Improving nonparameteric Bayesian inference: experiments on unsupervised word segmentation with adaptor grammars

Mark Johnson[1] Sharon Goldwater[2]

[1]Brown University
Providence, RI
Mark_Johnson@Brown.edu

[2]University of Edinburgh
Edinburgh EH8 9AB
sgwater@inf.ed.ac.uk

NAACL 2009 Conference

Outline

- *Non-parametric models* learn which parameters are relevant
  - Unsupervised word segmentation: identify the words in an unsegmented stream of phonemes
- *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
- This paper describes improvements to adaptor grammars and their MCMC inference procedure
  - *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 models

Outline

Unsupervised word segmentation

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

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

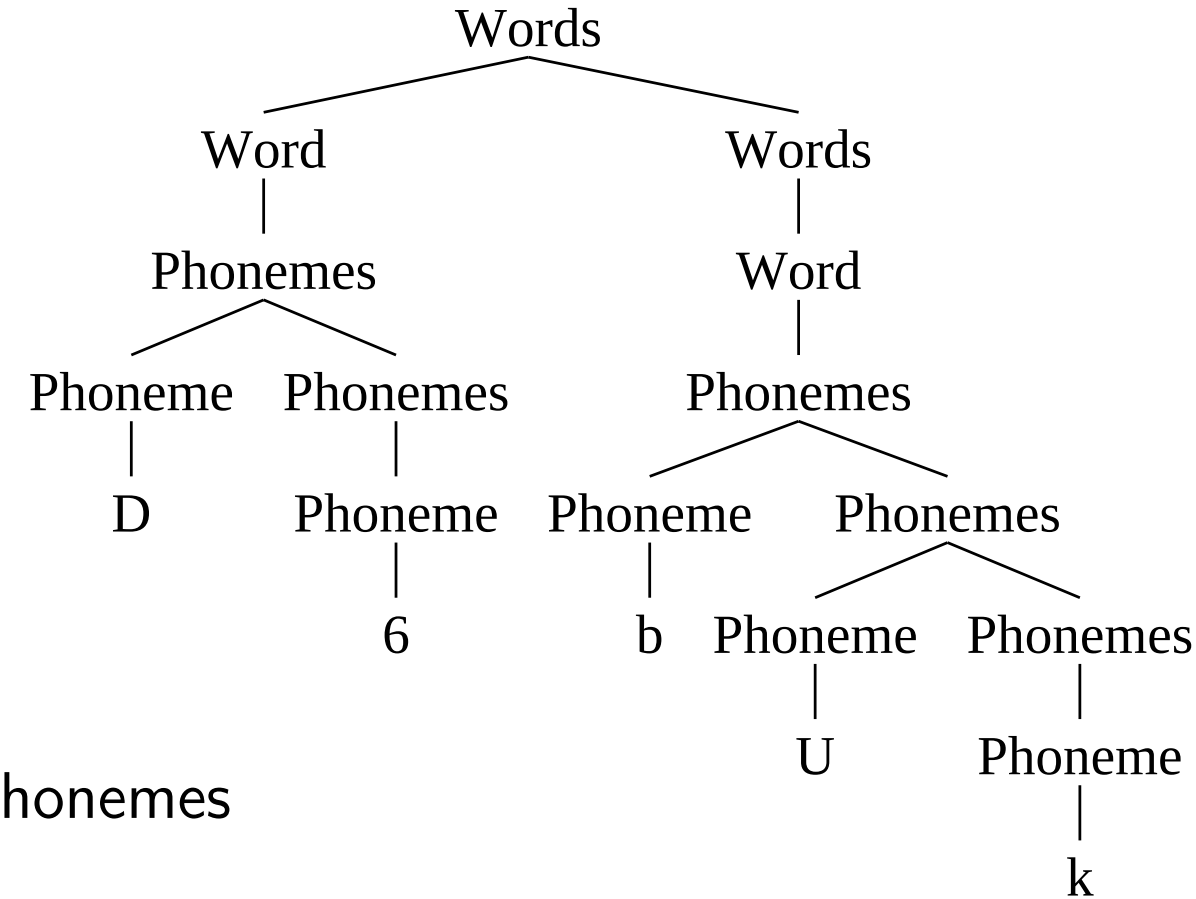
- Useful cues for word segmentation:
  - Phonotactics (Fleck)
  - Inter-word dependencies (Goldwater)

Word segmentation with PCFGs (1)

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

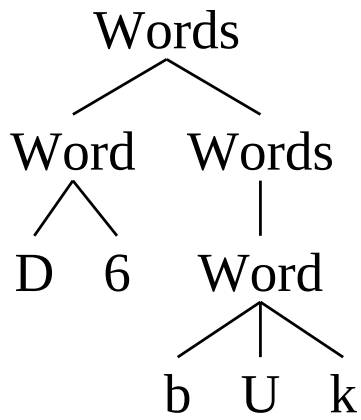
which abbreviates

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



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

Outline

Multinomials and Chinese Restaurant Processes

- Multinomials (e.g., dice, HMMs, PCFGs)
  - Observations $\mathbf{z} = (z_1, \ldots, z_n)$ ranging over outcomes $1, \ldots, m$ (e.g., sides of a die), with outcome k observed $n_k(\mathbf{z})$ times
  - *Predictive distribution* with uniform Dirichlet prior:

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

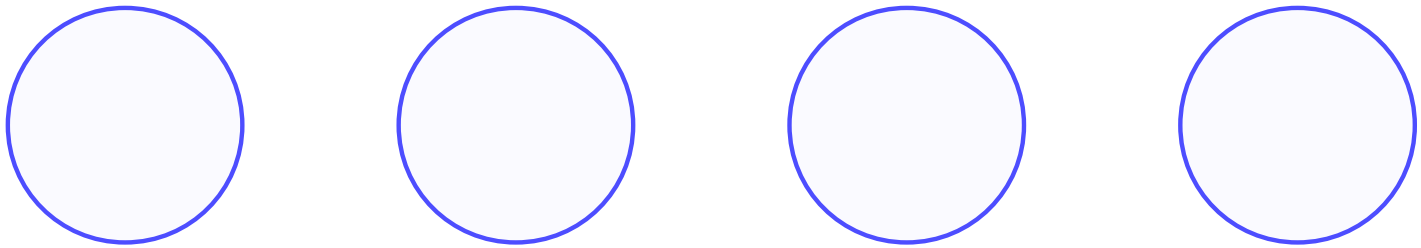
- Let $m \to \infty$

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

- Insist outcomes are enumerated *in order*
  $\Rightarrow$ *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)

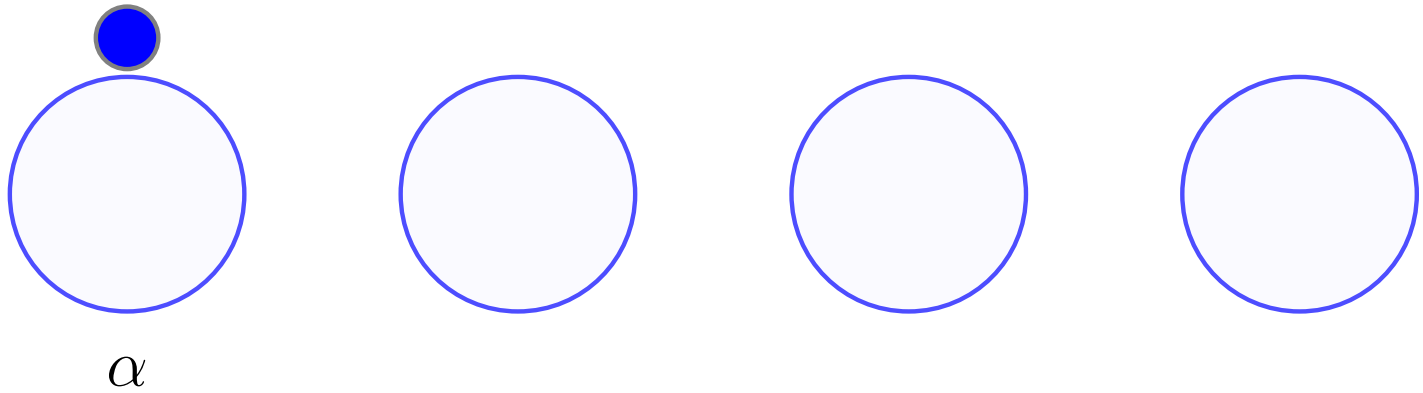


- Customer $\rightarrow$ table mapping $\mathbf{z} =$
- $\mathrm{P}(\mathbf{z}) = 1$

- Next customer chooses a table according to:

  $\mathrm{P}(\text{occupied table}) \propto \text{number of customers at table}$
  $\mathrm{P}(\text{next unoccupied table}) \propto \alpha$

Chinese Restaurant Process (1)

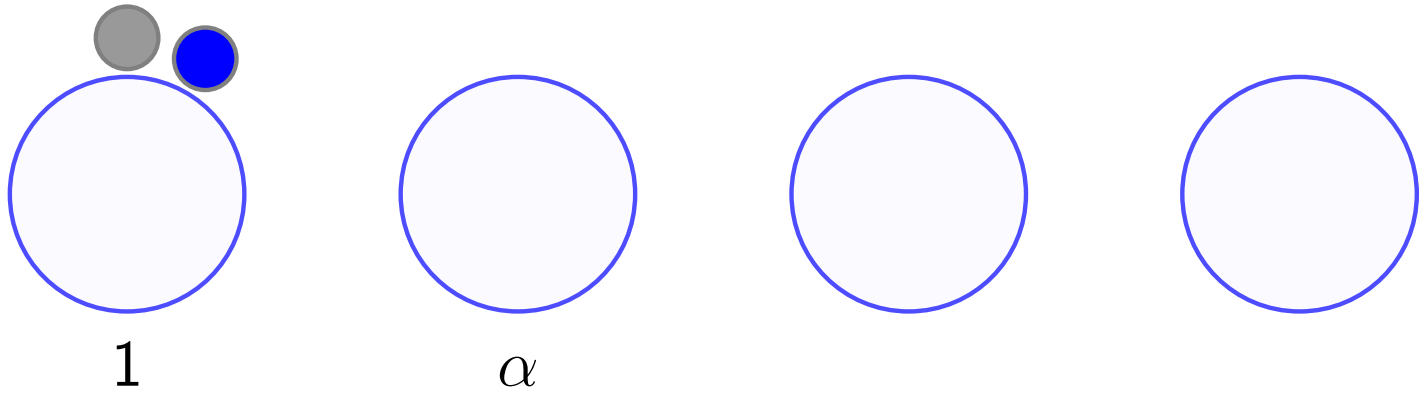


- Customer $\rightarrow$ table mapping $\mathbf{z} = 1$
- $\mathrm{P}(\mathbf{z}) = \alpha/\alpha$

- Next customer chooses a table according to:

  P(occupied table) $\propto$ number of customers at table
  P(next unoccupied table) $\propto$ α

Chinese Restaurant Process (2)

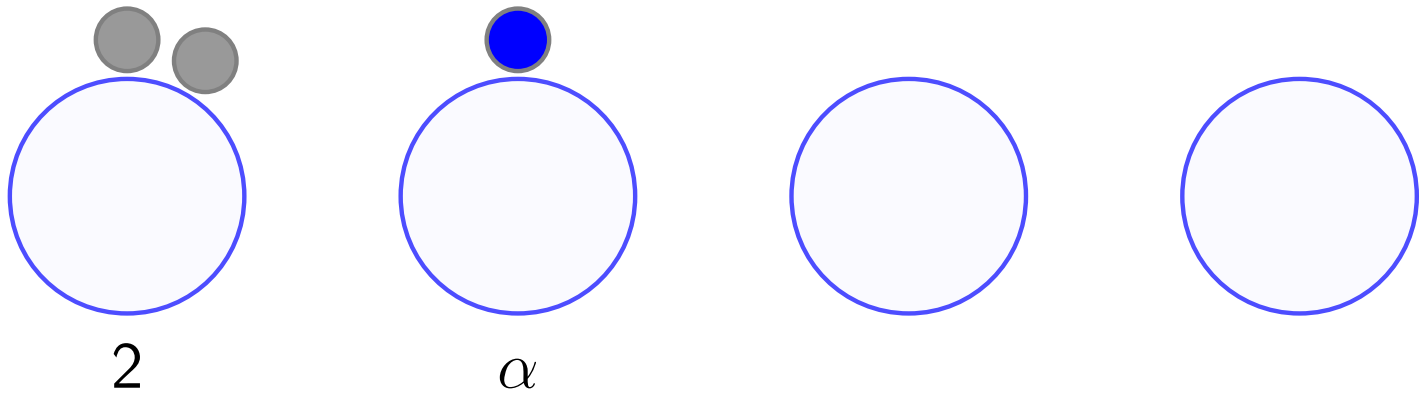


- Customer $\rightarrow$ table mapping $\mathbf{z} = 1, 1$
- $\mathrm{P}(\mathbf{z}) = \alpha/\alpha \times 1/(1+\alpha)$

- Next customer chooses a table according to:

  P(occupied table) $\propto$ number of customers at table
  P(next unoccupied table) $\propto$ α

Chinese Restaurant Process (3)

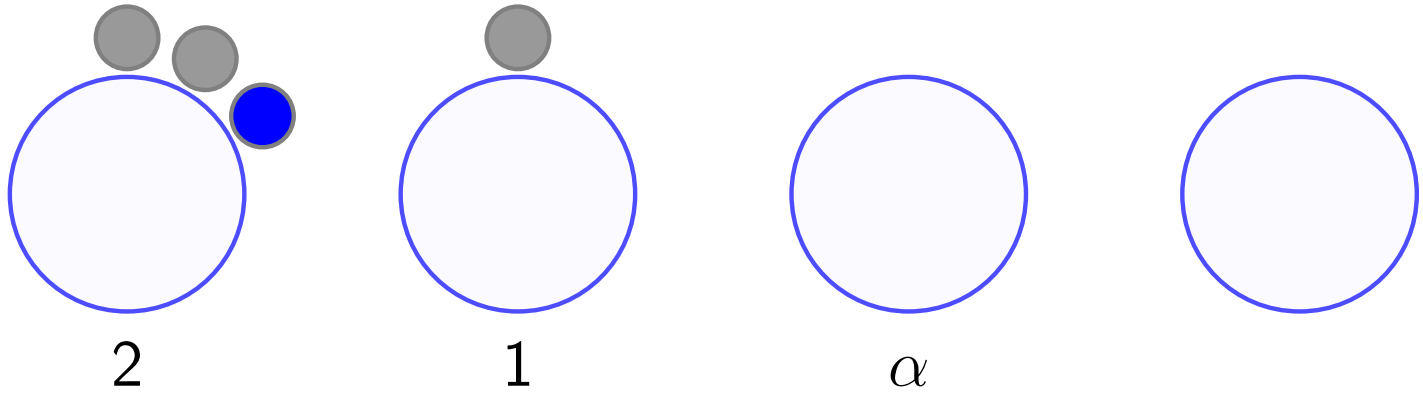


- Customer $\rightarrow$ table mapping $\mathbf{z} = 1, 1, 2$
- $\mathrm{P}(\mathbf{z}) = \alpha/\alpha \times 1/(1+\alpha) \times \alpha/(2+\alpha)$

- Next customer chooses a table according to:

  P(occupied table) $\propto$ number of customers at table
  P(next unoccupied table) $\propto$ α

Chinese Restaurant Process (4)

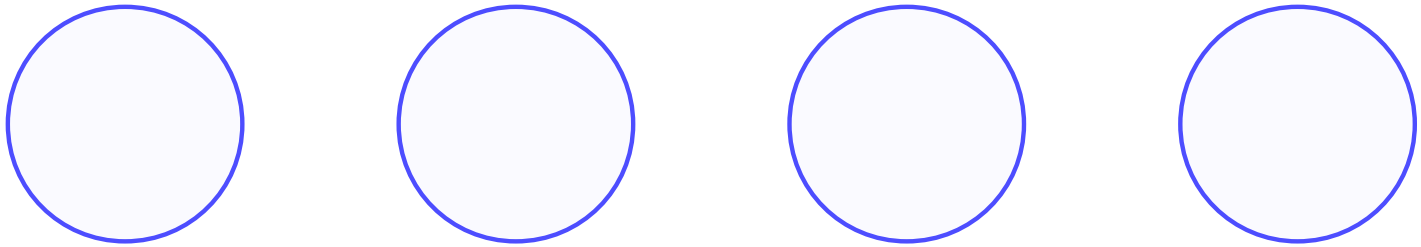


- Customer $\rightarrow$ table mapping $\mathbf{z} = 1, 1, 2, 1$
- $\mathsf{P}(\mathbf{z}) = \alpha/\alpha \times 1/(1+\alpha) \times \alpha/(2+\alpha) \times 2/(3+\alpha)$

- Next customer chooses a table according to:

  P(occupied table) $\propto$ number of customers at table
  P(next unoccupied table) $\propto$ α

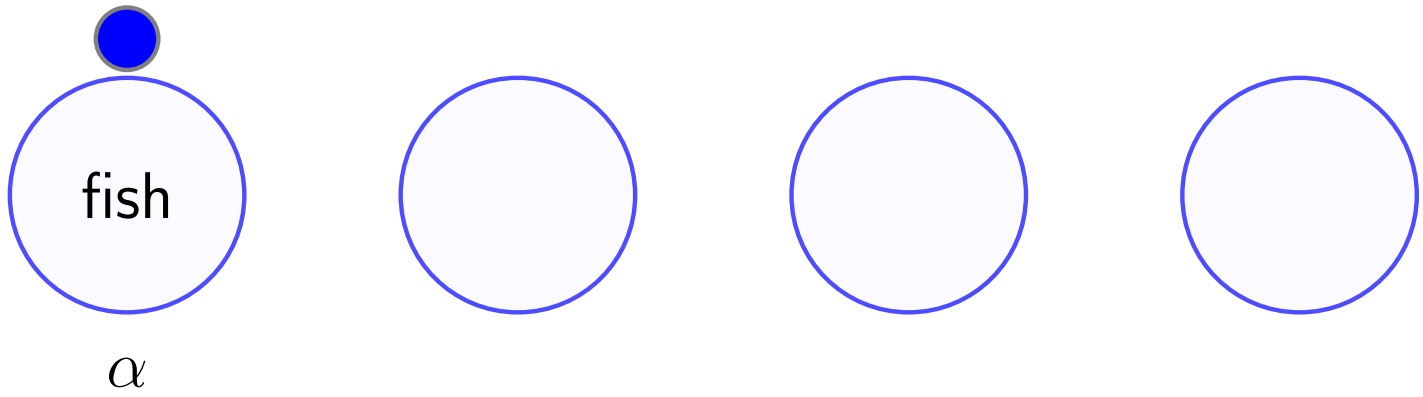
Labelled Chinese Restaurant Process (0)



- 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 $x_i = y_{z_i}$

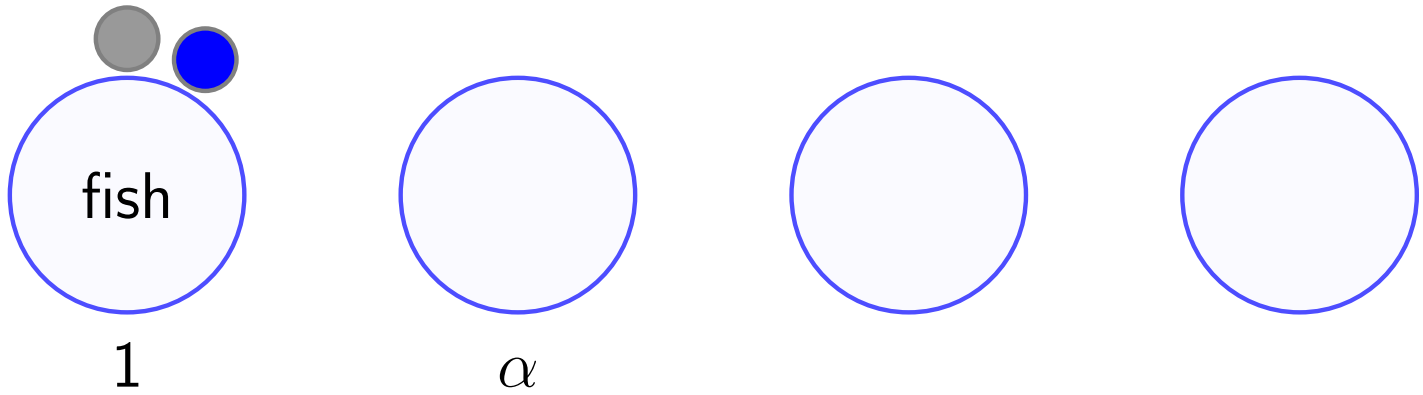
Labelled Chinese Restaurant Process (1)



- 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 $x_i = y_{z_i}$

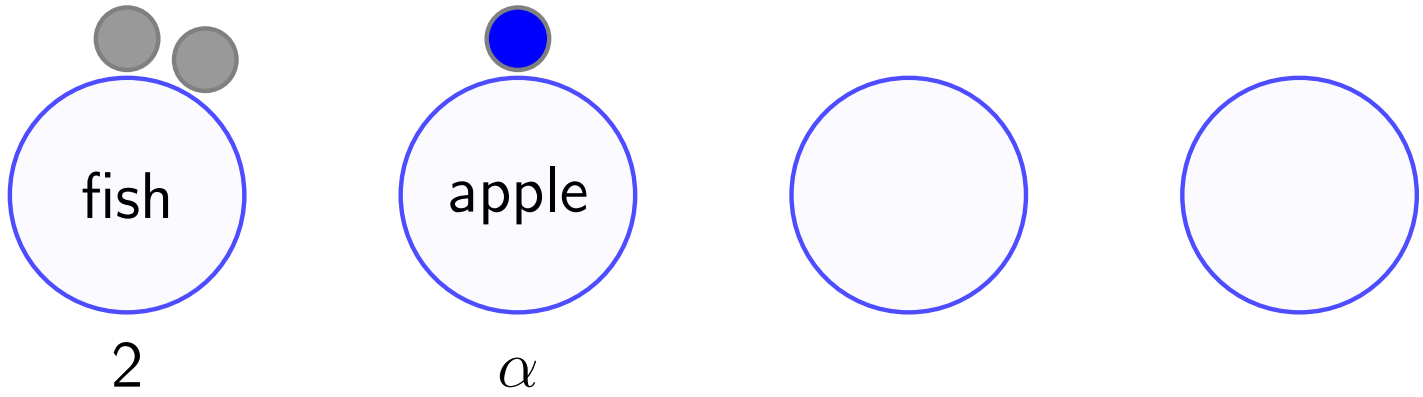
Labelled Chinese Restaurant Process (2)



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

- *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 $x_i = y_{z_i}$

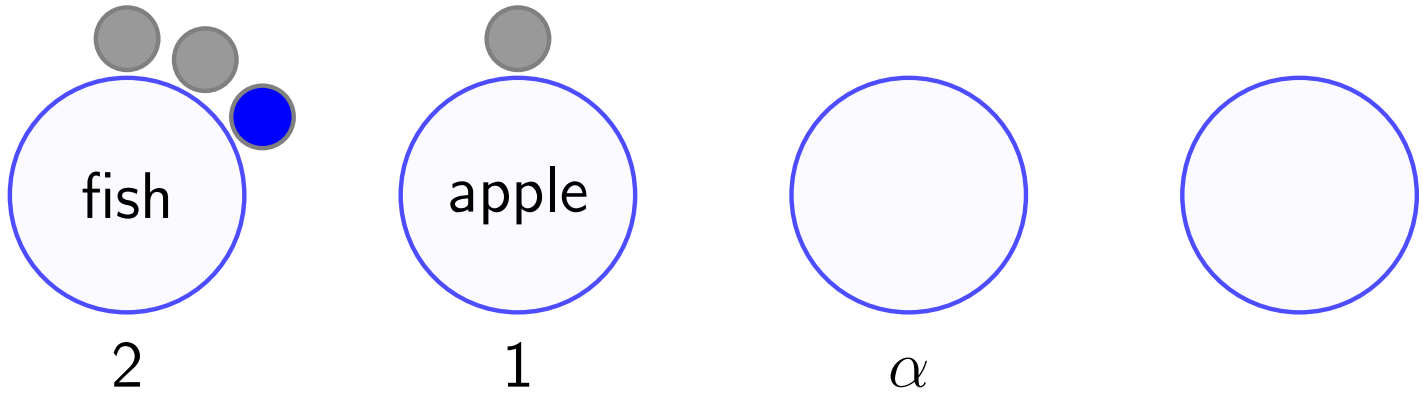
Labelled Chinese Restaurant Process (3)



- 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 $x_i = y_{z_i}$

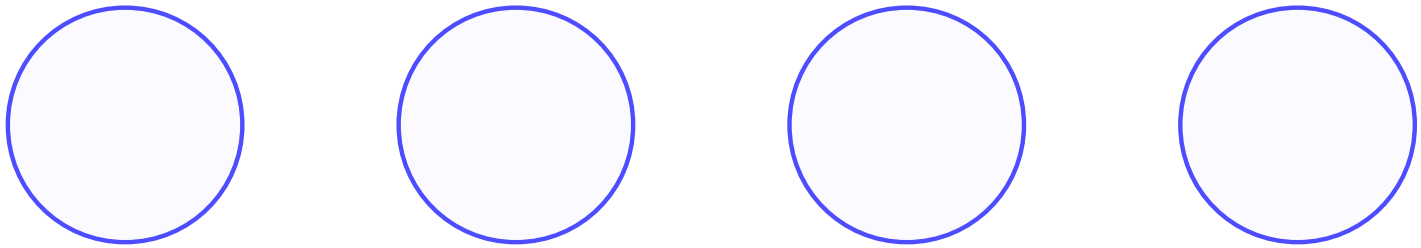
Labelled Chinese Restaurant Process (4)



- 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 $x_i = y_{z_i}$

Pitman-Yor Process (0)

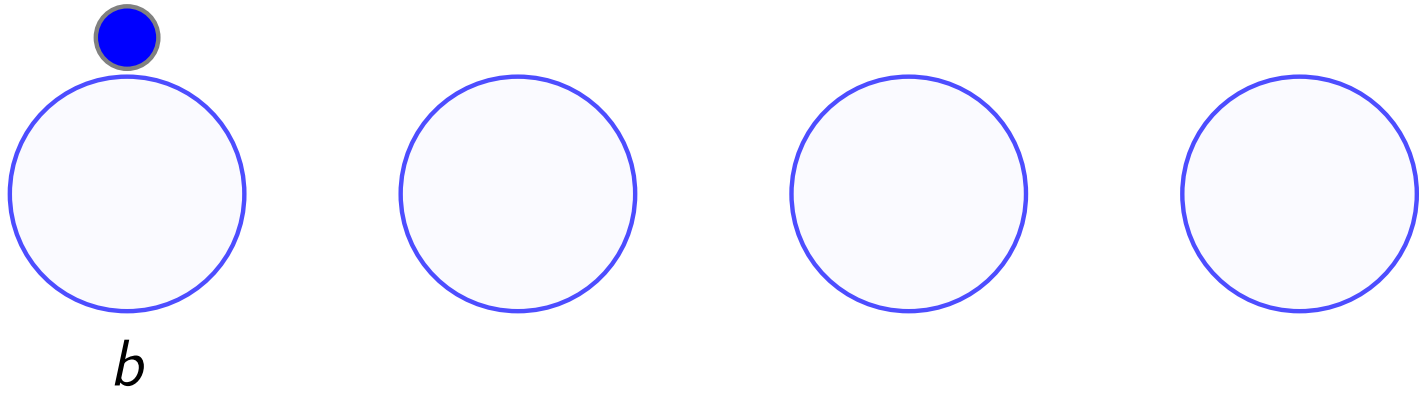


- Customer $\longrightarrow$ table mapping $\mathbf{z} =$
- $\mathrm{P}(\mathbf{z}) = 1$

- 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)

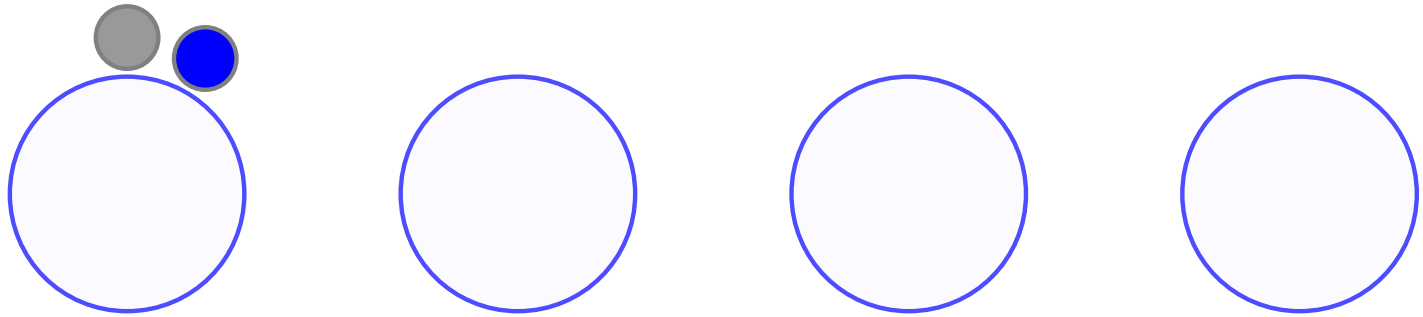


- Customer $\longrightarrow$ table mapping $\mathbf{z} = 1$
- $\mathrm{P}(\mathbf{z}) = b/b$

- 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)

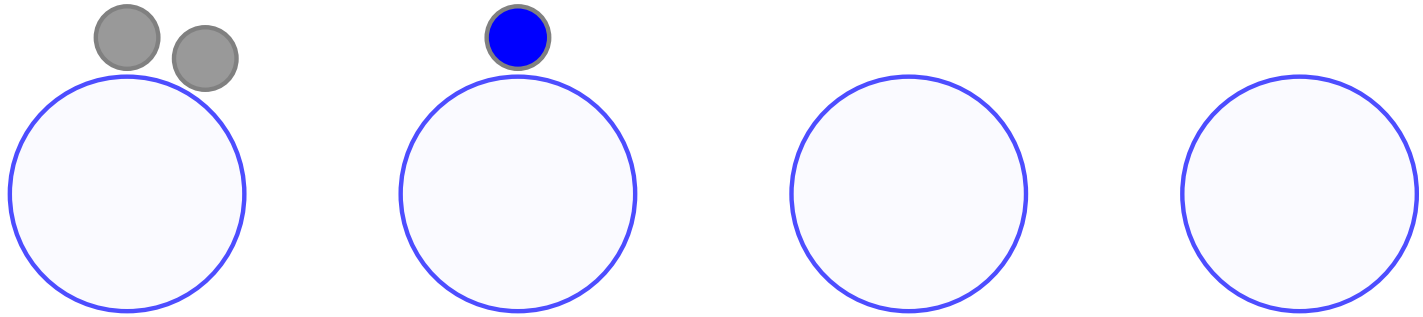


- Customer $\longrightarrow$ table mapping $\mathbf{z} = 1, 1$
- $\mathrm{P}(\mathbf{z}) = b/b \times (1-a)/(1+b)$

- 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)

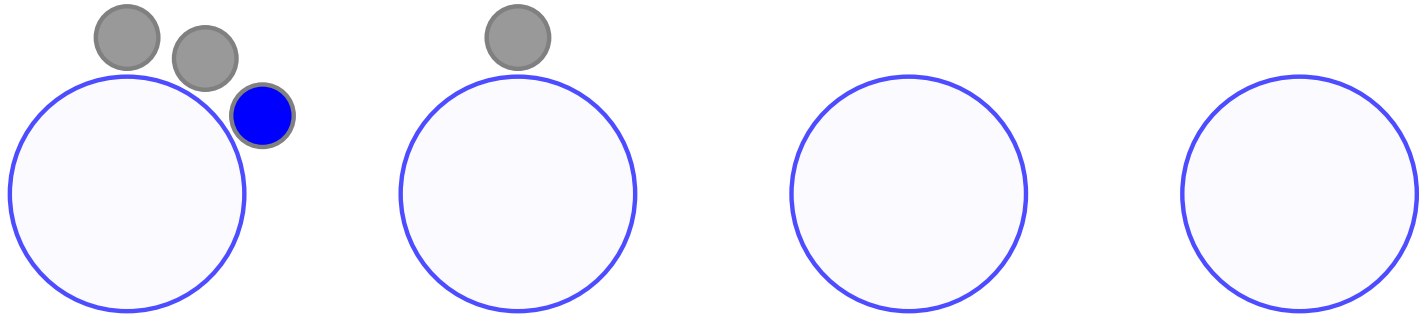


- Customer $\longrightarrow$ table mapping $\mathbf{z} = 1, 1, 2$
- $\mathrm{P}(\mathbf{z}) = b/b \times (1-a)/(1+b) \times (a+b)/(2+b)$

- 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

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

Summary: Chinese Restaurant Processes and Pitman-Yor Processes

- *Chinese Restaurant Processes* (CRPs) generalize Dirichlet-Multinomials to an *unbounded number of outcomes*
  - *concentration parameter* α 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
- *Labelled CRPs and PYPs* use a *base distribution* to label each table
  - base distribution can have *infinite support*
  - concentrates mass on a countable subset

Outline

Adaptor grammars: informal description

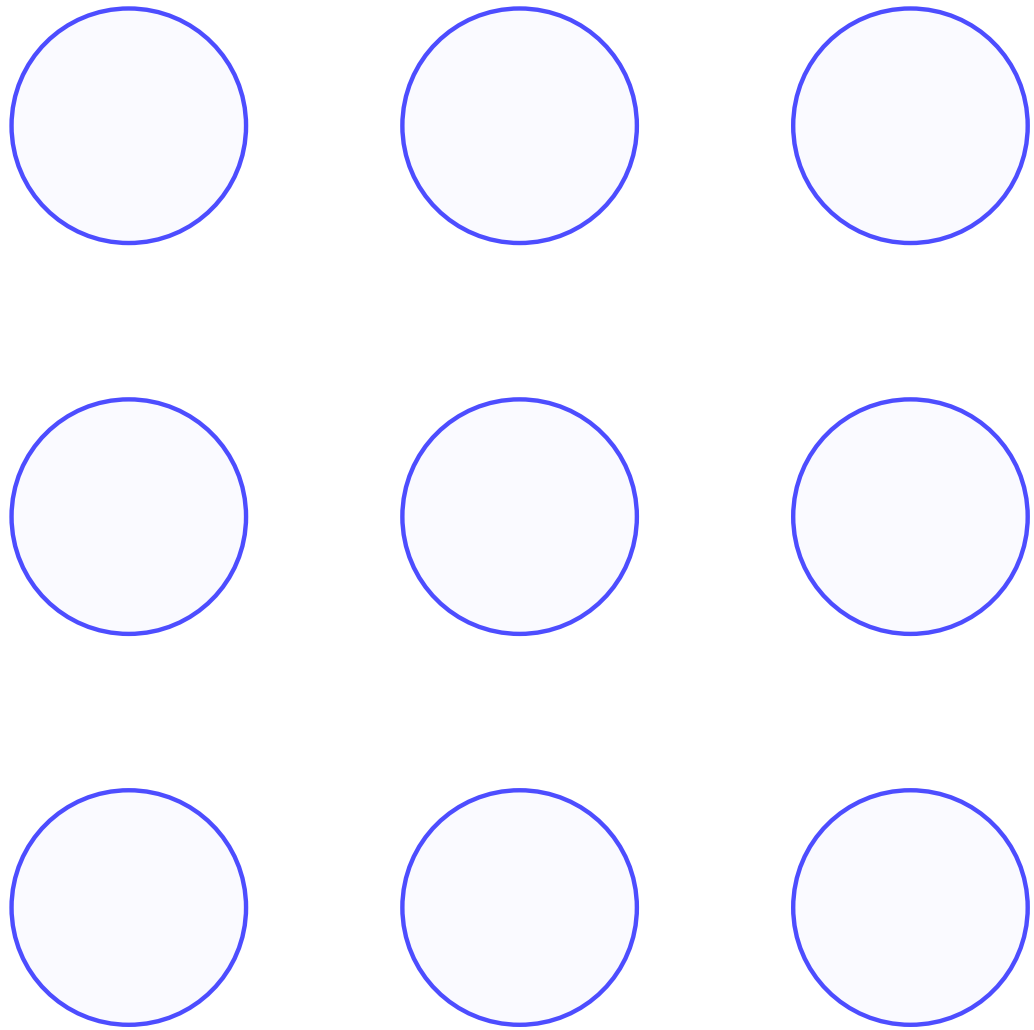
- The trees generated by an adaptor grammar are defined by CFG rules as in a CFG
- A subset of the nonterminals are *adapted*
- *Unadapted nonterminals* expand by picking a rule and recursively expanding its children, as in a PCFG
- *Adapted nonterminals* can expand in two ways:
  - by picking a rule 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*



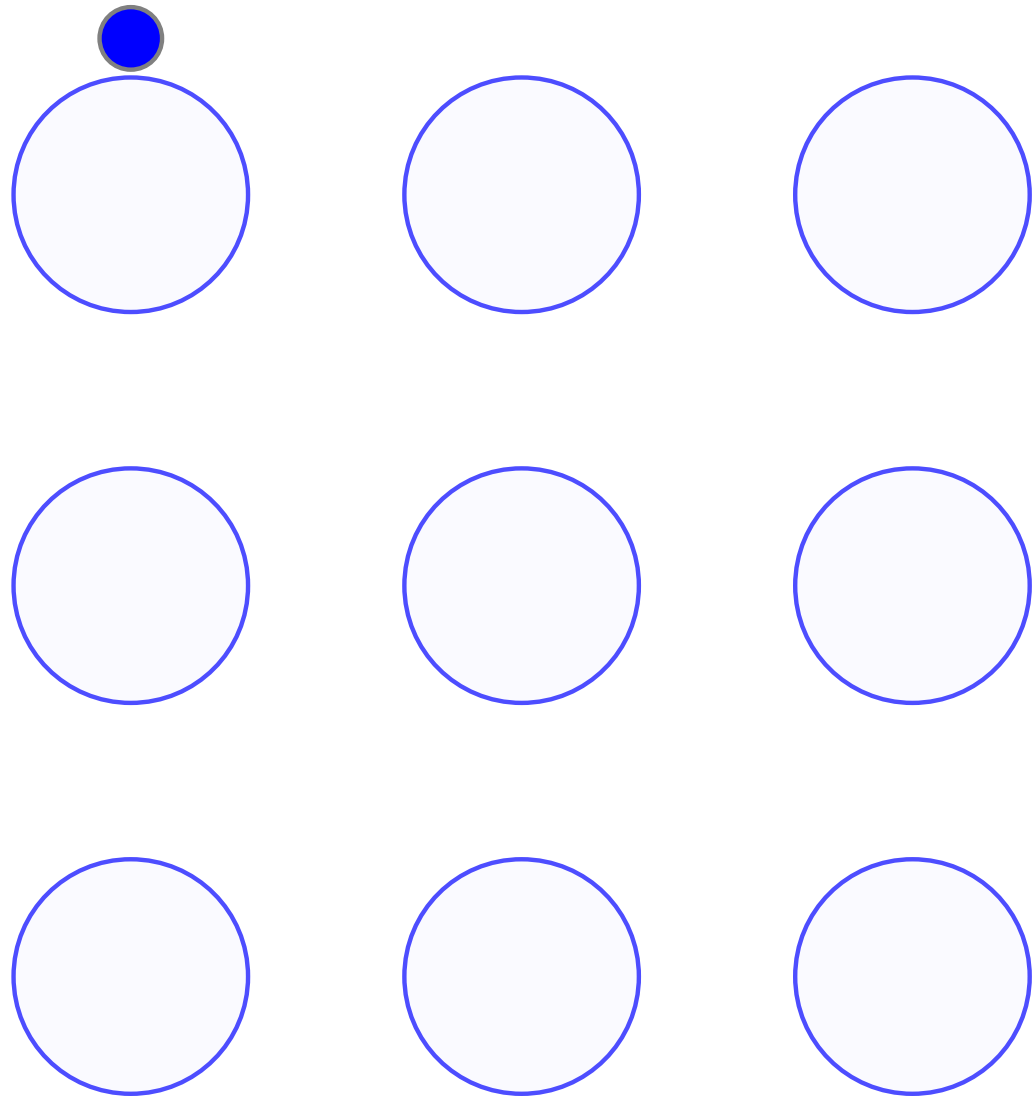
Generated words:

Adaptor grammar for stem-suffix morphology (1a)

Word $\rightarrow$ Stem Suffix

Stem $\rightarrow$ Phoneme^{+}

Suffix $\rightarrow$ Phoneme*



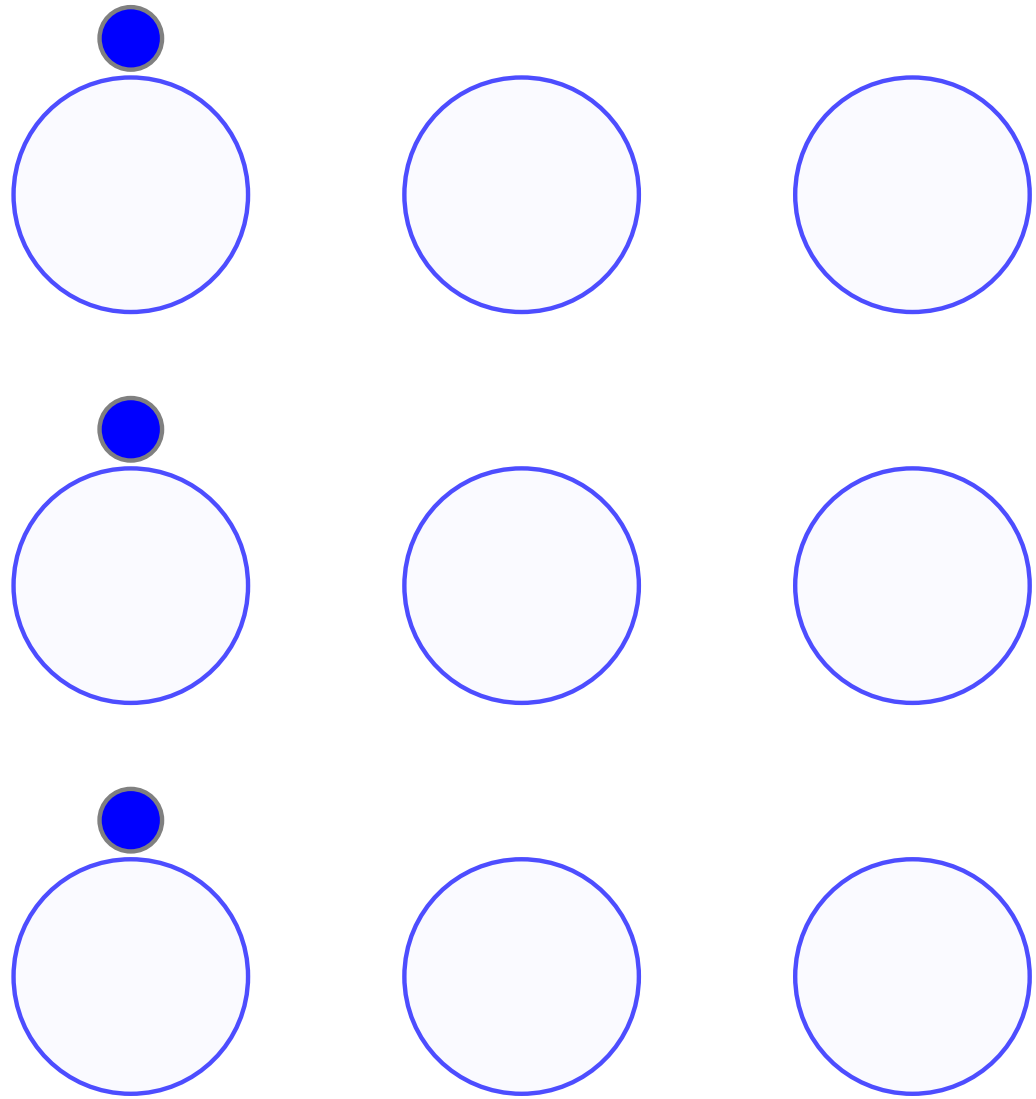
Generated words:

Adaptor grammar for stem-suffix morphology (1b)

Word $\rightarrow$ Stem Suffix

Stem $\rightarrow$ Phoneme^{+}

Suffix $\rightarrow$ Phoneme*



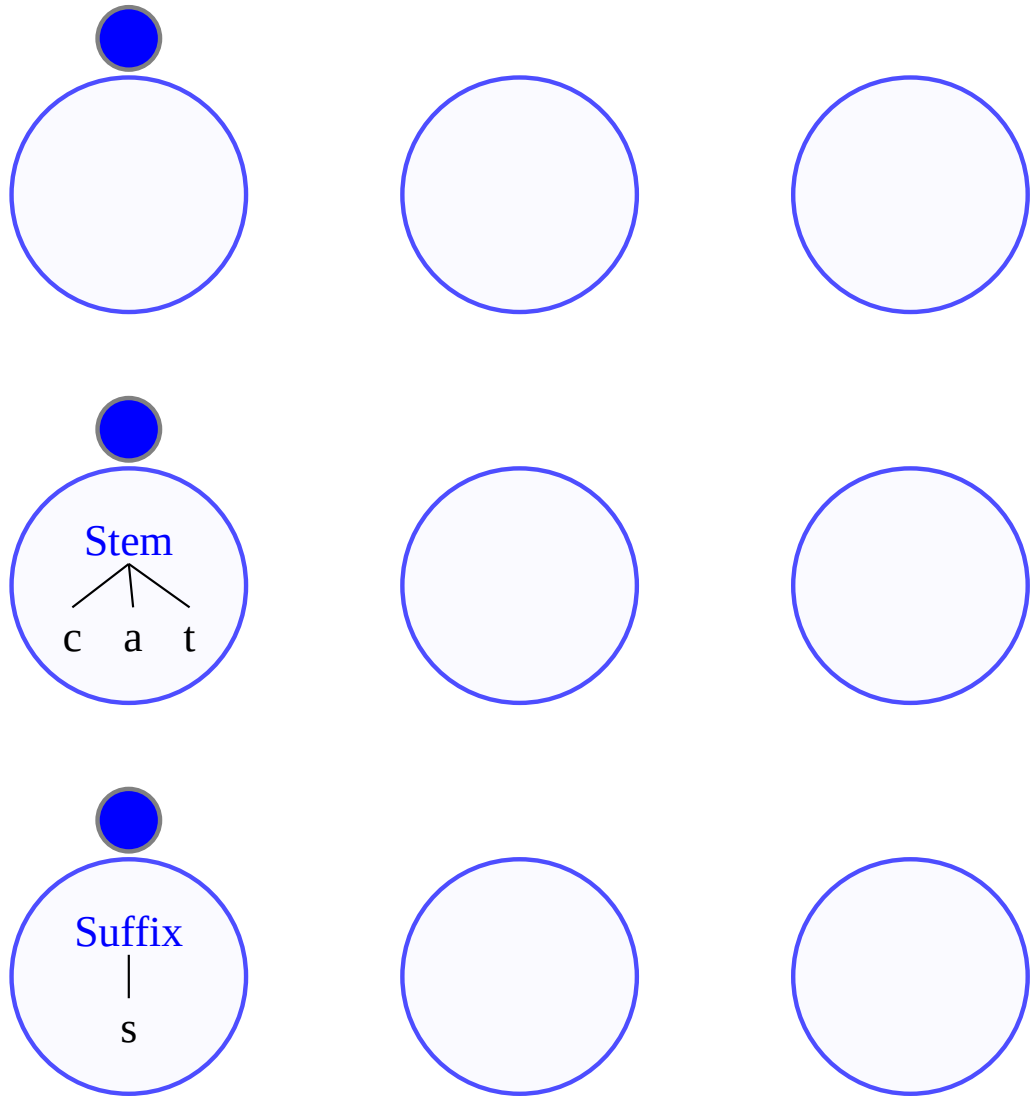
Generated words:

Adaptor grammar for stem-suffix morphology (1c)

Word → Stem Suffix

Stem → Phoneme^{+}

Suffix → Phoneme*



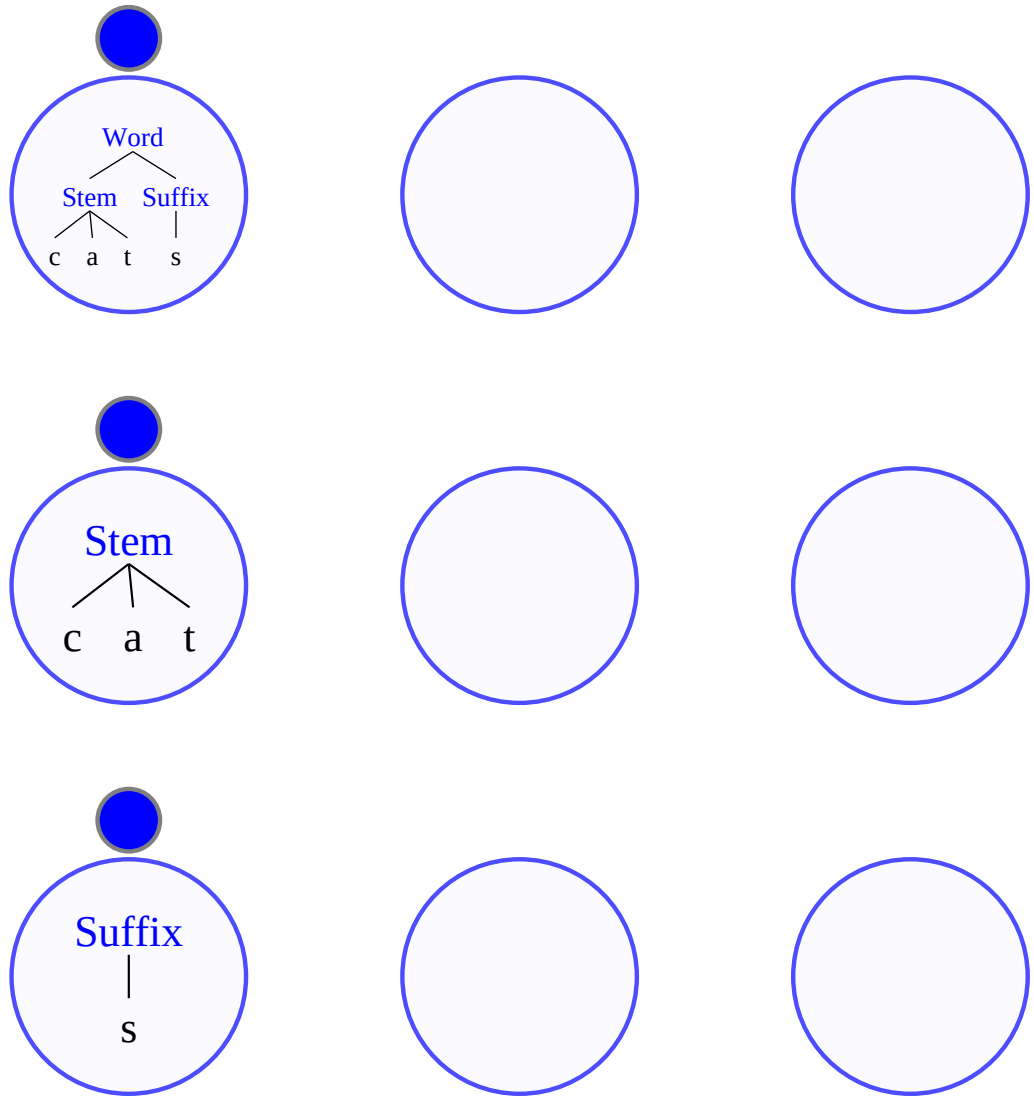
Generated words:

Adaptor grammar for stem-suffix morphology (1d)

Word → Stem Suffix

Stem → Phoneme^{+}

Suffix → Phoneme*



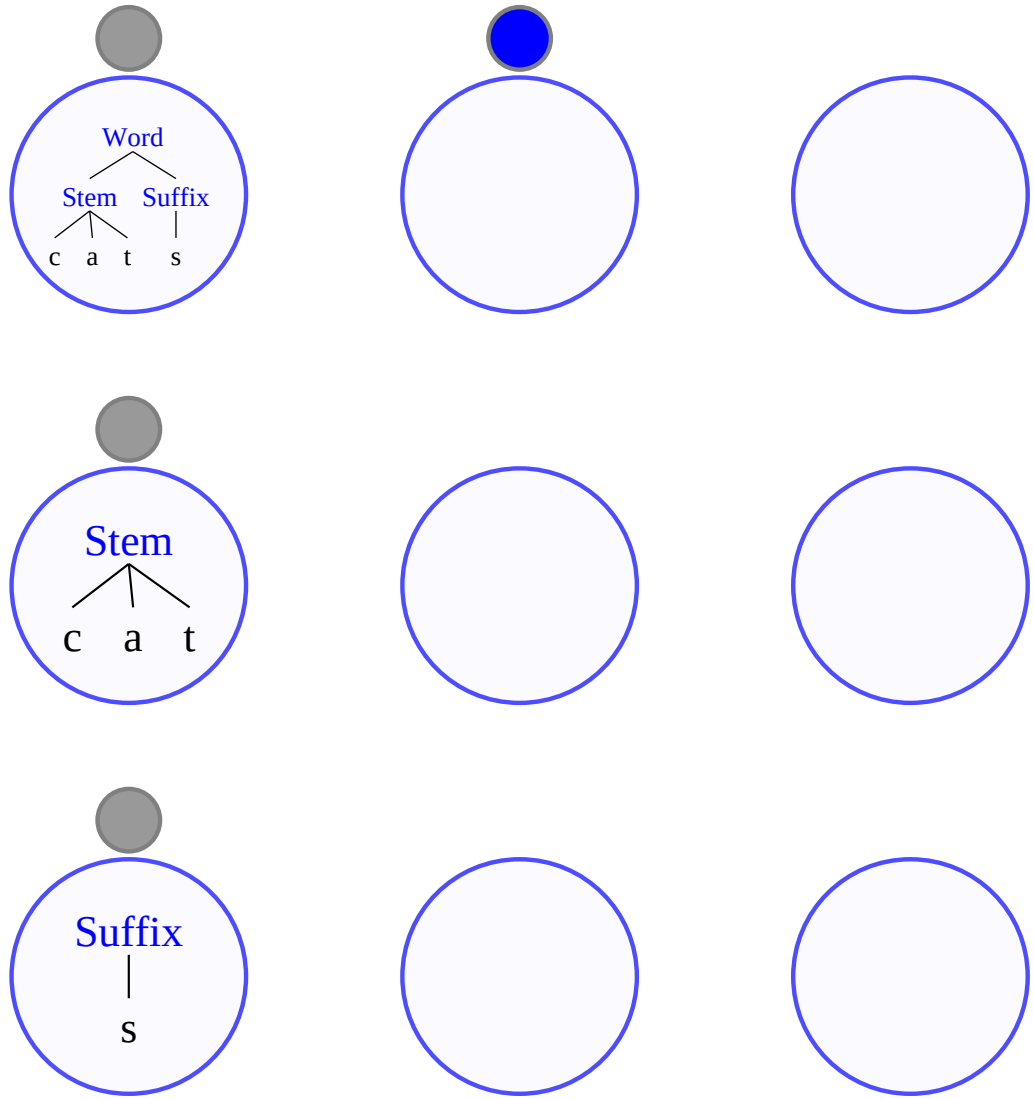
Generated words: cats

Adaptor grammar for stem-suffix morphology (2a)

Word → Stem Suffix

Stem → Phoneme^{+}

Suffix → Phoneme*



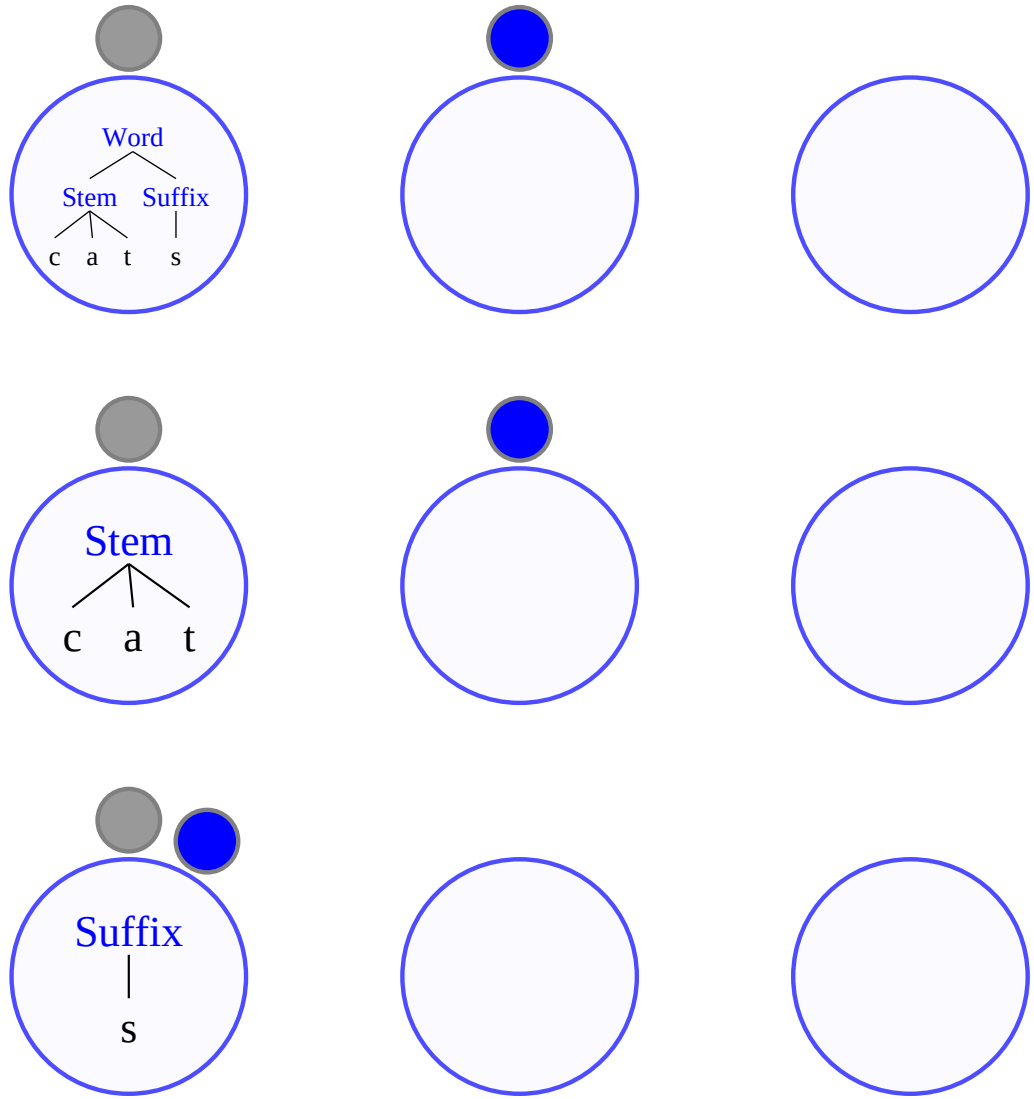
Generated words: cats

Adaptor grammar for stem-suffix morphology (2b)

Word → Stem Suffix

Stem → Phoneme^{+}

Suffix → Phoneme*



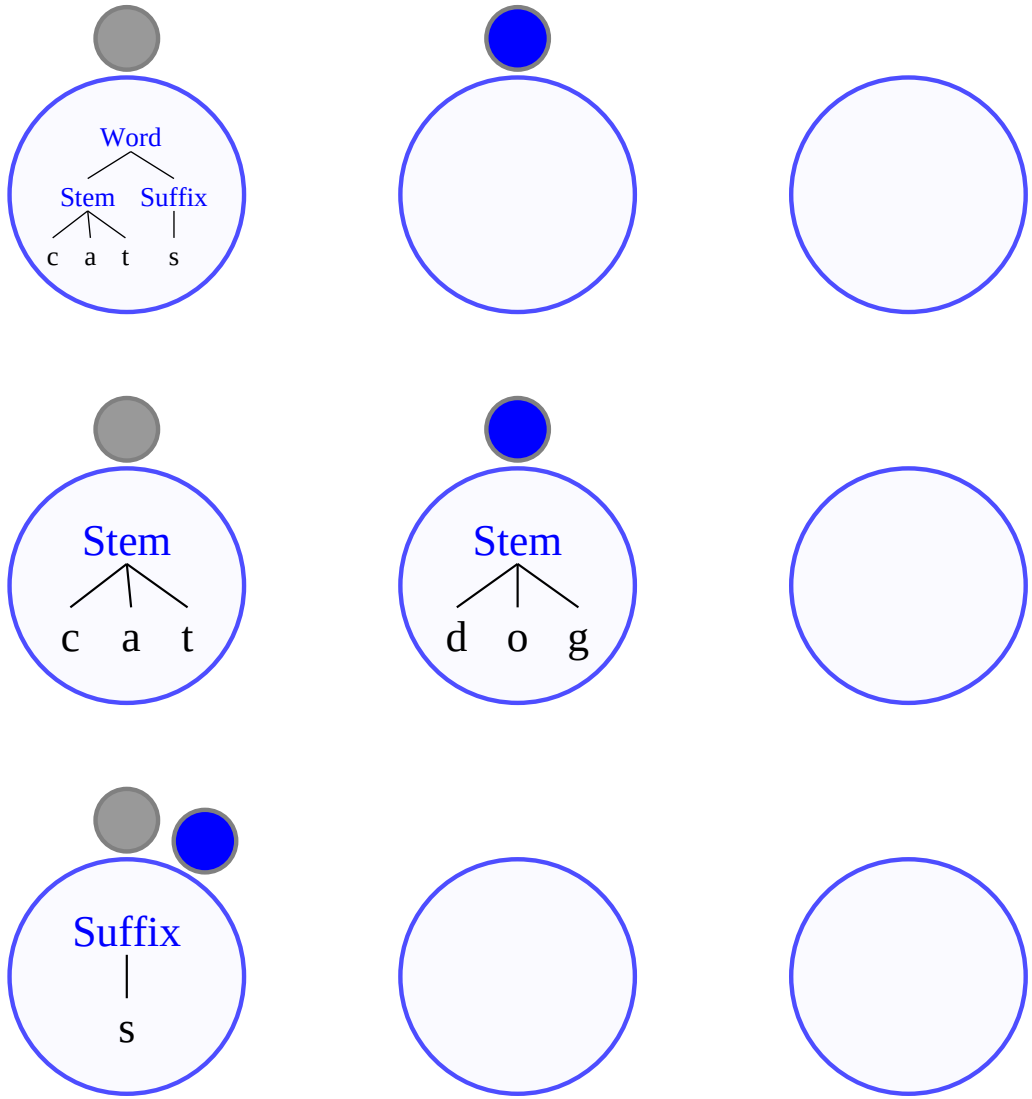
Generated words: cats

Adaptor grammar for stem-suffix morphology (2c)

Word → Stem Suffix

Stem → Phoneme^{+}

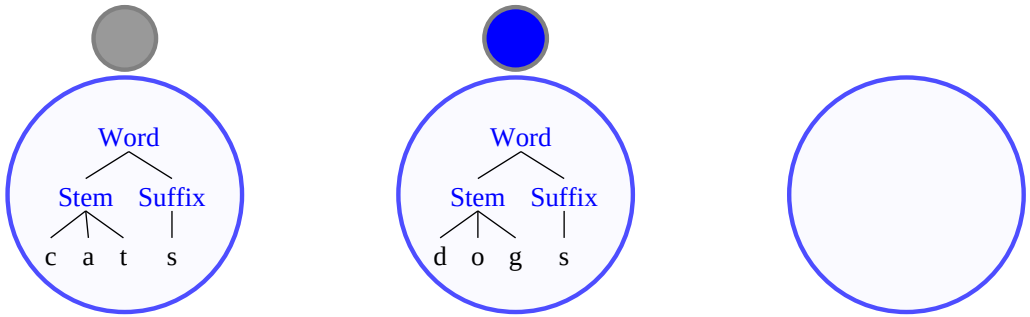
Suffix → Phoneme*



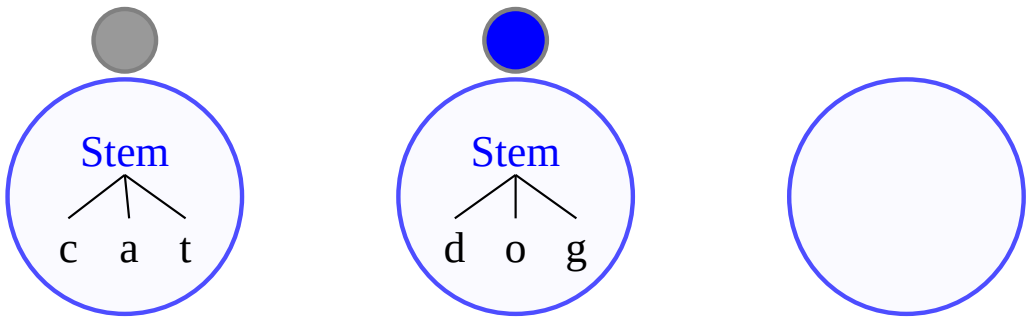
Generated words: cats

Adaptor grammar for stem-suffix morphology (2d)

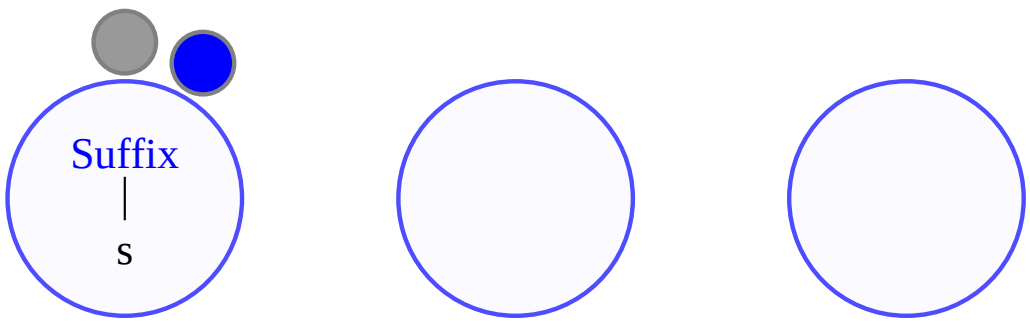
Word → Stem Suffix



Stem → Phoneme^{+}



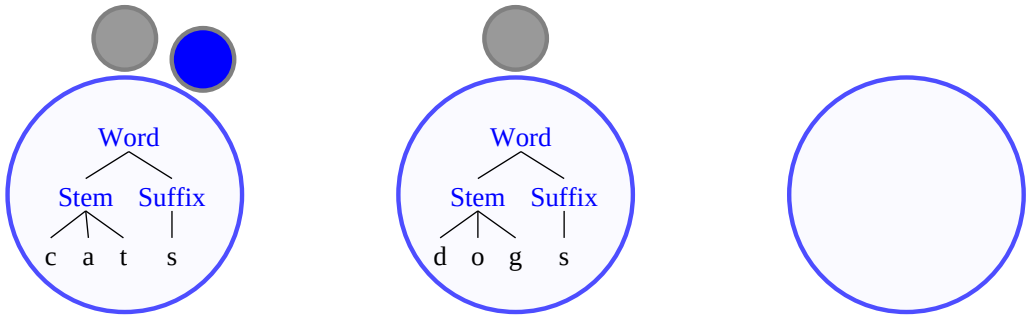
Suffix → Phoneme*



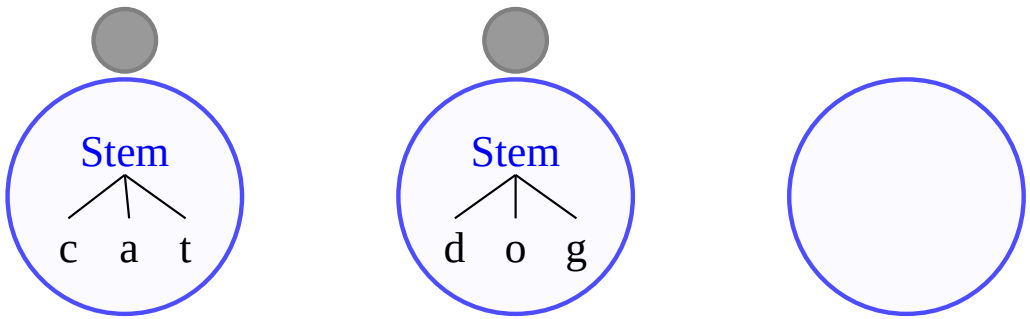
Generated words: cats, dogs

Adaptor grammar for stem-suffix morphology (3)

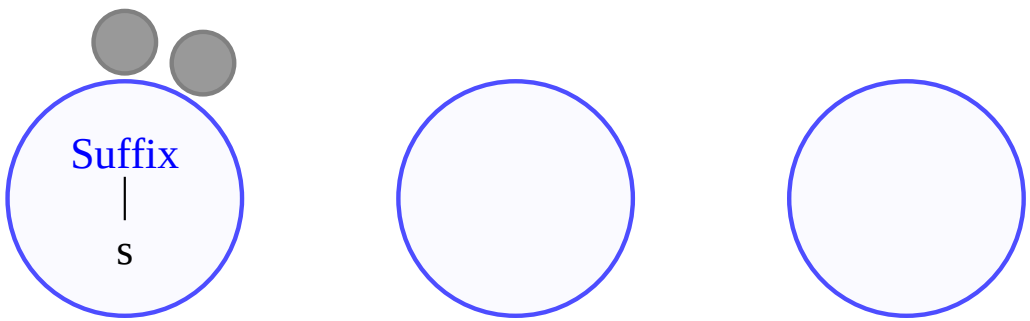
Word → Stem Suffix



Stem → Phoneme^{+}



Suffix → Phoneme*



Generated words: cats, dogs, cats

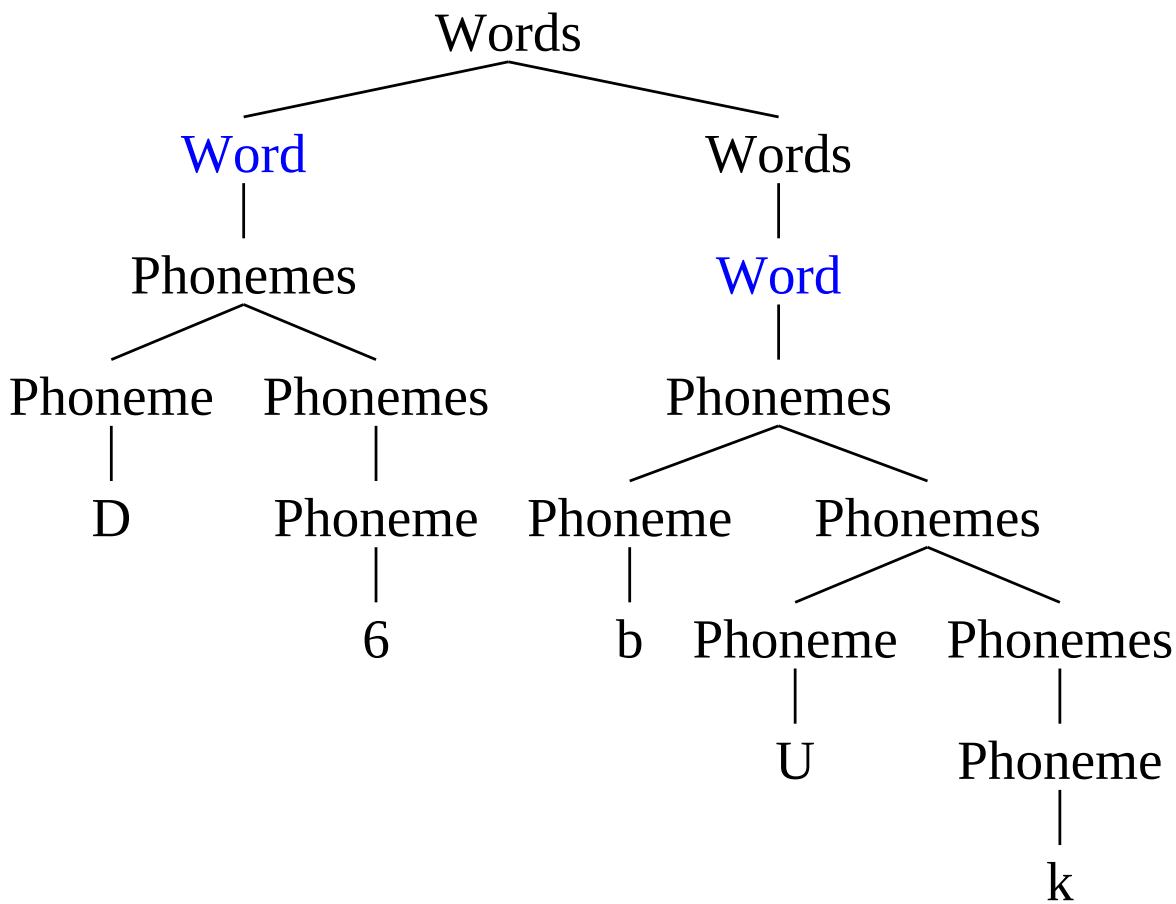
Adaptor grammars as generative processes

- 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 τ rooted in A with probability proportional to the number of times τ was previously generated
  - using $A \rightarrow \beta$ with probability proportional to $\alpha_A \theta_{A \rightarrow \beta}$

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

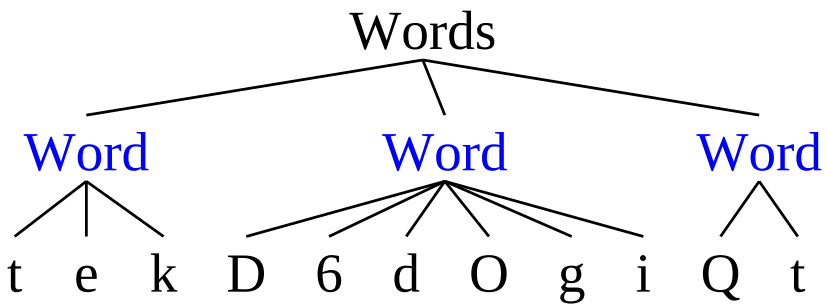
$$\text{Words} \rightarrow \text{Word}^+$$
$$\underline{\text{Word}} \rightarrow \text{Phon}^+$$

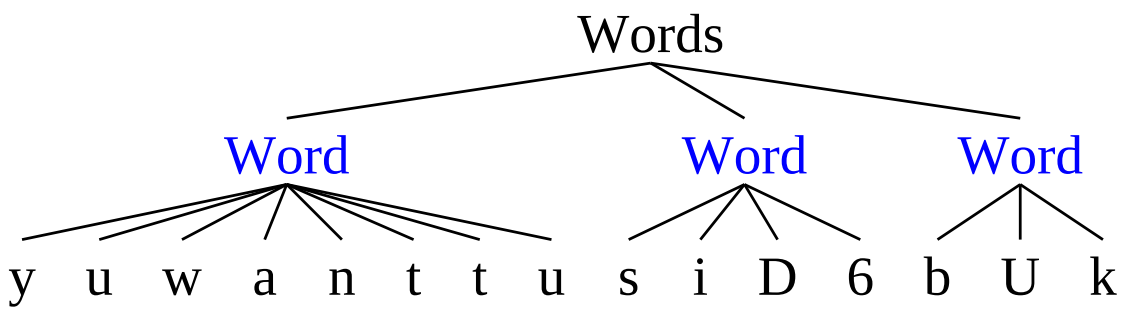
the learnt adapted grammar contains 1,712 rules such as:

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

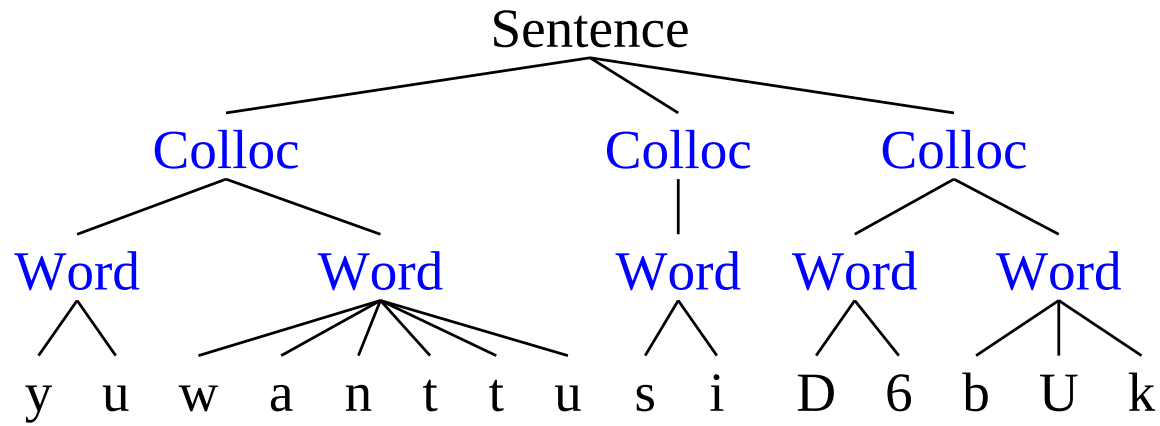
- Unigram word segmentation model assumes each word is generated independently
- But there are strong inter-word dependencies (collocations)
- Unigram model can only capture such dependencies by analyzing collocations as words (Goldwater 2006)





colloc: Collocations ⇒ Words

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

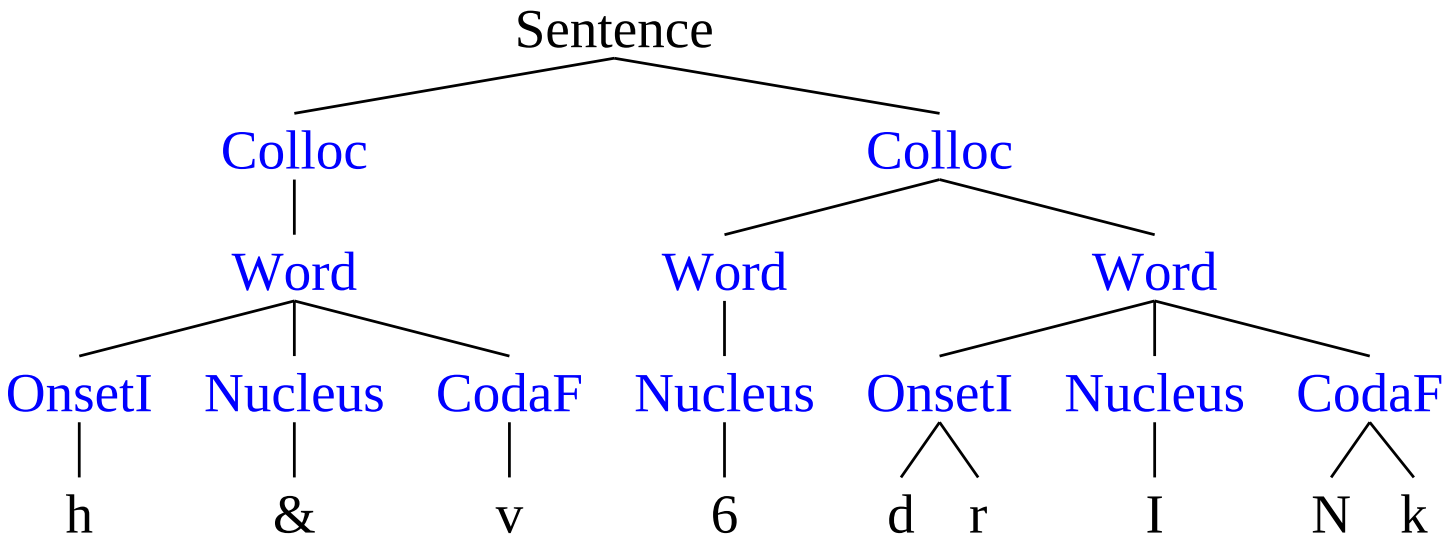


- A Colloc(ation) consists of one or more words
- Both Words and Collocs are adapted (learnt)
- Significantly improves word segmentation accuracy over unigram model (76% f-score; ≈ Goldwater's bigram model)

colloc-syll: Collocations ⇒ Words ⇒ Syllables

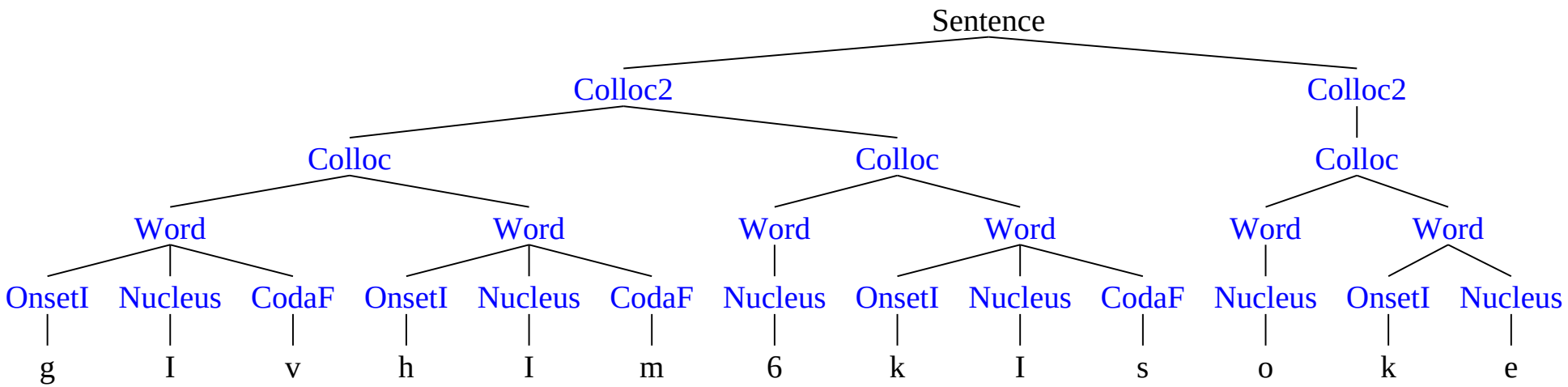
Sentence → Colloc$^+$
Word → SyllableIF
Word → SyllableI (Syllable) (Syllable) SyllableF
Onset → Consonant$^+$
Nucleus → Vowel$^+$

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)

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

Outline

Unsupervised inference via 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) $\mathrm{P}(\mathbf{x}, \mathbf{t})$
- Gibbs sampling algorithm:

  initialize $\mathbf{t}$ somehow (e.g., random trees)
  repeat forever:
    pick an index $j \in 1, \ldots, n$ at random
    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 $\mathrm{P}(\mathbf{t} \mid \mathbf{x})$

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

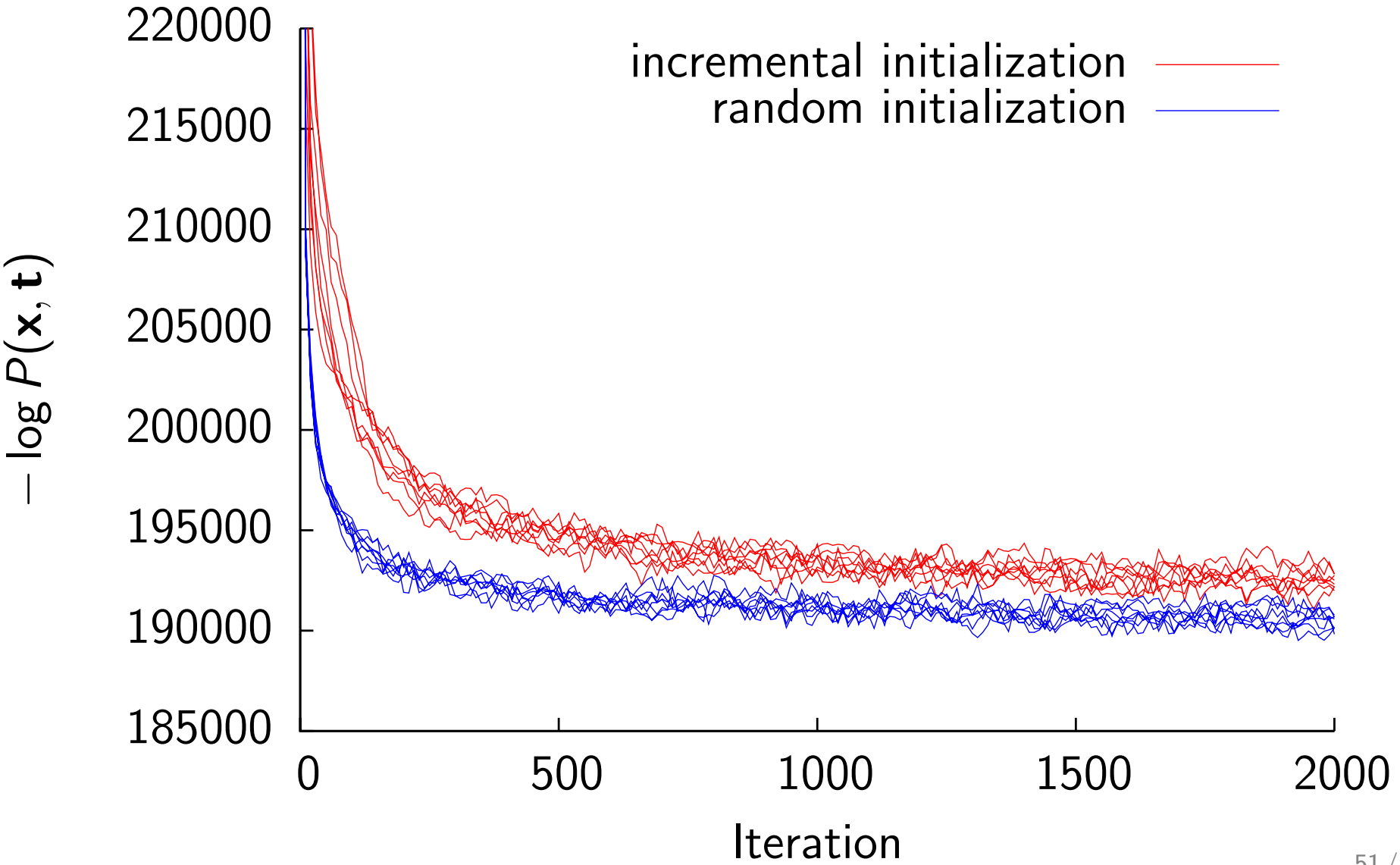
  Random initialization: Assign each string x_i a random parse t_i generated by base PCFG

  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

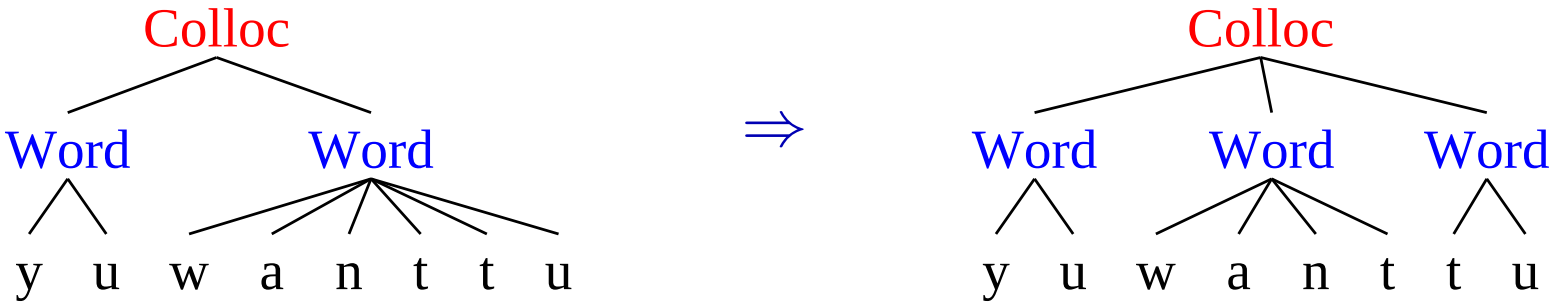
- Each adapted non-terminal has a CRP with tables labelled with parses
- "Rich get richer" $\Rightarrow$ resampling a sentence's parse reuses the same cached subtrees
- *Resample table labels* as well sentence parses
  - A table label may be used in many sentence parses

  $\Rightarrow$ 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

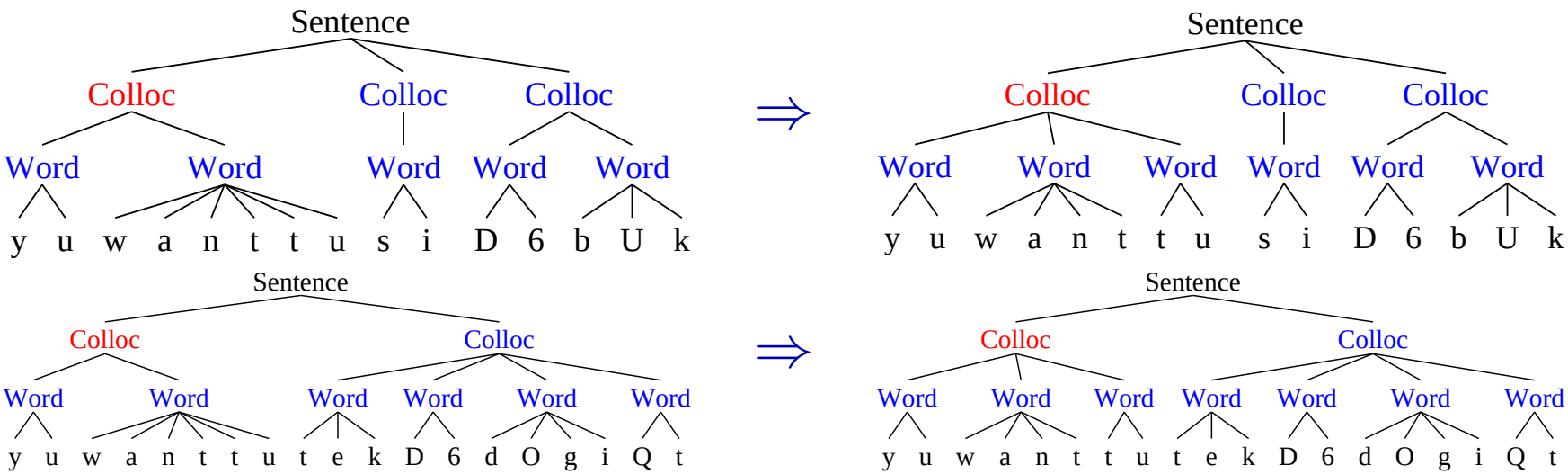
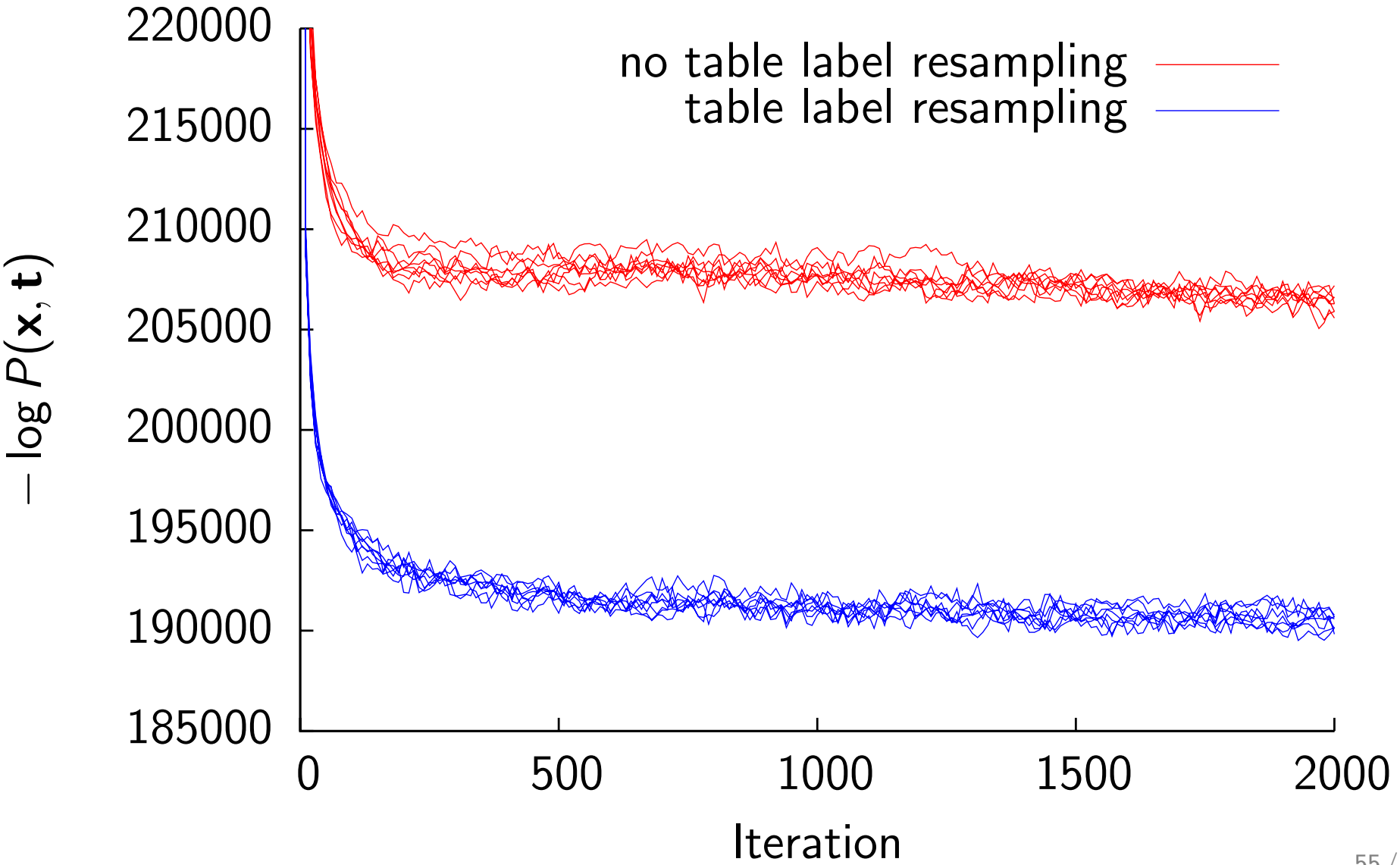


Table label resampling produces much higher-probability parses



Outline

Conclusion

- MCMC sampling is a natural approach to inference in non-parametric models
- Put Bayesian priors on PYP hyperparameters and slice sample
  - improves colloc-syll model by 9%
- Modal decoding from multiple samples
  - improves colloc-syll model by 2%
- Random initialization instead of incremental initialization
  - hurts colloc-syll model by 2%, but produces higher probability parses
- Table label resampling in hierarchical CRPs/PYPs
  - improves colloc-syll model by 20%
- Taken together, performance of colloc-syll model improves from last year's 78% to 87% token f-score.