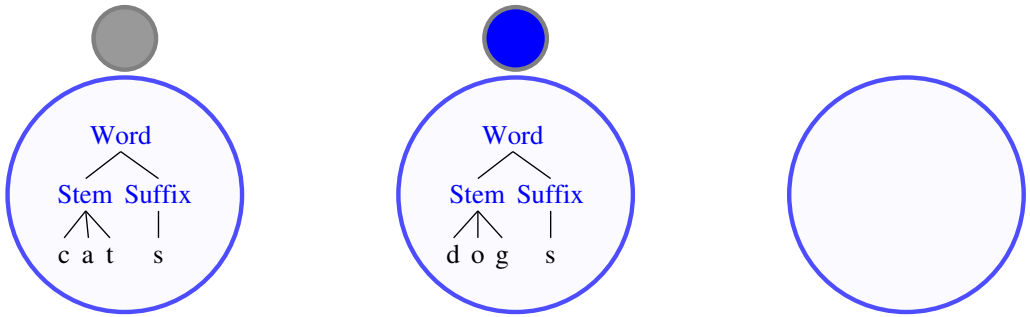
Suffix → Phoneme$^{\star}$



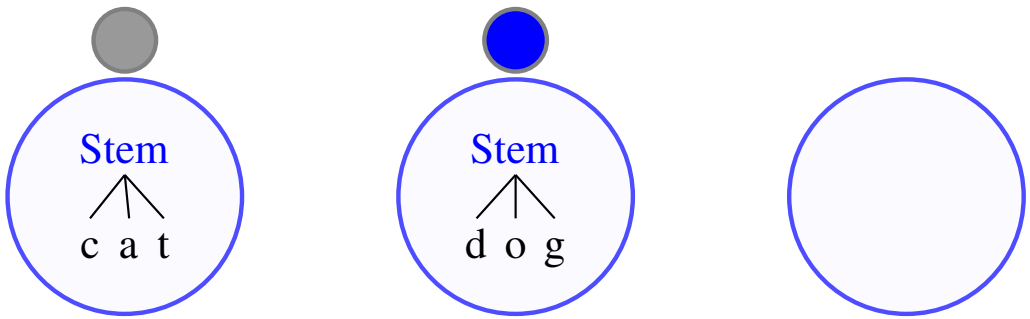

Generated words: cats

# Adaptor grammar for stem-suffix morphology (2d)

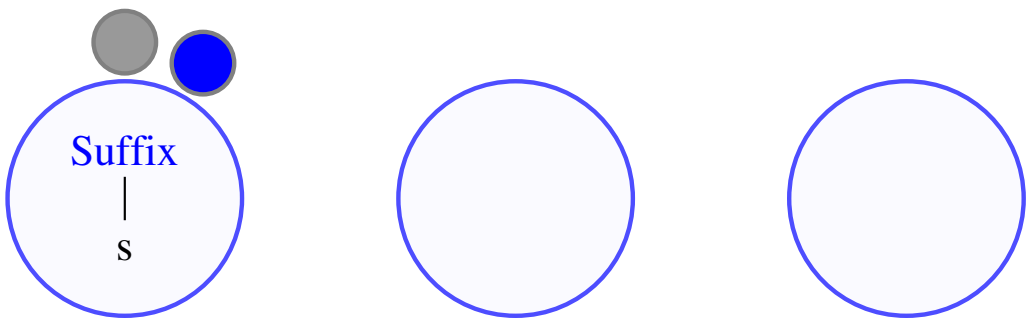
Word → Stem Suffix




Stem → Phoneme$^{+}$



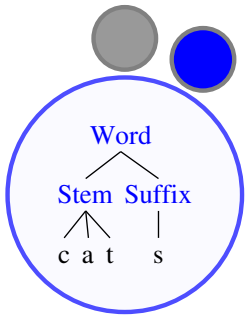

Suffix → Phoneme$^{\star}$

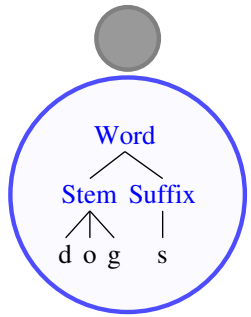


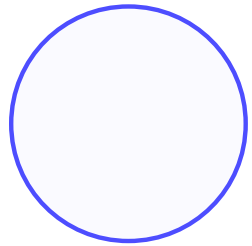

Generated words: cats, dogs

# Adaptor grammar for stem-suffix morphology (3)

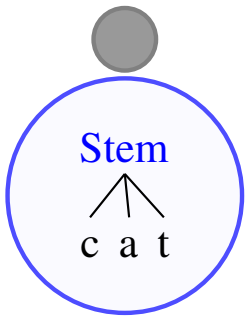
Word → Stem Suffix

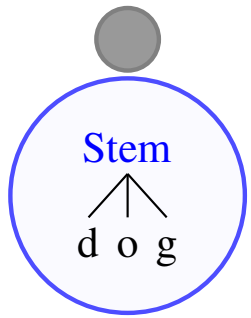


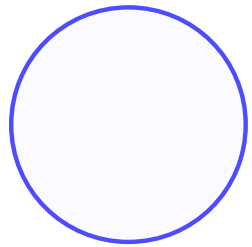



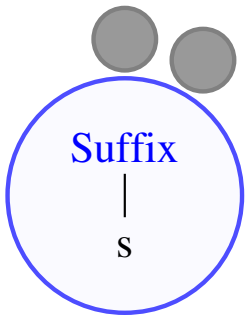
Stem → Phoneme$^{+}$

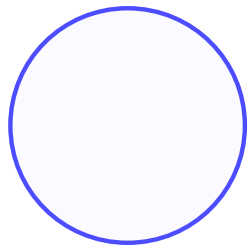


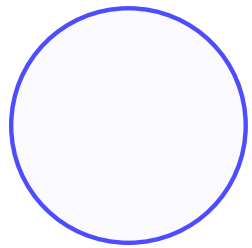



Suffix → Phoneme$^{\star}$





Generated words: cats, dogs, cats

# 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* (important for sampling)
- An *unadapted nonterminal* $A$ expands using $A \rightarrow \beta$ with probability $\theta_{A \rightarrow \beta}$
- Each adapted nonterminal $A$ is associated with a CRP (or PYP) that caches previously generated subtrees rooted in $A$
- An *adapted nonterminal* $A$ expands:
  - to a subtree $\tau$ rooted in $A$ with probability proportional to the number of times $\tau$ was previously generated
  - using $A \rightarrow \beta$ with probability proportional to $\alpha_A \theta_{A \rightarrow \beta}$

# Properties of adaptor grammars

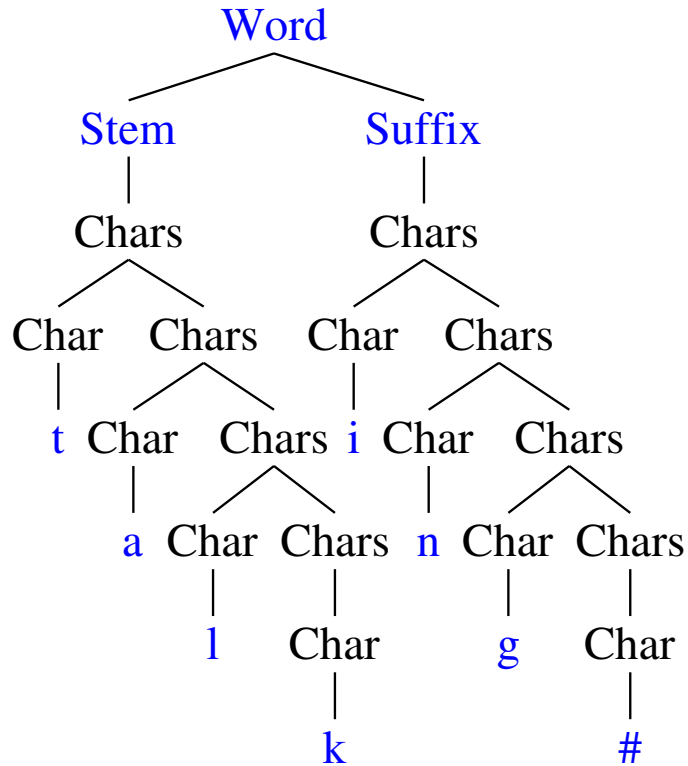
- Possible trees generated by CFG rules
  but the probability of each adapted tree is estimated separately
- Probability of a subtree $\tau$ is proportional to:
  - the number of times $\tau$ was seen before
    $\Rightarrow$ "rich get richer" dynamics (Zipf distributions)
  - plus $\alpha_A$ times prob.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



# Outline

# Unsupervised word segmentation

- Input: phoneme sequences with *sentence boundaries* (Brent)
- Task: identify *word boundaries*, and hence words

$$\mathrm{y}\,\triangle\,\mathrm{u}\,\blacktriangle\,\mathrm{w}\,\triangle\,\mathrm{a}\,\triangle\,\mathrm{n}\,\triangle\,\mathrm{t}\,\blacktriangle\,\mathrm{t}\,\triangle\,\mathrm{u}\,\blacktriangle\,\mathrm{s}\,\triangle\,\mathrm{i}\,\blacktriangle\,\mathrm{D}\,\triangle\,6\,\blacktriangle\,\mathrm{b}\,\triangle\,\mathrm{U}\,\triangle\,\mathrm{k}$$

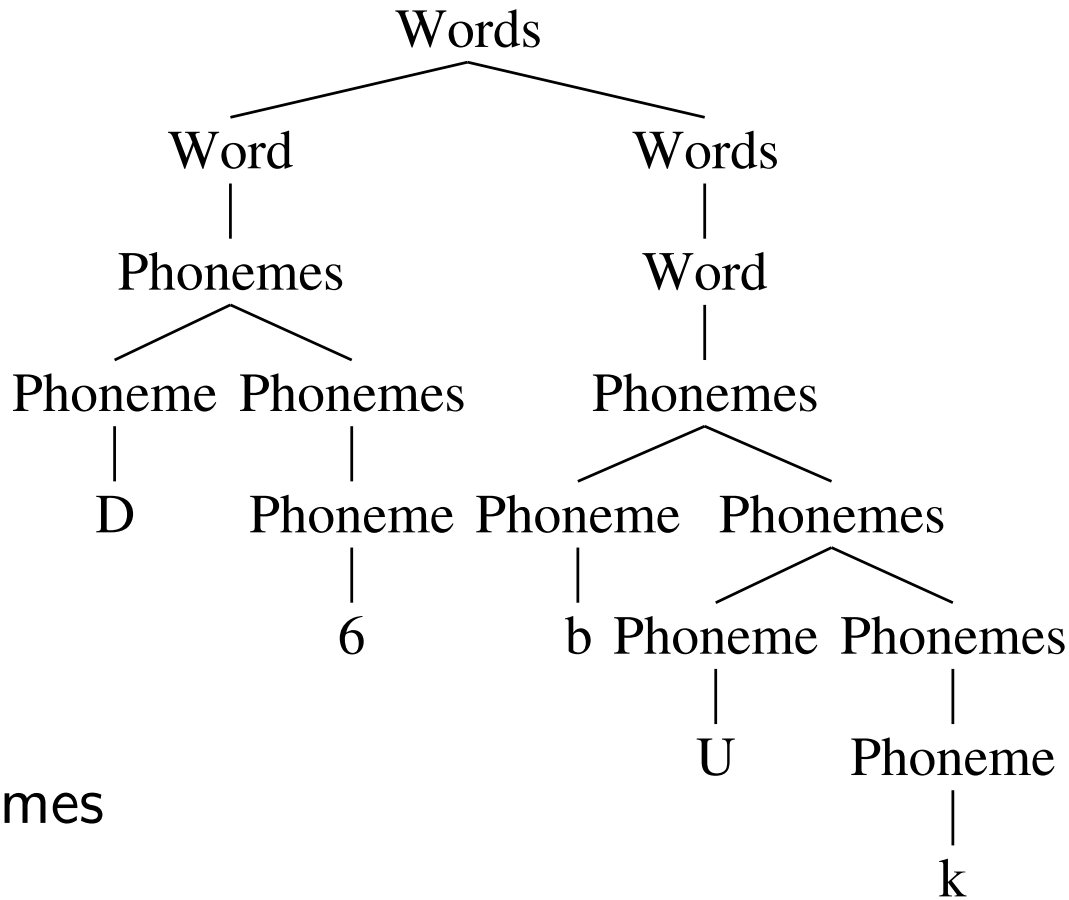
- Useful cues for word segmentation:
  - Phonotactics (Fleck)
  - Inter-word dependencies (Goldwater)

# Word segmentation with PCFGs (1)

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

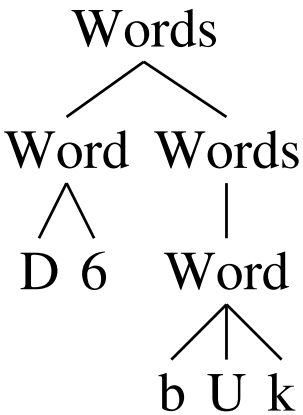
which abbreviates

$$\text{Sentence} \rightarrow \text{Words}$$
$$\text{Words} \rightarrow \text{Word Words}$$
$$\text{Word} \rightarrow \text{Phonemes}$$
$$\text{Phonemes} \rightarrow \text{Phoneme Phonemes}$$
$$\text{Phonemes} \rightarrow \text{Phoneme}$$
$$\text{Phoneme} \rightarrow a \mid \ldots \mid z$$



# Word segmentation with PCFGs (1)

Sentence $\longrightarrow$ Word$^+$
Word $\longrightarrow$ all possible phoneme strings



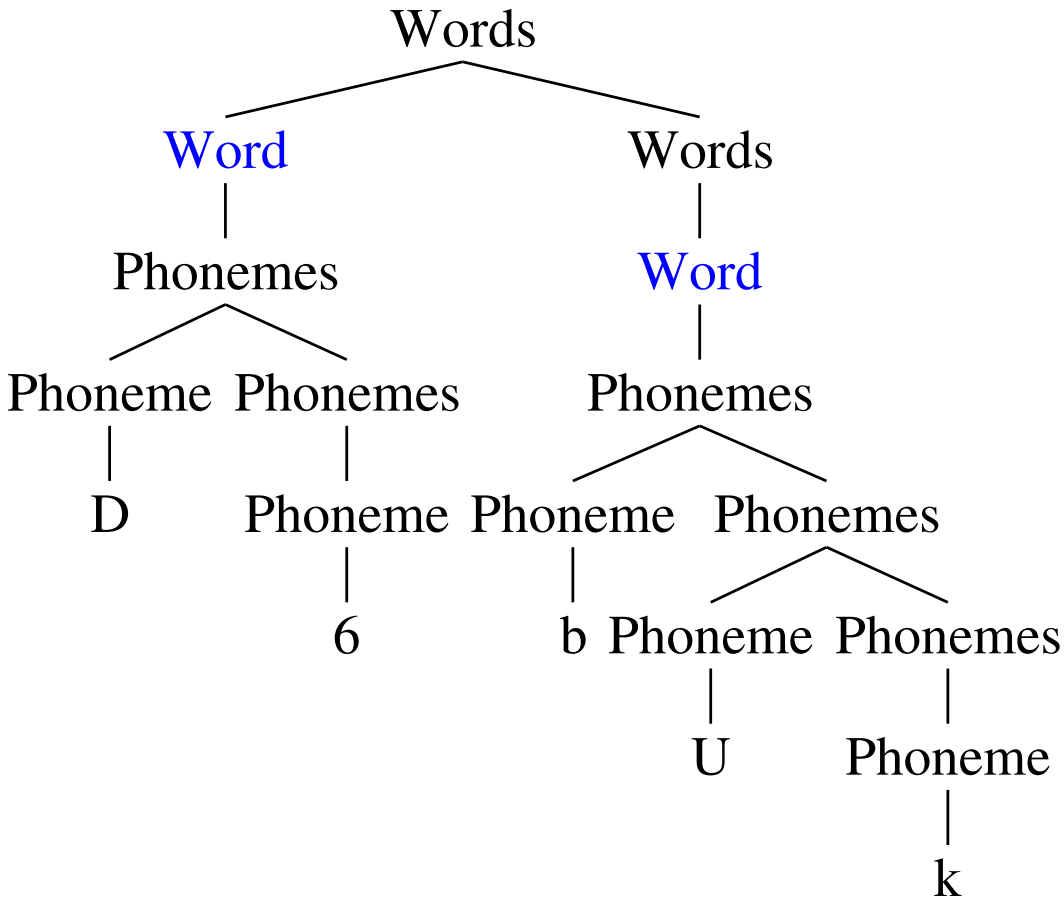

- But now there are an infinite number of PCFG rules!
  - once we see our (finite) training data, only finitely many are useful

  $\Rightarrow$ the set of parameters (rules) should be chosen based on training data

# Unigram word segmentation adaptor grammar

Sentence $\rightarrow$ Word$^+$
Word $\rightarrow$ Phoneme$^+$

- *Adapted nonterminals* indicated by underlining




- Adapting Words means that the grammar learns the probability of each Word subtree independently
- Unigram word segmentation on Brent corpus: 56% token f-score

# Unigram adaptor grammar after learning

- Given the Brent corpus and the unigram adaptor grammar

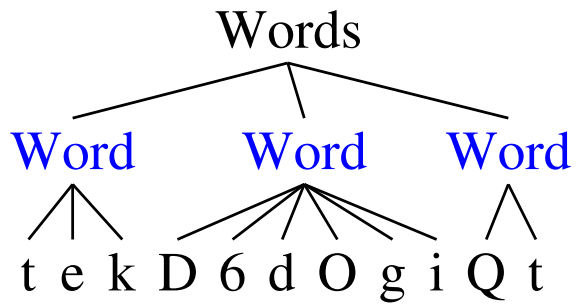
$$\text{Words} \rightarrow \text{Word}^+$$
$$\underline{\text{Word}} \rightarrow \text{Phon}^+$$

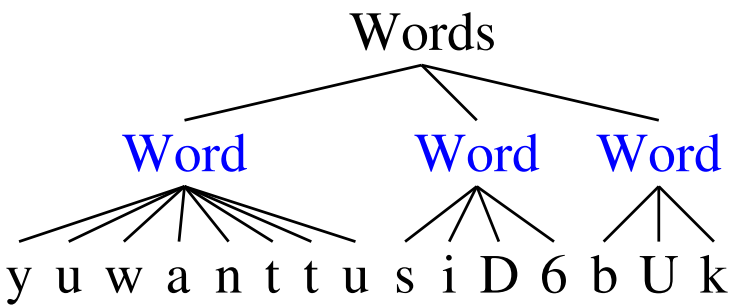
the learnt adapted grammar contains 1,712 rules such as:

| | |
|---|---|
| 15758 | Words $\rightarrow$ Word Words |
| 9791 | Words $\rightarrow$ Word |
| 1660 | Word $\rightarrow$ Phon$^+$ |
| 402 | Word $\rightarrow$ *y u* |
| 137 | Word $\rightarrow$ *I n* |
| 111 | Word $\rightarrow$ *w I T* |
| 100 | Word $\rightarrow$ *D 6 d O g i* |
| 45 | Word $\rightarrow$ *I n D 6* |
| 20 | Word $\rightarrow$ *I n D 6 h Q s* |

# unigram: Words

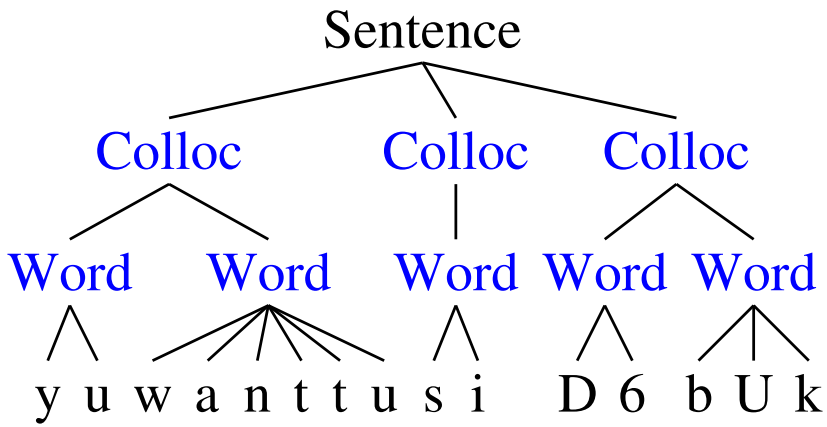
- Unigram word segmentation model assumes each word is generated independently
- But there are strong inter-word dependencies (collocations)
- Unigram model can only capture such dependencies by analyzing collocations as words (Goldwater 2006)



# colloc: Collocations ⇒ Words

$$\text{Sentence} \rightarrow \text{Colloc}^+$$
$$\underline{\text{Colloc}} \rightarrow \text{Word}^+$$
$$\underline{\text{Word}} \rightarrow \text{Phon}^+$$

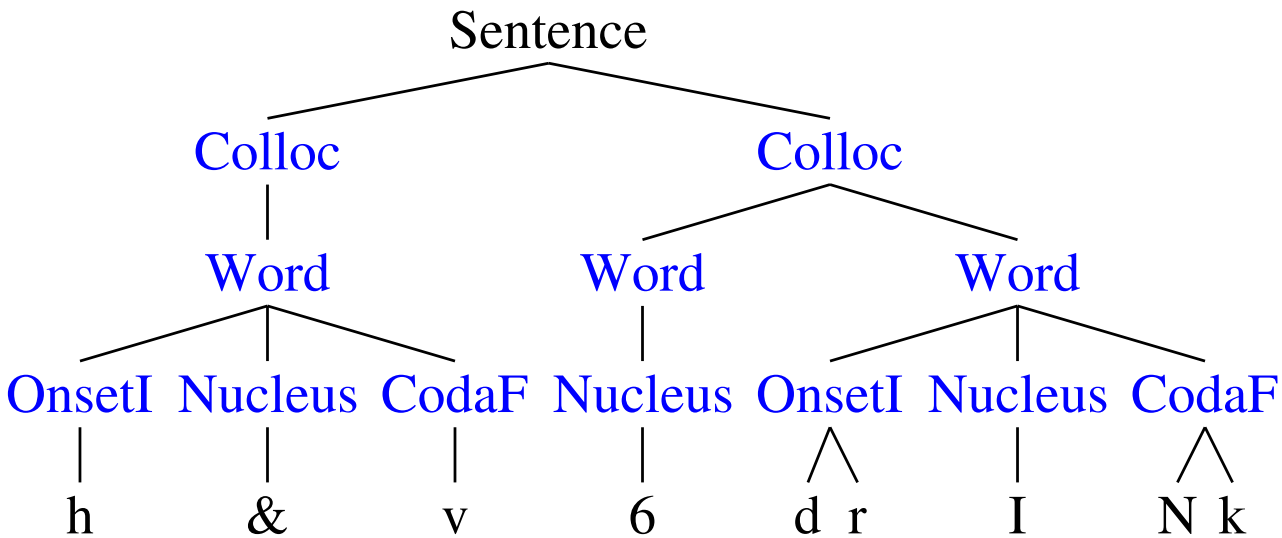



- A Colloc(ation) consists of one or more words
- Both Words and Collocs are adapted (learnt)
- Significantly improves word segmentation accuracy over unigram model (76% f-score; ≈ Goldwater's bigram model)

# colloc-syll: Collocations ⇒ Words ⇒ Syllables

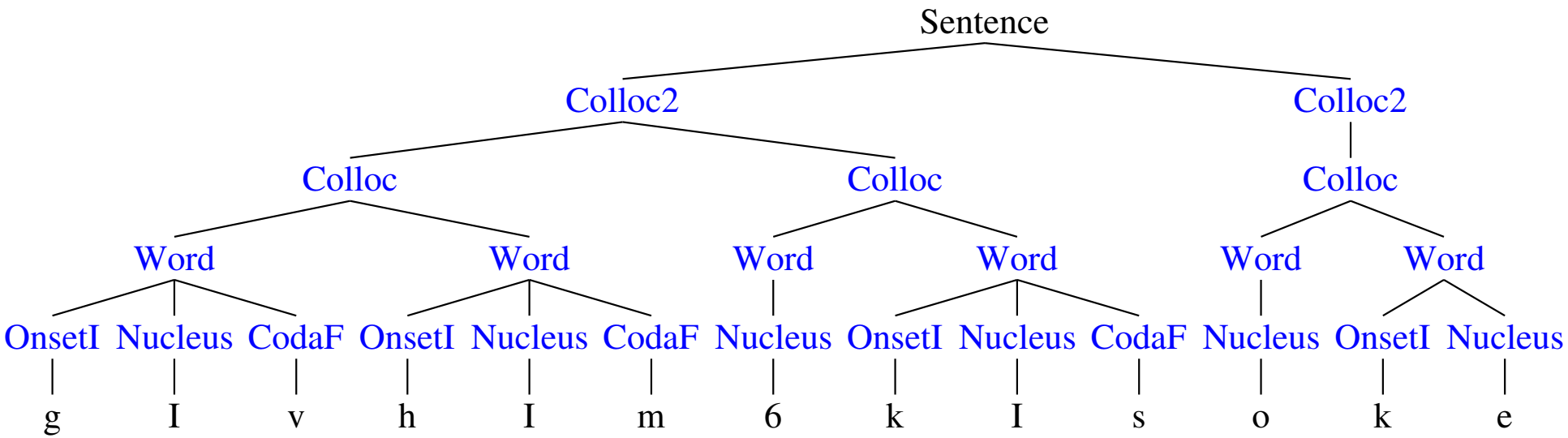
Sentence → Colloc$^+$
Word → SyllableIF
Word → SyllableI (Syllable) (Syllable) SyllableF
Onset → Consonant$^+$
Nucleus → Vowel$^+$

Colloc → Word$^+$
Syllable → (Onset) Rhyme
Rhyme → Nucleus (Coda)
Coda → Consonant$^+$




- With 2 Collocation levels, f-score = 87%

# colloc-syll: Collocations ⇒ Words ⇒ Syllables



# Another application of adaptor grammars: Learning structure in names

- Many different kinds of names
  - Person names, e.g., *Mr. Sam Spade Jr.*
  - Company names, e.g., *United Motor Manufacturing Corp.*
  - Other names, e.g., *United States of America*
- At least some of these are structured; e.g., *Mr* is an honorific, *Sam* is first name, *Spade* is a surname, etc.
- Penn treebanks assign flat structures to base NPs (including names)
- Data set: 10,787 unique lowercased sequences of base NP proper nouns, containing 23,392 words
- Can we automatically learn the structure of these names?

# Adaptor grammar for names

$\text{NP} \rightarrow \text{Unordered}^+$
$\text{NP} \rightarrow (\text{A0})\ (\text{A1})\ \ldots\ (\text{A6})$
$\underline{\text{A0}} \rightarrow \text{Word}^+$
$\ldots$
$\underline{\text{A6}} \rightarrow \text{Word}^+$

$\underline{\text{Unordered}} \rightarrow \text{Word}^+$
$\text{NP} \rightarrow (\text{B0})\ (\text{B1})\ \ldots\ (\text{B6})$
$\underline{\text{B0}} \rightarrow \text{Word}^+$
$\ldots$
$\underline{\text{B6}} \rightarrow \text{Word}^+$

- *Warning: hand-selected output, no evaluation!*

(A0 barrett) (A3 smith)
(A0 albert) (A2 j.) (A3 smith) (A4 jr.)
(A0 robert) (A2 b.) (A3 van dover)
(B0 aim) (B1 prime rate) (B2 plus) (B5 fund) (B6 inc.)
(B0 balfour) (B1 maclaine) (B5 international) (B6 ltd.)
(B0 american express) (B1 information services) (B6 co)
(U abc) (U sports)
(U sports illustrated)
(U sports unlimited)

# Outline

# Estimating adaptor grammars

- Need to estimate:
  - parse trees $\mathbf{t} = (t_1, \ldots, t_n)$ for strings $\mathbf{x} = (x_1, \ldots, x_n)$
  - cached subtrees $\tau$ for adapted nonterminals
  - (optional) DP parameters $\boldsymbol{\alpha}$ for adapted nonterminals
  - (optional) probabilities $\boldsymbol{\theta}$ of base grammar rules
- Component-wise Metropolis-within-Gibbs sampler for parse trees
  - sample parse tree $t_j$ from $\mathrm{P}(T \mid x_j, \mathbf{t}_{-j})$
  - sampling directly from conditional distribution of parses seems intractable
  - construct PCFG *proposal grammar* $G(\mathbf{t}_{-j})$ on the fly
  - each table label $\tau$ corresponds to a production in PCFG approximation
  - Use accept/reject to convert samples from PCFG approx to samples from adaptor grammar

# Component-wise Metropolis-within-Gibbs sampling

- Observations (terminal strings) $\mathbf{x} = (x_1, \ldots, x_n)$
  Hidden labels (parse trees) $\mathbf{t} = (t_1, \ldots, t_n)$
  Probabilistic model (adaptor grammar) $\mathrm{P}(\mathbf{x}, \mathbf{t})$
- Metropolis-within-Gibbs sampling algorithm:

  initialize $\mathbf{t}$ somehow (e.g., random trees)
  repeat forever:
  - pick an index $j \in 1, \ldots, n$ at random
  - construct PCFG proposal grammar $G(\mathbf{t}_{-j})$ on the fly
    replace $t_j$ with a random sample from $\mathrm{P}(T \mid x_j, \mathbf{t}_{-j})$
    where $\mathbf{t}_{-j} = (t_1, \ldots, t_{j-1}, t_{j+1}, \ldots, t_n)$
- After *burn-in* the samples $\mathbf{t}$ are distributed according to $\mathrm{P}(\mathbf{T} \mid \mathbf{x})$

# Metropolis-with-Gibbs sampler

- Collapsed Gibbs sampler: resample parse $T_j$ given $w_j$ and $\mathbf{t}_{-j}$
- Table counts change within a parse tree
  - $\Rightarrow$ grammar is not context-free
  - $\Rightarrow$ breaks standard dynamic programming
  - $\Rightarrow$ Metropolis accept/reject for each Gibbs sample
- PCFG can express probability of selecting a next table given $\mathbf{t}_{-j}$
  - ignores changing table counts within single parse
- Rules of PCFG proposal grammar $G(\mathbf{t}_{-j})$ 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 $\tau$ in $A$'s restaurant: $\theta'_{A \rightarrow \textsc{Yield}(\tau)} \propto n_\tau$, the number of customers at table $\tau$
- Parses of $G'$ can be mapped back to adaptor grammar parses

# Bayesian inference for PYP parameters

- Adaptor grammars have 1 (CRP) or 2 (PYP) hyper-parameters for each adapted non-terminal $X$
- Previous work used CRP adaptors with *tied parameters*
- Bayesian prior: for each adapted nonterminal $X$

$$\begin{aligned} a_X &\sim Beta(1, 1) \\ b_X &\sim Gamma(10, 0.1) \end{aligned}$$

  - $Gamma(10, 0.1)$ is a vague Gamma prior (MacKay 2003)
  - permits $a_X$ and $b_X$ to *vary with adapted nonterminal $X$*
- Estimate with *slice sampling* (no proposal distribution; Neal 2003)
- *Biggest improvement on complex models*, e.g., colloc-syll:
  - tied parameters 78% token f-score
  - $a_X = 0$, sampling $b_X$ (i.e., CRP adaptors) 84% token f-score
  - sampling $a_X$ and $b_X$ (i.e., PYP adaptors) 87% token f-score

# Finding the modal word segmentation

- Previous work decoded using last sampled trees (2,000 epochs)
- After burn-in, samples are distributed according to $P(\mathbf{t} \mid \mathbf{x})$
  $\Rightarrow$ use samples to identify *modal word segmentation*
- Modal decoding:
  - For each sentence $x_i$ collect *sample parses* $\mathbf{s}_i = (s_i^{(1)}, \ldots, s_i^{(800)})$
    (every 10th epoch from epochs 1,000–2,000 from 8 runs)
  - Compute *word segmentations* $\mathbf{w}_i = (w_i^{(1)}, \ldots, w_i^{(800)})$ from parses
  - Compute *modal segmentation* $\widehat{w}_i = \mathit{argmax}_w \, n_w(\mathbf{w}_i)$,
    where $n_w(\mathbf{w}_i)$ is the number of times $w$ appears in $\mathbf{w}_i$
- Improves word segmentation token f-score in all models

| Model | Average | Max-modal |
|---|---|---|
| unigram | 55% | 56% |
| colloc | 74% | 76% |
| colloc-syll | 85% | 87% |

- Goodman (1998) max-marginal decoding should also be possible

# Random vs incremental initialization

- The Gibbs sampler parse trees $\mathbf{t}$ needs to be initialized somehow

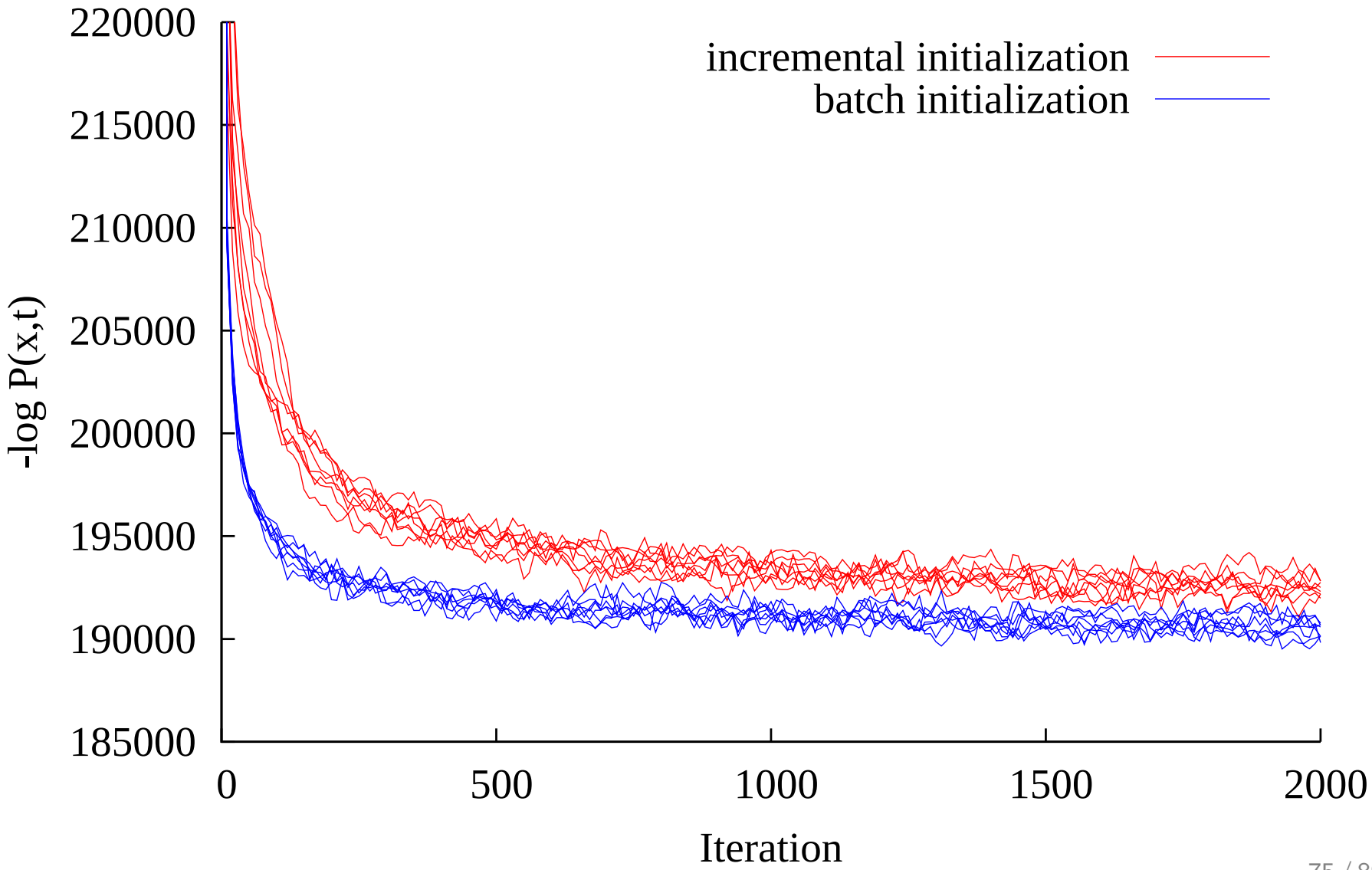
  Random initialization: Assign each string $x_i$ a random parse $t_i$ generated by base PCFG

  Incremental initialization: Sample $t_i$ from $\mathrm{P}(t \mid x_i, \mathbf{t}_{1:i-1})$
- Incremental initialization is easy to implement in a Gibbs sampler
- Incremental initialization improves token f-score in all models, especially on simple models

| Model | Random | Incremental |
|---|---|---|
| unigram | 56% | 81% |
| colloc | 76% | 86% |
| colloc-syll | 87% | 89% |

*but see caveats on next slide!*

# Incremental initialization produces low-probability parses



# Why incremental initialization produces low-probability parses

- Incremental initialization produces sample parses $\mathbf{t}$ with lower probability $P(\mathbf{t} \mid \mathbf{x})$
- Possible explanation: (Goldwater's 2006 analysis of Brent's model)
  - All the models tend to *undersegment* (i.e., find collocations instead of words)
  - Incremental initialization *greedily searches for common substrings*
  - Shorter strings are more likely to be recurr early than longer ones

# Table label resampling

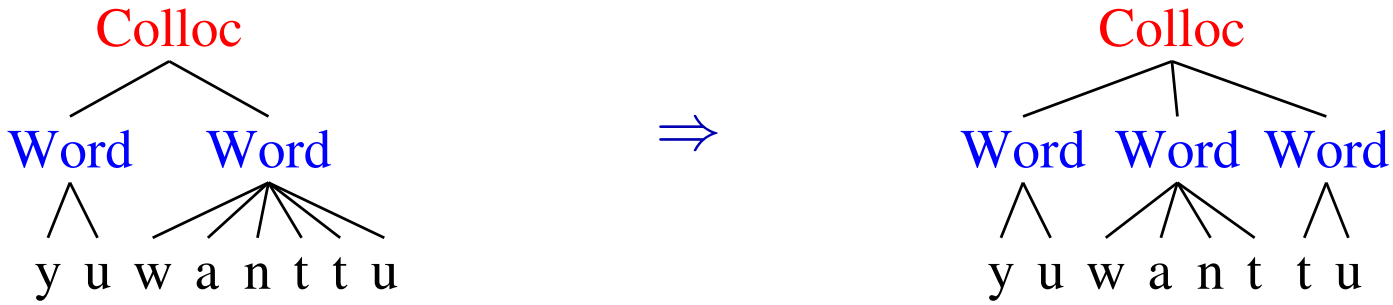
- Each adapted non-terminal has a CRP with tables labeled with parses
- "Rich get richer" $\Rightarrow$ resampling a sentence's parse reuses the same cached subtrees
- *Resample table labels* as well sentence parses
  - A table label may be used in many sentence parses

  $\Rightarrow$ Resampling a single table label may change the parses of a single sentence

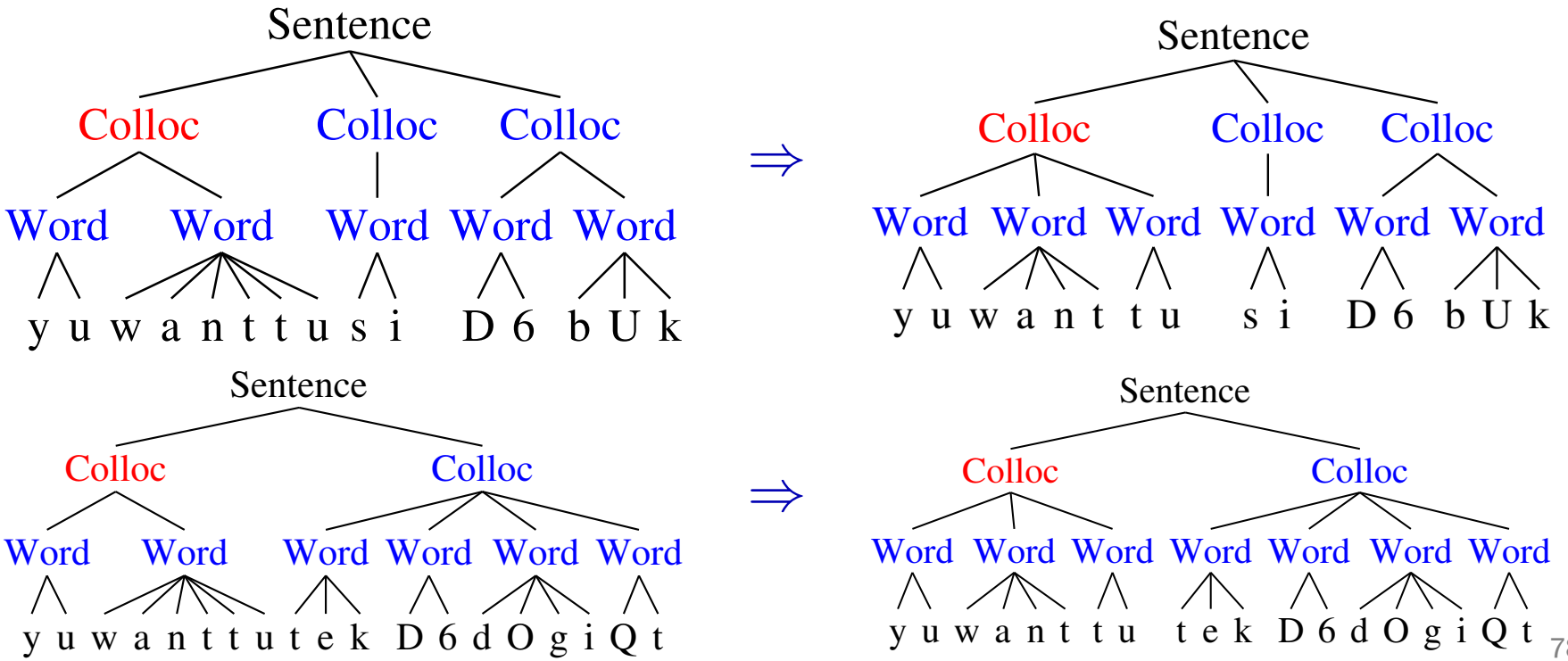
  $\Rightarrow$ table label resampling can improve mobility with grammars with a hierarchy of adapted non-terminals
- Essential for grammars with a complex hierarchical structure

# Table label resampling example

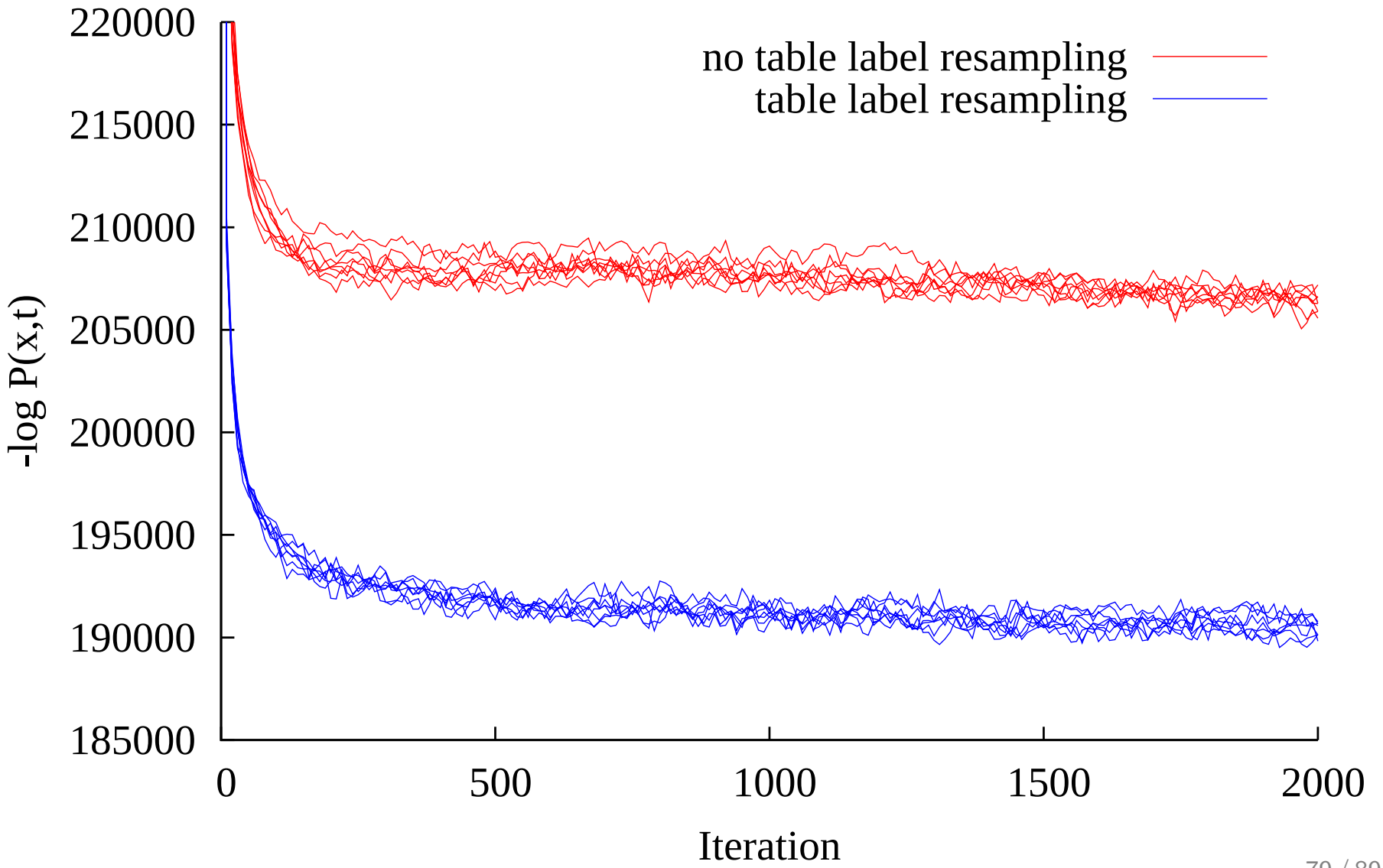
Label on table in Chinese Restaurant for colloc




Resulting changes in parse trees



# Table label resampling produces much higher-probability parses



# Outline

# Summary and future work

- Adaptor Grammars (AG) "adapt" to the strings they generate
- AGs learn probability of whole subtrees (not just rules)
- AGs are *non-parametric* because cached subtrees depend on the data
- MCMC sampling is a natural approach to AG inference
  - Slice sampling hyperparameters improves accuracy by 10%
  - Modal decoding improves accuracy by 2%
  - Incremental initialization improves accuracy by 2%, but hurts posterior probability
  - Table label resampling improves performance by 20%

  Taken together, colloc-syll word segmentation model improves from last year's 78% to 87% token f-score

# Outline

# Issues with adaptor grammars

- Recursion *through adapted nonterminals* seems problematic
  - New tables are created as each node is encountered top-down
  - But the tree labeling the table is only known after the whole subtree has been completely generated
  - If adapted nonterminals are recursive, might pick a table whose label we are currently constructing. What then?
- Extend adaptor grammars so adapted fragments can end at nonterminals a la DOP (currently always go to terminals)
  - Adding "exit probabilities" to each adapted nonterminal
  - In some approaches, fragments can grow "above" existing fragments, but can't grow "below" (O'Donnell)
- Adaptor grammars *conflate grammatical and Bayesian hierarchies*
  - Might be useful to disentangle them with *meta-grammars*

# Context-free grammars

A *context-free grammar* (CFG) consists of:

- a finite set $N$ of *nonterminals*,
- a finite set $W$ of *terminals* disjoint from $N$,
- a finite set $R$ of *rules* $A \to \beta$, where $A \in N$ and $\beta \in (N \cup W)^\star$
- a *start symbol* $S \in N$.

Each $A \in N \cup W$ *generates* a set $\mathcal{T}_A$ of trees.

These are the smallest sets satisfying:

- If $A \in W$ then $\mathcal{T}_A = \{A\}$.
- If $A \in N$ then:

$$\mathcal{T}_A = \bigcup_{A \to B_1 \ldots B_n \in R_A} \textsc{Tree}_A(\mathcal{T}_{B_1}, \ldots, \mathcal{T}_{B_n})$$

where $R_A = \{A \to \beta : A \to \beta \in R\}$, and

$$\textsc{Tree}_A(\mathcal{T}_{B_1}, \ldots, \mathcal{T}_{B_n}) = \left\{ \overbrace{t_1 \ \ldots \ t_n}^{A} : \begin{array}{l} t_i \in \mathcal{T}_{B_i}, \\ i = 1, \ldots, n \end{array} \right\}$$

The set of trees generated by a CFG is $\mathcal{T}_S$.

# Probabilistic context-free grammars

A *probabilistic context-free grammar* (PCFG) is a CFG and a vector $\boldsymbol{\theta}$, where:

- $\theta_{A \to \beta}$ is the probability of expanding the nonterminal $A$ using the production $A \to \beta$.

It defines distributions $G_A$ over trees $\mathcal{T}_A$ for $A \in N \cup W$:

$$G_A = \begin{cases} \delta_A & \text{if } A \in W \\ \sum_{A \to B_1 \ldots B_n \in R_A} \theta_{A \to B_1 \ldots B_n} \mathrm{TD}_A(G_{B_1}, \ldots, G_{B_n}) & \text{if } A \in N \end{cases}$$

where $\delta_A$ puts all its mass onto the singleton tree $A$, and:

$$\mathrm{TD}_A(G_1, \ldots, G_n) \left( \begin{array}{c} A \\ \diagup \quad \diagdown \\ t_1 \ \ldots \ t_n \end{array} \right) = \prod_{i=1}^{n} G_i(t_i).$$

$\mathrm{TD}_A(G_1, \ldots, G_n)$ is a distribution over $\mathcal{T}_A$ where each subtree $t_i$ is generated independently from $G_i$.

# DP adaptor grammars

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

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

$$H_A = \sum_{A \to B_1 \ldots B_n \in R_A} \theta_{A \to B_1 \ldots B_n} \mathrm{TD}_A(G_{B_1}, \ldots, G_{B_n})$$

The grammar generates the distribution $G_S$.
One Dirichlet Process for each adapted non-terminal $A$ (i.e., $\alpha_A > 0$).

# Recursion in adaptor grammars

- The probability of joint distributions $(\mathbf{G}, \mathbf{H})$ is defined by:

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

$$H_A = \sum_{A \to B_1 \ldots B_n \in R_A} \theta_{A \to B_1 \ldots B_n} \mathrm{TD}_A(G_{B_1}, \ldots, G_{B_n})$$

- This holds *even if adaptor grammar is recursive*
- Question: when does this define a *distribution* over $(\mathbf{G}, \mathbf{H})$?

# Adaptive fragment grammars

- Disentangle syntactic and Bayesian hierarchy
  - *Adaptive metagrammar* generates *fragment distributions*
  - which plug together as in tree substitution grammar
- Tree fragment sets $\mathcal{P}_A, A \in N$ are smallest sets satisfying:

$$\mathcal{P}_A = \bigcup_{A \to B_1 \ldots B_n \in R_A} \textsc{Tree}_A(\{B_1\} \cup \mathcal{P}_{B_1}, \ldots, \{B_n\} \cup \mathcal{P}_{B_n})$$

- Grammar's distributions $G_A$ over $\mathcal{T}_A$ defined using *fragment distributions* $F_A$ over $\mathcal{P}_A$ (*generalized PCFG rules*)

$$G_A = \sum_{\substack{A \\ B_1 \ldots B_n} \in \mathcal{P}_A} F_A(\begin{array}{c} A \\ \triangle \\ B_1 \ldots B_n \end{array}) \quad \mathrm{TD}_A(\begin{array}{c} \\ \triangle \\ B_1 \ldots B_n \end{array}, G_{B_1}, \ldots, G_{B_n})$$

- A fragment grammar generates the distribution $G_S$

# Adaptive fragment distributions

- $H_A$ is a PCFG distribution over $\mathcal{P}_A$

$$H_A = \sum_{A \to B_1 \ldots B_n \in R_A} \theta_{A \to B_1 \ldots B_n} \mathrm{TD}_A(\eta\, \delta_{B_1} + (1-\eta) H_{B_1}, \ldots)$$

  where $\eta$ is the *fragment exit probability*
- Obtain $F_A$ by *adapting the $H_A$ distribution*

$$F_A \sim \mathrm{DP}(\alpha_A, H_A)$$

- This construction can be *iterated*, i.e., replace $\boldsymbol{\theta}$ with another fragment distribution
- Question: if we iterate this, when does the *fixed point* exist, and what is it?