Grammars and Topic Models

Mark Johnson
Macquarie University
Sydney
Australia

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

December 2009

Talk summary

- Probabilistic Context-Free Grammars (PCFGs)
  - a way of specifying certain stochastic automata
- LDA topic models as PCFGs
- Non-parametric generalizations of PCFGs
  - "infinite" PCFGs
  - adaptor grammars
- Topic models and adaptor grammars
  - topical collocations
  - the structure of names
  - learning words and their meanings

Outline

Probabilistic Context-Free Grammars

LDA topic models as PCFGs

Adaptor grammars

Adaptor grammars and topic models

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 A with a multinomial distribution $\boldsymbol{\theta}_A$ over the rules $R_A = \{A \rightarrow \alpha\}$ that expand it
- Probability of a tree is the *product of the probabilities of the rules* used to construct it

Rule r	θ_r	Rule r	θ_r
S $\rightarrow$ NP VP	1.0		
NP $\rightarrow$ Sam	0.75	NP $\rightarrow$ Sandy	0.25
VP $\rightarrow$ barks	0.6	VP $\rightarrow$ snores	0.4

$$\mathrm{P}\left(\,[_{\mathrm{S}}\ [_{\mathrm{NP}}\ \mathrm{Sam}]\ [_{\mathrm{VP}}\ \mathrm{barks}]]\,\right) = 0.45 \qquad \mathrm{P}\left(\,[_{\mathrm{S}}\ [_{\mathrm{NP}}\ \mathrm{Sandy}]\ [_{\mathrm{VP}}\ \mathrm{snores}]]\,\right) = 0.1$$

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 $\beta \in (N \cup W)^\star$
- 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:

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

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

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

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

Probabilistic context-free grammars

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

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

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:

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

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

Bayesian inference for rule probabilities

- A PCFG specifies a *multinomial* distribution $\boldsymbol{\theta}_A$ for each nonterminal A over the rules $A \to \alpha$ expanding A
  - $\theta_{A \to \alpha}$ is probability of A expanding to α
  - probability of a parse t is a *product of multinomials*
- Conjugate prior: $\boldsymbol{\theta}_A \mid \boldsymbol{\alpha}_A \sim \mathrm{Dir}(\boldsymbol{\alpha}_A)$ for each nonterminal A
- Given a corpus of *parse trees* $\boldsymbol{t}$, posterior distribution for $\boldsymbol{\theta}_A \mid \boldsymbol{t}, \boldsymbol{\alpha}_A \sim \mathrm{Dir}(\boldsymbol{\alpha}_A + \boldsymbol{n}_A(\boldsymbol{t}))$
- Given a corpus of *strings* $\boldsymbol{w}$ (the *yields* of $\boldsymbol{t}$), joint posterior distribution over
  - rule probabilities $\boldsymbol{\theta}$
  - parse trees $\boldsymbol{t}$

  is intractable, but can be approximated by:
  - variational Bayes (mean field approximation)
  - Markov chain Monte Carlo

Outline

Probabilistic Context-Free Grammars

LDA topic models as PCFGs

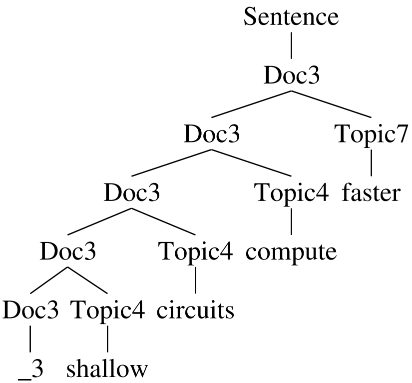
Adaptor grammars

Adaptor grammars and topic models

Conclusion

LDA topic models as PCFGs

- Each document i generates a distribution over m topics
- Each topic j generates a (unigram) distribution over vocabulary $\mathcal{X}$.
- Preprocess input by *prepending a document id to every sentence*

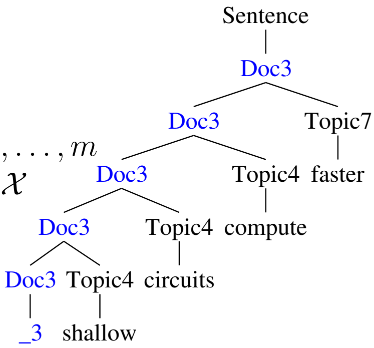


$$
\begin{array}{lll}
\text{Sentence} \rightarrow & \text{Doc}_i & i \in 1, \ldots, n \\
\text{Doc}_i \rightarrow & \_i & i \in 1, \ldots, n \\
\text{Doc}_i \rightarrow & \text{Doc}_i \ \text{Topic}_j & i \in 1, \ldots, n; j \in 1, \ldots, m \\
\text{Topic}_j \rightarrow & x & j \in 1, \ldots, m; x \in \mathcal{X}
\end{array}
$$

Left spine identifies document

- Left spine passes document id throughout sentence

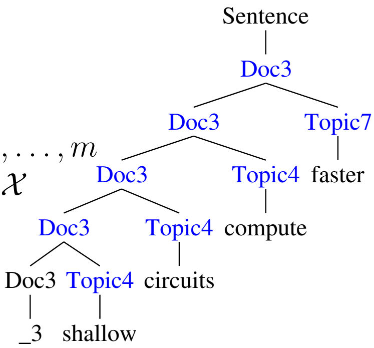
$$
\begin{array}{lll}
\text{Sentence} \rightarrow & \text{Doc}_i & i \in 1, \ldots, n \\
\text{Doc}_i \rightarrow & \_i & i \in 1, \ldots, n \\
\text{Doc}_i \rightarrow & \text{Doc}_i \ \text{Topic}_j & i \in 1, \ldots, n; j \in 1, \ldots, m \\
\text{Topic}_j \rightarrow & x & j \in 1, \ldots, m; x \in \mathcal{X}
\end{array}
$$



Document → topic rules

- Document → topic rules specify probability of topic within document

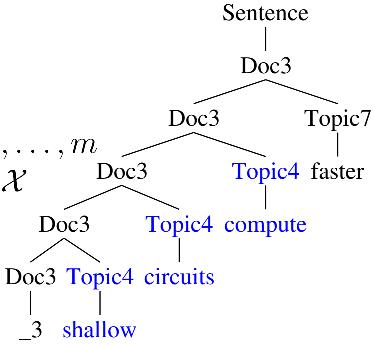
$\text{Sentence} \rightarrow$	Doc_i	$i \in 1, \ldots, n$
$\text{Doc}_i \rightarrow$	$\_i$	$i \in 1, \ldots, n$
$\text{Doc}_i \rightarrow$	$\text{Doc}_i\ \text{Topic}_j$	$i \in 1, \ldots, n; j \in 1, \ldots, m$
$\text{Topic}_j \rightarrow$	x	$j \in 1, \ldots, m; x \in \mathcal{X}$



Topic → word rules

- Topic → word rules specify probability of word within topic

$$
\begin{array}{lll}
\mathrm{Sentence} \to & \mathrm{Doc}_i & i \in 1, \ldots, n \\
\mathrm{Doc}_i \to & \_i & i \in 1, \ldots, n \\
\mathrm{Doc}_i \to & \mathrm{Doc}_i \ \mathrm{Topic}_j & i \in 1, \ldots, n; j \in 1, \ldots, m \\
\mathrm{Topic}_j \to & x & j \in 1, \ldots, m; x \in \mathcal{X}
\end{array}
$$

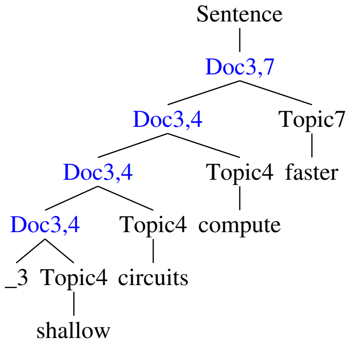


Bayesian inference for LDA PCFGs

- Dirichlet priors on Document $\rightarrow$ Topic and Topic $\rightarrow$ Word distributions
- General-purpose PCFG parsing/estimation algorithms require time cubic in length of sentence
  - not a good idea for long documents!
- More efficient algorithms for these kinds of grammars
  - (standard LDA inference algorithms)
  - predictive (e.g., Earley) parsing algorithms
  - identify compositions finite state automata/transducers that these grammars encode

Extended LDA PCFG (1): sticky topics

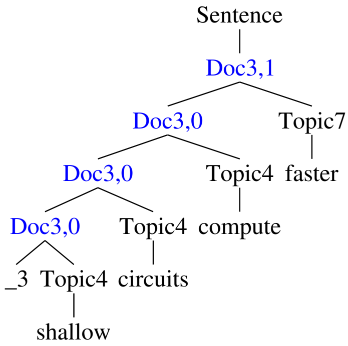
- HMM generating *sequences of topics* in each document
- Non-uniform Dirichlet prior over topic→topic transitions
  ⇒ sticky topics
- Grammar rule schemata given
  - document identifiers $\mathcal{D}$,
  - topics $\mathcal{T} = \{1, \ldots, m\}$, and
  - vocabulary $\mathcal{W}$



$\text{Sentence} \rightarrow \text{Doc}_{d,t}$	for each $d \in \mathcal{D}$ and $t \in \mathcal{T}$
$\text{Doc}_{d,t} \rightarrow d\ \text{Topic}_t$	for each $d \in \mathcal{D}$ and $t \in \mathcal{T}$
$\text{Doc}_{d,t} \rightarrow \text{Doc}_{d,t'}\ \text{Topic}_t$	for each $d \in \mathcal{D}$ and $t, t' \in \mathcal{T}$
$\text{Topic}_t \rightarrow w$	for each $t \in \mathcal{T}$ and $w \in \mathcal{W}$

Extended LDA PCFG (2): document segmentation

- Divide documents into *segments*
- Grammar rule schemata given
  - document identifiers $\mathcal{D}$,
  - *segments* $\mathcal{S} = \{1, \ldots, \ell\}$,
  - topics $\mathcal{T} = \{1, \ldots, m\}$, and
  - vocabulary $\mathcal{W}$



$$
\begin{array}{ll}
\text{Sentence} \rightarrow \text{Doc}_{d,s} & \text{for each } d \in \mathcal{D} \text{ and } s \in \mathcal{S} \\
\text{Doc}_{d,s} \rightarrow d & \text{for each } d \in \mathcal{D} \text{ and } s \in \mathcal{S} \\
\text{Doc}_{d,s} \rightarrow \text{Doc}_{d,s}\ \text{Topic}_t & \text{for each } d \in \mathcal{D},\ s \in \mathcal{S} \text{ and } t \in \mathcal{T} \\
\text{Doc}_{d,s} \rightarrow \text{Doc}_{d,s'} & \text{for each } d \in \mathcal{D} \text{ and } s, s' \in \mathcal{S} \\
\text{Topic}_t \rightarrow w & \text{for each } t \in \mathcal{T} \text{ and } w \in \mathcal{W}
\end{array}
$$

What's the point of using grammars?

- All of these models can be stated directly (without using grammars)
  - grammars don't make inference easier or produce better results
- Grammars provide another way of *describing complex models*
  - mutually recursive hierarchies of sequential structures
- There are generic algorithms for *PCFG parsing and inference*
  - rapid proto-typing of new models
  - may lead to (more) efficient implementation via *stochastic automata*

Outline

Nonparametric generalizations of PCFGs

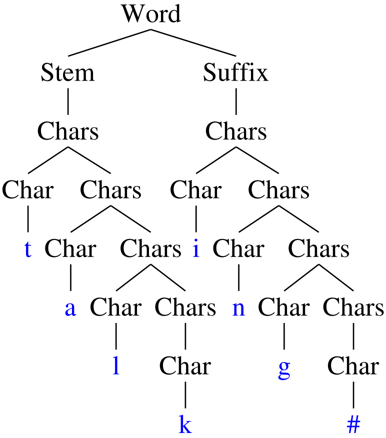
- Two obvious nonparametric extensions of PCFGs:
  - let the number of nonterminals N grow unboundedly
    - refine the nonterminals of an original grammar
      e.g., $\mathrm{S}_{35} \rightarrow \mathrm{NP}_{27}\ \mathrm{VP}_{17}$

    $\Rightarrow$ infinite PCFG
  - let the number of rules R 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 …

A CFG for stem-suffix morphology

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



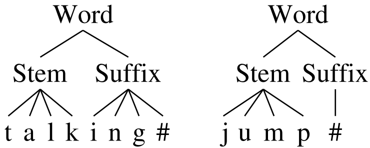
- Grammar's trees can represent any segmentation of words into stems and suffixes

⇒ Can *represent* true segmentation

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

A “CFG” with one rule per possible morpheme

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



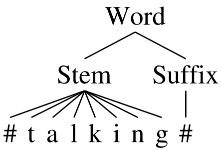
- A rule for each morpheme
  ⇒ “PCFG” can represent probability of each morpheme
- *Unbounded number of possible rules, so this is not a PCFG*
  - not a practical problem, as only a finite set of rules could possibly be used in any particular data set

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

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

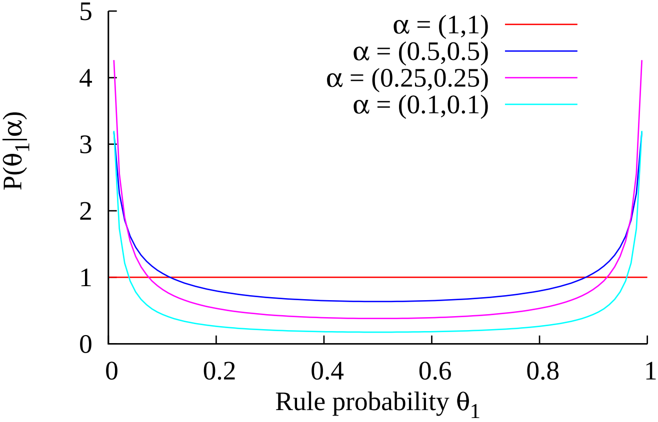
⇒ Saturated model is maximum likelihood estimate

- Maximum likelihood estimate does not find any suffixes



Forcing generalization via sparse Dirichlet priors

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



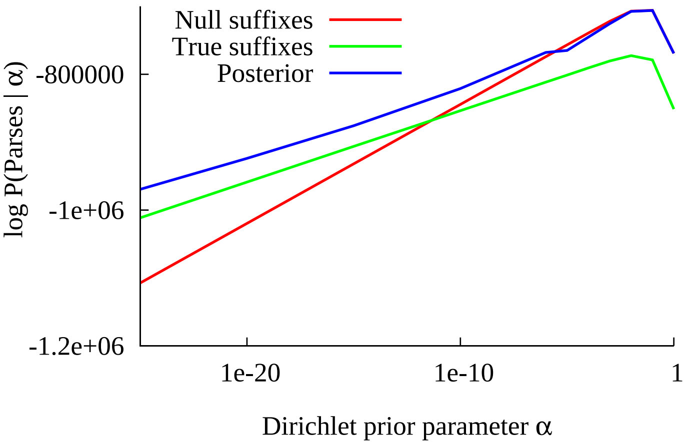
Morphological segmentation experiment

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

Posterior samples from WSJ verb tokens

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

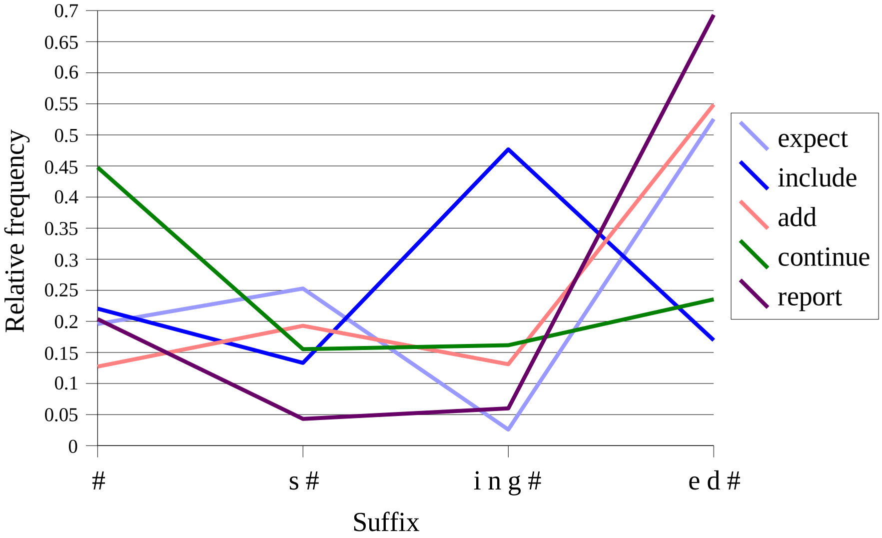
Log posterior for models on token data



- Correct solution is nowhere near as likely as posterior

⇒ model is wrong!

Relative frequencies of inflected verb forms



Types and tokens

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

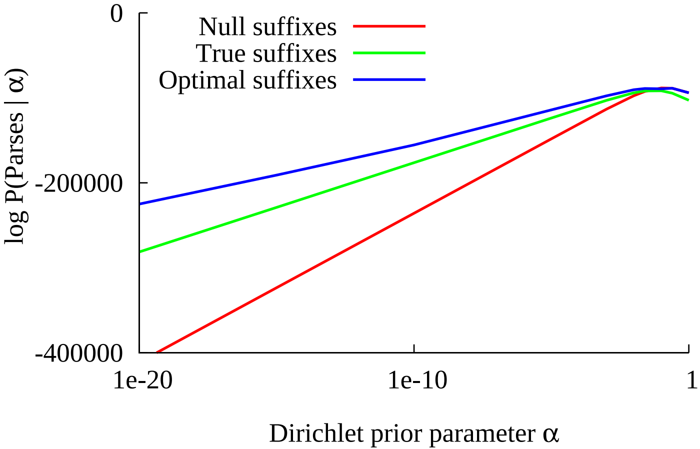
$$\begin{aligned} \text{Data} &= \text{“the cat chased the other cat”} \\ \text{Tokens} &= \text{“the”, “cat”, “chased”, “the”, “other”, “cat”} \\ \text{Types} &= \text{“the”, “cat”, “chased”, “other”} \end{aligned}$$

- Estimating θ from *word types* rather than word tokens eliminates (most) frequency variation
  - 4 common verb suffixes, so when estimating from verb types $\theta_{\text{Suffix}\rightarrow \text{i n g \#}} \approx 0.25$
- 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
report		report		repo	rt	rep	ort

Log posterior of models on type data



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

Desiderata for an extension of PCFGs

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

Adaptor grammars: informal description

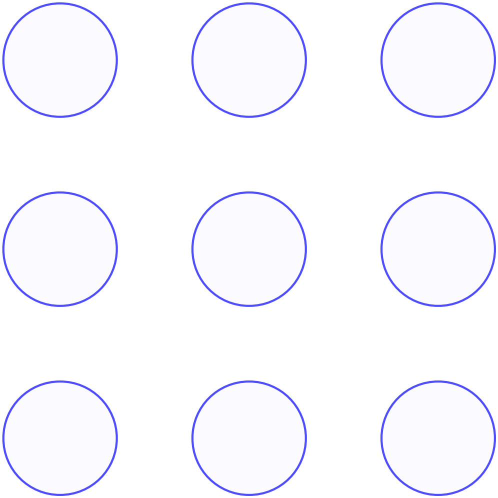
- The trees generated by an adaptor grammar are defined by CFG rules as in a CFG
- A subset of the nonterminals are *adapted*
- *Unadapted nonterminals* expand by picking a rule and recursively expanding its children, as in a PCFG
- *Adapted nonterminals* can expand in two ways:
  - by picking a rule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

Adaptor grammar for stem-suffix morphology (0)

Word → Stem Suffix

Stem → Phoneme$^+$

Suffix → Phoneme*



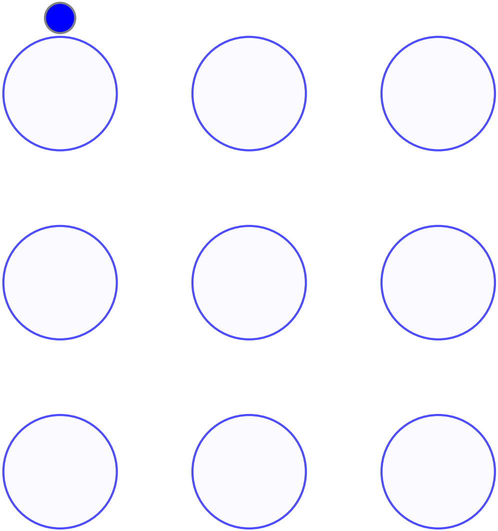
Generated words:

Adaptor grammar for stem-suffix morphology (1a)

Word $\rightarrow$ Stem Suffix

Stem $\rightarrow$ Phoneme^{+}

Suffix $\rightarrow$ Phoneme*



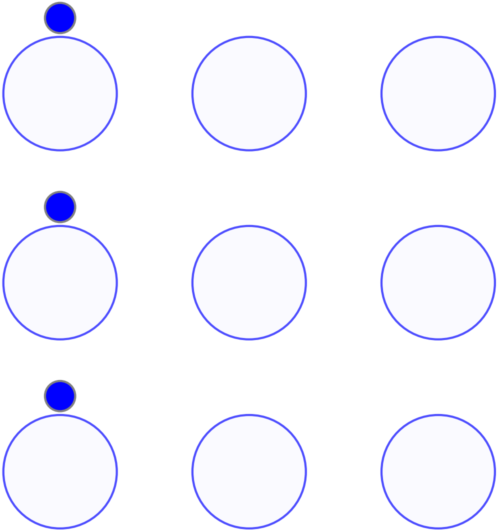
Generated words:

Adaptor grammar for stem-suffix morphology (1b)

Word $\rightarrow$ Stem Suffix

Stem $\rightarrow$ Phoneme^{+}

Suffix $\rightarrow$ Phoneme*



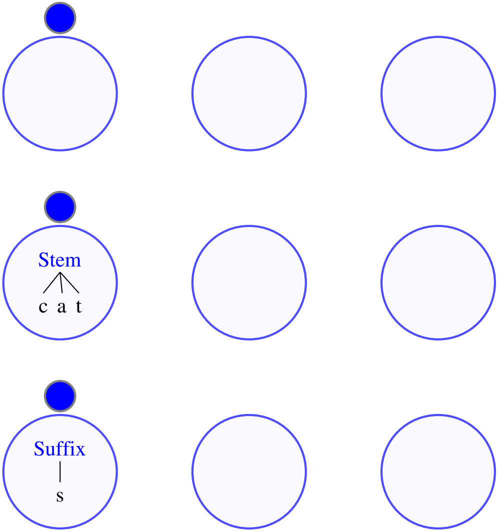
Generated words:

Adaptor grammar for stem-suffix morphology (1c)

Word → Stem Suffix

Stem → Phoneme$^+$

Suffix → Phoneme*



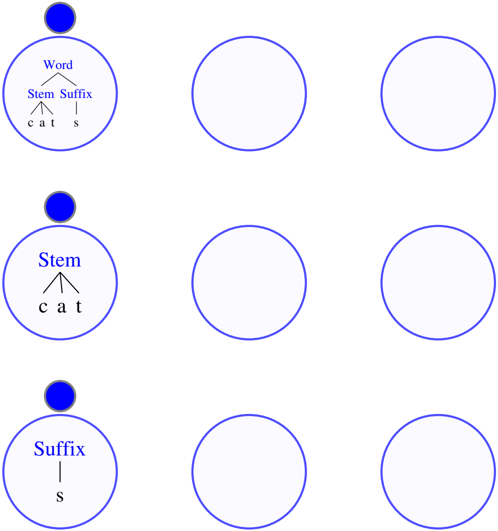
Generated words:

Adaptor grammar for stem-suffix morphology (1d)

<u>Word</u> → Stem Suffix

<u>Stem</u> → Phoneme$^+$

<u>Suffix</u> → Phoneme*



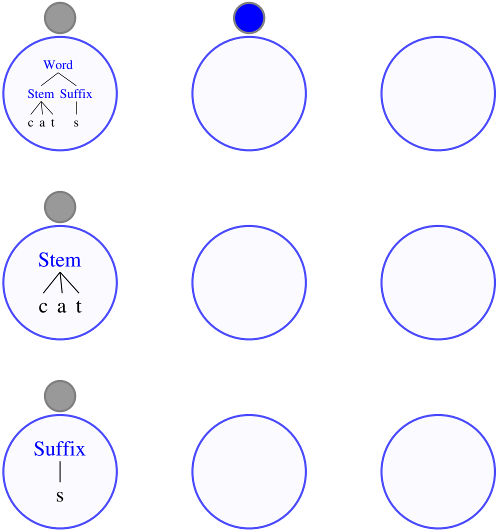
Generated words: cats

Adaptor grammar for stem-suffix morphology (2a)

Word → Stem Suffix

Stem → Phoneme$^+$

Suffix → Phoneme*



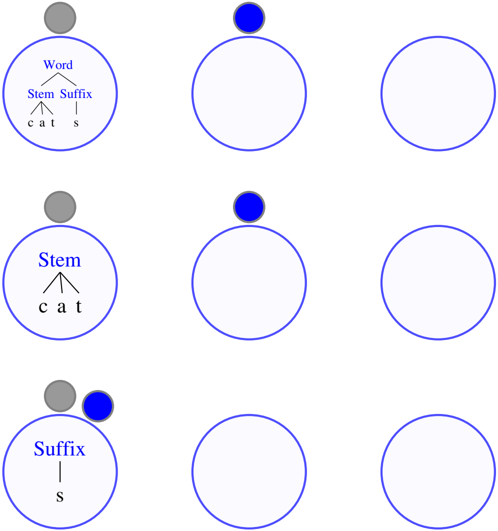
Generated words: cats

Adaptor grammar for stem-suffix morphology (2b)

Word → Stem Suffix

Stem → Phoneme^{+}

Suffix → Phoneme*



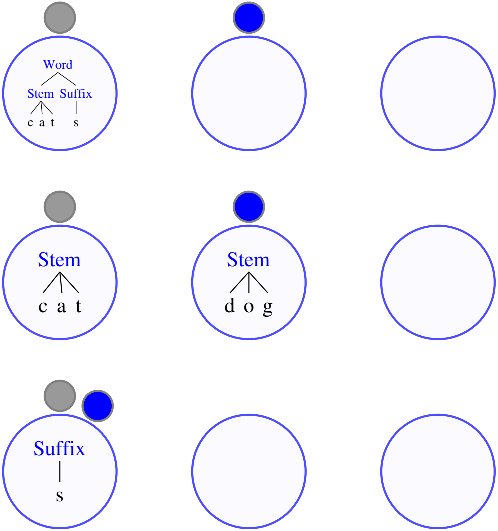
Generated words: cats

Adaptor grammar for stem-suffix morphology (2c)

Word $\rightarrow$ Stem Suffix

Stem $\rightarrow$ Phoneme^{+}

Suffix $\rightarrow$ Phoneme*



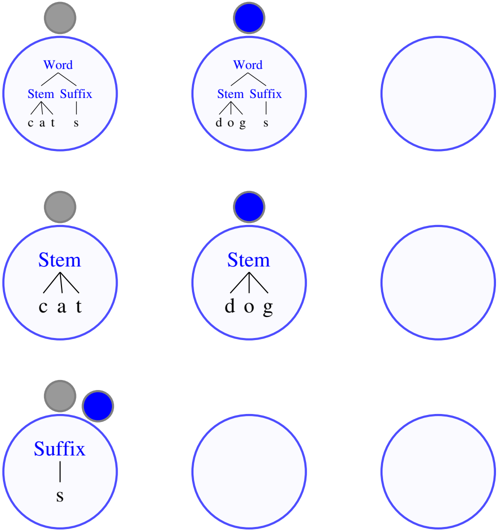
Generated words: cats

Adaptor grammar for stem-suffix morphology (2d)

Word → Stem Suffix

Stem → Phoneme^{+}

Suffix → $\text{Phoneme}^{\star}$



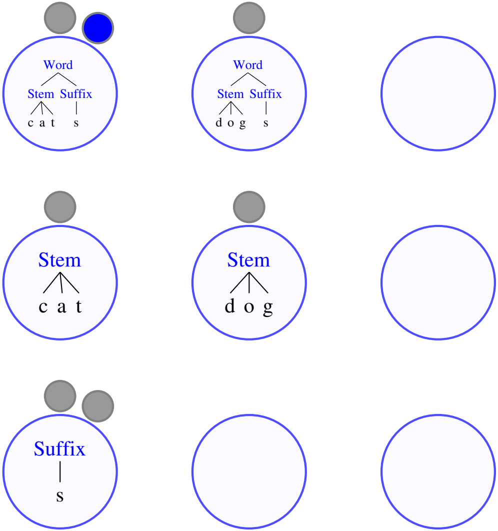
Generated words: cats, dogs

Adaptor grammar for stem-suffix morphology (3)

Word → Stem Suffix

Stem → Phoneme^{+}

Suffix → Phoneme*



Generated words: cats, dogs, cats

Adaptor grammars as generative processes

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

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 concentration parameter of the Dirichlet process associated with A.

$$
\begin{aligned}
G_A &\sim \mathrm{DP}(\alpha_A, H_A) \ \text{if } \alpha_A > 0 \text{, i.e., } A \text{ is } \textit{adapted} \\
&= H_A \qquad\qquad\qquad \text{otherwise}
\end{aligned}
$$

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

The grammar generates the distribution G_S.
One Dirichlet Process for each adapted non-terminal A.

Properties of adaptor grammars

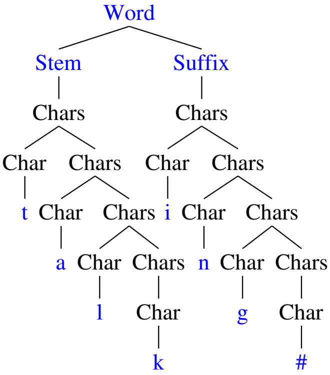
- Possible trees are generated by CFG rules
  but the probability of each adapted tree is learned separately
- Probability of adapted subtree τ is proportional to:
  - the number of times τ was seen before
    $\Rightarrow$ "rich get richer" dynamics (Zipf distributions)
  - plus α_A times prob. of generating it via PCFG expansion

$\Rightarrow$ Useful compound structures can be *more probable than their parts*

- PCFG rule probabilities estimated *from table labels*
  $\Rightarrow$ effectively *learns from types*, not tokens
  $\Rightarrow$ makes learner less sensitive to frequency variation in input

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



Unsupervised word segmentation

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

y △ u ▲ w △ a △ n △ t ▲ t △ u ▲ s △ i ▲ D △ 6 ▲ b △ U △ k

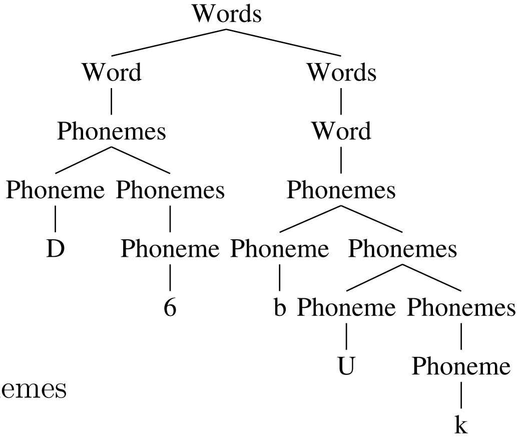
- Useful cues for word segmentation:
  - Phonotactics (Fleck)
  - Inter-word dependencies (Goldwater)

Word segmentation with PCFGs (1)

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

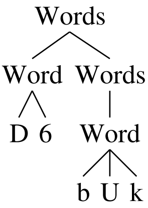
which abbreviates

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



Word segmentation with PCFGs (1)

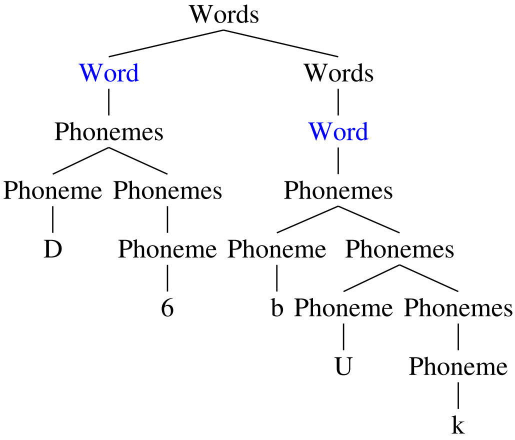
Sentence $\rightarrow$ Word$^+$
Word $\rightarrow$ all possible phoneme strings



- But now there are an infinite number of PCFG rules!
  - ▸ once we see our (finite) training data, only finitely many are useful
  - $\Rightarrow$ the set of parameters (rules) should be chosen based on training data

Unigram word segmentation adaptor grammar

Sentence → Word$^+$
Word → Phoneme$^+$



- *Adapted nonterminals* indicated by underlining

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

Adaptor grammar learnt from Brent corpus

- *Initial grammar*

1	Sentence → Word Sentence	1	Sentence → Word
1	Word → Phons		
1	Phons → Phon Phons	1	Phons → Phon
1	Phon → D	1	Phon → G
1	Phon → A	1	Phon → E

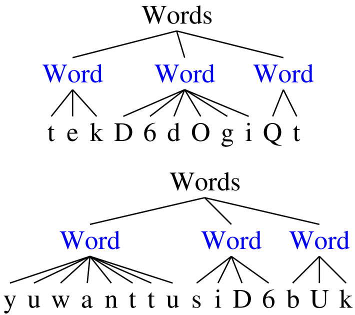
- *A grammar learnt from Brent corpus*

16625	Sentence → Word Sentence	9791	Sentence → Word
1	Word → Phons		
4962	Phons → Phon Phons	1575	Phons → Phon
134	Phon → D	41	Phon → G
180	Phon → A	152	Phon → E
460	Word → (Phons (Phon *y*) (Phons (Phon *u*)))		
446	Word → (Phons (Phon *w*) (Phons (Phon *A*) (Phons (Phon *t*))))		
374	Word → (Phons (Phon *D*) (Phons (Phon *6*)))		
372	Word → (Phons (Phon *&*) (Phons (Phon *n*) (Phons (Phon *d*))))		

Words (unigram model)

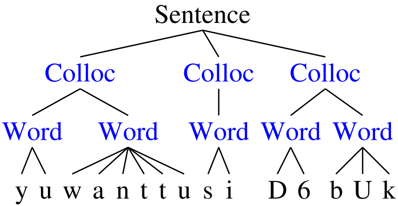
$$\text{Sentence} \rightarrow \text{Word}^+ \qquad \text{Word} \rightarrow \text{Phoneme}^+$$

- 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 $\rightarrow$ Colloc^{+}
Colloc $\rightarrow$ Word^{+}
Word $\rightarrow$ Phon^{+}

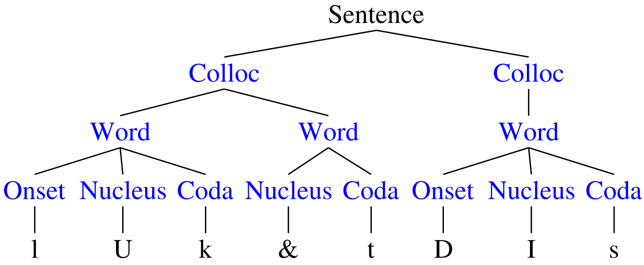


- A Colloc(ation) consists of one or more words
- Both Words and Collocs are adapted (learnt)
- Significantly improves word segmentation accuracy over unigram model (76% f-score; ≈ Goldwater's bigram model)

Collocations ⇒ Words ⇒ Syllables

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

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

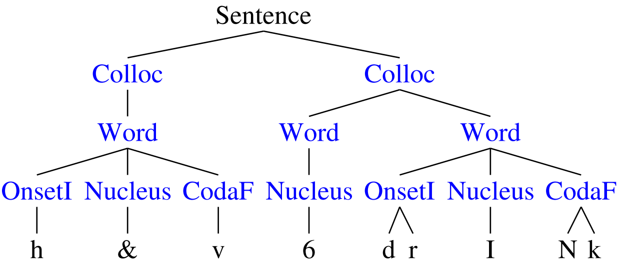


- With no supra-word generalizations, f-score = 68%
- With 2 Collocation levels, f-score = 82%

Distinguishing internal onsets/codas helps

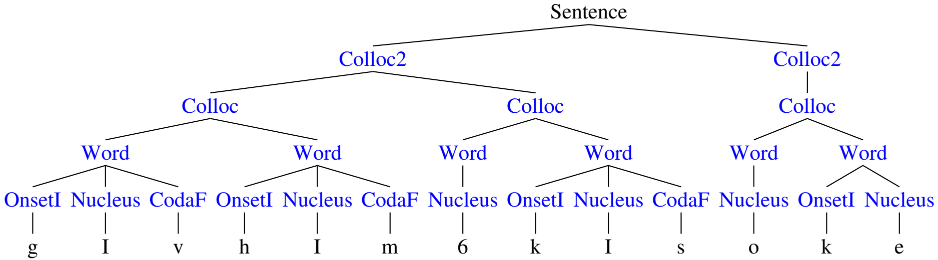
Sentence → Colloc$^+$
Word → SyllableIF
Word → SyllableI Syllable SyllableF
OnsetI → Consonant$^+$
Nucleus → Vowel$^+$

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



- Without distinguishing initial/final clusters, f-score = 82%
- Distinguishing initial/final clusters, f-score = 84%
- With 2 Collocation levels, f-score = 87%

Collocations^2 ⇒ Words ⇒ Syllables

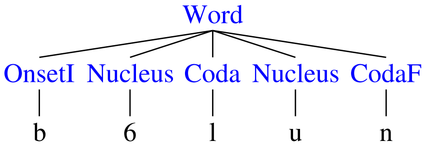


Syllabification learnt by adaptor grammars

- Grammar has no reason to prefer to parse word-internal intervocalic consonants as onsets

$$1 \quad \text{Syllable} \rightarrow \text{Onset Rhyme} \qquad 1 \quad \text{Syllable} \rightarrow \text{Rhyme}$$

- The learned grammars consistently analyse them as either Onsets or Codas $\Rightarrow$ learns wrong grammar half the time

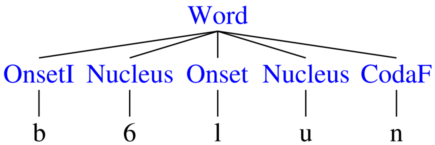


- Syllabification accuracy is relatively poor
  Syllabification given true word boundaries: f-score = 83%
  Syllabification learning word boundaries: f-score = 74%

Preferring Onsets improves syllabification

2 Syllable → Onset Rhyme   1 Syllable → Rhyme

- Changing the prior to prefer word-internal Syllables with Onsets dramatically improves segmentation accuracy
- "Rich get richer" property of Chinese Restaurant Processes
  ⇒ all ambiguous word-internal consonants analysed as Onsets



- Syllabification accuracy is much higher than without bias
  Syllabification given true word boundaries: f-score = 97%
  Syllabification learning word boundaries: f-score = 90%

Modelling sonority classes improves syllabification

$$\begin{array}{ll} \text{Onset} \rightarrow \text{Onset}_{\text{Stop}} & \text{Onset} \rightarrow \text{Onset}_{\text{Fricative}} \\ \text{Onset}_{\text{Stop}} \rightarrow \text{Stop} & \text{Onset}_{\text{Stop}} \rightarrow \text{Stop}\ \text{Onset}_{\text{Fricative}} \\ \text{Stop} \rightarrow \text{p} & \text{Stop} \rightarrow \text{t} \end{array}$$

- Five consonant sonority classes
- $\text{Onset}_{\text{Stop}}$ generates a consonant cluster with a Stop at left edge
- Prior prefers transitions compatible with sonority hierarchy (e.g., $\text{Onset}_{\text{Stop}} \rightarrow \text{Stop}\ \text{Onset}_{\text{Fricative}}$) to transitions that aren't (e.g., $\text{Onset}_{\text{Fricative}} \rightarrow \text{Fricative}\ \text{Onset}_{\text{Stop}}$)
- Same transitional probabilities used for initial and non-initial Onsets (maybe not a good idea for English?)
- Word-internal Onset bias still necessary
- Syllabification given true boundaries: f-score = 97.5%
  Syllabification learning word boundaries: f-score = 91%

Summary: Adaptor grammars for word segmentation

- Easy to define adaptor grammars that are sensitive to:

Generalization	Accuracy
words as units (unigram)	56%
+ associations between words (collocations)	76%
+ syllable structure	87%

- word segmentation *improves when you learn other things as well*
  - *explain away* potentially misleading generalizations

Outline

Finding topical collocations

- Modify LDA PCFG so each topic generates *sequences of words*
  - c.f. unigram word segmentation model

$$
\begin{array}{ll}
\text{Sentence} \to \text{Doc}_d & \text{for each } d \in \mathcal{D} \\
\text{Doc}_d \to d & \text{for each } d \in \mathcal{D} \\
\text{Doc}_d \to \text{Doc}_d \ \text{Topic}_t & \text{for each } d \in \mathcal{D} \text{ and } t \in \mathcal{T} \\
\underline{\text{Topic}_t} \to w^+ & \text{for each } t \in \mathcal{T} \text{ and } w \in \mathcal{W}
\end{array}
$$

where:

- document identifiers $\mathcal{D}$,
- *topics* $\mathcal{T} = \{1, \ldots, m\}$, and
- *vocabulary* $\mathcal{W}$

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 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

Learning the structure of names

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

Adaptor grammar for names

$\mathrm{NP} \to (\mathrm{A0})\ (\mathrm{A1})\ \ldots\ (\mathrm{A6})$	$\mathrm{NP} \to (\mathrm{B0})\ (\mathrm{B1})\ \ldots\ (\mathrm{B6})$
$\underline{\mathrm{A0}} \to \mathrm{Word}^+$	$\underline{\mathrm{B0}} \to \mathrm{Word}^+$
$\ldots$	$\ldots$
$\underline{\mathrm{A6}} \to \mathrm{Word}^+$	$\underline{\mathrm{B6}} \to \mathrm{Word}^+$
$\mathrm{NP} \to \mathrm{Unordered}^+$	$\underline{\mathrm{Unordered}} \to \mathrm{Word}^+$

- *Sample parses:*

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

Learning words and their referents

$\text{PIG|DOG} \quad I_{\triangle} z_{\blacktriangle} D_{\triangle} \&_{\triangle} t_{\blacktriangle} D_{\triangle} 6_{\blacktriangle} \underbrace{p_{\triangle} I_{\triangle} g}_{\text{PIG}}$

- Input: unsegmented phoneme sequence and *objects in nonlinguistic context*
- Goal: segment into words and *learn word-object relationship*

AG for (unigram) segmentation and reference

- Given *possible referents* $\mathcal{R}$, the grammar contains rules:

$$
\begin{array}{ll}
\text{Sentence} \rightarrow \text{Referent}_r & \text{for each } r \in \mathcal{R} \\
\text{Referent}_r \rightarrow s & \text{for each } r \in \mathcal{R},\ r \in s \in 2^{\mathcal{R}} \\
\text{Referent}_r \rightarrow \text{Referent}_r\ \text{Word}_r & \text{for each } r \in \mathcal{R} \\
\text{Referent}_r \rightarrow \text{Referent}_r\ \text{Word}_\emptyset & \text{for each } r \in \mathcal{R} \\
\underline{\text{Word}}_r \rightarrow \text{Phonemes}_r & \text{for each } r \in \mathcal{R} \cup \{\emptyset\} \\
\text{Phonemes}_r \rightarrow \text{Phoneme}^+ & \text{for each } r \in \mathcal{R} \cup \{\emptyset\}
\end{array}
$$

- Sample parses:
  T_dog|pig (Word I z D & t) (Word_dog D 6 d O g i)
  T_dog|pig (Word D E r z) (Word_dog D 6 d O g i)
  T_dog|pig (Word D & t s 6) (Word_pig p I g)

Joint segmentation and reference results

- Simultaneously learning word segmentation and reference does not seem to improve word segmentation
  - non-linguistic context is very impoverished
  - relatively few words are referential
- Referential words are segmented better when referents are provided
- Referential words are segmented better when at utterance ends
  - consistent with Frank's artificial language learning experiments

Outline

Probabilistic Context-Free Grammars

LDA topic models as PCFGs

Adaptor grammars

Adaptor grammars and topic models

Conclusion

Conclusion

- Grammars provide an alternative way of formulating complex models
- General-purpose inference procedures
- LDA topic models are PCFGs with Dirichlet priors on rule probabilities
- Two non-parametric generalizations of PCFGs
  - split non-terminals (states)
  - extend the set of possible rules (adaptor grammars)
- Adaptor grammars and topic models
  - topical collocations
  - structure in named entities
  - learning referents of words