Bayesian models of language acquisition, or Where do the rules come from?

Mark Johnson

joint work with Katherine Demuth, Michael Frank, Sharon Goldwater, Tom Griffiths and Bevan Jones

Macquarie University
Sydney, Australia

June 2011

MACQUARIE UNIVERSITY

The drunk under the lamppost

Late one night, a drunk guy is crawling around under a lamppost. A cop comes up and asks him what he's doing.

"*I'm looking for my keys*," the drunk says. "*I lost them about three blocks away*."

"*So why aren't you looking for them where you dropped them?*" the cop asks.

The drunk looks at the cop, amazed that he'd ask so obvious a question. "*Because the light is so much better here.*"

Language acquisition as Bayesian inference

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

- Likelihood measures *how well grammar describes data*
- Prior expresses knowledge of grammar before data is seen
  - can be very specific (e.g., Universal Grammar)
  - can be very general (e.g., prefer shorter grammars)
- Prior can also express *markedness preferences* ("soft universals")
- Posterior is a *product* of both likelihood and prior
  - a grammar must do well on both to have high posterior probability
- Posterior is a *distribution* over grammars
  - captures *learner's uncertainty* about which grammar is correct

Outline

MACQUARIE UNIVERSITY

Probabilistic context-free grammars

- Probabilistic context-free grammars (PCFGs) define *probability distributions over trees*
- Each *nonterminal node* expands by
  - choosing a rule expanding that nonterminal, and
  - recursively expanding any nonterminal children it contains
- Probability of tree is *product of probabilities of rules* used to construct it

Probability θ_r	*Rule r*
1	S $\rightarrow$ NP VP
0.7	NP $\rightarrow$ *Sam*
0.3	NP $\rightarrow$ *Sandy*
1	VP $\rightarrow$ V NP
0.8	V $\rightarrow$ *likes*
0.2	V $\rightarrow$ *hates*

S

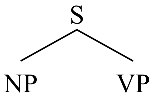
$$\mathrm{P}(\text{Tree}) \quad =$$

MACQUARIE UNIVERSITY

Probabilistic context-free grammars

- Probabilistic context-free grammars (PCFGs) define *probability distributions over trees*
- Each *nonterminal node* expands by
  - choosing a rule expanding that nonterminal, and
  - recursively expanding any nonterminal children it contains
- Probability of tree is *product of probabilities of rules* used to construct it

Probability θ_r	*Rule r*
1	S $\rightarrow$ NP VP
0.7	NP $\rightarrow$ *Sam*
0.3	NP $\rightarrow$ *Sandy*
1	VP $\rightarrow$ V NP
0.8	V $\rightarrow$ *likes*
0.2	V $\rightarrow$ *hates*

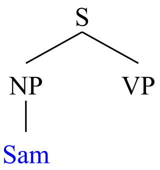


$$\mathrm{P}(\text{Tree}) \;=\; 1 \times$$

Probabilistic context-free grammars

- Probabilistic context-free grammars (PCFGs) define *probability distributions over trees*
- Each *nonterminal node* expands by
  - choosing a rule expanding that nonterminal, and
  - recursively expanding any nonterminal children it contains
- Probability of tree is *product of probabilities of rules* used to construct it

Probability θ_r	*Rule r*
1	S $\rightarrow$ NP VP
0.7	NP $\rightarrow$ *Sam*
0.3	NP $\rightarrow$ *Sandy*
1	VP $\rightarrow$ V NP
0.8	V $\rightarrow$ *likes*
0.2	V $\rightarrow$ *hates*

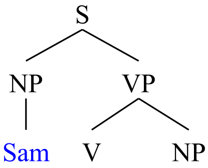


$$\mathrm{P}(\text{Tree}) = 1 \times 0.7 \times$$

Probabilistic context-free grammars

- Probabilistic context-free grammars (PCFGs) define *probability distributions over trees*
- Each *nonterminal node* expands by
  - choosing a rule expanding that nonterminal, and
  - recursively expanding any nonterminal children it contains
- Probability of tree is *product of probabilities of rules* used to construct it

Probability θ_r	*Rule* r
1	S $\rightarrow$ NP VP
0.7	NP $\rightarrow$ *Sam*
0.3	NP $\rightarrow$ *Sandy*
1	VP $\rightarrow$ V NP
0.8	V $\rightarrow$ *likes*
0.2	V $\rightarrow$ *hates*

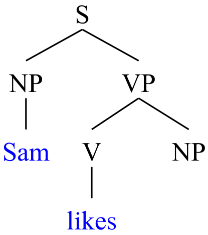


$$\mathrm{P}(\text{Tree}) \;=\; 1 \times 0.7 \times 1 \times$$

Probabilistic context-free grammars

- Probabilistic context-free grammars (PCFGs) define *probability distributions over trees*
- Each *nonterminal node* expands by
  - choosing a rule expanding that nonterminal, and
  - recursively expanding any nonterminal children it contains
- Probability of tree is *product of probabilities of rules* used to construct it

Probability θ_r	*Rule r*
1	S $\rightarrow$ NP VP
0.7	NP $\rightarrow$ *Sam*
0.3	NP $\rightarrow$ *Sandy*
1	VP $\rightarrow$ V NP
0.8	V $\rightarrow$ *likes*
0.2	V $\rightarrow$ *hates*

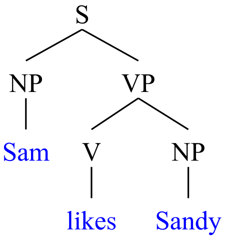


$$\mathrm{P}(\text{Tree}) = 1 \times 0.7 \times 1 \times 0.8 \times$$

Probabilistic context-free grammars

- Probabilistic context-free grammars (PCFGs) define *probability distributions over trees*
- Each *nonterminal node* expands by
  - choosing a rule expanding that nonterminal, and
  - recursively expanding any nonterminal children it contains
- Probability of tree is *product of probabilities of rules* used to construct it

Probability θ_r	*Rule* r
1	S $\rightarrow$ NP VP
0.7	NP $\rightarrow$ *Sam*
0.3	NP $\rightarrow$ *Sandy*
1	VP $\rightarrow$ V NP
0.8	V $\rightarrow$ *likes*
0.2	V $\rightarrow$ *hates*



$$\mathrm{P}(\text{Tree}) \;=\; 1 \times 0.7 \times 1 \times 0.8 \times 0.3$$

Parametric and non-parametric learners

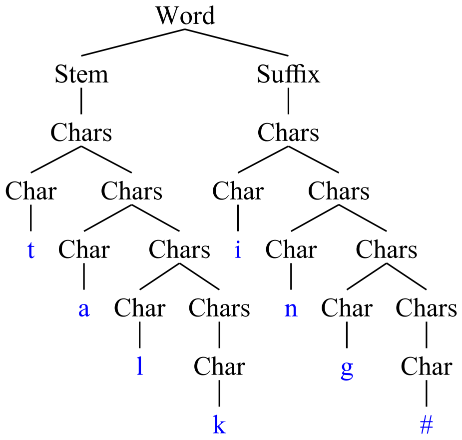
- Standard algorithms for learning PCFGs from strings alone are *parametric learners*
  - input: a corpus of strings and *a set of rules*
  - output: probabilities for the rules
- Can we learn the rules as well?
- *Parametric learners* optimise a *predetermined finite vector of parameters*
- *Non-parametric learners* can't be viewed as optimising a finite parameter vector
- Learning the rules as non-parametric learning:
  - infinite list enumerating all possible CF rules
  - learner has to choose the right subset of rules, as well as their probabilities

Plan for rest of talk

- Learning structure is hard!
  - Bayesian PCFG estimation works well on toy data, but
  - results are disappointing on real data
- Strategy: study simpler cases
  - morphological segmentation (e.g., *walking* = *walk*+*ing*)
  - segmenting utterances into words, i.e., learning word pronunciations
  - learning the relationship between words and the objects they refer to
- (Even hard-core rationalists agree these are learned)

A CFG for stem-suffix morphology

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



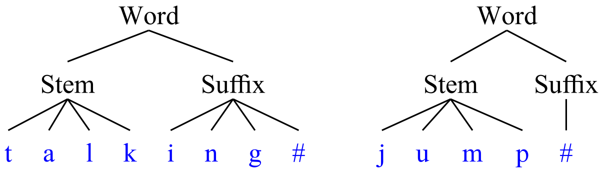
- Grammar's trees can represent any segmentation of words into stems and suffixes

$\Rightarrow$ Can *represent* true segmentation

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

A "CFG" with one rule per possible morpheme

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

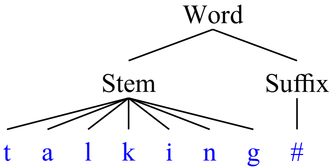
MACQUARIE UNIVERSITY

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

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

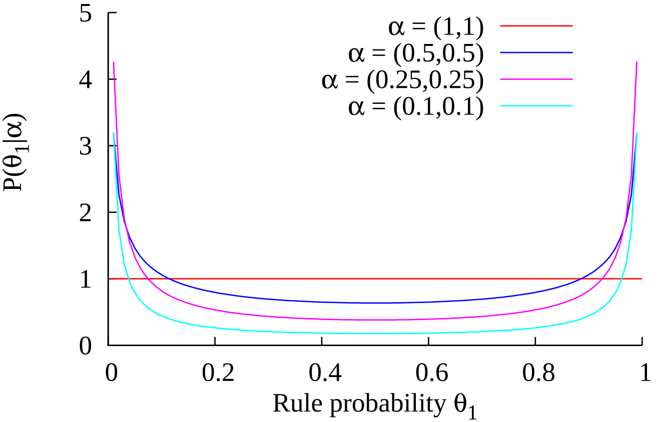
$\Rightarrow$ Saturated model is maximum likelihood estimate

- Maximum likelihood estimate does not find any suffixes



Forcing generalization via sparse priors

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



MACQUARIE UNIVERSITY

Morphological segmentation experiment

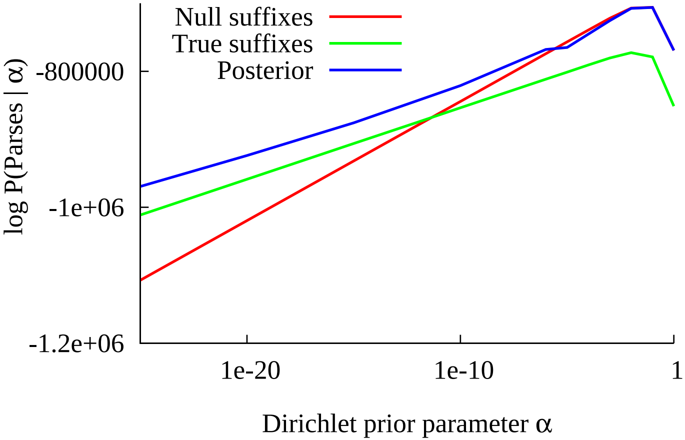
- Trained on orthographic verbs from U Penn. Wall Street Journal treebank
- Uniform Dirichlet prior prefers sparse solutions as $\alpha \to 0$
- Gibbs sampler samples from posterior distribution of parses
  - reanalyses each word based on parses of the other words

MACQUARIE UNIVERSITY

Posterior samples from WSJ verb tokens

$\alpha = 0.1$		$\alpha = 10^{-5}$		$\alpha = 10^{-10}$		$\alpha = 10^{-15}$	
expect		expect		expect		expect	
expects		expects		expects		expects	
expected		expected		expected		expected	
expecting		expect	ing	expect	ing	expect	ing
include		include		include		include	
includes		includes		includ	es	includ	es
included		included		includ	ed	includ	ed
including		including		including		including	
add		add		add		add	
adds		adds		adds		add	s
added		added		add	ed	added	
adding		adding		add	ing	add	ing
continue		continue		continue		continue	
continues		continues		continue	s	continue	s
continued		continued		continu	ed	continu	ed
continuing		continuing		continu	ing	continu	ing
report		report		report		report	

Log posterior for models on token data

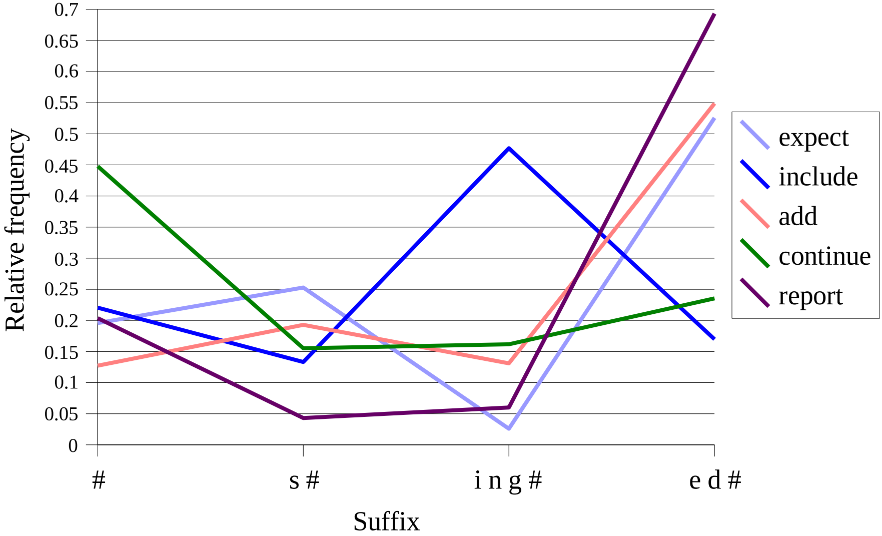


- Correct solution is nowhere near as likely as posterior

$\Rightarrow$ model is wrong!

MACQUARIE UNIVERSITY

Relative frequencies of inflected verb forms



Types and tokens

- A word *type* is a distinct word shape
- A word *token* is an occurrence of a word

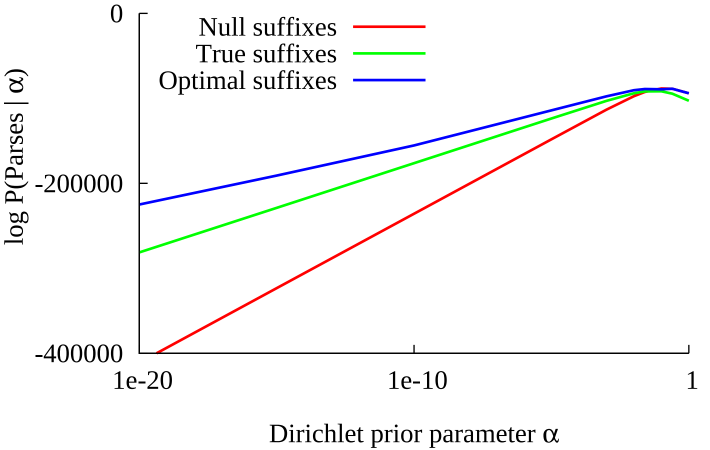
$$\begin{aligned} \text{Data} &= \text{“the cat chased the other cat”} \\ \text{Tokens} &= \text{“the”, “cat”, “chased”, “the”, “other”, “cat”} \\ \text{Types} &= \text{“the”, “cat”, “chased”, “other”} \end{aligned}$$

- Estimating θ from *word types* rather than word tokens eliminates (most) frequency variation
  - 4 common verb suffixes, so when estimating from verb types $\theta_{\text{Suffix}\to\text{i n g \#}} \approx 0.25$
- Several psycholinguists believe that humans learn morphology from word types
- Adaptor grammar mimics Goldwater et al “Interpolating between Types and Tokens” morphology-learning model

Posterior samples from WSJ verb *types*

$\alpha = 0.1$		$\alpha = 10^{-5}$		$\alpha = 10^{-10}$		$\alpha = 10^{-15}$	
expect		expect		expect		exp	ect
expects		expect	s	expect	s	exp	ects
expected		expect	ed	expect	ed	exp	ected
expect	ing	expect	ing	expect	ing	exp	ecting
include		includ	e	includ	e	includ	e
include	s	includ	es	includ	es	includ	es
included		includ	ed	includ	ed	includ	ed
including		includ	ing	includ	ing	includ	ing
add		add		add		add	
adds		add	s	add	s	add	s
add	ed	add	ed	add	ed	add	ed
adding		add	ing	add	ing	add	ing
continue		continu	e	continu	e	continu	e
continue	s	continu	es	continu	es	continu	es
continu	ed	continu	ed	continu	ed	continu	ed
continuing		continu	ing	continu	ing	continu	ing
repo[illegible]		report		repo	rt	rep	ort

Log posterior of models on type data



- Correct solution is close to optimal at $\alpha = 10^{-3}$

Desiderata for an extension of PCFGs

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

Outline

Bayesian inference for Dirichlet-multinomials

- Probability of next event with *uniform Dirichlet prior* with mass α over m outcomes and observed data $\mathbf{Z}_{1:n} = (Z_1, \ldots, Z_n)$

$$\mathrm{P}(Z_{n+1} = k \mid \mathbf{Z}_{1:n}, \alpha) \quad \propto \quad n_k(\mathbf{Z}_{1:n}) + \alpha/m$$

  where $n_k(\mathbf{Z}_{1:n})$ is number of times k appears in $\mathbf{Z}_{1:n}$
- Example: Coin ($m = 2$), $\alpha = 1$, $\mathbf{Z}_{1:2} = (\text{heads}, \text{heads})$
  - $\mathrm{P}(Z_3 = \text{heads} \mid \mathbf{Z}_{1:2}, \alpha) \propto 2.5$
  - $\mathrm{P}(Z_3 = \text{tails} \mid \mathbf{Z}_{1:2}, \alpha) \propto 0.5$

Dirichlet-multinomials with many outcomes

- Predictive probability:

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

- Suppose the number of outcomes $m \gg n$. Then:

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

- But *most outcomes will be unobserved*, so:

$$\mathrm{P}(Z_{n+1} \notin \mathbf{Z}_{1:n} \mid \mathbf{Z}_{1:n}, \alpha) \;\propto\; \alpha$$

From Dirichlet-multinomials to Chinese Restaurant Processes

...

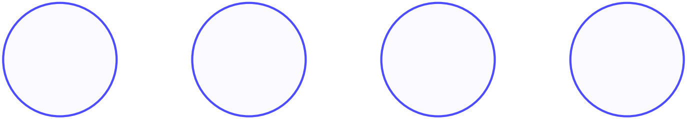
- Suppose *number of outcomes is unbounded* but *we* pick the event labels
- If we number event types in order of occurrence
  $\Rightarrow$ *Chinese Restaurant Process*

$$Z_1 = 1$$

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

MACQUARIE UNIVERSITY

Chinese Restaurant Process (0)



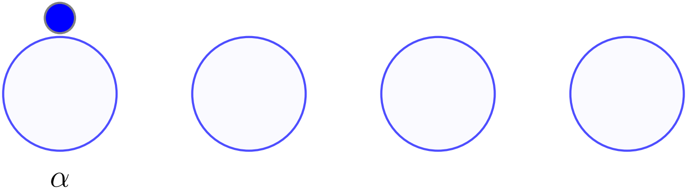
- Customer $\rightarrow$ table mapping $\mathbf{Z} =$
- $\mathrm{P}(\mathbf{z}) = 1$

- Next customer chooses a table according to:

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

MACQUARIE UNIVERSITY

Chinese Restaurant Process (1)

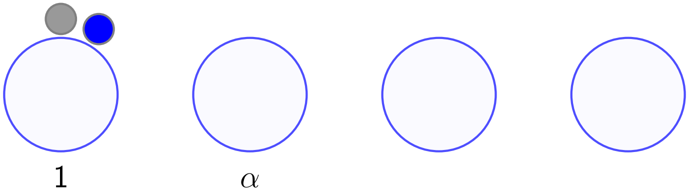


- Customer $\rightarrow$ table mapping $\mathbf{Z} = 1$
- $\mathrm{P}(\mathbf{z}) = \alpha/\alpha$

- Next customer chooses a table according to:

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

Chinese Restaurant Process (2)



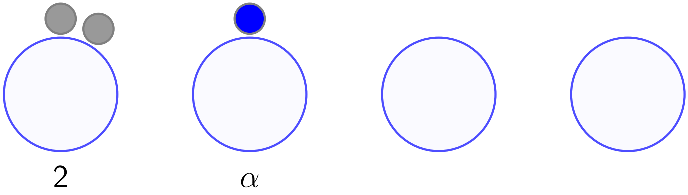
- Customer $\rightarrow$ table mapping $\mathbf{Z} = 1, 1$
- $\mathrm{P}(\mathbf{z}) = \alpha/\alpha \times 1/(1+\alpha)$

- Next customer chooses a table according to:

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

MACQUARIE UNIVERSITY

Chinese Restaurant Process (3)

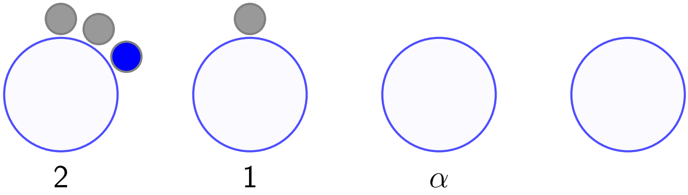


- Customer $\rightarrow$ table mapping $\mathbf{Z} = 1, 1, 2$
- $\mathrm{P}(\mathbf{z}) = \alpha/\alpha \times 1/(1+\alpha) \times \alpha/(2+\alpha)$

- Next customer chooses a table according to:

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

Chinese Restaurant Process (4)

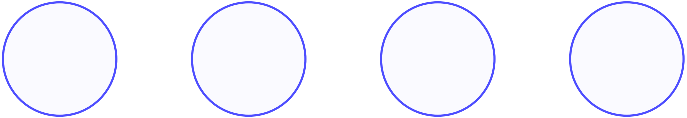


- Customer $\rightarrow$ table mapping $\mathbf{Z} = 1, 1, 2, 1$
- $\mathrm{P}(\mathbf{z}) = \alpha/\alpha \times 1/(1+\alpha) \times \alpha/(2+\alpha) \times 2/(3+\alpha)$

- Next customer chooses a table according to:

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

Labeled Chinese Restaurant Process (0)

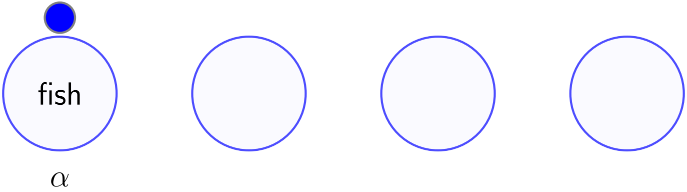


- Table $\rightarrow$ label mapping $\mathbf{Y} =$
- Customer $\rightarrow$ table mapping $\mathbf{Z} =$
- Output sequence $\mathbf{X} =$
- $\mathrm{P}(\mathbf{X}) = 1$

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

MACQUARIE UNIVERSITY

Labeled Chinese Restaurant Process (1)

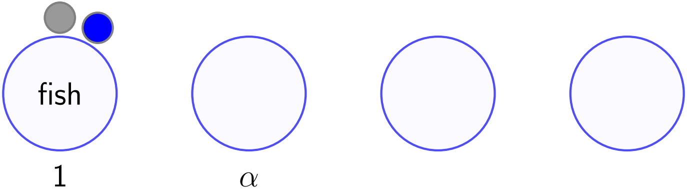


- Table $\rightarrow$ label mapping $\mathbf{Y} = \text{fish}$
- Customer $\rightarrow$ table mapping $\mathbf{Z} = 1$
- Output sequence $\mathbf{X} = \text{fish}$
- $\mathrm{P}(\mathbf{X}) = \alpha/\alpha \times \mathrm{P}_0(\text{fish})$

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

MACQUARIE UNIVERSITY

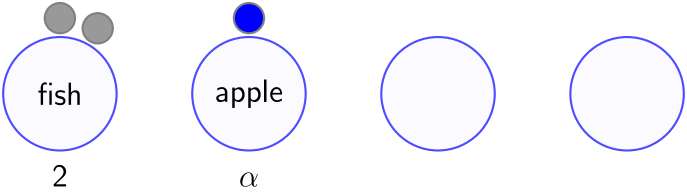
Labeled Chinese Restaurant Process (2)



- Table → label mapping $\mathbf{Y}$ = fish
- Customer → table mapping $\mathbf{Z} = 1, 1$
- Output sequence $\mathbf{X}$ = fish,fish
- $\mathrm{P}(\mathbf{X}) = \mathrm{P}_0(\text{fish}) \times 1/(1 + \alpha)$

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

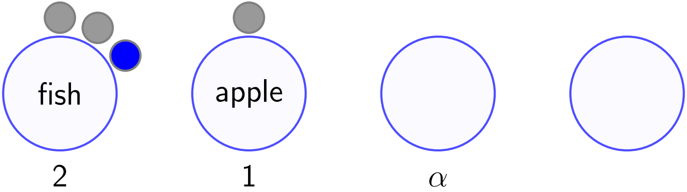
Labeled Chinese Restaurant Process (3)



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

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

Labeled Chinese Restaurant Process (4)



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

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

MACQUARIE UNIVERSITY

Summary: Chinese Restaurant Processes

- *Chinese Restaurant Processes* (CRPs) generalise Dirichlet-Multinomials to an *unbounded number of outcomes*
  - *concentration parameter* α controls how likely a new outcome is
  - CRPs exhibit a *rich get richer* power-law behaviour
- *Pitman-Yor Processes* (PYPs) generalise CRPs with an additional concentration parameter
  - this parameter specifies the asymptotic power-law behaviour
- *Labeled CRPs* use a *base distribution* to define distributions over arbitrary objects
  - base distribution *"labels the tables"*
  - base distribution can have *infinite support*
  - concentrates mass on a countable subset
  - power-law behaviour $\Rightarrow$ Zipfian distributions

Nonparametric extensions of PCFGs

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

    $\Rightarrow$ infinite PCFG (Finkel et al 2007, Liang et al 2007)
  - let *the number of rules grow unboundedly*
    - "new" rules are compositions of several rules from base grammar
    - equivalent to *caching tree fragments*

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

Outline

Learning Probabilistic Context-Free Grammars

Chinese Restaurant Processes

Adaptor grammars

Adaptor grammars for unsupervised word segmentation

Synergies in learning syllables and words

Adaptor grammars for Sesotho morphology

Topic models and learning the referents of words

Learning collocations in LDA topic models

Bayesian inference for adaptor grammars

Conclusion

MACQUARIE UNIVERSITY

Adaptor grammars: informal description

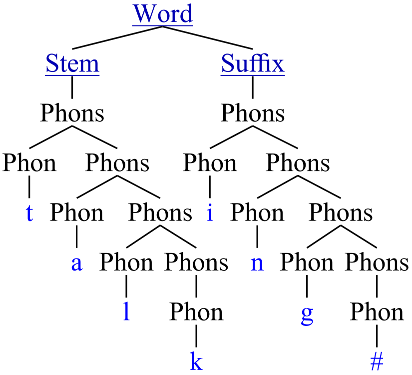
- The trees generated by an adaptor grammar are defined by CFG rules as in a CFG
- A subset of the nonterminals are *adapted*
- *Unadapted nonterminals* expand by picking a rule and recursively expanding its children, as in a PCFG
- *Adapted nonterminals* can expand in two ways:
  - by picking a rule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

From PCFGs to Adaptor grammars

- An adaptor grammar is a PCFG where a subset of the nonterminals are *adapted*
- **Adaptor grammar generative process:**
  - to expand an *unadapted nonterminal* B: (just as in PCFG)
    - select a *rule* $B \to \beta \in R$ with prob. $\theta_{B \to \beta}$, and recursively expand nonterminals in β
  - to expand an *adapted nonterminal* B:
    - select a *previously generated subtree* T_B with prob. $\propto$ number of times T_B was generated, or
    - select a *rule* $B \to \beta \in R$ with prob. $\propto \alpha_B \, \theta_{B \to \beta}$, and recursively expand nonterminals in β

MACQUARIE UNIVERSITY

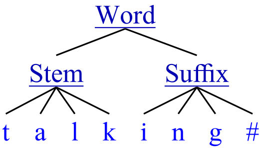
Adaptor grammar for stem-suffix morphology



Word → Stem Suffix
Stem → Phons
Suffix → Phons
Phons → Phon
Phons → Phon Phons

or in *abbreviated form* with
non-adapted nonterminals suppressed

Word → Stem Suffix
Stem → Phon^{+}
Suffix → Phon^{+}



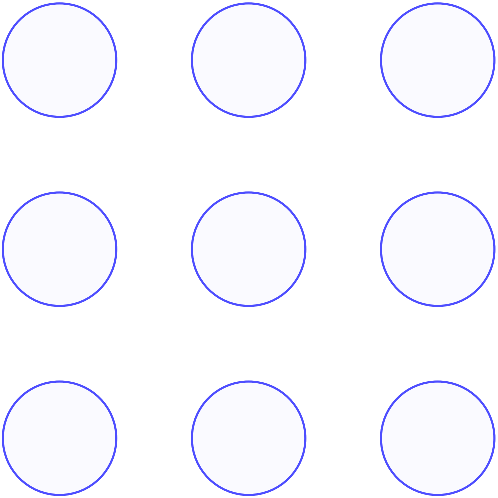
MACQUARIE
UNIVERSITY

Adaptor grammar for stem-suffix morphology (0)

Word $\rightarrow$ Stem Suffix

Stem $\rightarrow$ Phoneme^{+}

Suffix $\rightarrow$ Phoneme*



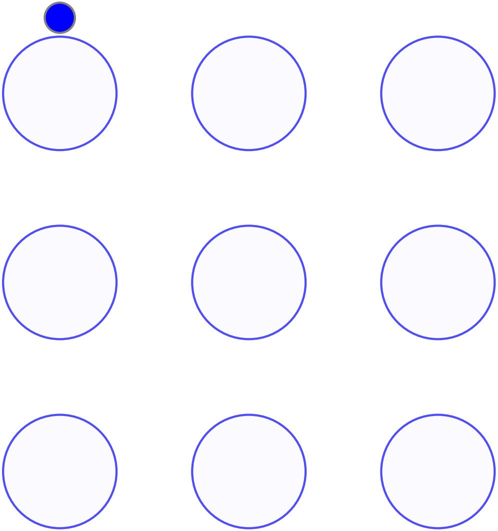
Generated words:

Adaptor grammar for stem-suffix morphology (1a)

Word $\rightarrow$ Stem Suffix

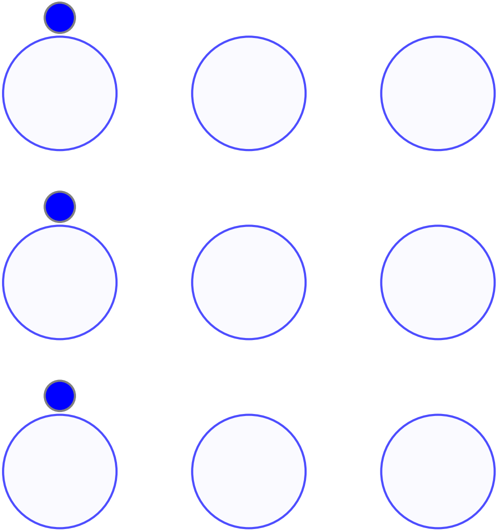
Stem $\rightarrow$ Phoneme^{+}

Suffix $\rightarrow$ Phoneme*



Generated words:

Adaptor grammar for stem-suffix morphology (1b)



Word $\rightarrow$ Stem Suffix

Stem $\rightarrow$ Phoneme^{+}

Suffix $\rightarrow$ Phoneme*

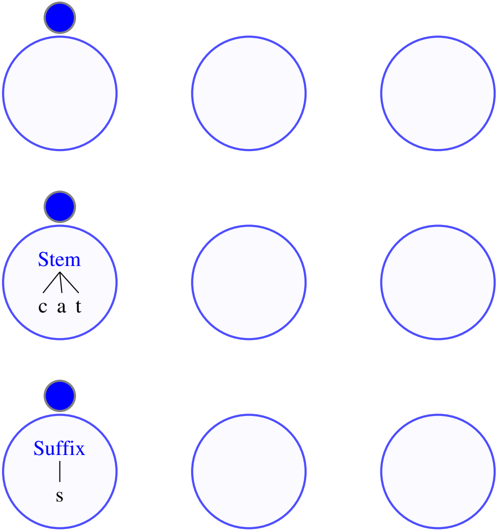
Generated words:

Adaptor grammar for stem-suffix morphology (1c)

Word $\rightarrow$ Stem Suffix

Stem $\rightarrow$ Phoneme^{+}

Suffix $\rightarrow$ Phoneme*



Generated words:

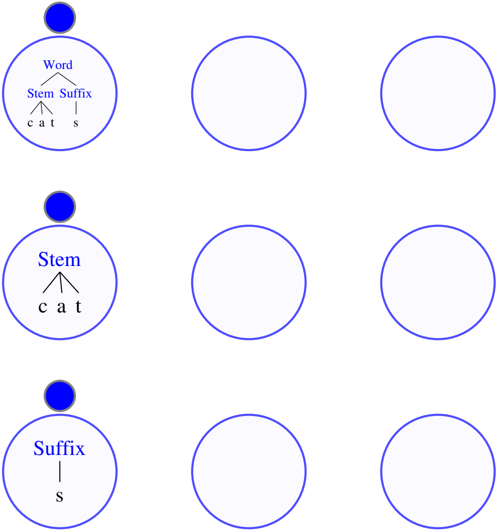
MACQUARIE UNIVERSITY

Adaptor grammar for stem-suffix morphology (1d)

Word $\rightarrow$ Stem Suffix

Stem $\rightarrow$ Phoneme^{+}

Suffix $\rightarrow$ Phoneme*



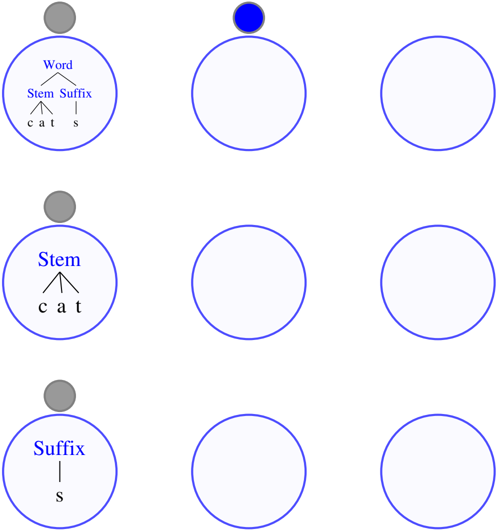
Generated words: cats

Adaptor grammar for stem-suffix morphology (2a)

Word → Stem Suffix

Stem → Phoneme$^+$

Suffix → Phoneme*



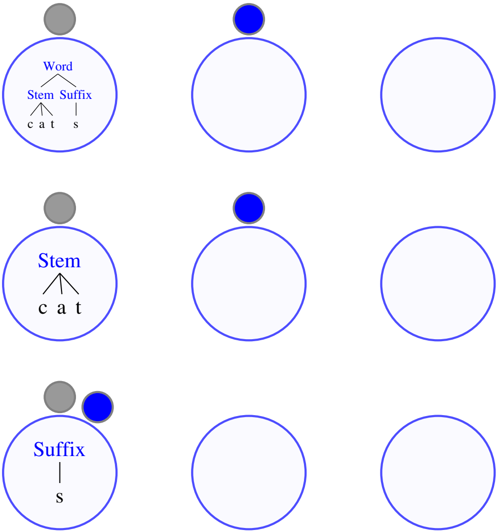
Generated words: cats

Adaptor grammar for stem-suffix morphology (2b)

Word $\rightarrow$ Stem Suffix

Stem $\rightarrow$ Phoneme^{+}

Suffix $\rightarrow$ Phoneme*



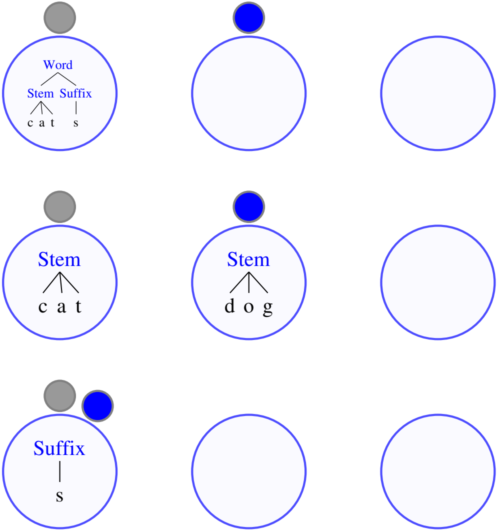
Generated words: cats

Adaptor grammar for stem-suffix morphology (2c)

$\underline{\text{Word}} \rightarrow$ Stem Suffix

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

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



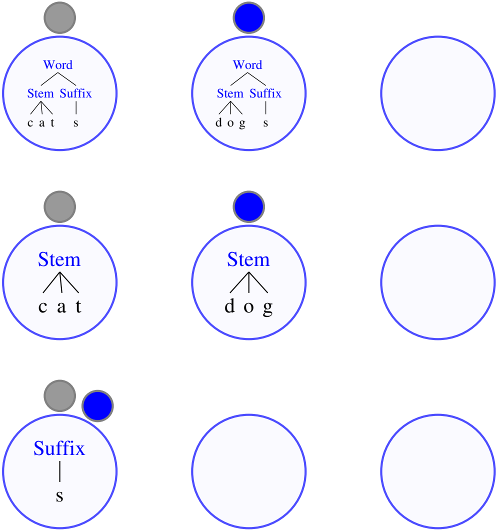
Generated words: cats

Adaptor grammar for stem-suffix morphology (2d)

Word $\rightarrow$ Stem Suffix

Stem $\rightarrow$ Phoneme^{+}

Suffix $\rightarrow$ Phoneme*



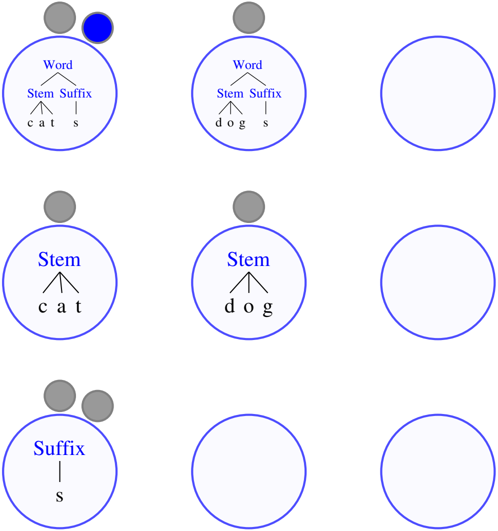
Generated words: cats, dogs

Adaptor grammar for stem-suffix morphology (3)

Word → Stem Suffix

Stem → Phoneme^{+}

Suffix → Phoneme*



Generated words: cats, dogs, cats

MACQUARIE UNIVERSITY

Adaptor grammars as generative processes

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

MACQUARIE UNIVERSITY

Adaptor grammars as non-parametric PCFGs

- An adaptor grammar *reuses whole previously-generated subtrees* T_A of adapted nonterminals A
- This is equivalent to *adding a rule* $A \rightarrow w$ to the grammar, where w is the yield of T_A
  - for implementation efficiency, adaptor grammars constrain w to *only consist of terminals*
  - *Fragment Grammars* (O'Donnell 2009) lift this restriction
- If the base CFG generates an *infinite number of trees* T_A for A, then the adaptor grammar is *non-parametric*
- But any set of sample parses for a *finite training corpus* only contains a *finite number of number of adapted subtrees*

  $\Rightarrow$ *sampling methods* (e.g., MCMC) are a natural approach to learning and parsing adaptor grammars

  - in implementation terms, an adaptor grammar is like a PCFG with a *constantly changing set of rules*

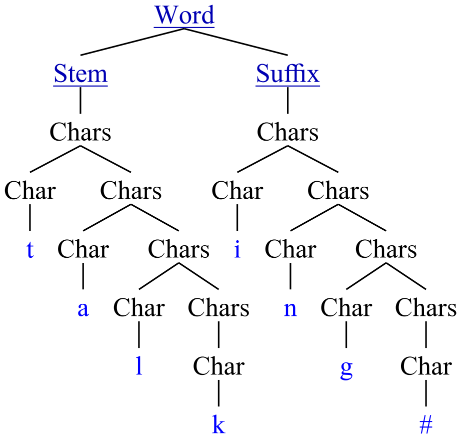
Properties of adaptor grammars

- Probability of reusing an adapted subtree T_A $\propto$ number of times T_A was previously generated
  - adapted subtrees are *not independent*
    - an adapted subtree can be *more probable* than the rules used to construct it
  - but they are *exchangable* $\Rightarrow$ efficient sampling algorithms
  - "rich get richer" $\Rightarrow$ Zipf power-law distributions
- Each adapted nonterminal is associated with a *Chinese Restaurant Process* or *Pitman-Yor Process*
  - CFG rules define *base distribution* of CRP or PYP
- CRP/PYP parameters (e.g., α_A) can themselves be estimated (e.g., slice sampling)

MACQUARIE UNIVERSITY

Bayesian hierarchy inverts grammatical hierarchy

- Grammatically, a Word is composed of a Stem and a Suffix, which are composed of Chars
- To generate a new Word from an Adaptor Grammar:
  - reuse an old Word, or
  - generate a fresh one from the base distribution, i.e., generate a Stem and a Suffix



- *Lower in the tree ⇒ higher in Bayesian hierarchy*

MACQUARIE UNIVERSITY

Outline

MACQUARIE UNIVERSITY

Unsupervised word segmentation

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

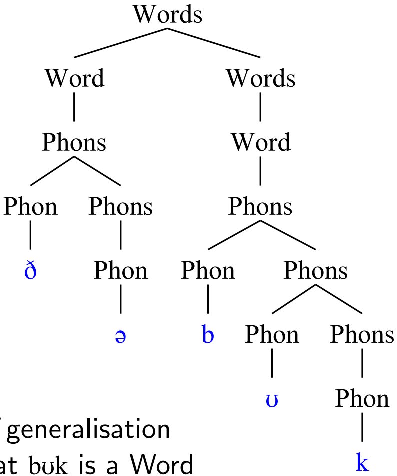
j △ u ▲ w △ ɑ △ n △ t ▲ t △ u ▲ s △ i ▲ ð △ ə ▲ b △ ʊ △ k
"you want to see the book"

- Ignoring phonology and morphology, this involves learning the pronunciations of the lexical items in the language

MACQUARIE UNIVERSITY

CFG models of word segmentation

Words → Word
Words → Word Words
Word → Phons
Phons → Phon
Phons → Phon Phons
Phon → *a* | *b* | ...

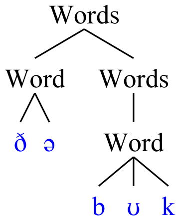


- CFG trees can *describe* segmentation, but
- PCFGs *can't distinguish* good segmentations from bad ones
  - PCFG rules are *too small* a unit of generalisation
  - need to learn e.g., probability that bʊk is a Word

MACQUARIE UNIVERSITY

Towards non-parametric grammars

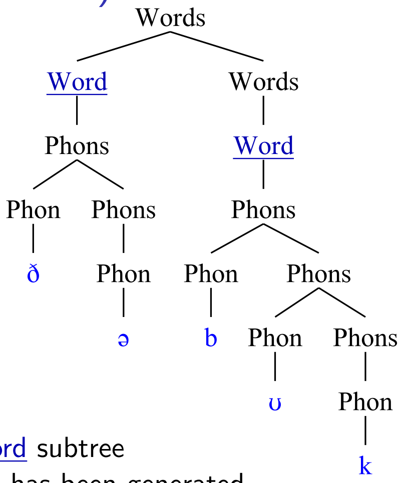
Words $\rightarrow$ Word
Words $\rightarrow$ Word Words
Word $\rightarrow$ *all possible phoneme sequences*



- Learn probability Word $\rightarrow$ b ʊ k
- But *infinitely many possible Word expansions*
  $\Rightarrow$ this grammar is *not a PCFG*
- Given *fixed training data*, only finitely many useful rules
  $\Rightarrow$ use data to choose Word rules as well as their probabilities
- An adaptor grammar can do precisely this!

Unigram adaptor grammar (Brent)

Words → Word
Words → Word Words
Word → Phons
Phons → Phon
Phons → Phon Phons



- Word nonterminal is adapted

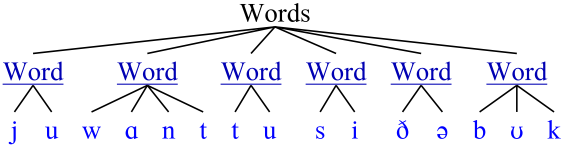
⇒ To generate a Word:

- select a previously generated Word subtree with prob. $\propto$ number of times it has been generated
- expand using Word → Phons rule with prob. $\propto \alpha_{\text{Word}}$ and recursively expand Phons

Unigram model of word segmentation

- Unigram "bag of words" model (Brent):
  - generate a *dictionary*, i.e., a set of words, where each word is a random sequence of phonemes
    - Bayesian prior prefers smaller dictionaries
  - generate each utterance by choosing each word at random from dictionary
- Brent's unigram model as an adaptor grammar:

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



- Accuracy of word segmentation learnt: *56% token f-score* (same as Brent model)
- But we can construct many more word segmentation models using AGs

Adaptor grammar learnt from Brent corpus

- **Initial grammar**

1	Words → <u>Word</u> Words	1	Words → <u>Word</u>
1	<u>Word</u> → Phon		
1	Phons → Phon Phons	1	Phons → Phon
1	Phon → D	1	Phon → G
1	Phon → A	1	Phon → E

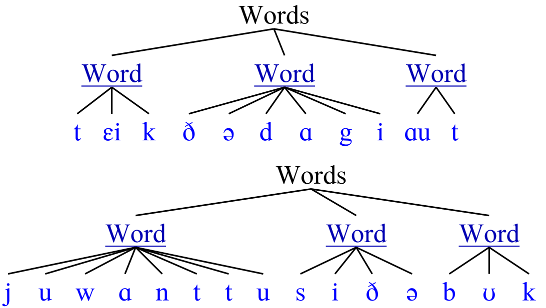
- **A grammar learnt from Brent corpus**

16625	Words → <u>Word</u> Words	9791	Words → <u>Word</u>
1575	<u>Word</u> → Phons		
4962	Phons → Phon Phons	1575	Phons → Phon
134	Phon → D	41	Phon → G
180	Phon → A	152	Phon → E
460	<u>Word</u> → (Phons (Phon y) (Phons (Phon u)))		
446	<u>Word</u> → (Phons (Phon w) (Phons (Phon A) (Phons (Phon t)))		
374	<u>Word</u> → (Phons (Phon D) (Phons (Phon 6)))		
372	<u>Word</u> → (Phons (Phon $\&$) (Phons (Phon n) (Phons (Phon d)))		

Undersegmentation errors with Unigram model

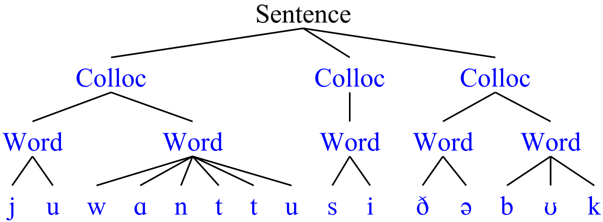
Words $\rightarrow$ Word$^+$ Word $\rightarrow$ Phon$^+$

- 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)



Collocations ⇒ Words

Sentence → Colloc$^+$
Colloc → Word$^+$
Word → Phon$^+$



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

Outline

MACQUARIE UNIVERSITY

Two hypotheses about language acquisition

1. Pre-programmed *staged acquisition* of linguistic components
   - Conventional view of *lexical acquisition*, e.g., Kuhl (2004)
     - child first learns the phoneme inventory, which it then uses to learn
     - phonotactic cues for word segmentation, which are used to learn
     - phonological forms of words in the lexicon, ...
2. *Interactive acquisition* of all linguistic components together
   - corresponds to *joint inference* for all components of language
   - stages in language acquisition might be due to:
     - child's input may contain more information about some components
     - some components of language may be learnable with less data

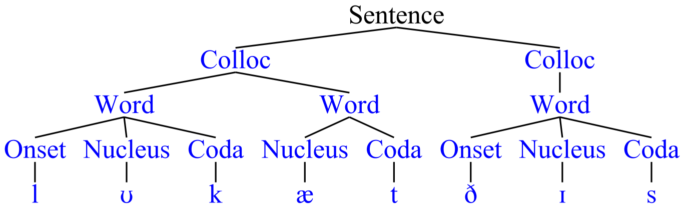
Synergies: an advantage of interactive learning

- An *interactive learner* can take advantage of *synergies in acquisition*
  - partial knowledge of component A provides information about component B
  - partial knowledge of component B provides information about component A
- A staged learner can only take advantage of one of these dependencies
- An interactive or *joint learner* can benefit from a positive feedback cycle between A and B
- Are there synergies in *learning how to segment words* and *learning the referents of words*?

MACQUARIE UNIVERSITY

Jointly learning words and syllables

Sentence $\rightarrow$ Colloc$^+$	Colloc $\rightarrow$ Word$^+$
Word $\rightarrow$ Syllable$^{\{1:3\}}$	Syllable $\rightarrow$ (Onset) Rhyme
Onset $\rightarrow$ Consonant$^+$	Rhyme $\rightarrow$ Nucleus (Coda)
Nucleus $\rightarrow$ Vowel$^+$	Coda $\rightarrow$ Consonant$^+$

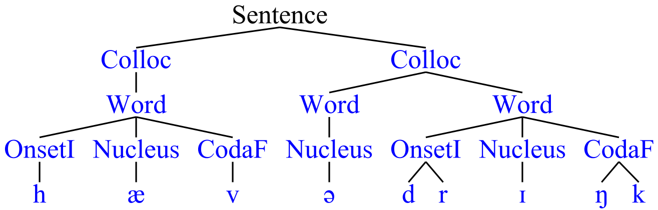


- Rudimentary syllable model (an improved model might do better)
- With 2 Collocation levels, f-score = 84%

Distinguishing internal onsets/codas helps

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

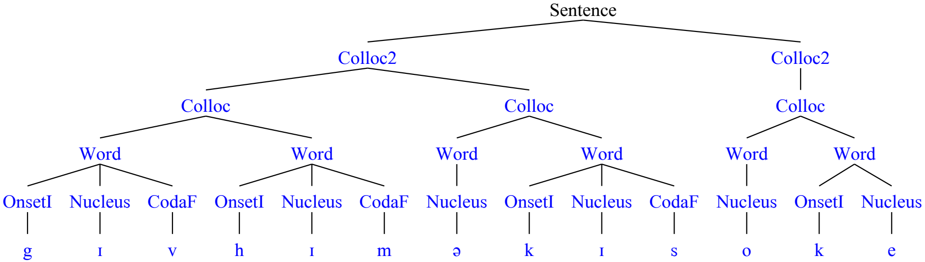
Colloc → Word^+
Word → SyllableI SyllableF
SyllableIF → (OnsetI) RhymeF
RhymeF → Nucleus (CodaF)
CodaF → Consonant^+



- With 2 Collocation levels, not distinguishing initial/final clusters, f-score = 84%
- With 3 Collocation levels, distinguishing initial/final clusters, f-score = 87%

MACQUARIE UNIVERSITY

Collocations^2 ⇒ Words ⇒ Syllables



MACQUARIE UNIVERSITY

Summary of English word segmentation

- Word segmentation accuracy depends on the kinds of generalisations learnt.

Generalization	**Accuracy**
words as units (unigram)	56%
+ associations between words (collocations)	76%
+ syllable structure	84%
+ interaction between segmentation and syllable structure	87%

- *Synergies in learning words and syllable structure*
  - joint inference permits the learner to *explain away* potentially misleading generalizations

MACQUARIE UNIVERSITY

Outline

The Sesotho corpus

- Sesotho is a Bantu language spoken in southern Africa
- Orthography is (roughly) phonemic
  ⇒ use orthographic forms as broad phonemic representations
- Rich agglutinative morphology (especially in verbs)

  u- e- nk- il- e kae
  SM-OM-take-PERF-IN where
  "You took it from where?"
- The Demuth Sesotho corpus (1992) contains transcripts of child and child-directed speech
- We used a subset of size roughly comparable to Brent corpus of infant-directed speech

	Brent	Demuth
utterances	9,790	8,503
word tokens	33,399	30,200
phonemes	95,809	100,113

MACQUARIE UNIVERSITY

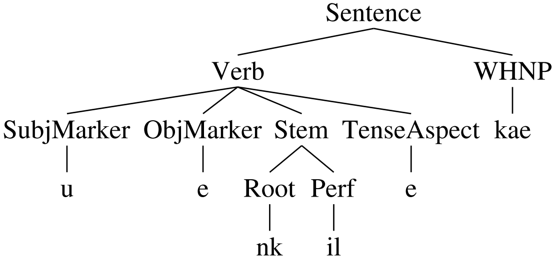
Sesotho verbs are morphologically complex

u- e- nk- il- e kae
SM-OM-take-PERF-IN where
"You took it from where?"

- Input:

u △ e △ n △ k △ i △ l △ e △ k △ a △ e

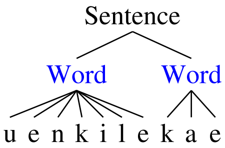
- What I'd like to be able to learn eventually:



Unigram segmentation grammar – word

u- e- nk- il- e kae
SM-OM-take-PERF-IN where
"You took it from where?"

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

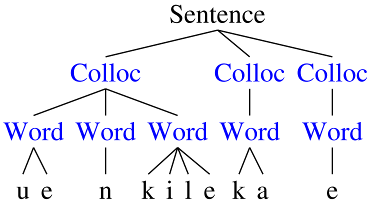


- The word grammar has a word segmentation f-score of 43%
- Lower than 56% f-score on the Brent corpus.
- Sesotho words are longer and more complex.

MACQUARIE UNIVERSITY

Collocation grammar – colloc

Sentence $\rightarrow$ Colloc$^+$
Colloc $\rightarrow$ Word$^+$
Word $\rightarrow$ Phon$^+$



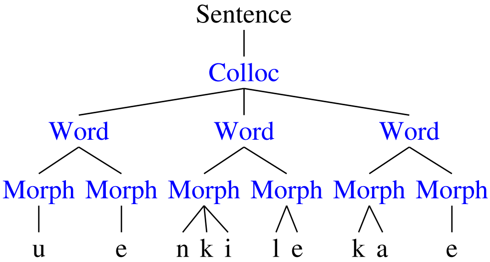
- Learning Collocations improves word segmentation in English; will it help in Sesotho?
- If we treat lower-level units as Words, f-score = 27%
- If we treat upper-level units as Words, f-score = 48%
- English improves by learning dependencies above words, but Sesotho improves by learning generalizations below words

MACQUARIE UNIVERSITY

Adding more levels – colloc2

u- e- nk- il- e kae
SM-OM-take-PERF-IN where
"You took it from where?"

Sentence → Colloc^+
Colloc → Word^+
Word → Morph^+
Morph → Phon^+

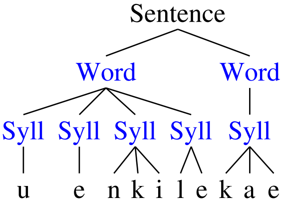


- If two levels are good, maybe three would be better?
- Word segmentation f-score drops to 47%
- Doesn't seem to be much value in adding dependencies above Word level in Sesotho

Using syllable structure – word-syll

u- e- nk- il- e kae
SM-OM-take-PERF-IN where
"You took it from where?"

Sentence → Word$^+$
Word → Syll$^+$
Syll → (Onset) Nuc (Coda)
Syll → SC
Onset → C$^+$
Nuc → V$^+$
Coda → C$^+$



- SC (syllabic consonants) are '*l*', '*m*' '*n*' and '*r*'
- Word segmentation f-score = 50%
- Assuming that words are composed of syllables does improve Sesotho word segmentation

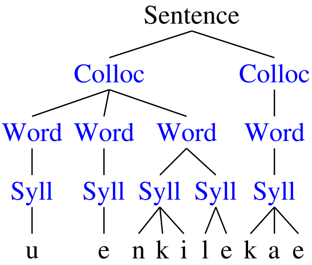
MACQUARIE UNIVERSITY

Using syllable structure – colloc-syll

u-	e-	nk-	il-	e	kae
SM-	OM-	take-	PERF-	IN	where

"You took it from where?"

Sentence → Colloc^+
Colloc → Word^+
Syll → (Onset) Nuc (Coda)
Syll → SC
Onset → C^+
Nuc → V^+
Coda → C^+

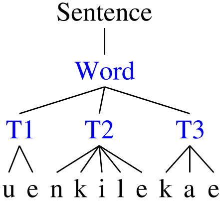


- Word segmentation f-score = 48%
- Additional collocation level doesn't help

Morpheme positions – word-morph

u- e- nk- il- e kae
SM-OM-take-PERF-IN where
"You took it from where?"

Sentence → Word$^+$
Word → T1 (T2 (T3 (T4 (T5))))
T1 → Phon$^+$
T2 → Phon$^+$
T3 → Phon$^+$
T4 → Phon$^+$
T5 → Phon$^+$



- Each word consists of 1–5 morphemes
- Learn separate morphemes for each position
- Improves word segmentation f-score to 53%

Building in language-specific information – word-smorph

u- e- nk- il- e kae

SM-OM-take-PERF-IN where

"You took it from where?"

Sentence → Word$^+$

Word → (P1 (P2 (P3))) T (S)

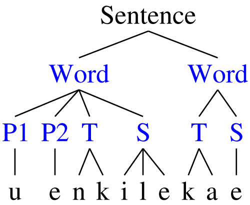
P1 → Phon$^+$

P2 → Phon$^+$

P3 → Phon$^+$

T → Phon$^+$

S → Phon$^+$



- In Sesotho many words consist of a stem T, an optional suffix S and up to 3 prefixes P1, P2 and P3
- Achieves highest f-score = 56%

Outline

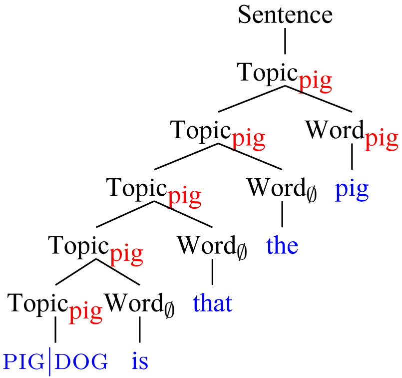
Prior work: mapping words to referents

- Input to learner:
  - word sequence: *Is that the pig?*
  - objects in nonlinguistic context: DOG, PIG
- Learning objectives:
  - identify utterance topic: PIG
  - identify word-topic mapping: *pig* $\mapsto$ PIG

MACQUARIE UNIVERSITY

Frank et al (2009) "topic models" as PCFGs

- Prefix sentences with *possible topic marker*, e.g., PIG|DOG
- PCFG rules *choose a topic* from topic marker and *propagate it through sentence*
- Each word is either generated from sentence topic or null topic ∅



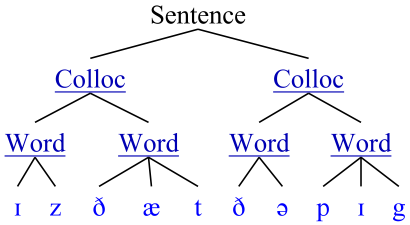
- Grammar can require *at most one topical word per sentence*
- Bayesian inference for PCFG rules and trees corresponds to Bayesian inference for word and sentence topics using topic model (Johnson 2010)

MACQUARIE UNIVERSITY

Word segmentation with adaptor grammars

- Adaptor grammars (AGs) can learn the probability of entire subtrees (as well as rules)
- AGs can express several different word segmentation models
- Learning collocations as well as words significantly improves segmentation accuracy

$$\begin{aligned} &\text{Sentence} \rightarrow \underline{\text{Colloc}}^{+} \\ &\underline{\text{Colloc}} \rightarrow \underline{\text{Word}}^{+} \\ &\underline{\text{Word}} \rightarrow \text{Phon}^{+} \end{aligned}$$

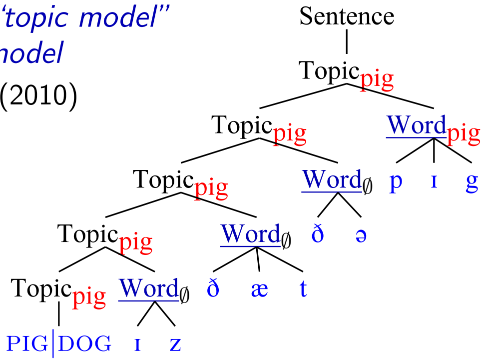


AGs for joint segmentation and referent-mapping

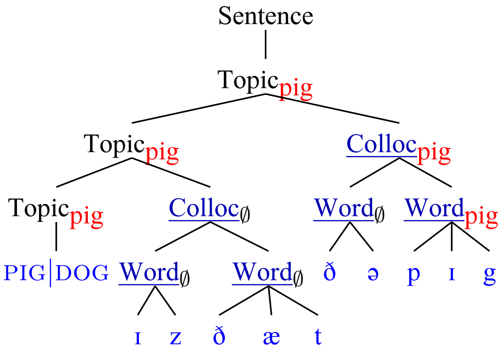
- Combine topic-model PCFG with word segmentation AGs
- Input consists of unsegmented phonemic forms prefixed with possible topics:

PIG|DOG ɪ z ð æ t ð ə p ɪ g

- E.g., combination of *Frank "topic model"* and *unigram segmentation model*
  - equivalent to Jones et al (2010)
- Easy to define *other combinations of topic models and segmentation models*



Collocation topic model AG



- Collocations are either "topical" or not
- Easy to modify this grammar so
  - at most one topical word per sentence, or
  - at most *one topical word per topical collocation*

MACQUARIE UNIVERSITY

Experimental set-up

- Input consists of unsegmented phonemic forms prefixed with possible topics:

  PIG|DOG ɪ z ð æ t ð ə p ɪ g

  - Child-directed speech corpus collected by Fernald et al (1993)
  - Objects in visual context annotated by Frank et al (2009)
- Bayesian inference for AGs using MCMC (Johnson et al 2009)
  - Uniform prior on PYP *a* parameter
  - "Sparse" Gamma$(100, 0.01)$ on PYP *b* parameter
- For each grammar we ran 8 MCMC chains for 5,000 iterations
  - collected word segmentation and topic assignments at every 10th iteration during last 2,500 iterations

    $\Rightarrow$ 2,000 sample analyses per sentence
  - computed and evaluated the modal (i.e., most frequent) sample analysis of each sentence

MACQUARIE UNIVERSITY

Does non-linguistic context help segmentation?

Model		word segmentation
segmentation	**topics**	**token f-score**
unigram	not used	0.533
unigram	any number	0.537
unigram	one per sentence	0.547
collocation	not used	0.695
collocation	any number	0.726
collocation	one per sentence	0.719
collocation	one per collocation	**0.750**

- Not much improvement with unigram model
  - consistent with results from Jones et al (2010)
- Larger improvement with collocation model
  - most gain with *one topical word per topical collocation* (this constraint cannot be imposed on unigram model)

Does better segmentation help topic identification?

- Task: identify object (if any) *this sentence* is about

Model		**sentence referent**	
segmentation	**topics**	**accuracy**	**f-score**
unigram	not used	0.709	0
unigram	any number	0.702	0.355
unigram	one per sentence	0.503	0.495
collocation	not used	0.709	0
collocation	any number	0.728	0.280
collocation	one per sentence	0.440	0.493
collocation	one per collocation	**0.839**	**0.747**

- The collocation grammar with *one topical word per topical collocation* is the only model clearly better than baseline

MACQUARIE UNIVERSITY

Does better segmentation help learning word-to-referent mappings?

- Task: identify *head nouns* of NPs referring to topical objects (e.g. pɪg ↦ PIG in input PIG | DOG ɪ z ð æ t ð ə p ɪ g)

Model		topical word
segmentation	**topics**	**f-score**
unigram	not used	0
unigram	any number	0.149
unigram	one per sentence	0.147
collocation	not used	0
collocation	any number	0.220
collocation	one per sentence	0.321
collocation	one per collocation	**0.636**

- The collocation grammar with one topical word per topical collocation is best at identifying head nouns of referring NPs

MACQUARIE UNIVERSITY

Summary of segmentation and word-to-referent mappings

- *Word to object mapping is learnt more accurately when words are segmented more accurately*
  - improving segmentation accuracy improves topic detection and acquisition of topical words
- *Word segmentation accuracy improves when exploiting non-linguistic context information*
  - incorporating word-topic mapping improves segmentation accuracy (at least with collocation grammars)

⇒ *There are synergies a learner can exploit when learning word segmentation and word-object mappings*
  - Caveat: results seem to depend on details of model

- Models limited by ability to simulate "feature-passing" in a PCFG

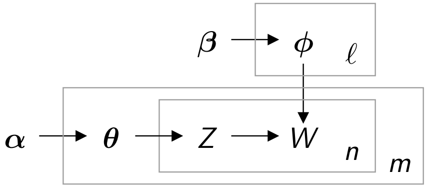
Outline

LDA topic models

- LDA topic models are *admixture models* of documents
  - topics are assigned to *words* (not sentences or documents)
- An LDA topic model learns:
  - the topics expressed in a document
  - the words characteristic of a topic
- Each topic i is a distribution over words $\boldsymbol{\phi}_i$
- Each document j has a *distribution* $\boldsymbol{\theta}_j$ over topics
- To generate document j:
  - for each word position in document:
    - choose a topic z according to $\boldsymbol{\theta}_j$, and then
    - choose a word belonging to that topic according to $\boldsymbol{\phi}_z$
- "Sparse priors" on $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$
  - $\Rightarrow$ most documents have few topics
  - $\Rightarrow$ most topics have few words

LDA topic models as Bayes nets

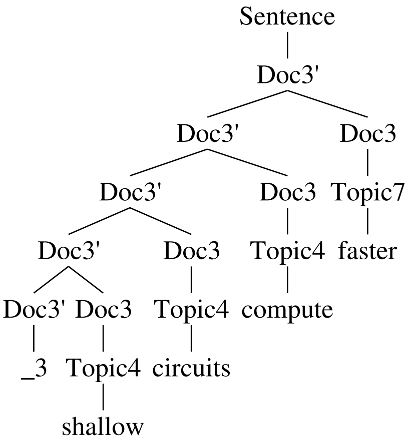
$$
\begin{array}{rcll}
\boldsymbol{\phi}_i & \sim & \mathrm{Dir}(\boldsymbol{\beta}) & i = 1, \ldots, \ell = \text{number of topics} \\
\boldsymbol{\theta}_j & \sim & \mathrm{Dir}(\boldsymbol{\alpha}) & j = 1, \ldots, m = \text{number of documents} \\
z_{j,k} & \sim & \boldsymbol{\theta}_j & j = 1, \ldots, m \\
 & & & k = 1, \ldots, n = \text{number of words in a document} \\
w_{j,k} & \sim & \boldsymbol{\phi}_{z_{j,k}} & j = 1, \ldots, m \\
 & & & k = 1, \ldots, n
\end{array}
$$



LDA topic models as PCFGs (1)

- Prefix strings from document j with a *document identifier* "$\_j$"

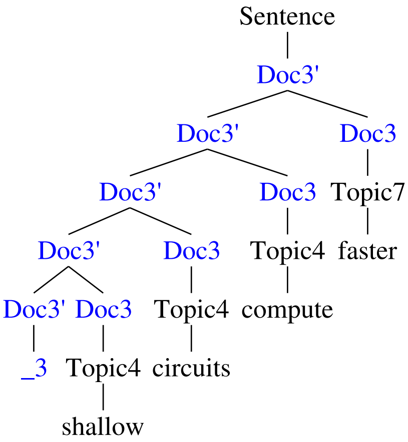
$$
\begin{array}{lll}
\text{Sentence} \rightarrow \ \text{Doc}'_j & & j \in 1, \ldots, m \\
\text{Doc}'_j \rightarrow \ \_j & & j \in 1, \ldots, m \\
\text{Doc}'_j \rightarrow \ \text{Doc}'_j \ \text{Doc}_j & & j \in 1, \ldots, m \\
\text{Doc}_j \rightarrow \text{Topic}_i & & i \in 1, \ldots, \ell \\
 & & j \in 1, \ldots, m \\
\text{Topic}_i \rightarrow \ w & & i \in 1, \ldots, \ell \\
 & & w \in \mathcal{V}
\end{array}
$$



LDA topic models as PCFGs (2)

- Spine *propagates document id up through tree*

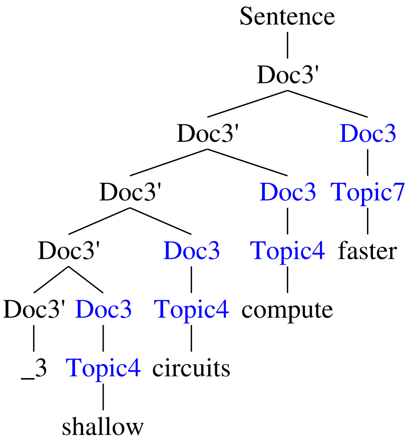
$$
\begin{array}{lll}
\text{Sentence} \rightarrow & \text{Doc}'_j & j \in 1, \ldots, m \\
\text{Doc}'_j \rightarrow & \_j & j \in 1, \ldots, m \\
\text{Doc}'_j \rightarrow & \text{Doc}'_j \ \text{Doc}_j & j \in 1, \ldots, m \\
\text{Doc}_j \rightarrow & \text{Topic}_i & i \in 1, \ldots, \ell \\
 & & j \in 1, \ldots, m \\
\text{Topic}_i \rightarrow & w & i \in 1, \ldots, \ell \\
 & & w \in \mathcal{V}
\end{array}
$$



LDA topic models as PCFGs (3)

- $\mathrm{Doc}_j \rightarrow \mathrm{Topic}_i$ rules map *documents to topics*

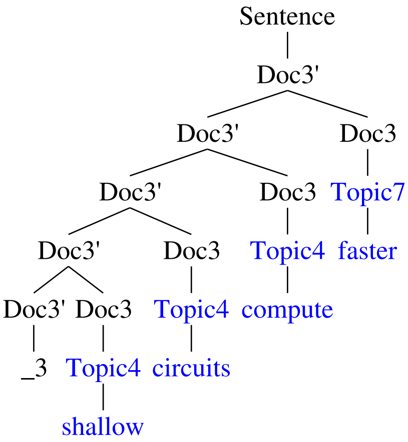
$$
\begin{array}{lll}
\mathrm{Sentence} \rightarrow & \mathrm{Doc}'_j & j \in 1, \ldots, m \\
\mathrm{Doc}'_j \rightarrow & \_j & j \in 1, \ldots, m \\
\mathrm{Doc}'_j \rightarrow & \mathrm{Doc}'_j \ \mathrm{Doc}_j & j \in 1, \ldots, m \\
\mathrm{Doc}_j \rightarrow & \mathrm{Topic}_i & i \in 1, \ldots, \ell \\
 & & j \in 1, \ldots, m \\
\mathrm{Topic}_i \rightarrow & w & i \in 1, \ldots, \ell \\
 & & w \in \mathcal{V}
\end{array}
$$



LDA topic models as PCFGs (4)

- $\mathrm{Topic}_i \to\ w$ rules map *topics to words*

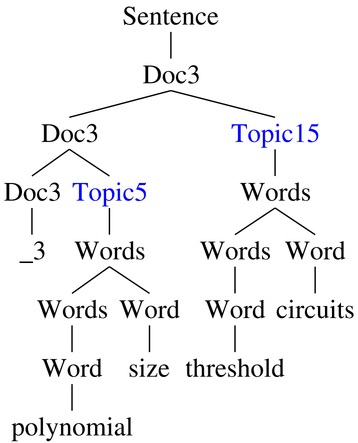
$$
\begin{array}{lll}
\mathrm{Sentence} \to\ \mathrm{Doc}'_j & & j \in 1, \ldots, m \\
\mathrm{Doc}'_j \to\ \_{}_j & & j \in 1, \ldots, m \\
\mathrm{Doc}'_j \to\ \mathrm{Doc}'_j\ \mathrm{Doc}_j & & j \in 1, \ldots, m \\
\mathrm{Doc}_j \to \mathrm{Topic}_i & & i \in 1, \ldots, \ell \\
 & & j \in 1, \ldots, m \\
\mathrm{Topic}_i \to\ w & & i \in 1, \ldots, \ell \\
 & & w \in \mathcal{V}
\end{array}
$$



Topic model with collocations

- Combines *PCFG topic model* and *segmentation adaptor grammar*

$$
\begin{array}{ll}
\text{Sentence} \to \text{Doc}_j & j \in 1, \ldots, m \\
\text{Doc}_j \to \_j & j \in 1, \ldots, m \\
\text{Doc}_j \to \text{Doc}_j \ \text{Topic}_i & i \in 1, \ldots, \ell; \\
 & j \in 1, \ldots, m \\
\underline{\text{Topic}_i} \to \text{Words} & i \in 1, \ldots, \ell \\
\text{Words} \to \text{Word} & \\
\text{Words} \to \text{Words Word} & \\
\text{Word} \to w & w \in \mathcal{V}
\end{array}
$$



Finding topical collocations in NIPS abstracts

- Run topical collocation adaptor grammar on NIPS corpus
- Run with $\ell = 20$ topics (i.e., 20 distinct Topic_i nonterminals)
- Corpus is segmented by punctuation
  - terminal strings are fairly short
  - $\Rightarrow$ inference is fairly efficient
- Used standard AG implementation
  - Pitman-Yor adaptors
  - sampled Pitman-Yor *a* and *b* parameters
  - flat and "vague Gamma" priors on Pitman-Yor *a* and *b* parameters

Sample output on NIPS corpus, 20 topics

- Multiword subtrees learned by adaptor grammar:

T_0 → gradient descent	T_1 → associative memory
T_0 → cost function	T_1 → standard deviation
T_0 → fixed point	T_1 → randomly chosen
T_0 → learning rates	T_1 → hamming distance
T_3 → membrane potential	T_10 → ocular dominance
T_3 → action potentials	T_10 → visual field
T_3 → visual system	T_10 → nervous system
T_3 → primary visual cortex	T_10 → action potential

- Sample skeletal parses:

  _3 (T_5 polynomial size) (T_15 threshold circuits)
  _4 (T_11 studied) (T_19 pattern recognition algorithms)
  _4 (T_2 feedforward neural network) (T_1 implements)
  _5 (T_11 single) (T_10 ocular dominance stripe) (T_12 low)
  (T_3 ocularity) (T_12 drift rate)

Outline

What do we have to learn?

- To learn an adaptor grammar, we need:
  - probabilities of grammar rules
  - adapted subtrees and their probabilities for adapted non-terminals
- If we knew the true parse trees for a training corpus, we could:
  - read off the adapted subtrees from the corpus
  - count rules and adapted subtrees in corpus
  - compute the rule and subtree probabilities from these counts
    - simple computation (smoothed relative frequencies)
- If we aren't given the parse trees:
  - there can be *infinitely many* possible adapted subtrees

  $\Rightarrow$ can't track the probability of all of them (as in EM)
  - but *sample parses of a finite corpus* only include finitely many
- Sampling-based methods learn the relevant subtrees as well as their weights

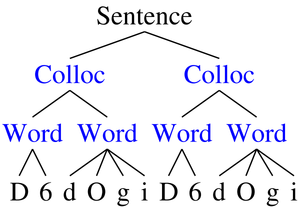
A Gibbs sampler for learning adaptor grammars

- Gibbs sampling for learning adaptor grammars
  - Assign (random) parse trees to each sentence, and compute rule and subtree counts
  - Repeat forever:
    - pick a sentence (and corresponding parse) at random
    - deduct the counts for the sentence's parse from current rule and subtree counts
    - sample a parse for sentence according to updated grammar
    - add sampled parse's counts to rule and subtree counts
- Sampled parse trees and grammar converges to Bayesian posterior distribution

Sampling parses from an adaptor grammar

- Sampling a parse tree for a sentence is computationally most demanding part of learning algorithm
- Component-wise Metropolis-within-Gibbs sampler for parse trees:
  - adaptor grammar rules and probabilities *change on the fly*
  - construct PCFG *proposal grammar* from adaptor grammar for previous sentences
  - sample a parse from PCFG proposal grammar
  - use accept/reject to convert samples from proposal PCFG to samples from adaptor grammar
- For particular adaptor grammars, there are often more efficient algorithms

Details about sampling parses



- Adaptor grammars are *not context-free*
- The probability of a rule (and a subtree) can change within a single sentence
    - breaks standard dynamic programming
- But with moderate or large corpora, the probabilities don't change by much
    - use Metropolis-Hastings accept/reject with a PCFG proposal distribution
- Rules of PCFG proposal grammar $G'(\mathbf{t}_{-j})$ consist of:
    - rules $A \rightarrow \beta$ from base PCFG: $\theta'_{A \rightarrow \beta} \ \propto \ \alpha_A \theta_{A \rightarrow \beta}$
    - A rule $A \rightarrow \textsc{Yield}(t)$ for each table t in A's restaurant: $\theta'_{A \rightarrow \textsc{Yield}(t)} \ \propto \ n_t$, the number of customers at table t
- Map parses using $G'(\mathbf{t}_{-j})$ back to adaptor grammar parses

MACQUARIE UNIVERSITY

Random vs incremental initialization

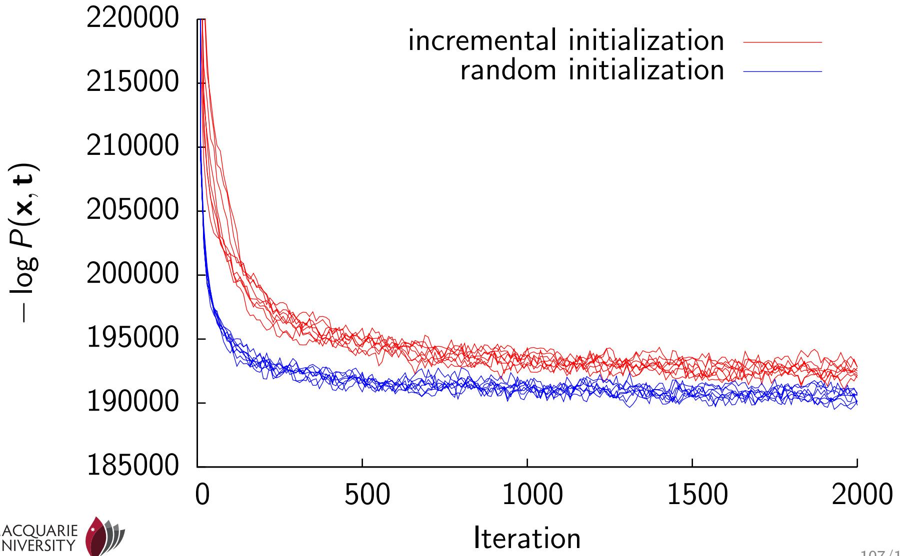
- The Gibbs sampler parse trees $\mathbf{t}$ needs to be initialized somehow
  Random initialization: Assign each string x_i a random parse t_i generated by base PCFG
  Incremental initialization: Sample t_i from $\mathrm{P}(t \mid x_i, \mathbf{t}_{1:i-1})$
- Incremental initialization is easy to implement in a Gibbs sampler
- Incremental initialization improves token f-score in all models, especially on simple models

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

but see caveats on next slide!

MACQUARIE UNIVERSITY

Incremental initialization produces low-probability parses



MACQUARIE UNIVERSITY

Why incremental initialization produces low-probability parses

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

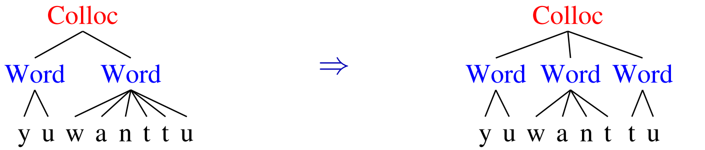
Table label resampling

- Each adapted non-terminal has a CRP with tables labelled with parses
- "Rich get richer" $\Rightarrow$ resampling a sentence's parse reuses the same cached subtrees
- *Resample table labels* as well sentence parses
  - A table label may be used in many sentence parses
  - $\Rightarrow$ Resampling a single table label may change the parses of a single sentence
  - $\Rightarrow$ table label resampling can improve mobility with grammars with a hierarchy of adapted non-terminals
- Essential for grammars with a complex hierarchical structure

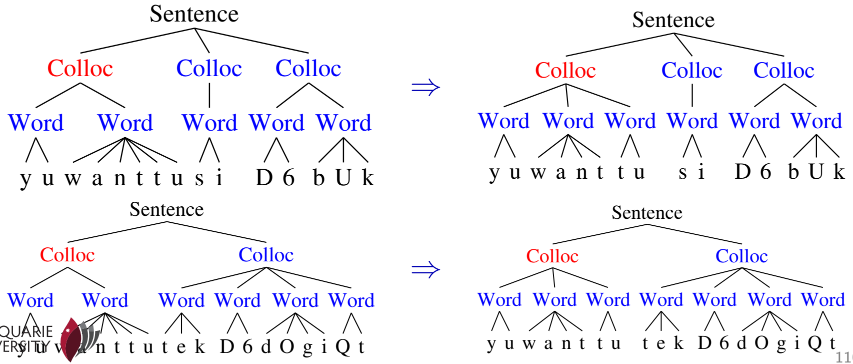
MACQUARIE UNIVERSITY

Table label resampling example

Label on table in Chinese Restaurant for colloc

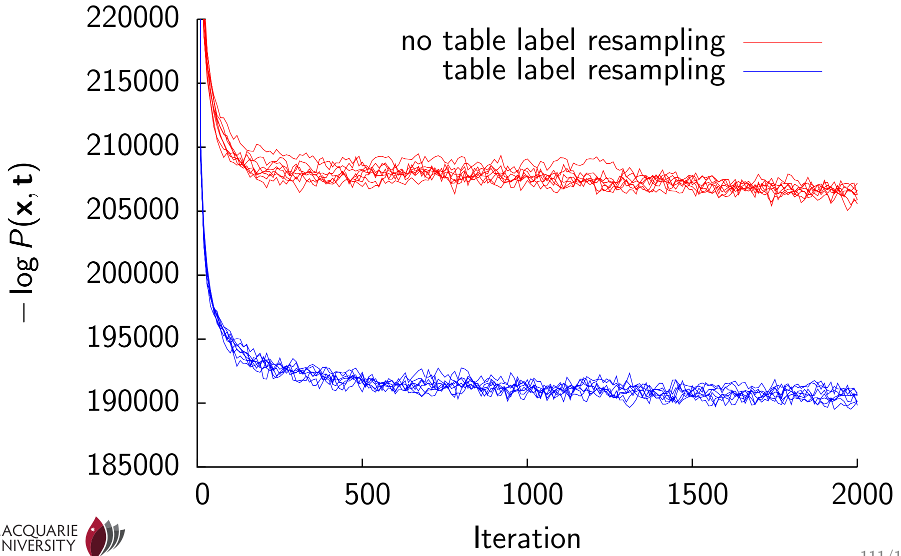


Resulting changes in parse trees



MACQUARIE UNIVERSITY

Table label resampling produces much higher-probability parses



Summary: learning adaptor grammars

- Unbounded number of possible cached subtrees $\Rightarrow$ Expectation Maximisation isn't sufficient
- *Gibbs sampler* batch learning algorithm
  - assign every sentence a (random) parse
  - repeatedly cycle through training sentences:
    - withdraw parse (decrement counts) for sentence
    - sample parse for current sentence and update counts
    - Metropolis-Hastings correction

MACQUARIE UNIVERSITY

Outline

Conclusions and future work

- Adaptor grammars can express a variety of useful HDP models
  - generic AG inference code makes it easy to explore a variety of models
- AGs have a variety of applications
  - unsupervised acquisition of morphology
  - unsupervised word segmentation
  - learning word to referent mappings
  - learning collocations in topic models
- *Joint learning* often uses information in the input more effectively than staged learning
- Future work:
  - extend expressive power of AGs (e.g., feature-passing)
  - richer data (e.g., more non-linguistic context)
  - more realistic data (e.g., stress, phonological variation)

The future of Bayesian models of language acquisition

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

- So far our grammars and priors don't encode much linguistic knowledge, but in principle they can!
  - how do we represent this knowledge?
  - how can we learn efficiently using this knowledge?
- Should permit us to *empirically investigate effects of specific universals on the course of language acquisition*
- My guess: the interaction between innate knowledge and learning will be *richer and more interesting* than either the rationalists or empiricists currently imagine!

Interested in statistical models, machine learning and computational linguistics?

Macquarie University is recruiting **PhD students** and **post-docs**!

Contact **Mark.Johnson@mq.edu.au** for more information.

MACQUARIE UNIVERSITY

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

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

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 $A \in N \cup W$:

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

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

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

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

MACQUARIE UNIVERSITY

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.

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

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

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