

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

⇒ Unsupervised morphological analysis:

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

⇒ Word segmentation: segment *broad phonemic* utterances

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

Outline

From PCFGs to Adaptor Grammars

Adaptor grammars for English word segmentation

Bayesian inference for Adaptor Grammars

Conclusion

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
$S \rightarrow NP VP$	1.0
$NP \rightarrow \text{Sam}$	0.75
$VP \rightarrow \text{barks}$	0.6

Rule r	θ_r
$NP \rightarrow \text{Sandy}$	0.25
$VP \rightarrow \text{snores}$	0.4

$$P \left(\begin{array}{c} S \\ \swarrow \quad \searrow \\ NP \quad VP \\ | \quad | \\ \text{Sam} \quad \text{barks} \end{array} \right) = 0.45$$

$$P \left(\begin{array}{c} S \\ \swarrow \quad \searrow \\ NP \quad VP \\ | \quad | \\ \text{Sandy} \quad \text{snores} \end{array} \right) = 0.1$$

A CFG for stem-suffix morphology

Word \rightarrow Stem Suffix

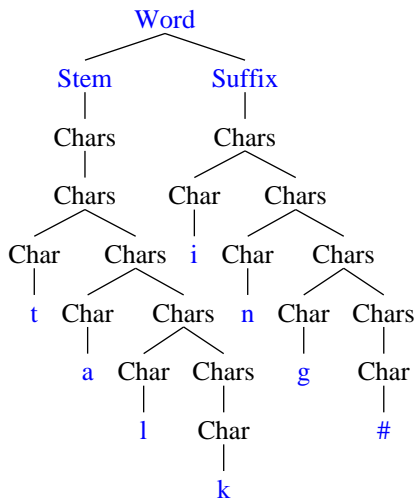
Stem \rightarrow Chars

Suffix \rightarrow Chars

Chars \rightarrow Char

Chars \rightarrow Char Chars

Char \rightarrow a | b | c | ...



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

\Rightarrow Can represent true segmentation

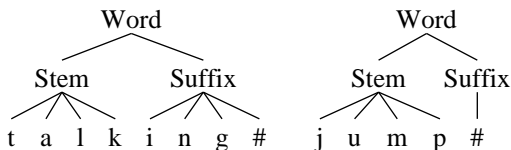
- But grammar's *units of generalization* (PCFG rules) are "too small" to learn morphemes

A “CFG” with one rule per possible morpheme

Word \rightarrow Stem Suffix

Stem \rightarrow all possible stems

Suffix \rightarrow all possible suffixes



- 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 \rightarrow Stem Suffix

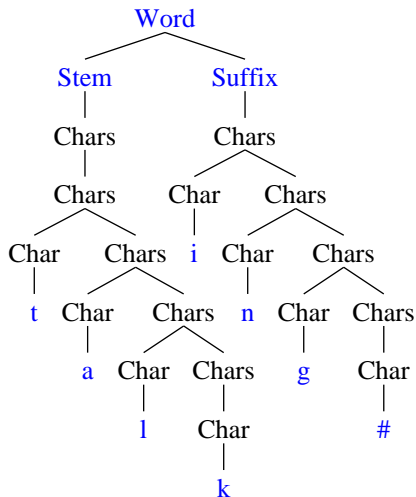
Stem \rightarrow Chars

Suffix \rightarrow Chars

Chars \rightarrow Char

Chars \rightarrow Char Chars

Char \rightarrow a | b | c | ...

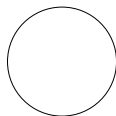
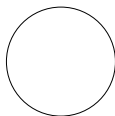
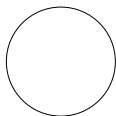
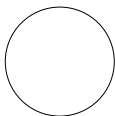


- Word, Stem and Suffix are *adapted*

\Rightarrow learns probabilities of whole Stem and Suffix and Word subtrees

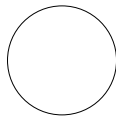
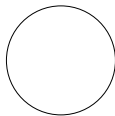
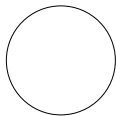
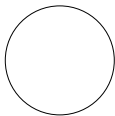
Morphology adaptor grammar (0)

Word restaurant
Word \rightarrow Stem Suffix



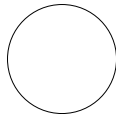
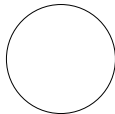
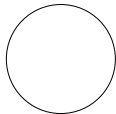
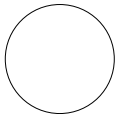
...

Stem restaurant
Stem \rightarrow Phoneme^{*}



...

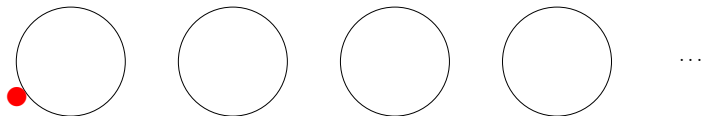
Suffix restaurant
Suffix \rightarrow Phoneme⁺



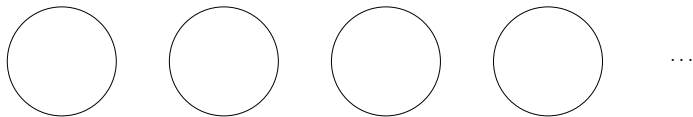
...

Morphology adaptor grammar (1a)

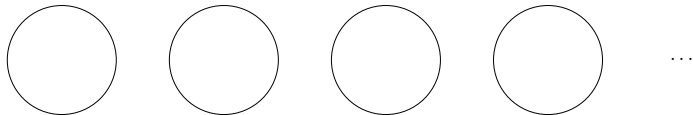
Word restaurant
Word \rightarrow Stem Suffix



Stem restaurant
Stem \rightarrow Phoneme^{*}

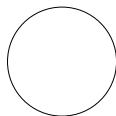
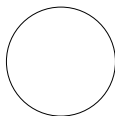
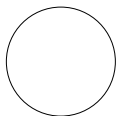
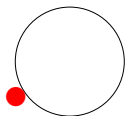


Suffix restaurant
Suffix \rightarrow Phoneme⁺



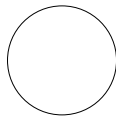
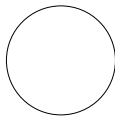
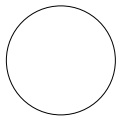
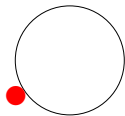
Morphology adaptor grammar (1b)

Word restaurant
Word \rightarrow Stem Suffix



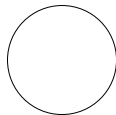
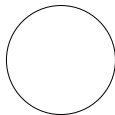
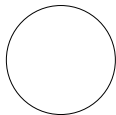
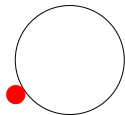
...

Stem restaurant
Stem \rightarrow Phoneme^{*}



...

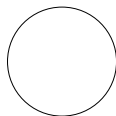
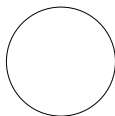
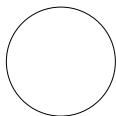
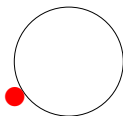
Suffix restaurant
Suffix \rightarrow Phoneme⁺



...

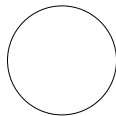
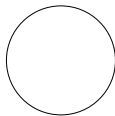
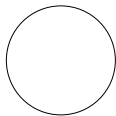
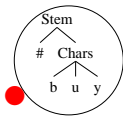
Morphology adaptor grammar (1c)

Word restaurant
Word \rightarrow Stem Suffix



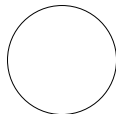
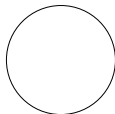
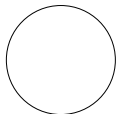
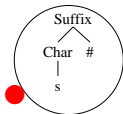
...

Stem restaurant
Stem \rightarrow Phoneme*



...

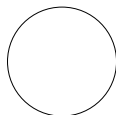
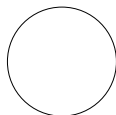
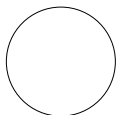
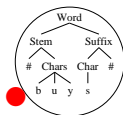
Suffix restaurant
Suffix \rightarrow Phoneme⁺



...

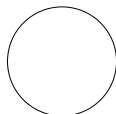
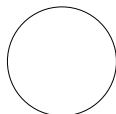
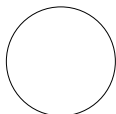
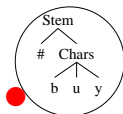
Morphology adaptor grammar (1d)

Word restaurant
Word \rightarrow Stem Suffix



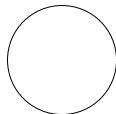
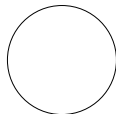
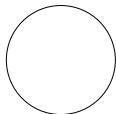
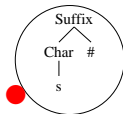
...

Stem restaurant
Stem \rightarrow Phoneme*



...

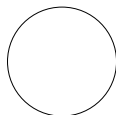
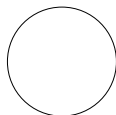
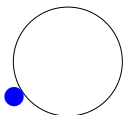
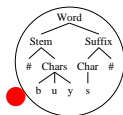
Suffix restaurant
Suffix \rightarrow Phoneme⁺



...

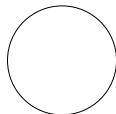
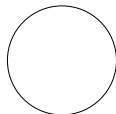
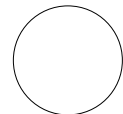
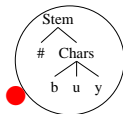
Morphology adaptor grammar (2a)

Word restaurant
Word \rightarrow Stem Suffix



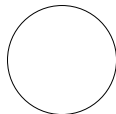
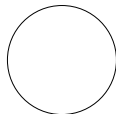
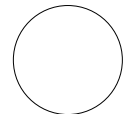
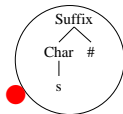
...

Stem restaurant
Stem \rightarrow Phoneme*



...

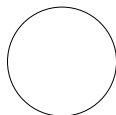
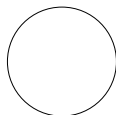
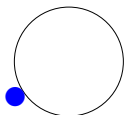
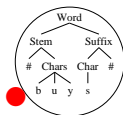
Suffix restaurant
Suffix \rightarrow Phoneme⁺



...

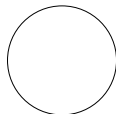
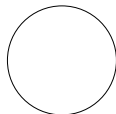
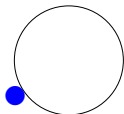
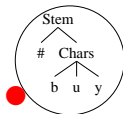
Morphology adaptor grammar (2b)

Word restaurant
Word \rightarrow Stem Suffix



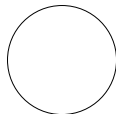
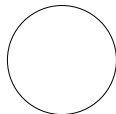
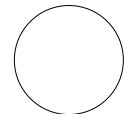
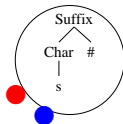
...

Stem restaurant
Stem \rightarrow Phoneme*



...

Suffix restaurant
Suffix \rightarrow Phoneme⁺

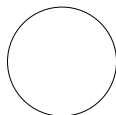
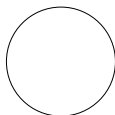
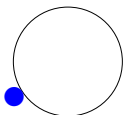
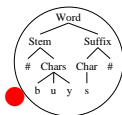


...

Morphology adaptor grammar (2c)

Word restaurant

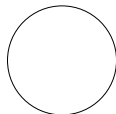
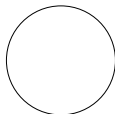
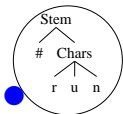
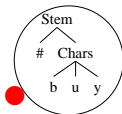
Word \rightarrow Stem Suffix



...

Stem restaurant

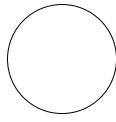
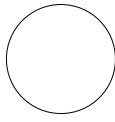
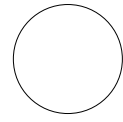
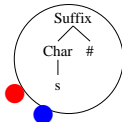
Stem \rightarrow Phoneme^{*}



...

Suffix restaurant

Suffix \rightarrow Phoneme⁺

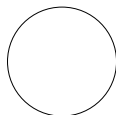
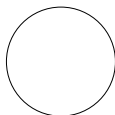
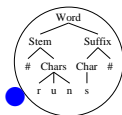
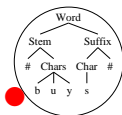


...

Morphology adaptor grammar (2d)

Word restaurant

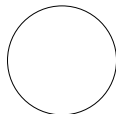
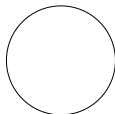
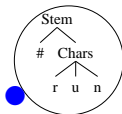
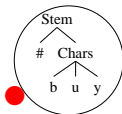
Word \rightarrow Stem Suffix



...

Stem restaurant

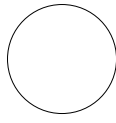
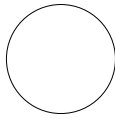
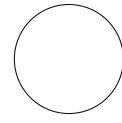
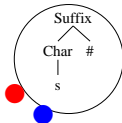
Stem \rightarrow Phoneme^{*}



...

Suffix restaurant

Suffix \rightarrow Phoneme⁺

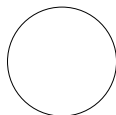
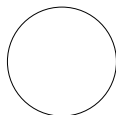
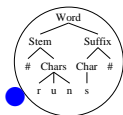
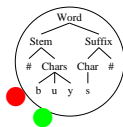


...

Morphology adaptor grammar (3)

Word restaurant

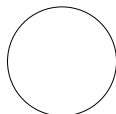
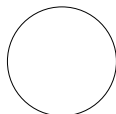
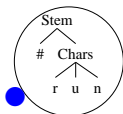
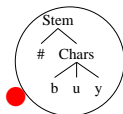
Word \rightarrow Stem Suffix



...

Stem restaurant

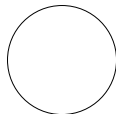
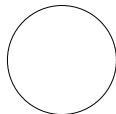
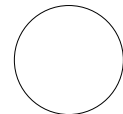
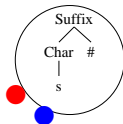
Stem \rightarrow Phoneme^{*}



...

Suffix restaurant

Suffix \rightarrow Phoneme⁺

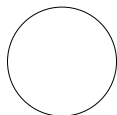
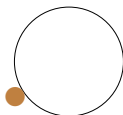
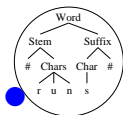
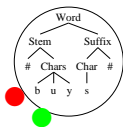


...

Morphology adaptor grammar (4a)

Word restaurant

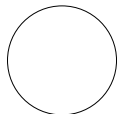
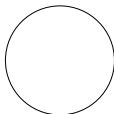
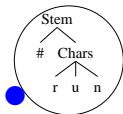
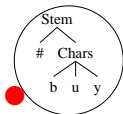
Word \rightarrow Stem Suffix



...

Stem restaurant

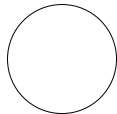
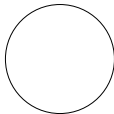
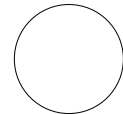
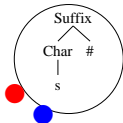
Stem \rightarrow Phoneme^{*}



...

Suffix restaurant

Suffix \rightarrow Phoneme⁺

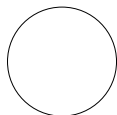
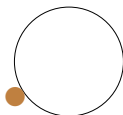
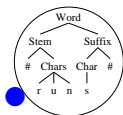
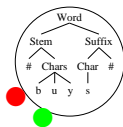


...

Morphology adaptor grammar (4b)

Word restaurant

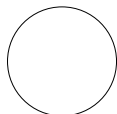
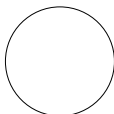
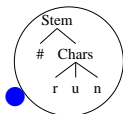
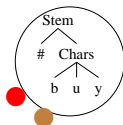
Word \rightarrow Stem Suffix



...

Stem restaurant

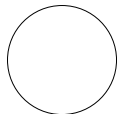
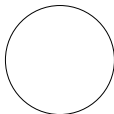
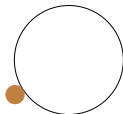
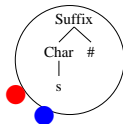
Stem \rightarrow Phoneme^{*}



...

Suffix restaurant

Suffix \rightarrow Phoneme⁺

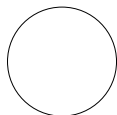
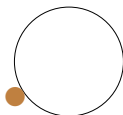
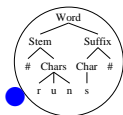
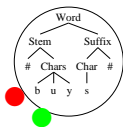


...

Morphology adaptor grammar (4c)

Word restaurant

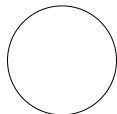
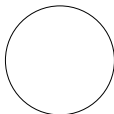
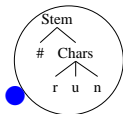
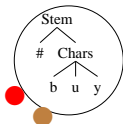
Word \rightarrow Stem Suffix



...

Stem restaurant

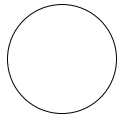
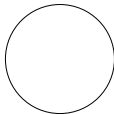
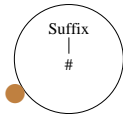
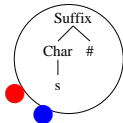
Stem \rightarrow Phoneme^{*}



...

Suffix restaurant

Suffix \rightarrow Phoneme⁺

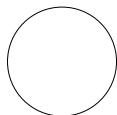
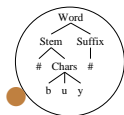
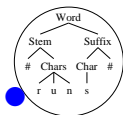
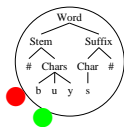


...

Morphology adaptor grammar (4d)

Word restaurant

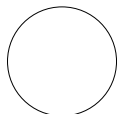
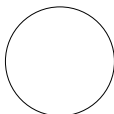
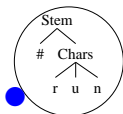
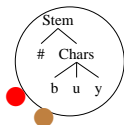
Word \rightarrow Stem Suffix



...

Stem restaurant

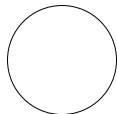
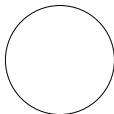
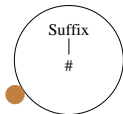
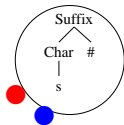
Stem \rightarrow Phoneme^{*}



...

Suffix restaurant

Suffix \rightarrow Phoneme⁺



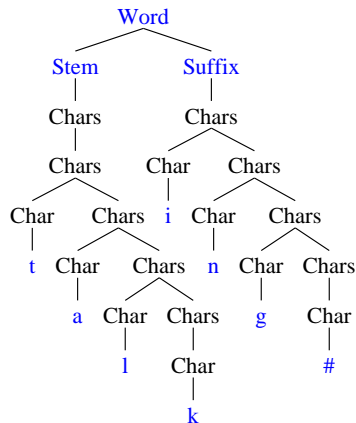
...

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. $\propto \alpha_{\text{Word}} P(\text{Stem}) P(\text{Suffix})$
- 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. $\propto \alpha_{\text{Stem}} P(\text{Phonemes})$
- 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
⇒ 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
⇒ “rich get richer” dynamics (Zipf distributions)
 - ▶ plus α_A times prob. of generating it via PCFG expansion
- ⇒ Useful compound structures can be *more probable than their parts*
- PCFG rule probabilities estimated *from table labels*
 - ⇒ learns from types, not tokens
 - ⇒ dampens frequency variation

Outline

From PCFGs to Adaptor Grammars

Adaptor grammars for English word segmentation

Bayesian inference for Adaptor Grammars

Conclusion

Unigram adaptor grammar for English

- Adaptor grammar (adapted nonterminals highlighted):

Sentence \rightarrow Words

Words \rightarrow Word

Words \rightarrow Word Words

Word \rightarrow Phonemes

Phonemes \rightarrow Phoneme

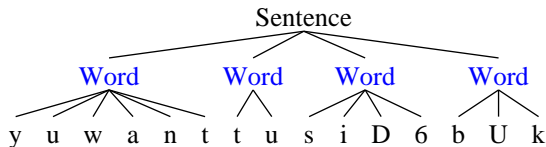
Phonemes \rightarrow Phoneme Phonemes

or in abbreviated format:

Sentence \rightarrow Word⁺

Word \rightarrow Phoneme⁺

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



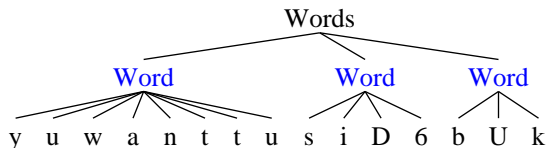
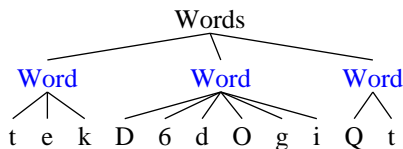
- Word segmentation f-score = 0.55 (same as Goldwater et al)

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

Words \rightarrow Word⁺

Word \rightarrow Phoneme⁺

the adapted grammar contains 1,712 rules such as:

15758 Words \rightarrow Word Words

9791 Words \rightarrow Word

1660 Word \rightarrow Phoneme⁺

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 morphology adaptor grammar

- Adaptor grammar memorizes Word, Stem and Suffix:

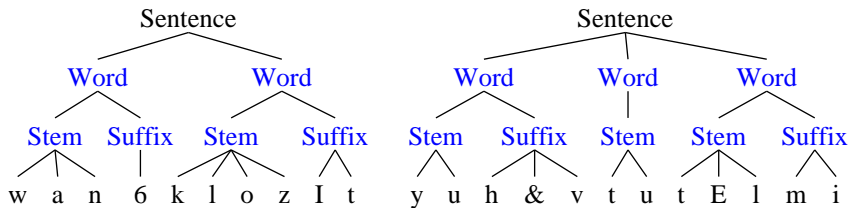
Sentence \rightarrow Word⁺

Word \rightarrow Stem (Suffix)

Stem \rightarrow Phoneme⁺

Suffix \rightarrow 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

Sentence \rightarrow Word⁺

Word \rightarrow Syllable⁺

Syllable \rightarrow (Onset) Rhyme

Onset \rightarrow Consonant⁺

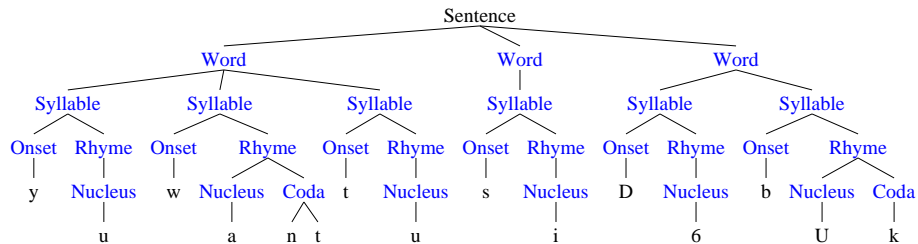
Rhyme \rightarrow Nucleus (Coda)

Nucleus \rightarrow Vowel⁺

Coda \rightarrow Consonant⁺

- 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

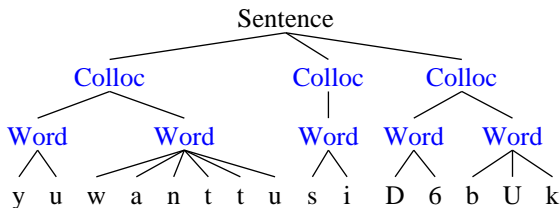


Modeling collocations improves segmentation

Sentence \rightarrow Colloc⁺

Colloc \rightarrow Word⁺

Word \rightarrow Phoneme^{*}



- 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; \approx Goldwater's bigram model)
- Two levels of **Collocations** improves slightly (76%)

Syllables + Collocations + Word segmentation

Sentence \rightarrow Colloc⁺

Word \rightarrow SyllableIF

Word \rightarrow SyllableI Syllable SyllableF

Onset \rightarrow Consonant⁺

Nucleus \rightarrow Vowel⁺

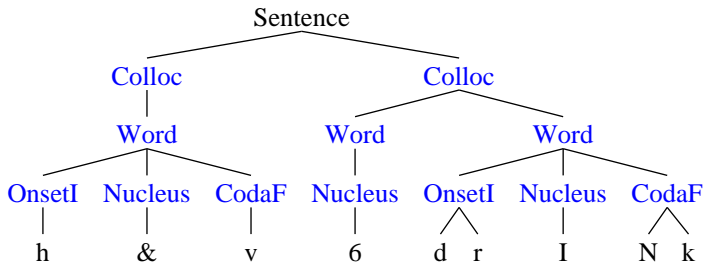
Colloc \rightarrow Word⁺

Word \rightarrow SyllableI SyllableF

Syllable \rightarrow (Onset) Rhyme

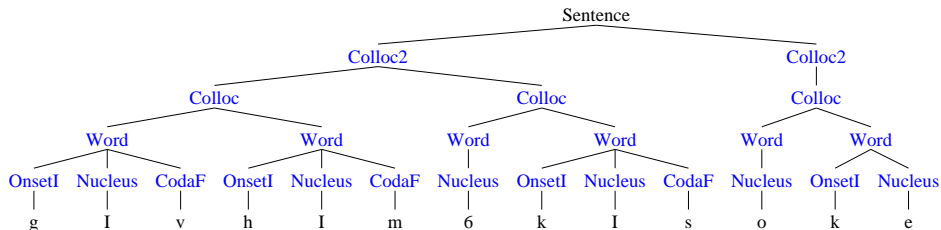
Rhyme \rightarrow Nucleus (Coda)

Coda \rightarrow 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

		<i>Collocation levels above the word</i>			
		none	1 level	2 levels	3 levels
<i>Below the word</i>	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

From PCFGs to Adaptor Grammars

Adaptor grammars for English word segmentation

Bayesian inference for Adaptor Grammars

Conclusion

Estimating adaptor grammars

- Need to estimate:
 - ▶ cached subtrees τ for adapted nonterminals
 - ▶ (optional) DP parameters α for adapted nonterminals
 - ▶ (optional) probabilities θ 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 $P(T|W_i, \mathbf{T}_{-i}, \alpha, \theta)$ for each sentence W_i in turn
- Sampling directly from conditional distribution of parses seems intractable
 - ▶ construct PCFG proposal grammar $G'(\mathbf{T}_{-i})$ 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 \mathbf{T}_{-i}
- Table counts change within a parse tree
 - ⇒ grammar is not context-free
 - ⇒ breaks standard dynamic programming
 - ⇒ Metropolis accept/reject for each Gibbs sample
- PCFG can express probability of selecting a table given \mathbf{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 $A \rightarrow \text{YIELD}(\tau)$ for each table τ in A 's restaurant:
 $\theta'_{A \rightarrow \text{YIELD}(\tau)} \propto n_\tau$, 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 $\theta_{A \rightarrow \beta}$ of base grammar rules $A \rightarrow \beta$
 - ▶ concentration parameters α_A of adapted nonterminals A
- Put Bayesian priors on these parameters
 - ▶ (Uniform) Dirichlet prior on base grammar rule probabilities θ
 - ▶ 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 $\mathbf{T}_{1,i-1}$
- Random initialization: initially assign parses T_i *randomly*
- Incremental initialization seems to get stuck in local optima

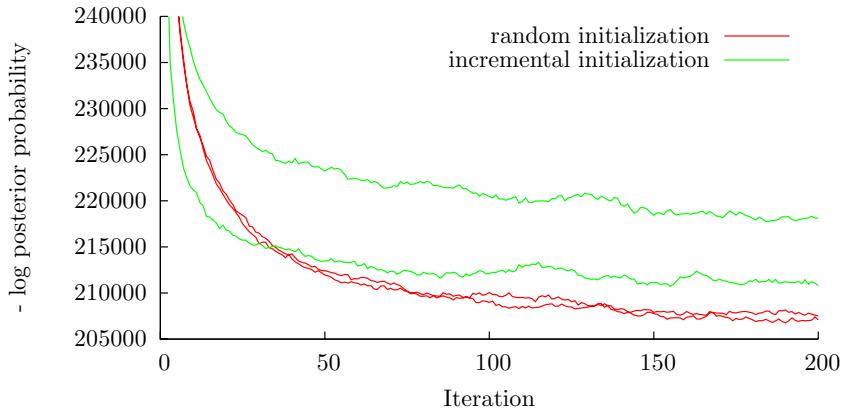


Table label resampling improves mobility

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

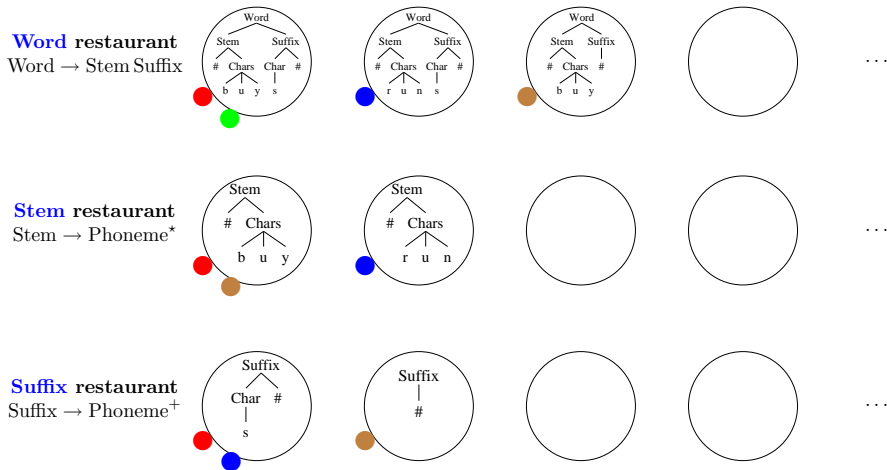
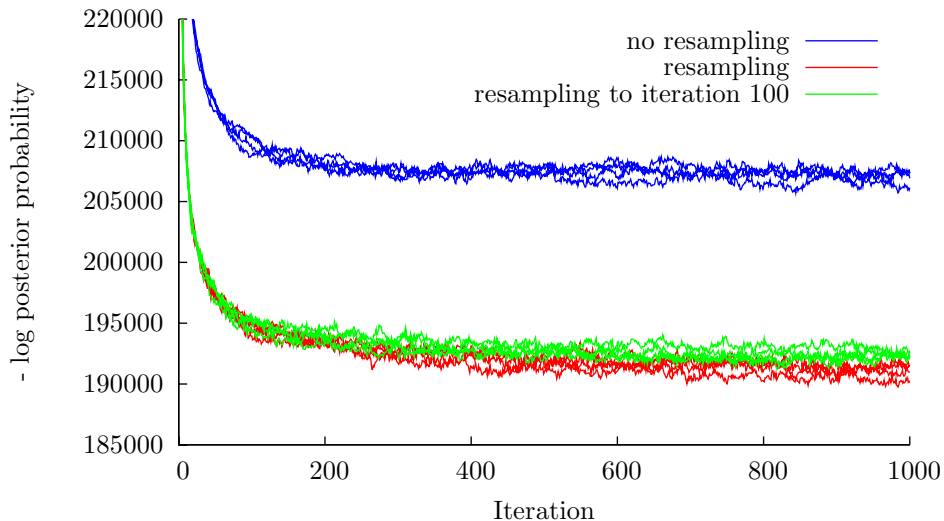
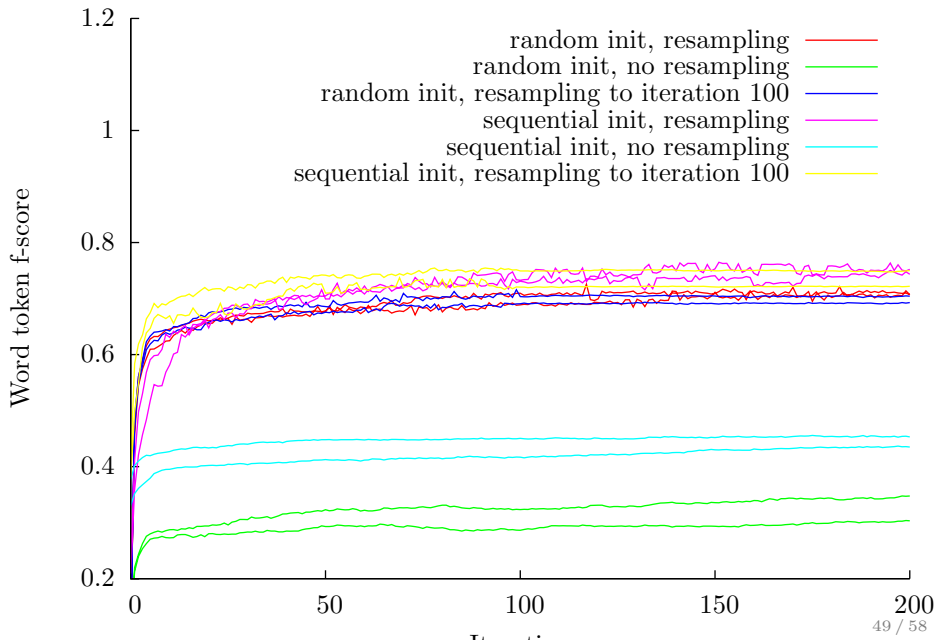


Table label resampling with Colloc grammar



Segmentation accuracy with Colloc grammar



Outline

From PCFGs to Adaptor Grammars

Adaptor grammars for English word segmentation

Bayesian inference for Adaptor Grammars

Conclusion

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* $A \rightarrow \beta$, where $A \in N$ and $\beta \in (N \cup W)^*$
- 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:

$$\mathcal{T}_A = \bigcup_{A \rightarrow B_1 \dots B_n \in R_A} \text{TREE}_A(\mathcal{T}_{B_1}, \dots, \mathcal{T}_{B_n})$$

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

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

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 θ , where:

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

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

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

where δ_A puts all its mass onto the singleton tree A , and:

$$\text{TD}_A(G_1, \dots, G_n) \left(\begin{array}{c} A \\ \swarrow \quad \searrow \\ t_1 \quad \dots \quad t_n \end{array} \right) = \prod_{i=1}^n G_i(t_i).$$

$\text{TD}_A(G_1, \dots, G_n)$ 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 .

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

$$H_A = \sum_{A \rightarrow B_1 \dots B_n \in R_A} \theta_{A \rightarrow B_1 \dots B_n} \text{TD}_A(G_{B_1}, \dots, G_{B_n})$$

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:

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

$$H_A = \sum_{A \rightarrow B_1 \dots B_n \in R_A} \theta_{A \rightarrow B_1 \dots B_n} \text{TD}_A(G_{B_1}, \dots, G_{B_n})$$

- 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:

$$\mathcal{P}_A = \bigcup_{A \rightarrow B_1 \dots B_n \in R_A} \text{TREE}_A(\{B_1\} \cup \mathcal{P}_{B_1}, \dots, \{B_n\} \cup \mathcal{P}_{B_n})$$

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

$$G_A = \sum_{\substack{A \\ \triangle \\ B_1 \dots B_n \in \mathcal{P}_A}} F_A\left(\begin{array}{c} A \\ \triangle \\ B_1 \dots B_n \end{array}\right) \text{TD}_A\left(\begin{array}{c} \triangle \\ B_1 \dots B_n \end{array}(G_{B_1}, \dots, G_{B_n})\right)$$

- A fragment grammar generates the distribution G_S

Adaptive fragment distributions

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

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

where η is the *fragment exit probability*

- Obtain F_A by *adapting the H_A distribution*

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

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