Learning rules with Adaptor Grammars
(workshop on Prior Knowledge)

Mark Johnson

based on joint work
with Sharon Goldwater and Tom Griffiths

July, 2008

Ideas behind talk

- Most successful statistical learning methods are parametric
  - PCFG learning: given data *and rules*, learn rule probabilities
- *Non-parametric learning* can potentially learn parameters (rules) as well as their values
- *Adaptor grammars*:
  - are a framework for specifying hierarchical nonparametric Bayesian models
  - are approximated by PCFGs, where *number and shape of rules depends on data*
- Prior knowledge (shapes of possible rules) plays a crucial role
  - in word segmentation, need to balance structure above and below the word

Questions this work tries to address

- Can non-parametric hierarchical Bayesian models help us understand language acquisition?
  - *Adaptor grammars* are a framework for easily constructing these models
- How useful are various potential information sources for language acquisition? (here, word segmentation)
  - Changing grammar changes the generalizations it can learn
  - How much do we have to "build in"?
- Are the information sources most useful for English also useful for other languages?
- Are there *synergies* in learning? Is it better to learn several things at once?

Why not study syntax?

- Bayesian PCFG estimation works well *on toy examples* but simple models (at least) don't work well on real data
- Lower level phenomena (word segmentation, phonology, morphology) may be:
  - less dependent on meaning,
  - learnt earlier by children, and
  - easier to learn than syntax

  yet still exhibit structural complexities

$\Rightarrow$ Unsupervised morphological analysis:

Example: $e_{\triangle}x_{\triangle}p_{\triangle}a_{\triangle}n_{\triangle}d_{\blacktriangle}e_{\triangle}d$

$\Rightarrow$ Word segmentation: segment *broad phonemic* utterances

Example: $y_{\triangle}u_{\blacktriangle}w_{\triangle}a_{\triangle}n_{\triangle}t_{\blacktriangle}t_{\triangle}u_{\blacktriangle}s_{\triangle}i_{\blacktriangle}D_{\triangle}6_{\blacktriangle}b_{\triangle}U_{\triangle}k$

Outline

Probabilistic context-free grammars

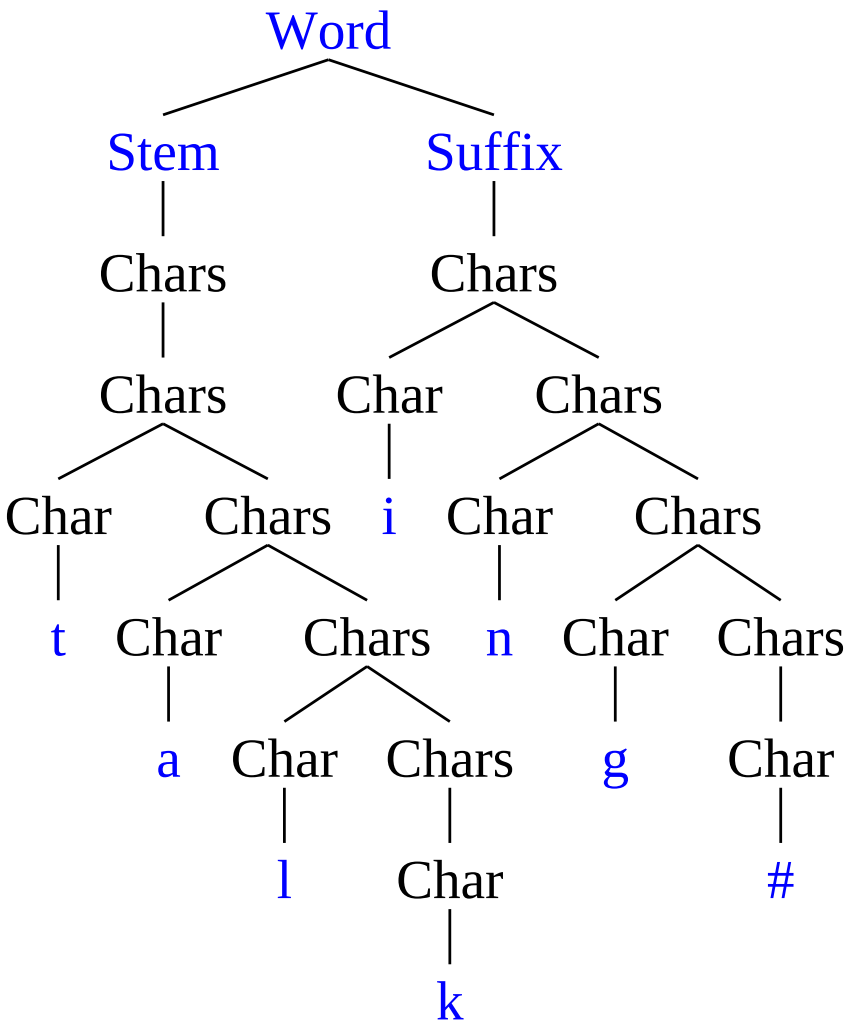
- Rules in *Context-Free Grammars* (CFGs) expand nonterminals into sequences of terminals and nonterminals
- A *Probabilistic CFG* (PCFG) associates each nonterminal with a multinomial distribution over the rules that expand it
- Probability of a tree is the *product of the probabilities of the rules*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

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

A CFG for stem-suffix morphology

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



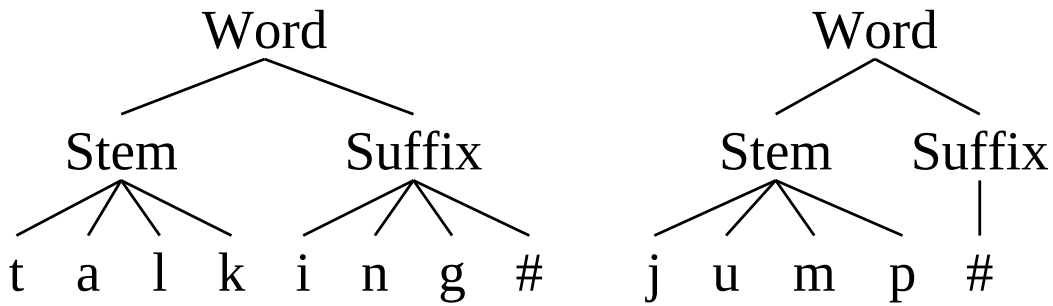
- Grammar's trees can represent any segmentation of words into stems and suffixes

$\Rightarrow$ 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

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



- 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

Nonparametric extensions of PCFGs

- Dirichlet Processes are a natural nonparametric extension of Dirichlet-multinomials that underlie Bayesian PCFGs
- 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 ...

Adaptor grammars: informal description

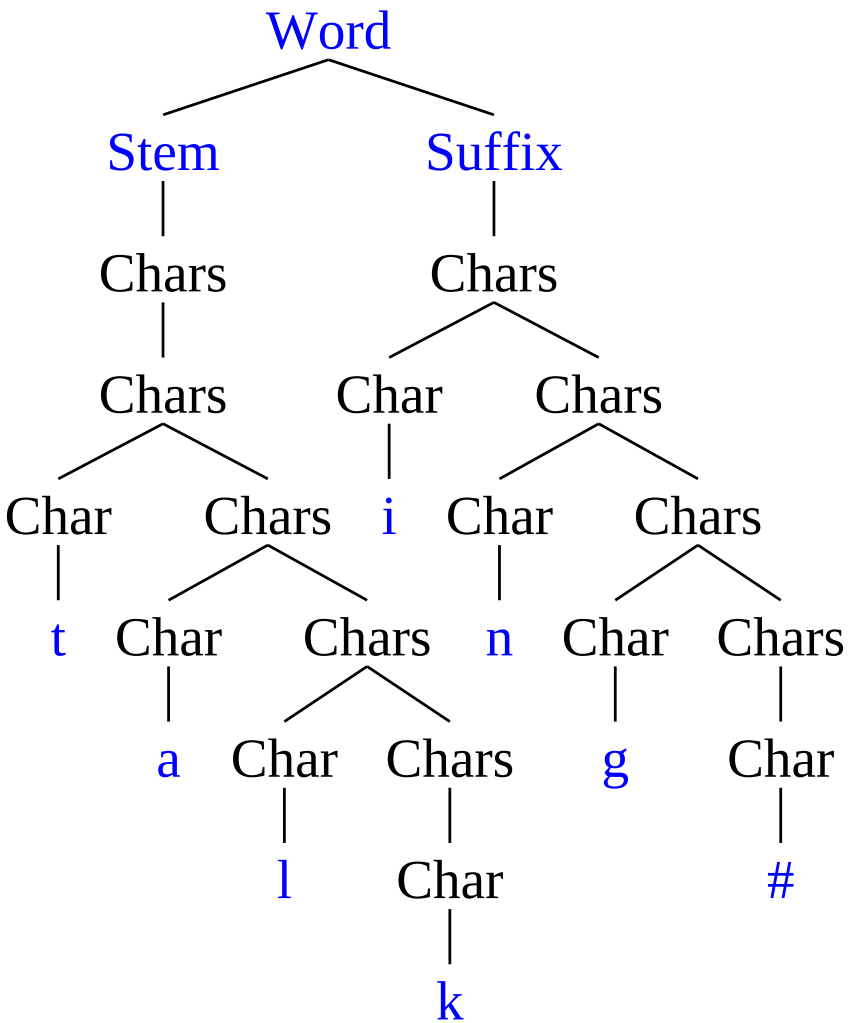
- An adaptor grammar has a set of CFG rules
- These determine the possible structures as in a CFG
- A subset of the nonterminals are *adapted*
- *Unadapted nonterminals* expand by picking a rule and recursively expanding its children, as in a PCFG
- *Adapted nonterminals* can expand in two ways:
  - 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 $\alpha_A \theta_{A \rightarrow \beta}$

An Adaptor Grammar for stem-suffix morphology

Word	→	Stem Suffix	Chars	→	Char
Stem	→	Chars	Chars	→	Char Chars
Suffix	→	Chars	Char	→	a | b | c | . . .

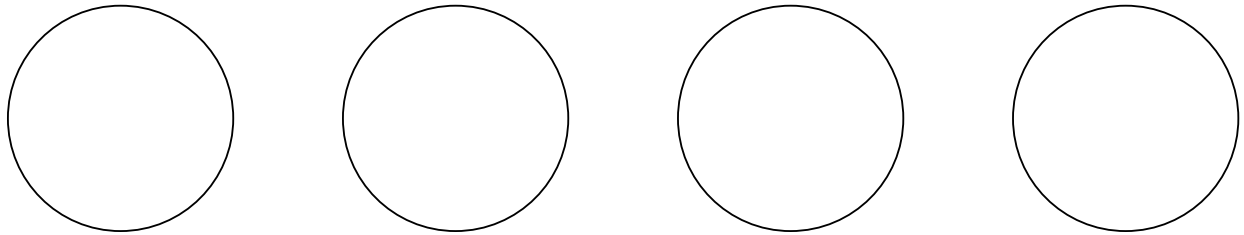


- Word, Stem and Suffix are *adapted*

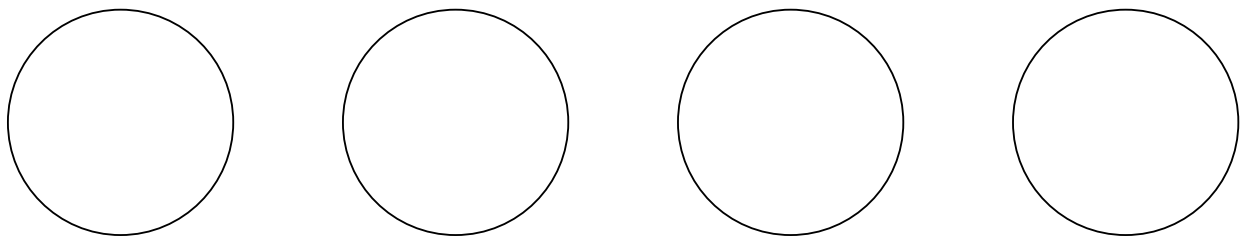
⇒ learns probabilities of whole Stem and Suffix and Word subtrees

Morphology adaptor grammar (0)

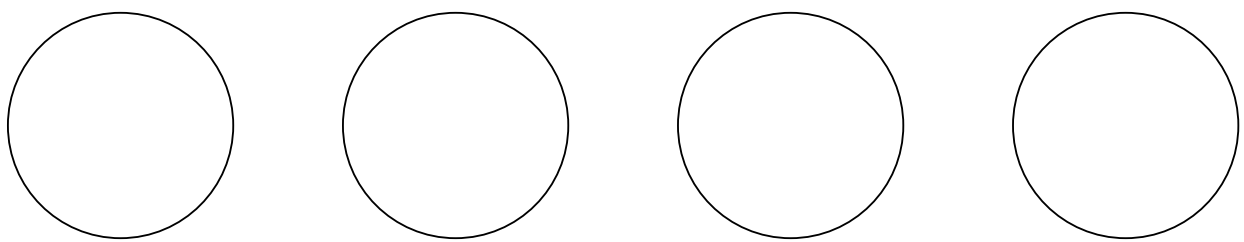
Word restaurant
Word → Stem Suffix



Stem restaurant
Stem → Phoneme*

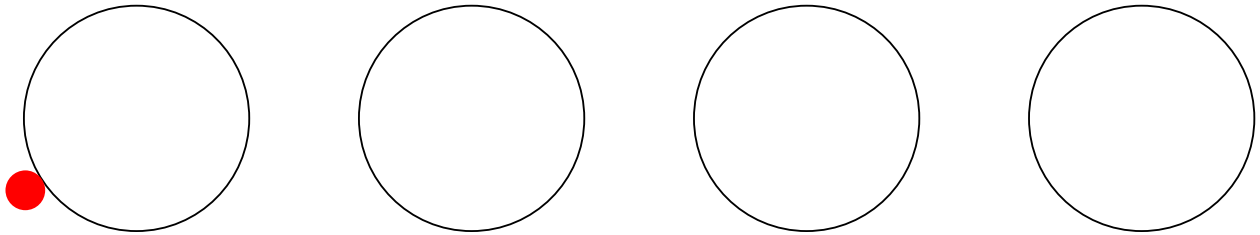


Suffix restaurant
Suffix → Phoneme^{+}



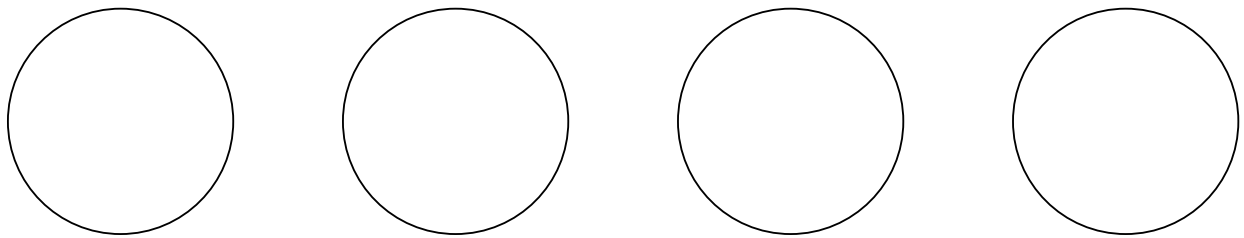
Morphology adaptor grammar (1a)

Word restaurant
Word → Stem Suffix



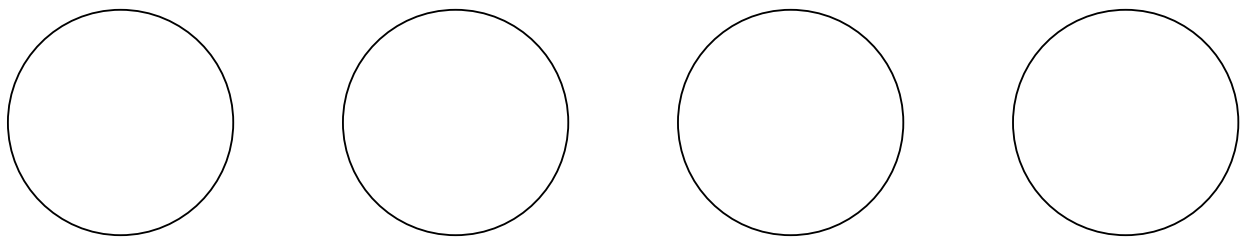
...

Stem restaurant
Stem → Phoneme*



...

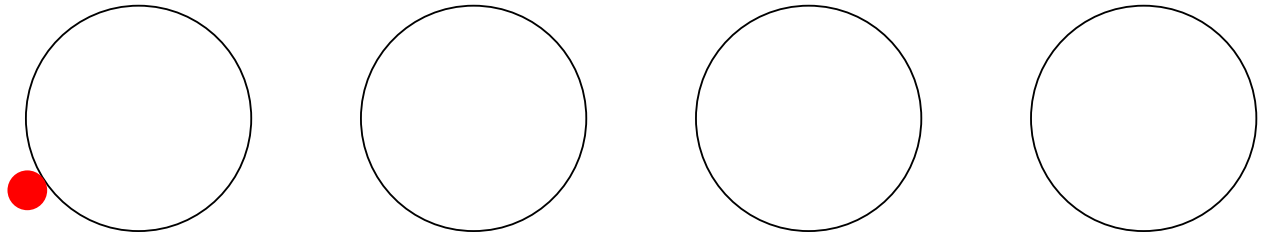
Suffix restaurant
Suffix → Phoneme^{+}



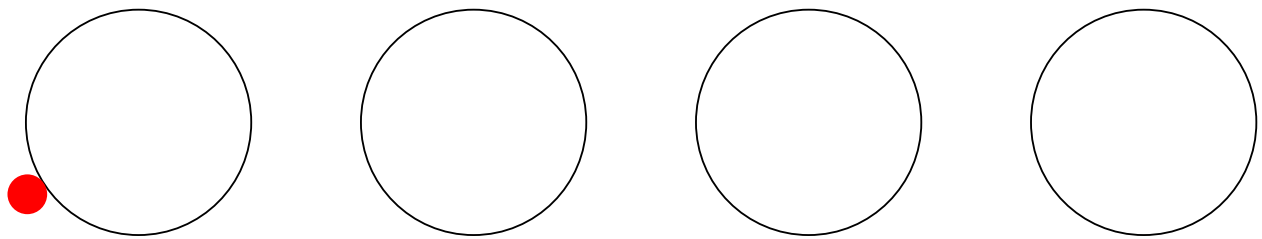
...

Morphology adaptor grammar (1b)

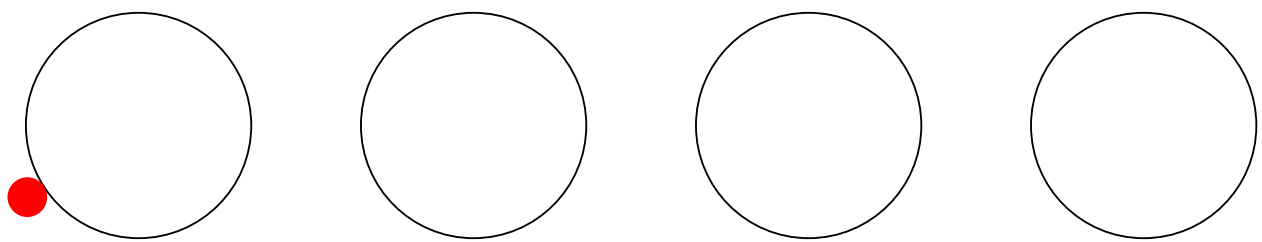
Word restaurant
Word → Stem Suffix



Stem restaurant
Stem → Phoneme*

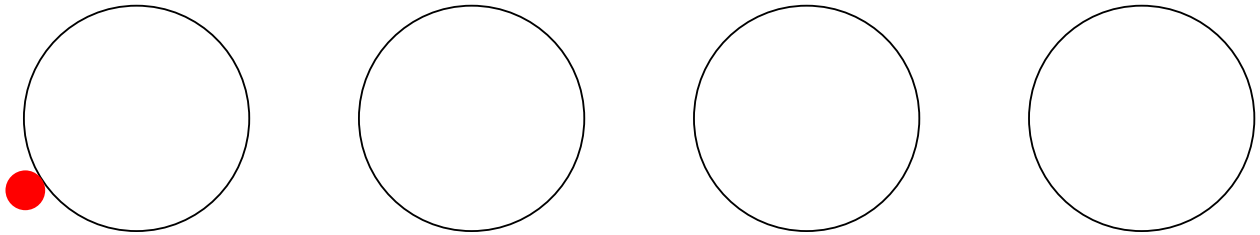


Suffix restaurant
Suffix → Phoneme^{+}

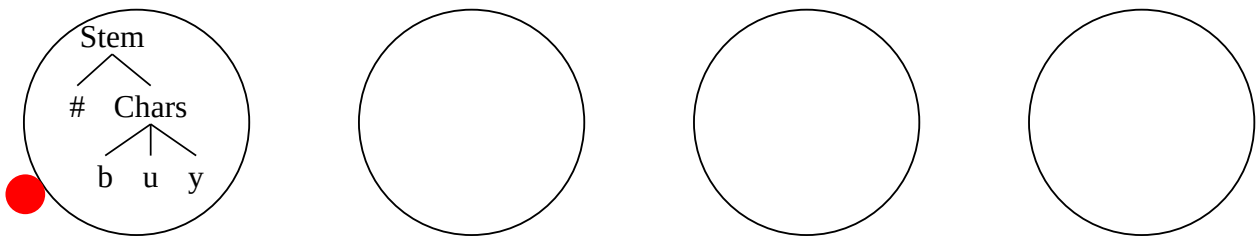


Morphology adaptor grammar (1c)

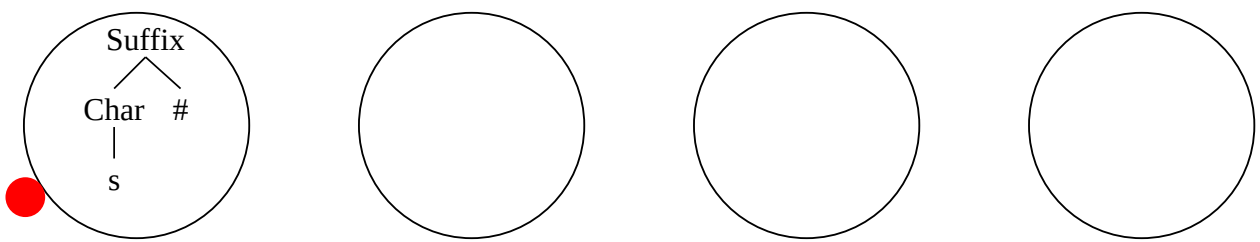
Word restaurant
Word → Stem Suffix



Stem restaurant
Stem → Phoneme*

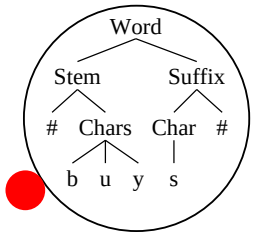


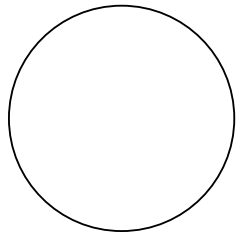
Suffix restaurant
Suffix → Phoneme^{+}

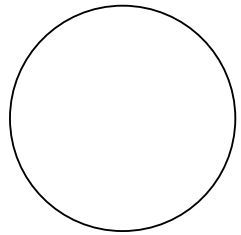


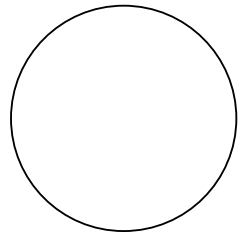
Morphology adaptor grammar (1d)

Word restaurant
Word → Stem Suffix



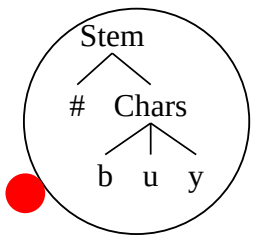


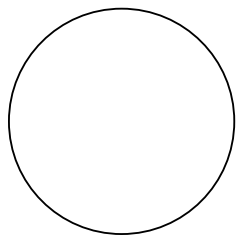


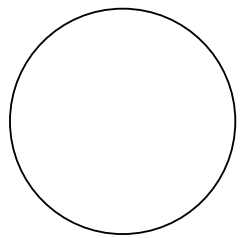


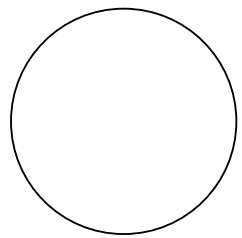
...

Stem restaurant
Stem → Phoneme*



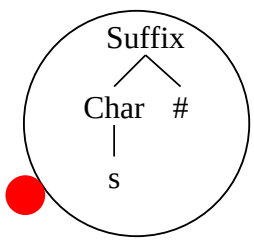


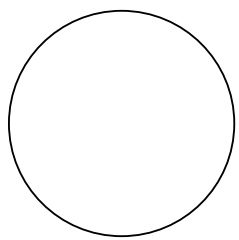


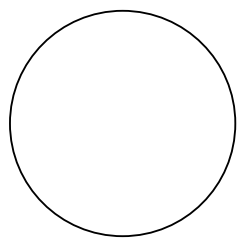


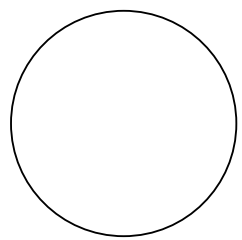
...

Suffix restaurant
Suffix → Phoneme^{+}





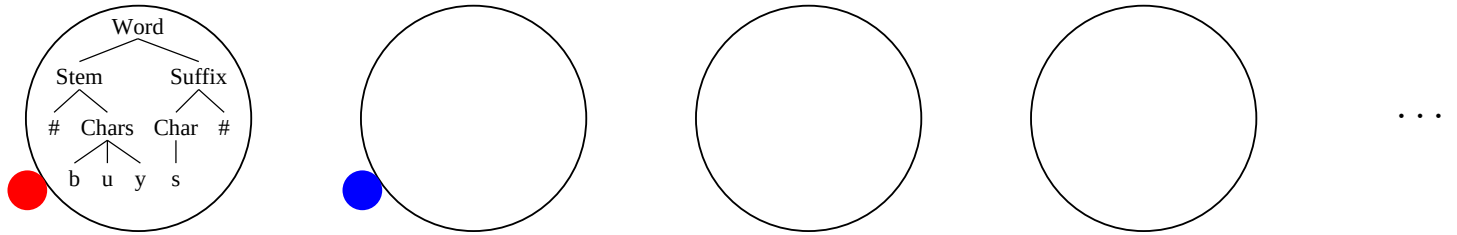




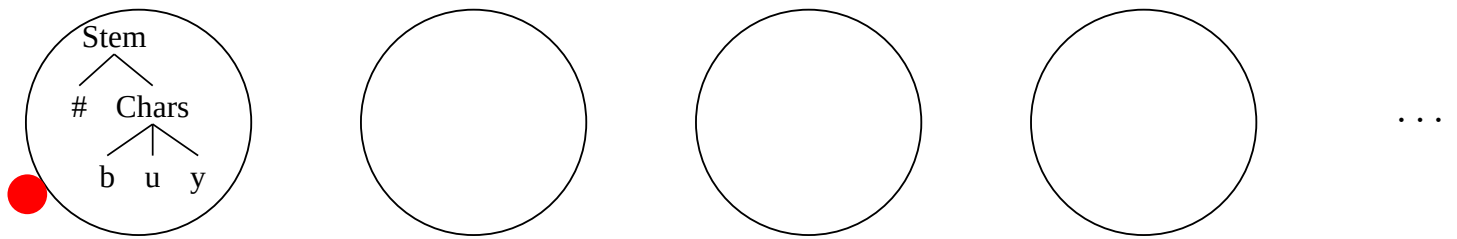
...

Morphology adaptor grammar (2a)

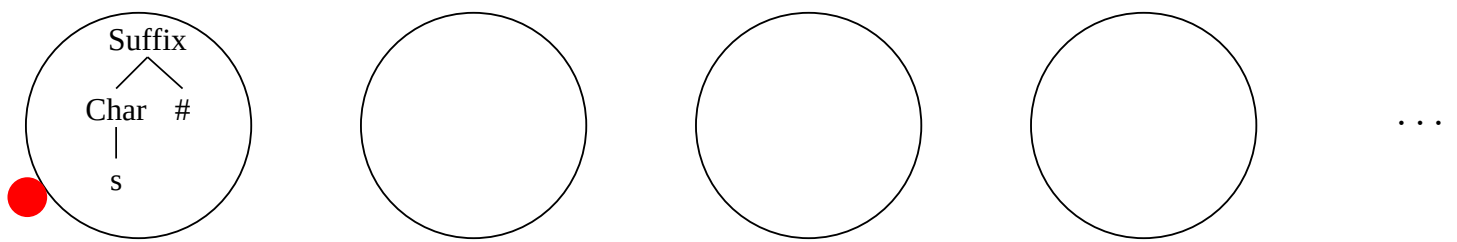
Word restaurant
Word → Stem Suffix



Stem restaurant
Stem → Phoneme*

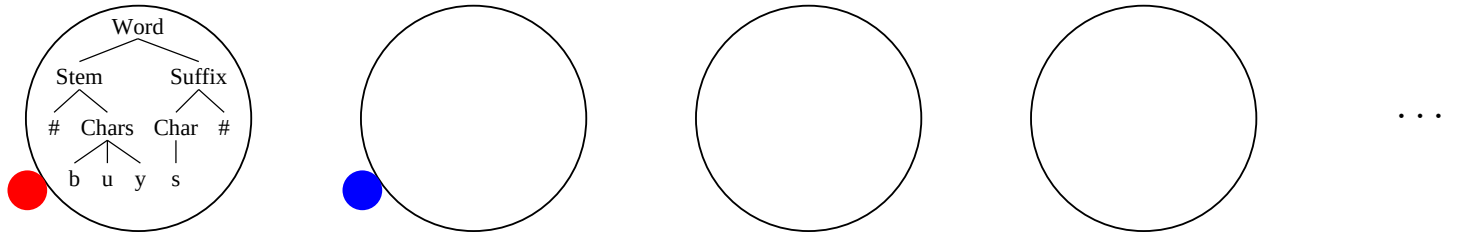


Suffix restaurant
Suffix → Phoneme^{+}

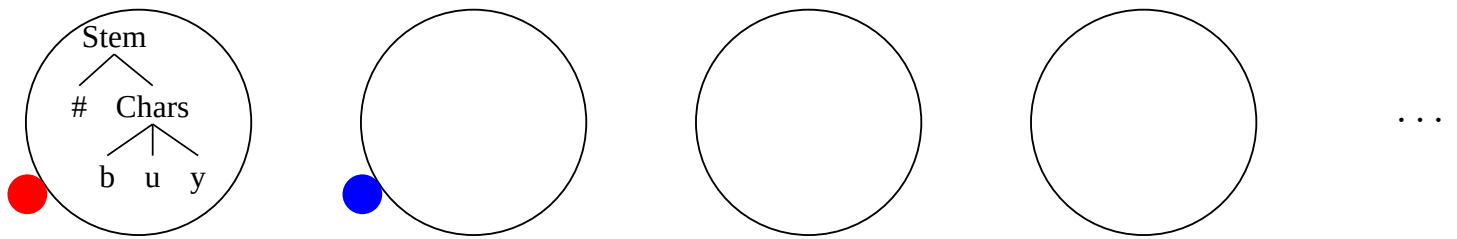


Morphology adaptor grammar (2b)

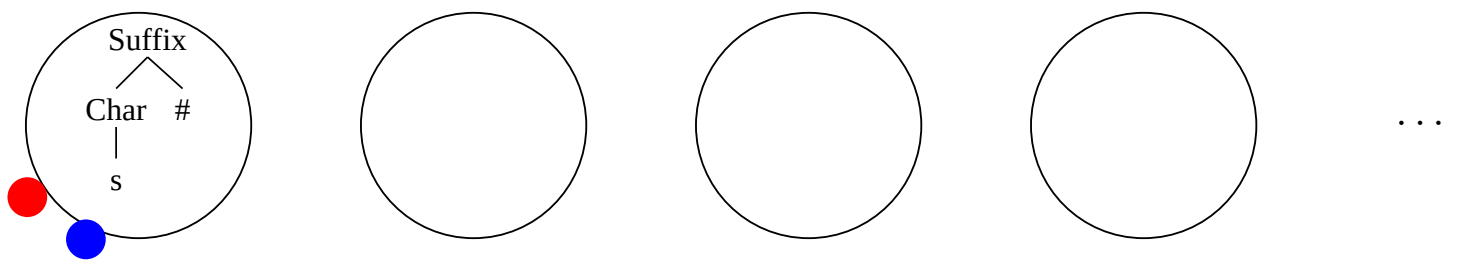
Word restaurant
Word → Stem Suffix



Stem restaurant
Stem → Phoneme*

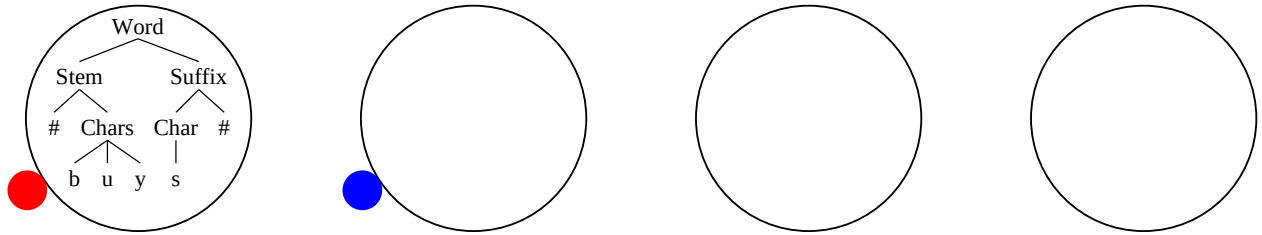


Suffix restaurant
Suffix → Phoneme^{+}

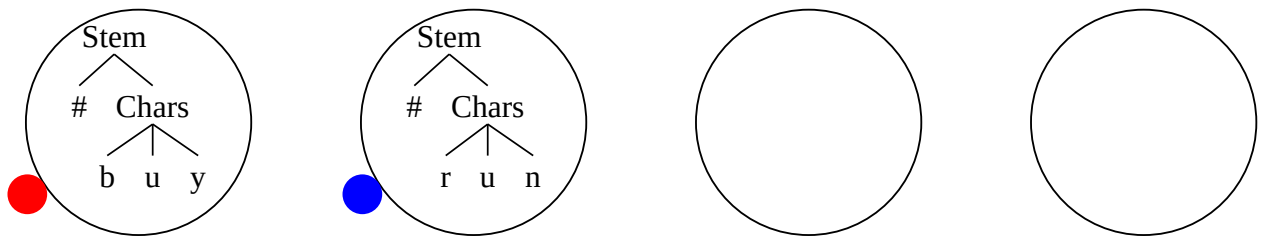


Morphology adaptor grammar (2c)

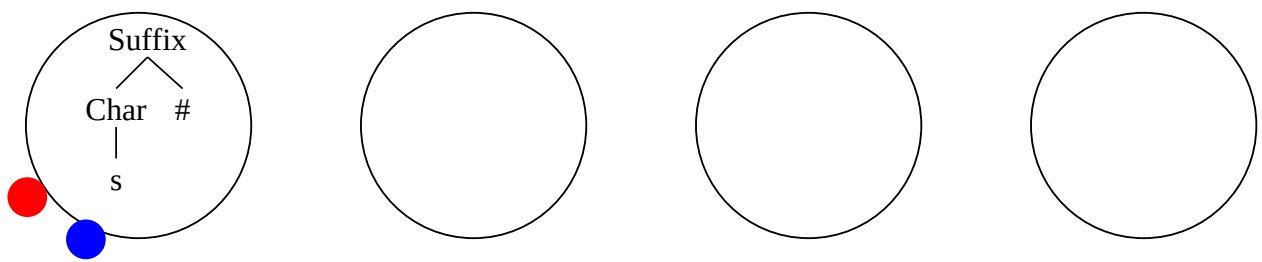
Word restaurant
Word → Stem Suffix



Stem restaurant
Stem → Phoneme*

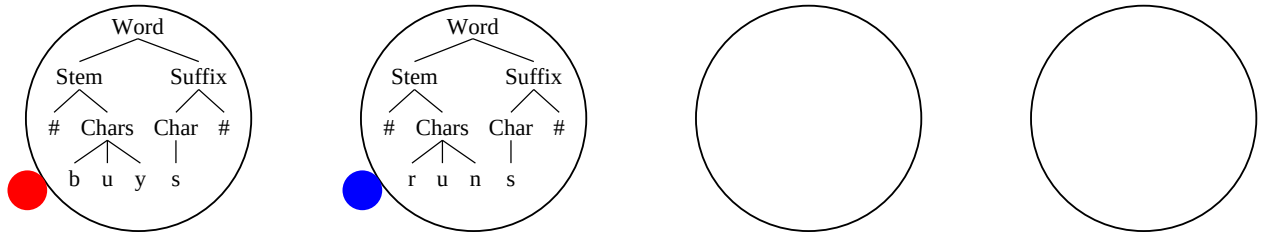


Suffix restaurant
Suffix → Phoneme^{+}

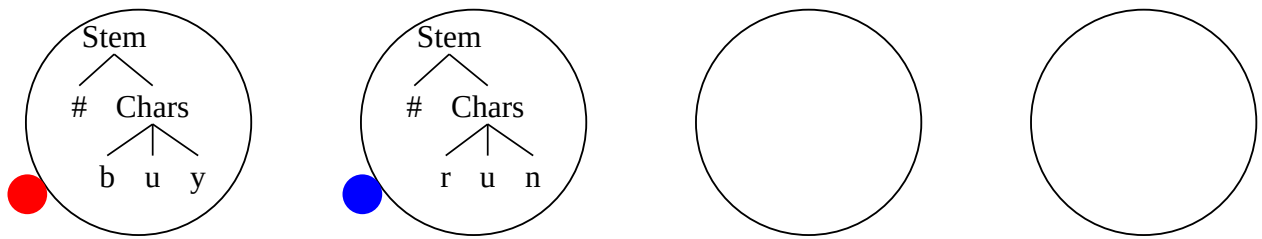


Morphology adaptor grammar (2d)

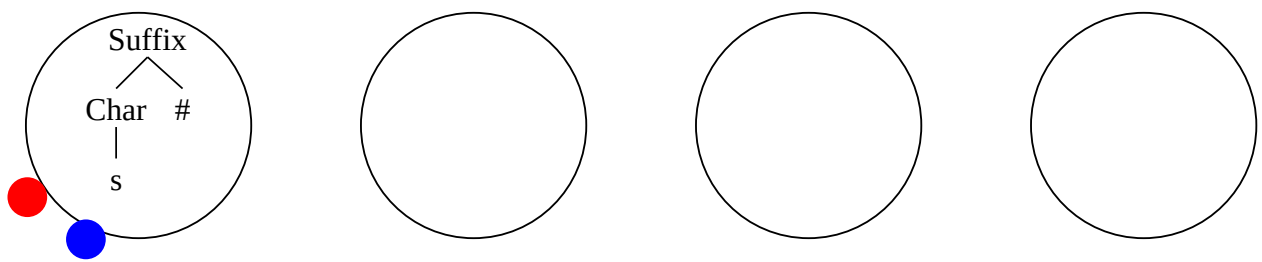
Word restaurant
Word → Stem Suffix



Stem restaurant
Stem → Phoneme*

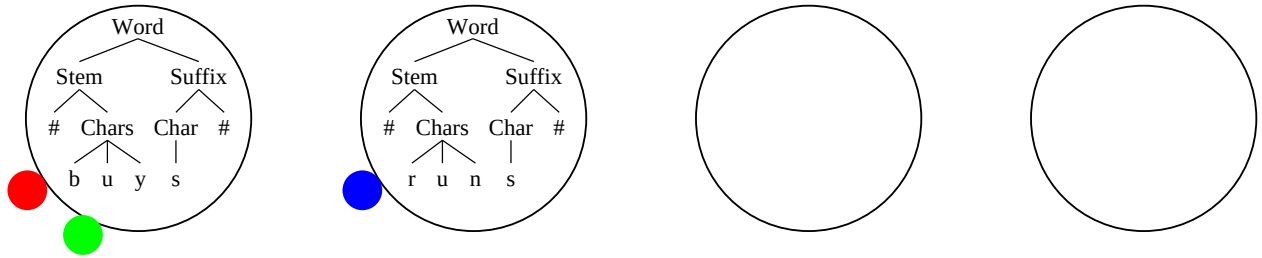


Suffix restaurant
Suffix → Phoneme^{+}

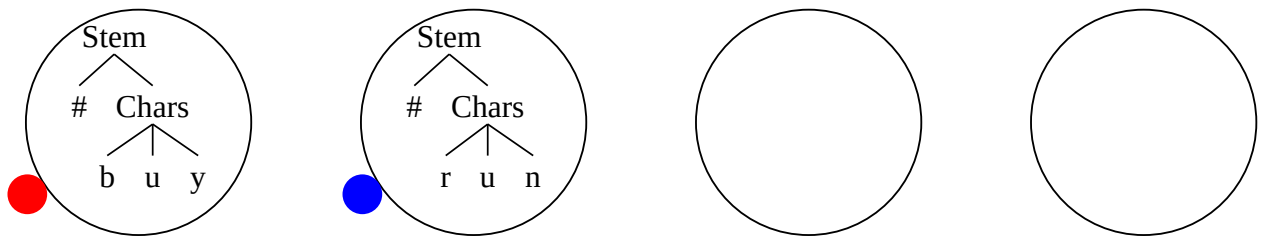


Morphology adaptor grammar (3)

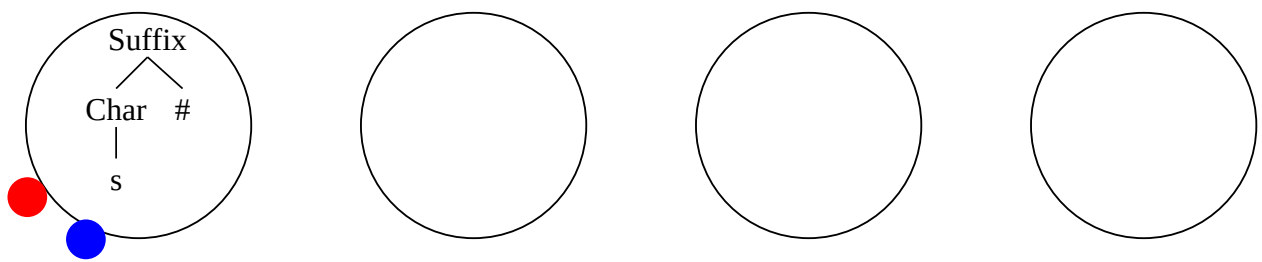
Word restaurant
Word → Stem Suffix



Stem restaurant
Stem → Phoneme*

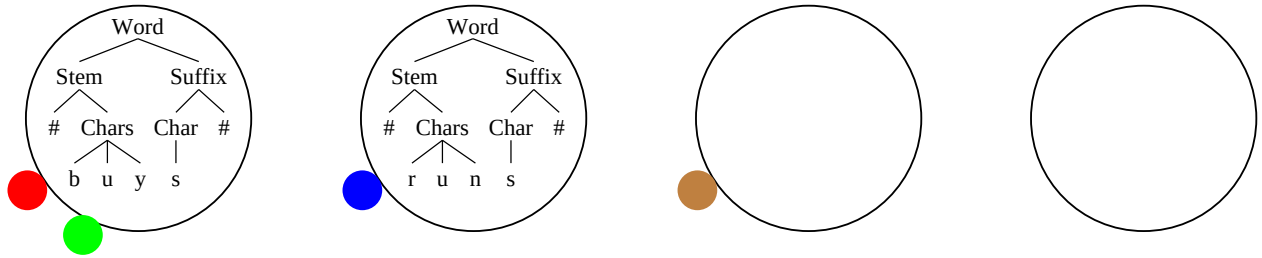


Suffix restaurant
Suffix → Phoneme^{+}

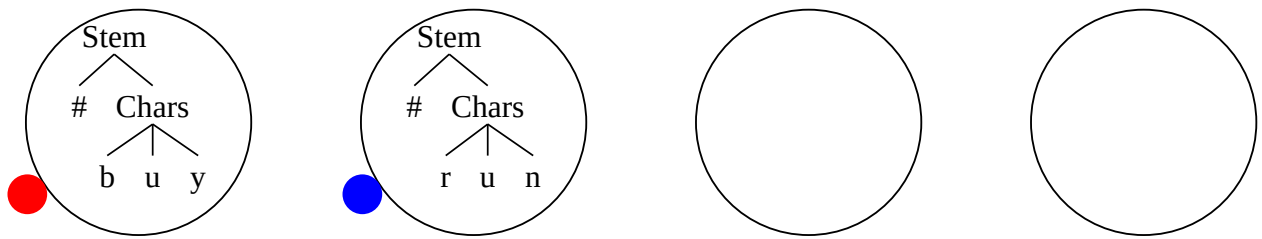


Morphology adaptor grammar (4a)

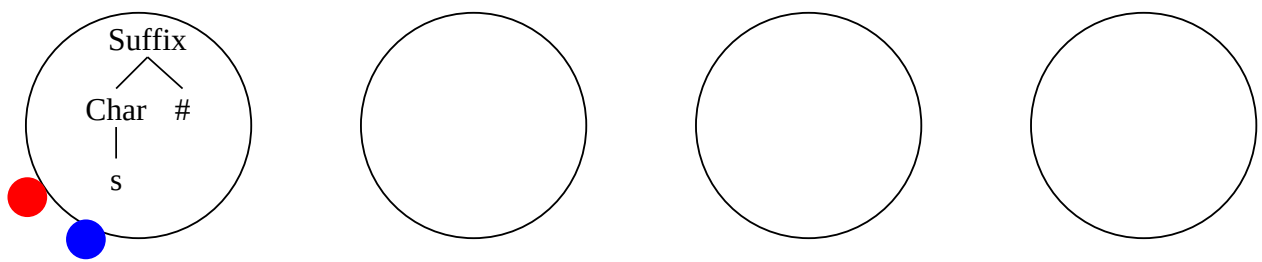
Word restaurant
Word → Stem Suffix



Stem restaurant
Stem → Phoneme*

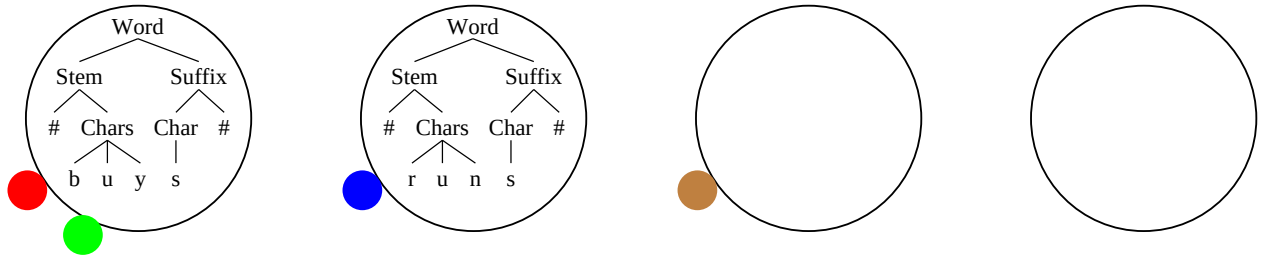


Suffix restaurant
Suffix → Phoneme^{+}

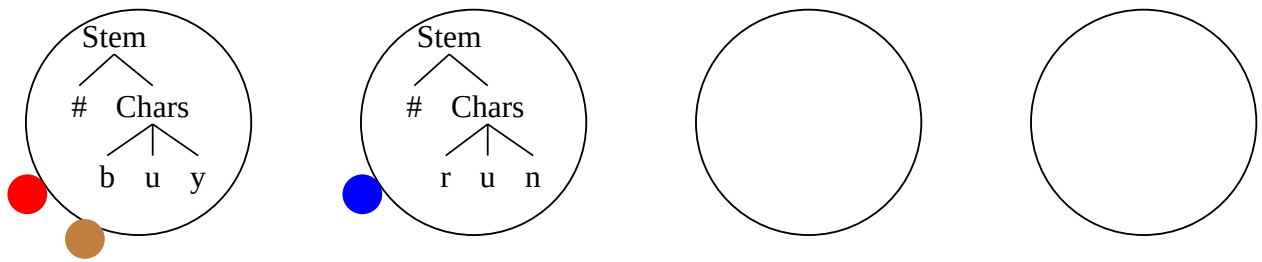


Morphology adaptor grammar (4b)

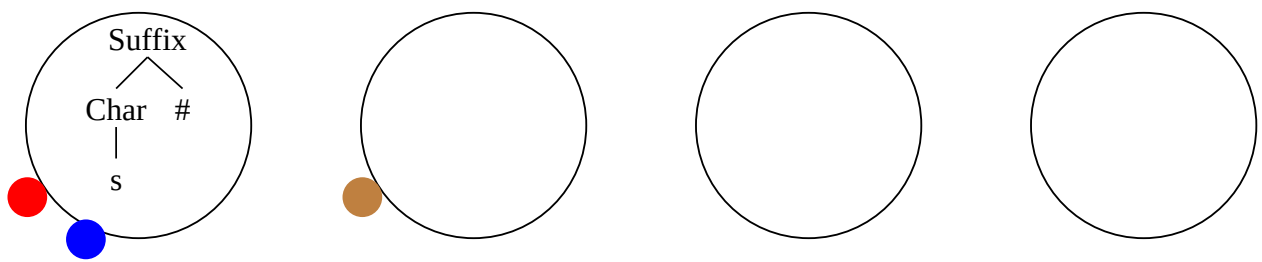
Word restaurant
Word → Stem Suffix



Stem restaurant
Stem → Phoneme*

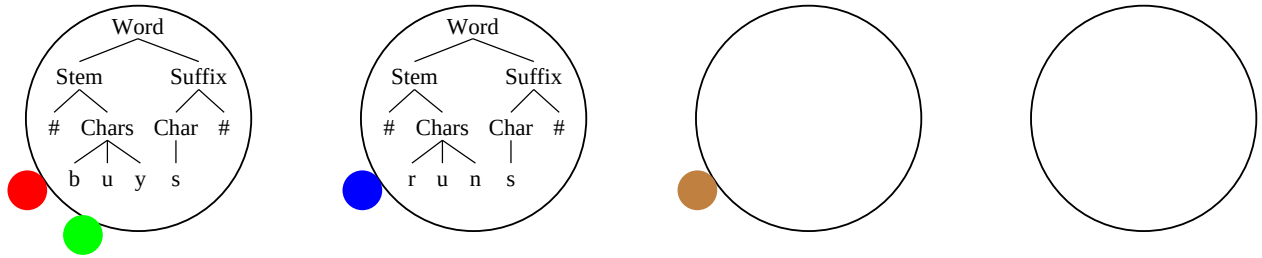


Suffix restaurant
Suffix → Phoneme^{+}

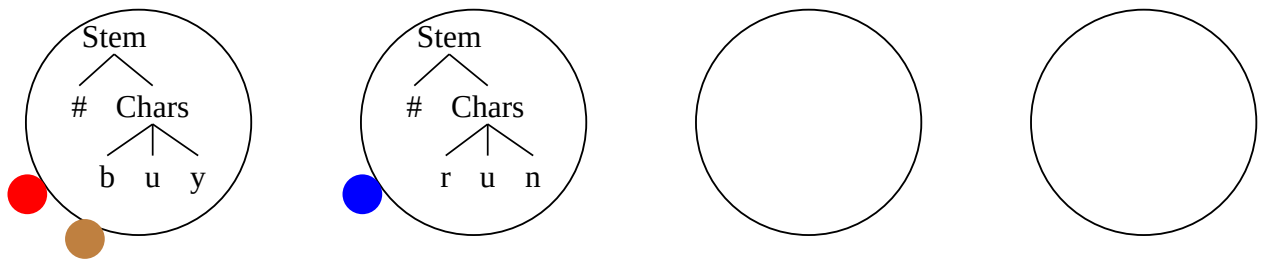


Morphology adaptor grammar (4c)

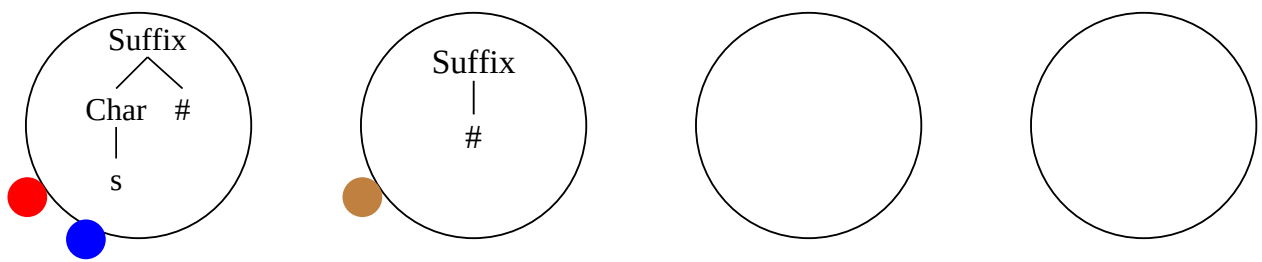
Word restaurant
Word → Stem Suffix



Stem restaurant
Stem → Phoneme*

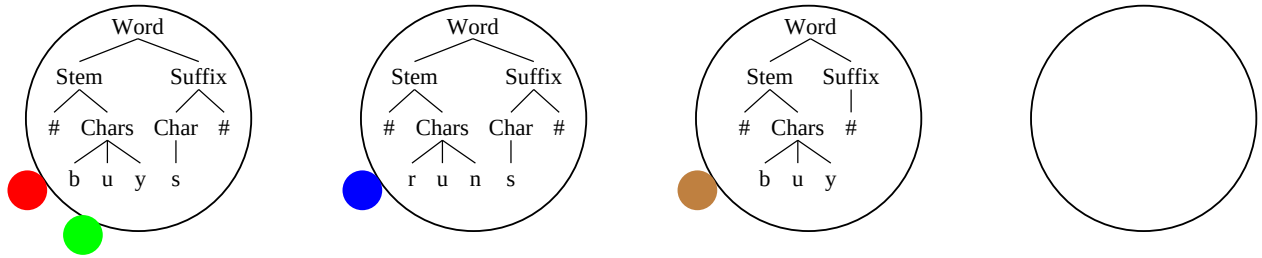


Suffix restaurant
Suffix → Phoneme^{+}

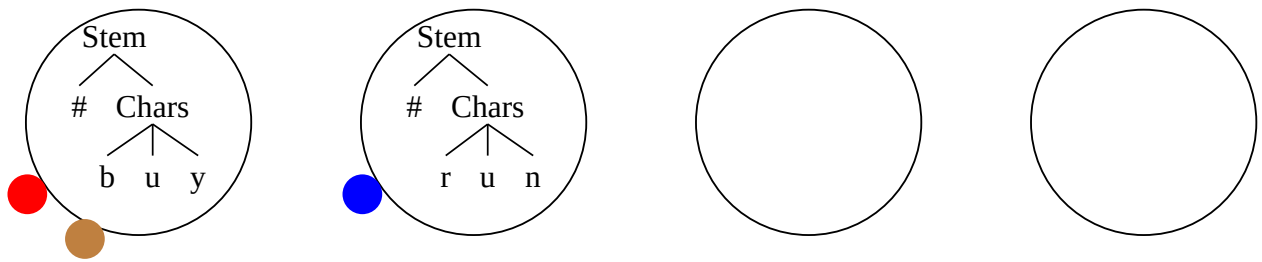


Morphology adaptor grammar (4d)

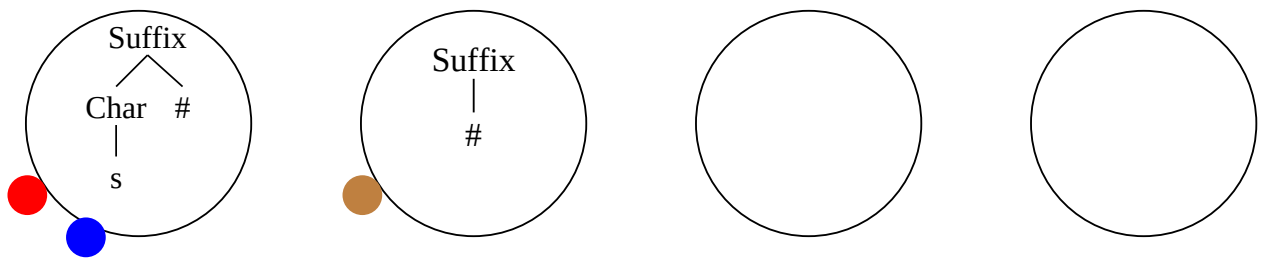
Word restaurant
Word → Stem Suffix



Stem restaurant
Stem → Phoneme*



Suffix restaurant
Suffix → Phoneme^{+}

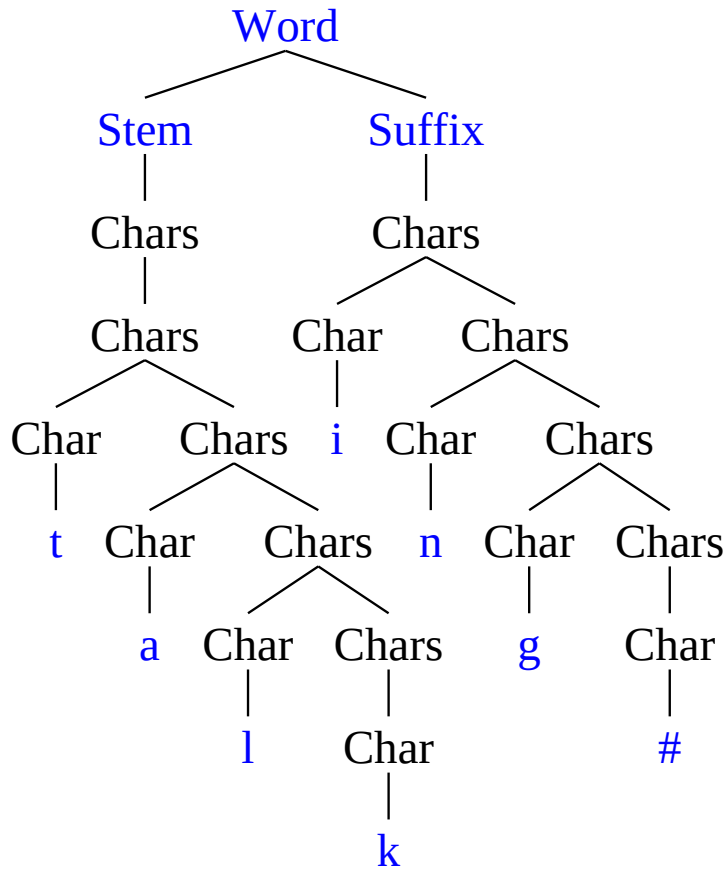


Morphology as a Hierarchical Dirichlet Process

- Expand each Word into:
  - a previously generated subtree with prob. $\propto$ number of times subtree was generated before
  - a Stem and Suffix with prob. $\propto \alpha_{\text{Word}}\mathsf{P}(\text{Stem})\mathsf{P}(\text{Suffix})$
- Expand Stem into:
  - a sequence of Phoneme with prob. $\propto$ number of times Stem expanded to this sequence before
  - a sequence of Phoneme generated by PCFG rules with prob. $\propto \alpha_{\text{Stem}}\mathsf{P}(\text{Phonemes})$
- Suffix expands in same way as Stem
- This is a *Hierarchical Dirichlet Process* where Stem and Suffix distributions define the base distribution for Word DP

Bayesian hierarchy inverts grammatical hierarchy

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



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

Outline

Unigram adaptor grammar for English

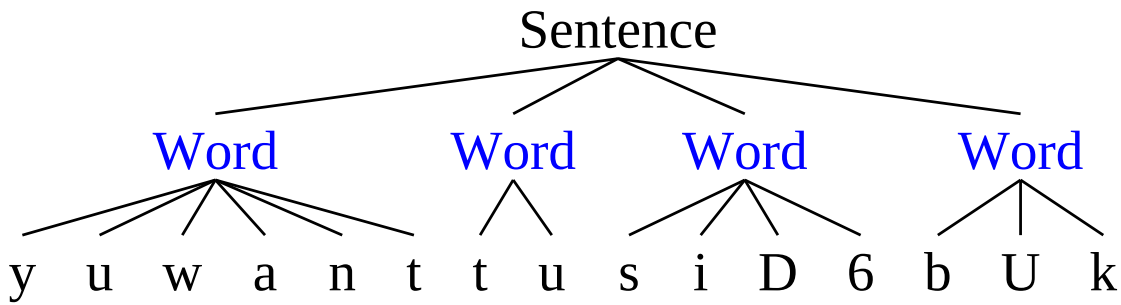
- Adaptor grammar (adapted nonterminals highlighted):

Sentence → Words
Words → Word
Words → Word Words
Word → Phonemes
Phonemes → Phoneme
Phonemes → Phoneme Phonemes

or in abbreviated format:

Sentence → Word^+
Word → Phoneme^+

- Sample parse (only showing root and adapted nonterminals):



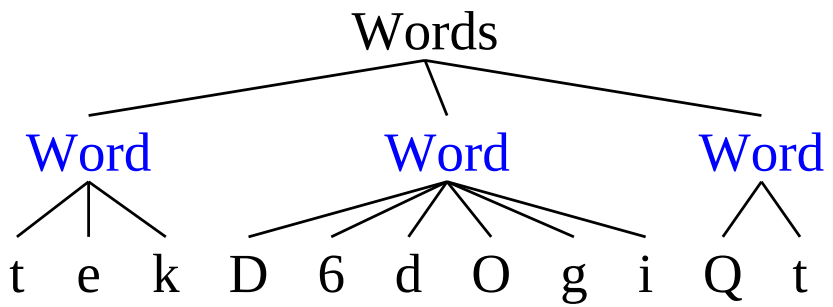
- Word segmentation f-score = 0.55 (same as Goldwater et al)

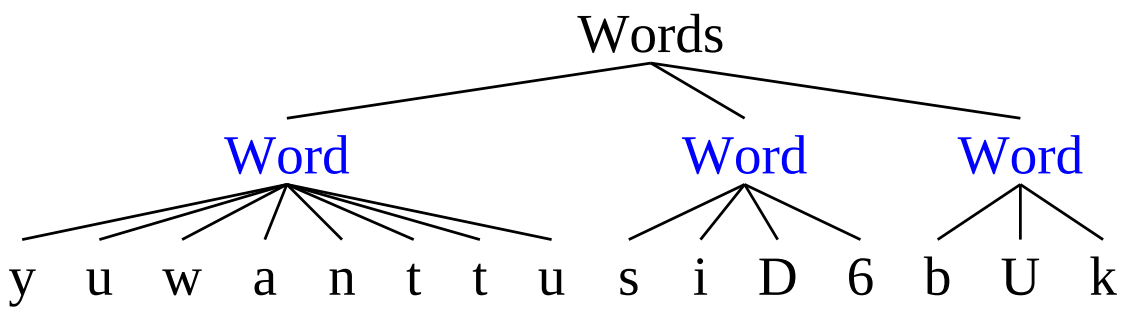
Unigram word grammar as a Dirichlet Process

- Unigram word grammar implements unigram word segmentation model of Goldwater et al (2006)
- Generative process:
  - expand Sentence into a sequence of Words using PCFG rules
  - expand each Word into:
    - a sequence of Phonemes with prob. $\propto$ number of times Word expanded to this sequence before
    - a sequence of phonemes generated by PCFG rules with prob. $\propto \alpha_{\text{Word}}$
- This is a *Dirichlet Process* where the PCFG rules expanding Word define the *base distribution*

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 (Goldwater 2006)





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:

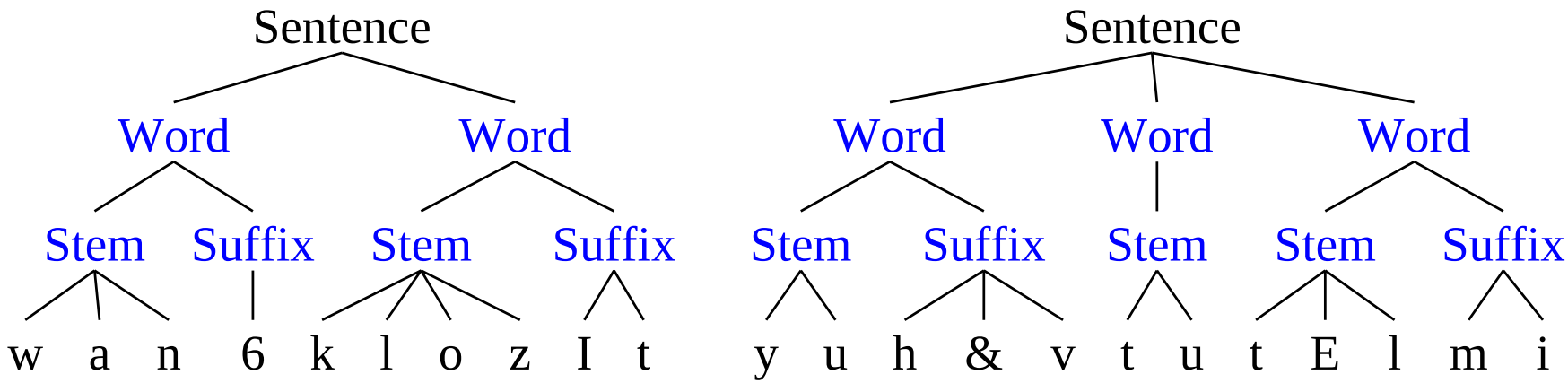
15758	Words → Word Words
9791	Words → Word
1660	Word → Phoneme^{+}
402	Word → y u
137	Word → I n
111	Word → w I T
100	Word → D 6 d O g i
45	Word → I n D 6
20	Word → I n D 6 h Q s

Unigram morphology adaptor grammar

- Adaptor grammar memorizes Word, Stem and Suffix:

$$\text{Sentence} \rightarrow \text{Word}^+$$
$$\text{Word} \rightarrow \text{Stem (Suffix)}$$
$$\text{Stem} \rightarrow \text{Phoneme}^+$$
$$\text{Suffix} \rightarrow \text{Phoneme}^+$$

- Sample parse:



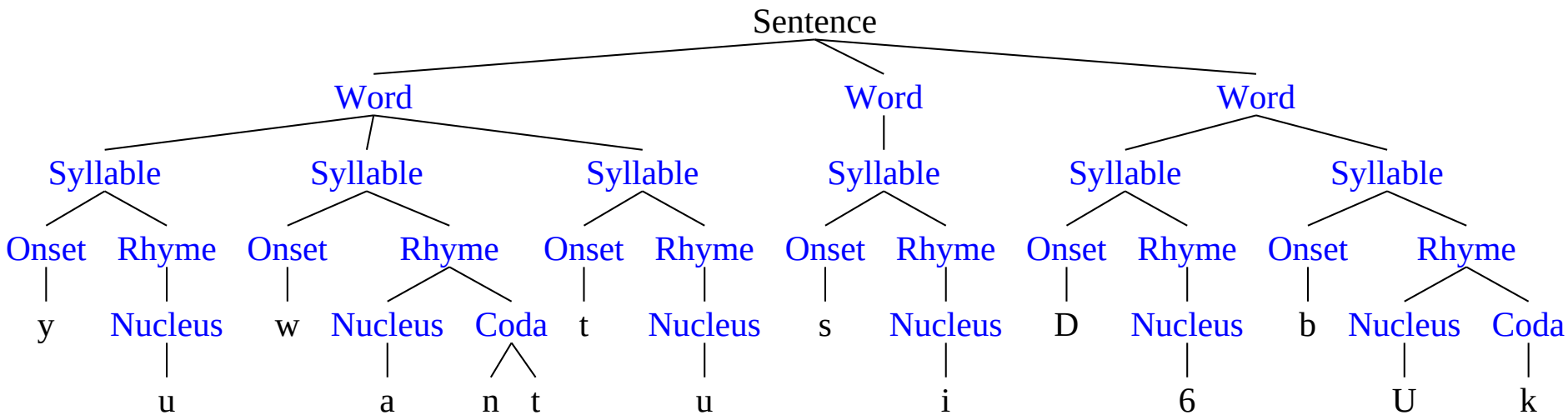
- Combines Goldwater's morphology and unigram model
- Word segmentation f-score = 0.46 (worse than unigram)
- Tends to misanalyse words as Stems or Suffixes

Simultaneously learning word segmentation and syllable structure

$$
\begin{aligned}
\text{Sentence} &\rightarrow \text{Word}^{+} \\
\text{Word} &\rightarrow \text{Syllable}^{+} \\
\text{Syllable} &\rightarrow (\text{Onset})\ \text{Rhyme} \\
\text{Onset} &\rightarrow \text{Consonant}^{+} \\
\text{Rhyme} &\rightarrow \text{Nucleus}\ (\text{Coda}) \\
\text{Nucleus} &\rightarrow \text{Vowel}^{+} \\
\text{Coda} &\rightarrow \text{Consonant}^{+}
\end{aligned}
$$

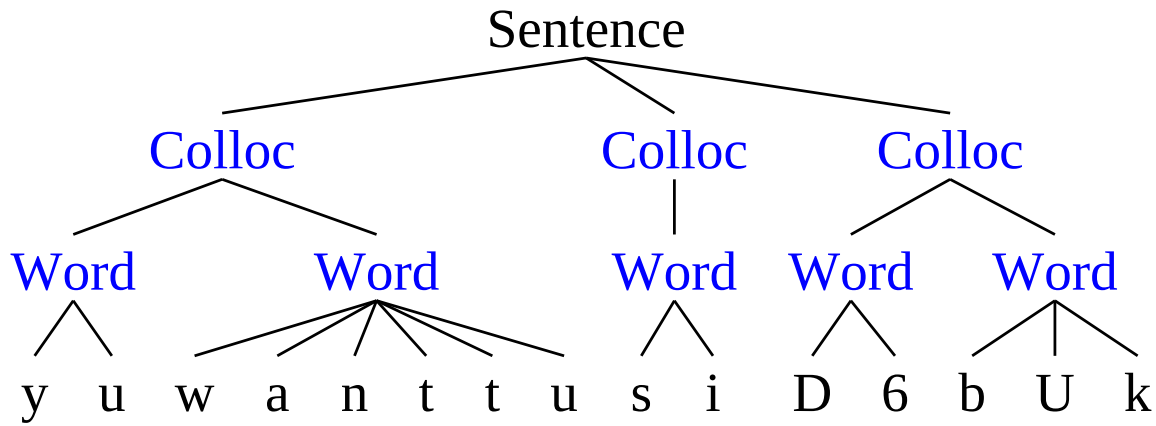
- Word, Onset, Nucleus and Coda are all adapted
- Seems to do a fairly good job of identifying syllable boundaries
- Does a poor job of word segmentation (32% token f-score)
  Words are undersegmented, just as in unigram model
- Segmentation gets better if we distinguish word-initial Onsets and word-final Codas (46% f-score)

Simultaneous word segmentation and syllable structure



Modeling collocations improves segmentation

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

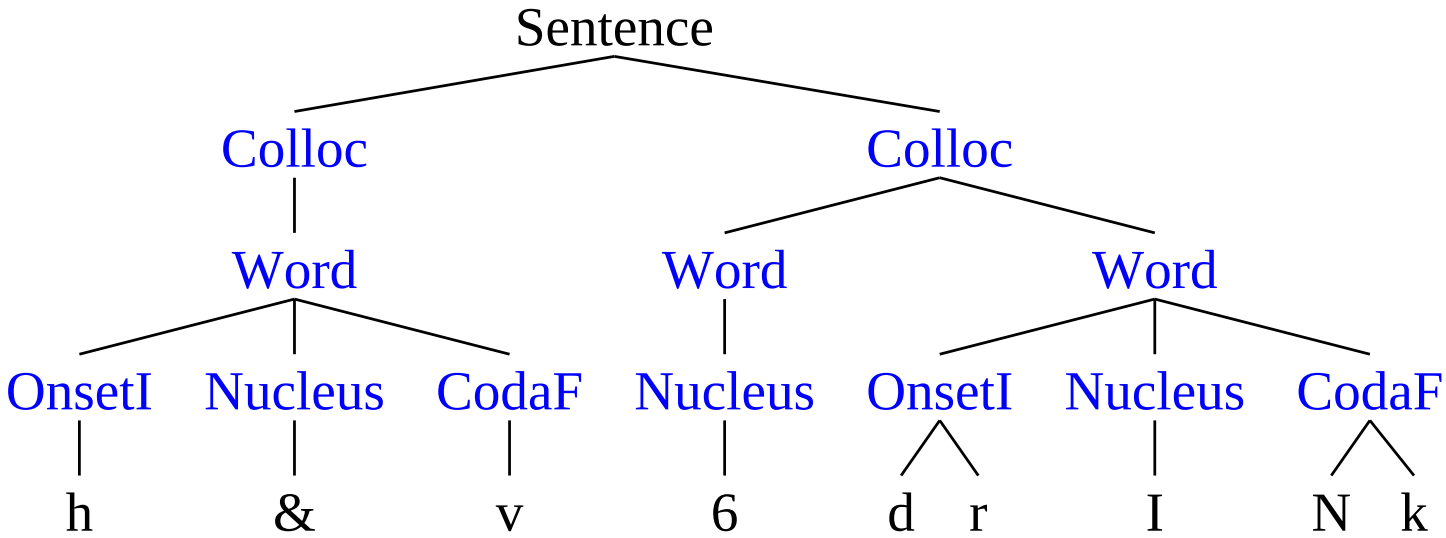


- A Colloc(ation) consists of one or more words
- Both Words and Collocs are adapted (learnt)
- Significantly improves word segmentation accuracy over unigram model (75% f-score; ≈ Goldwater's bigram model)
- Two levels of Collocations improves slightly (76%)

Syllables + Collocations + Word segmentation

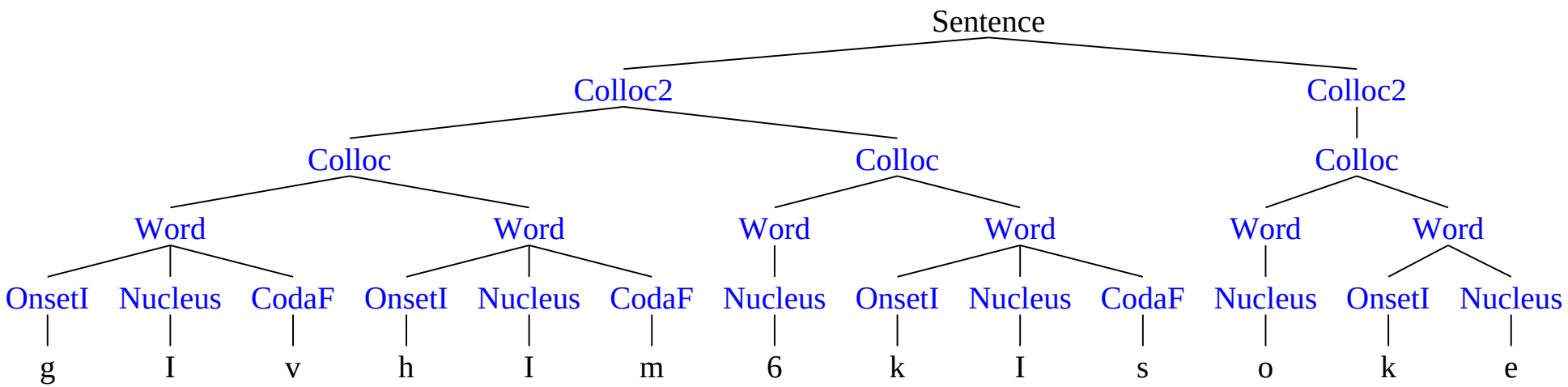
Sentence → Colloc$^+$
Word → SyllableIF
Word → SyllableI Syllable SyllableF
Onset → Consonant$^+$
Nucleus → Vowel$^+$

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



- With no supra-word generalizations, f-score = 68%
- With 2 Collocation levels, f-score = 84%
- Without distinguishing initial/final clusters, f-score = 82%

Syllables + 2-level Collocations + Word segmentation



Word segmentation results summary

		Collocation levels above the word			
		none	1 level	2 levels	3 levels
Below the word	none	0.55	0.73	0.75	0.74
	morphemes	0.35	0.55	0.79	0.78
	syllables	0.32	0.69	0.82	0.81
	syllables IF	0.46	0.68	**0.84**	**0.84**

- We can learn 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?
- Appropriately balancing structure above and below the Word seems to be crucial
  - even though all of the grammars are non-parametric

Outline

Estimating adaptor grammars

- Need to estimate:
  - cached subtrees τ 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
  - components are parse tree T_i for each string W_i
  - sample T_i from $\mathsf{P}(T|W_i, \boldsymbol{T}_{-i}, \boldsymbol{\alpha}, \boldsymbol{\theta})$ for each sentence W_i in turn
- 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

Metropolis-with-Gibbs sampler

- Collapsed Gibbs sampler: resample parse T_i given W_i and $\boldsymbol{T}_{-i}$
- 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 table given $\boldsymbol{T}_{-i}$
  - ignores changing table counts within single parse
- 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

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
- We use a Metropolis-Hastings sampler for a and b parameters
  - a is sampled from sequence of increasingly narrow Dirichlets
  - b is sampled from sequence of increasingly narrow Gammas
- Seems to improve performance with complicated grammars

Random initialization is better than incremental initialization

- Incremental initialization: assign parse for W_i based on $\boldsymbol{T}_{1,i-1}$
- Random initialization: initially assign parses T_i *randomly*
- Incremental initialization seems to get stuck in local optima

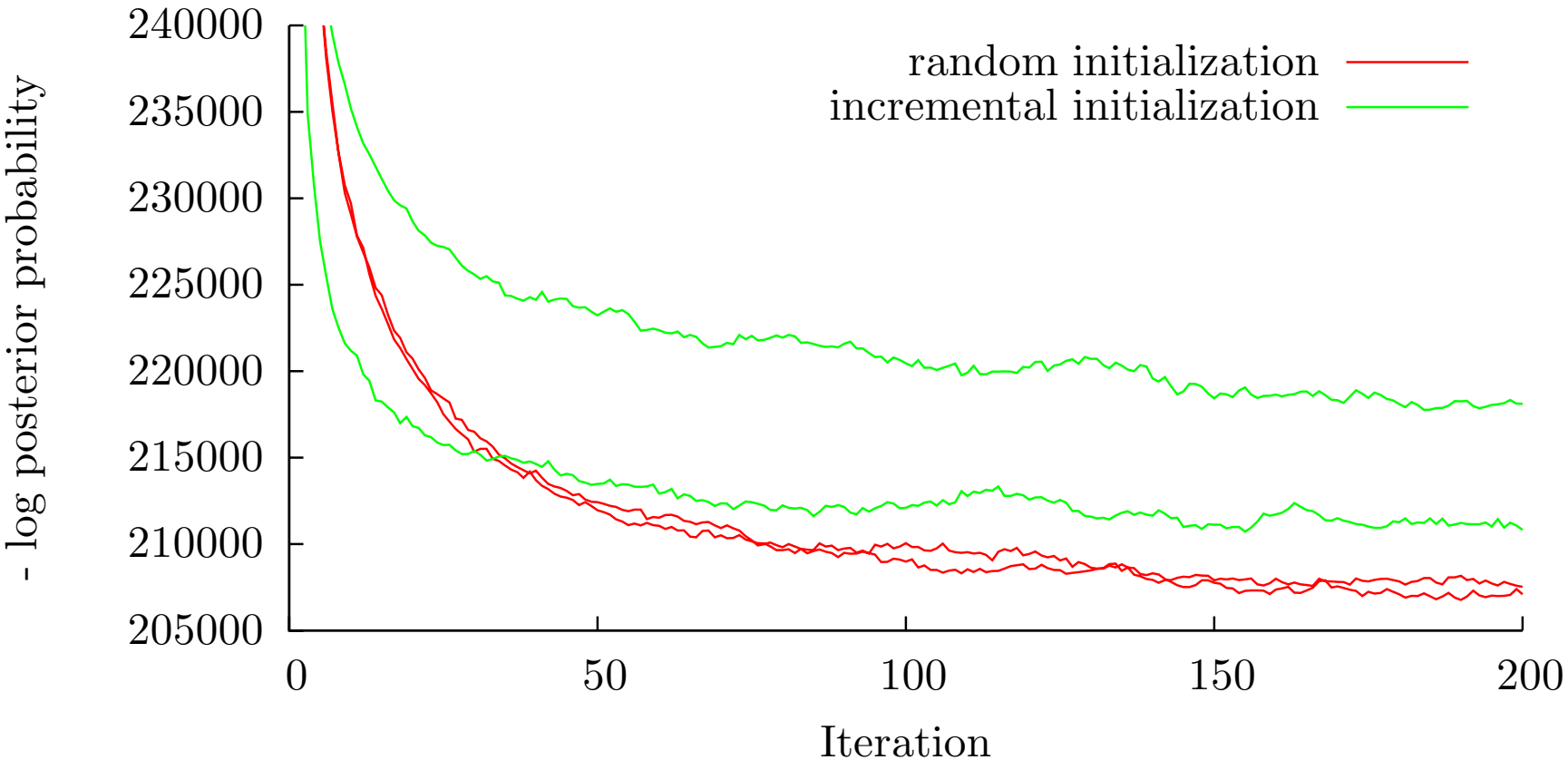
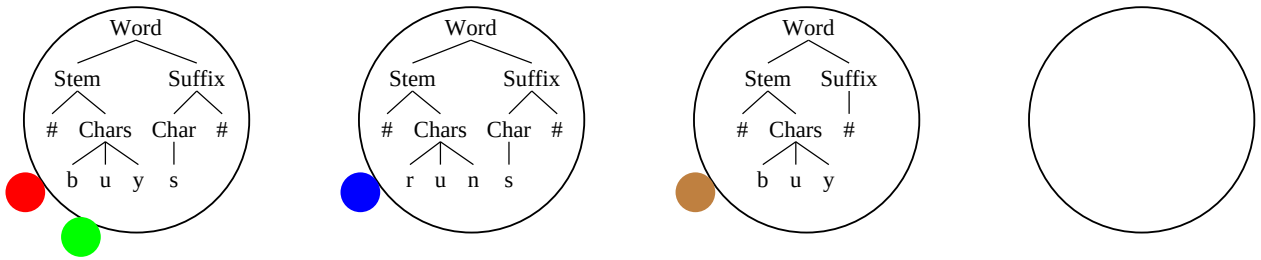


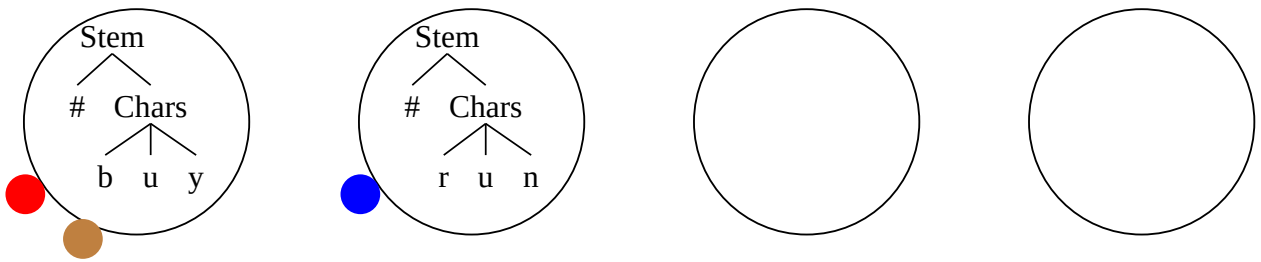
Table label resampling improves mobility

- Gibbs algorithm: resample T_i given W_i and $\boldsymbol{T}_{-i}$
- Table label resampling resamples the labels on each table
  - can change parses for many sentences at once

Word restaurant
Word → Stem Suffix



Stem restaurant
Stem → Phoneme*



Suffix restaurant
Suffix → Phoneme$^+$

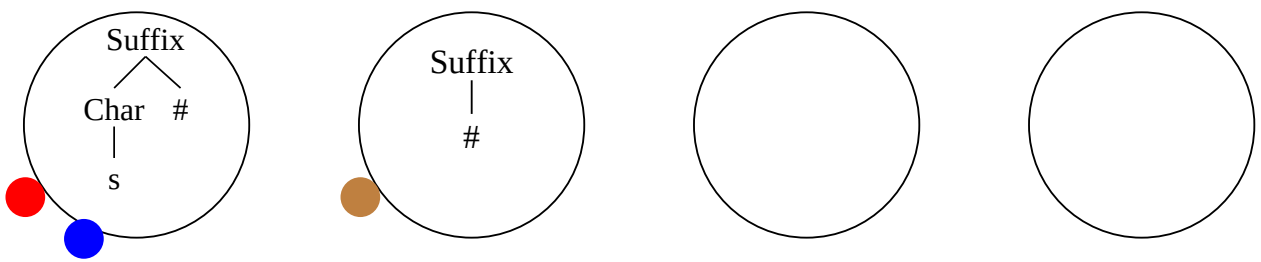
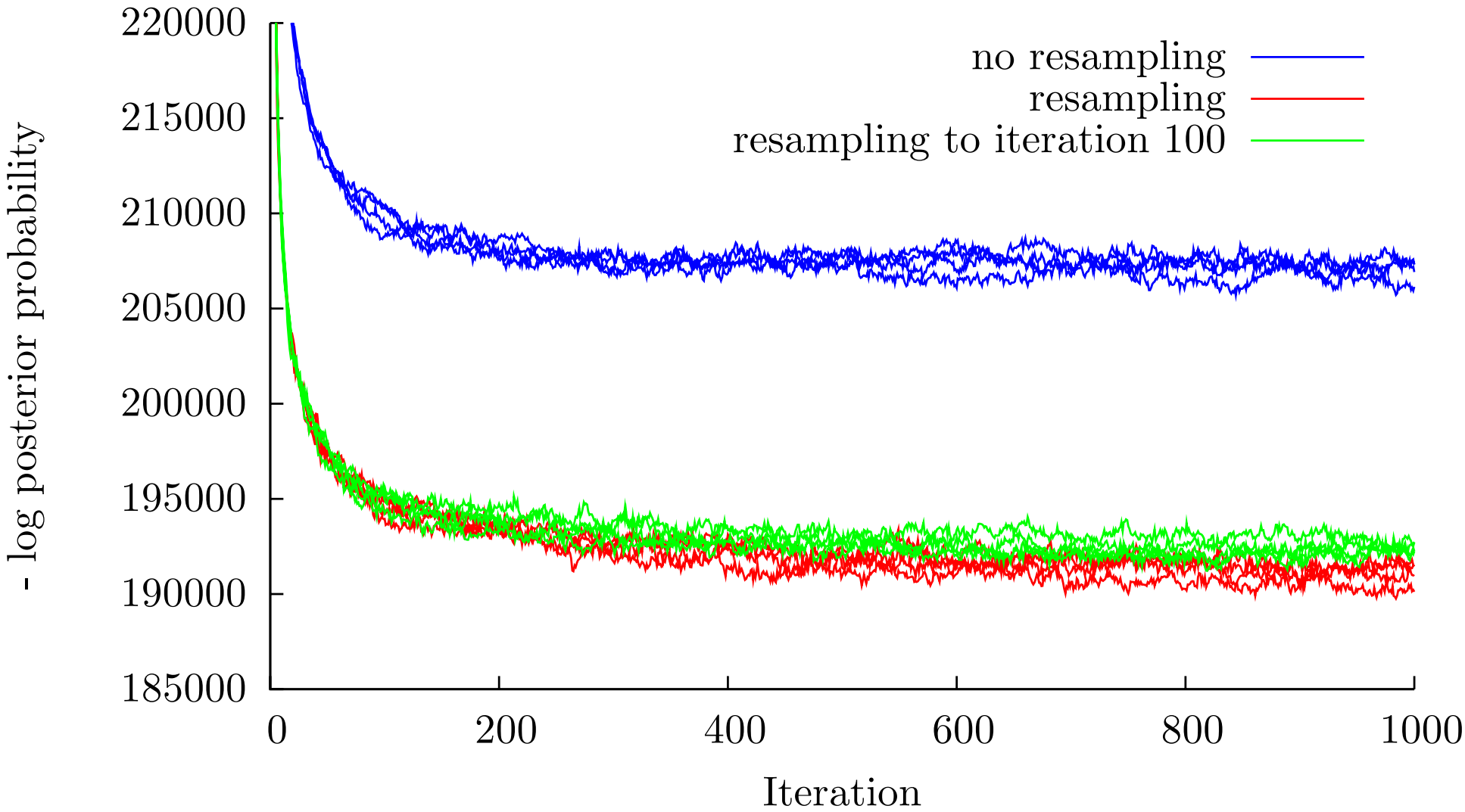
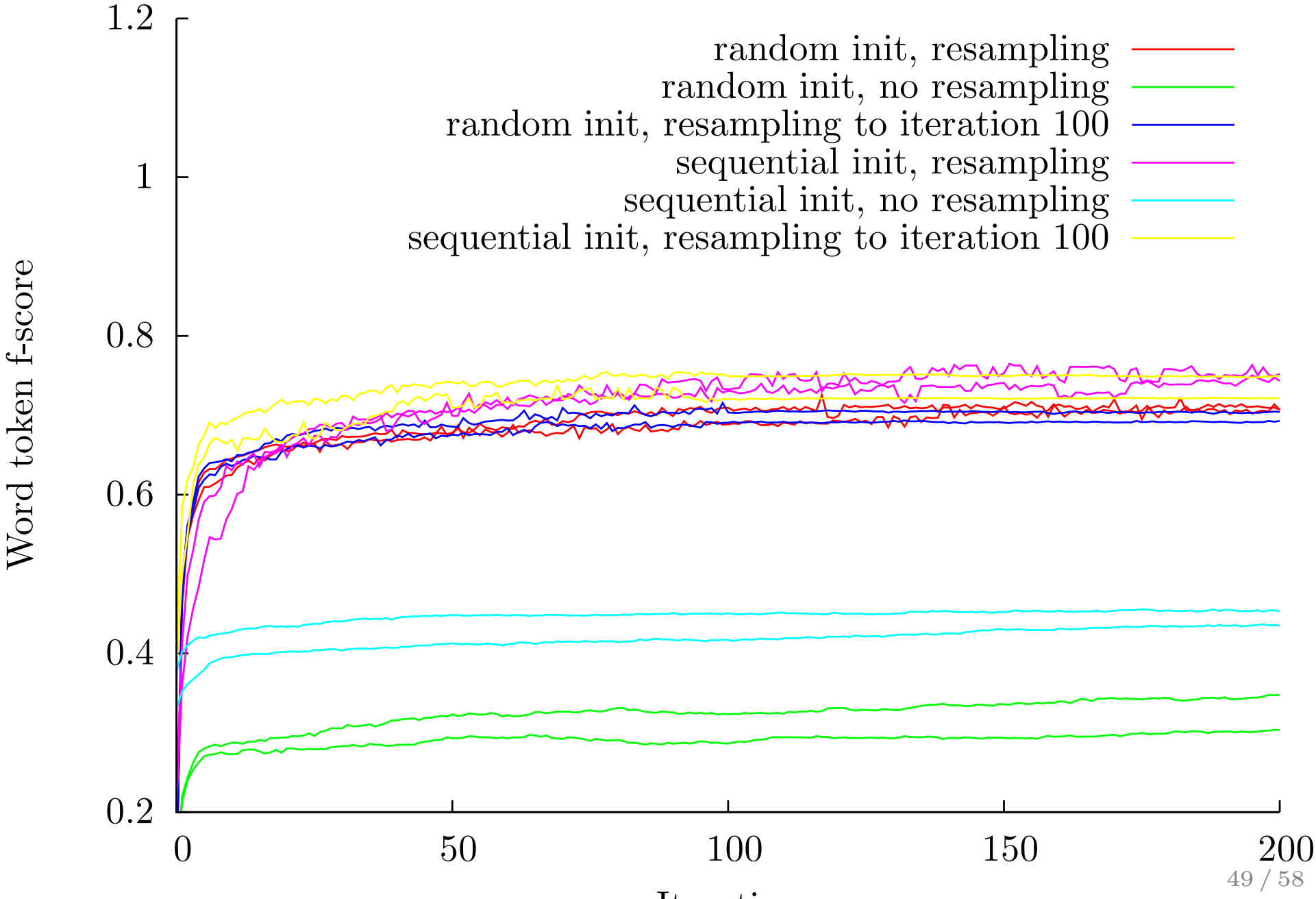


Table label resampling with Colloc grammar



Segmentation accuracy with Colloc grammar



Outline

Summary and future work

- Adaptor grammars can describe a variety of linguistic phenomena
- Grammars $\Rightarrow$ easy to design and compose
- Adaptor grammars "adapt" their distribution to the strings they have generated
  - learn probabilities of subtrees of adapted nonterminals

  $\Rightarrow$ adaptor grammars *non-parametric*; the subtrees they cache depends on trees they have generated

  $\Rightarrow$ grammar determines the generalizations it learns
- *Synergies in learning*
  - learning certain kinds of generalizations (Syllable, Collocation) "explains away" data that would otherwise interfere with learning other generalizations (Word)

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

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

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 δ_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 $A \in N$, α_A is the parameter of the Dirichlet process associated with A.

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

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

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 $(\boldsymbol{G}, \boldsymbol{H})$ is defined by:

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

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

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

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*)

$$G_A = \sum_{\substack{A \\ B_1 \ldots B_n} \in \mathcal{P}_A} F_A(\underset{B_1 \ldots B_n}{\overset{A}{\triangle}}) \; \mathrm{TD}_{\underset{B_1 \ldots B_n}{\overset{A}{\triangle}}}(G_{B_1}, \ldots, G_{B_n})$$

- 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 η is the *fragment exit probability*
- Obtain F_A by *adapting the H_A distribution*

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

- 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?