Where do the rules come from?

Mark Johnson

joint work with Tom Griffiths and Sharon Goldwater

April, 2008

Outline

Why probabilistic models?

- Computational linguistics studies the computational aspects of linguistic processes (comprehension, generation, parsing)
- Distributional evidence is very useful; people are exquisitely sensitive to it
- Algorithms are specifications of computational processes, but aren't always the best way of *understanding* them
- Probabilistic models abstract away from algorithms, and describe the dependencies between different types of information
  - mathematical theory e.g., about how to compose multiple probabilistic models
  - same model implemented by many different algorithms (often making different assumptions)

The big question

- *How do we come to know so much about the languages we speak?*
  1. We learn it somehow
  2. It's innate

  Obviously some combination of both is involved, but what exactly?
- Do recent advances in statistical learning have anything to add?
  - Currently most statistical learning is *parameter setting*
  - Learning *structure* or the *rules that generate it* is much harder
  - *Nonparametric Bayesian techniques offer new ways of understanding structure learning*
  - Eventually may be able to quantitatively measure information contained in different aspects of input
    and show it does (not) suffice for learning structures we find in human languages

Chomskyian linguists ought to be Bayesians

- Bayes rule combines *prior knowledge* with *likelihood*

$$\underbrace{\mathsf{P}(\text{Hypothesis}|\text{Data})}_{\text{Posterior}} \propto \underbrace{\mathsf{P}(\text{Data}|\text{Hypothesis})}_{\text{Likelihood}}\ \underbrace{\mathsf{P}(\text{Hypothesis})}_{\text{Prior}}$$

- *Bayesian priors can incorporate detailed linguistic information*
  - Heads are all at the left edge or at the right edge of phrases
  - Words consist of bimoraic feet

  but need not
  - Prefer grammars with fewer/shorter rules/lexical entries
- A prior can encode both inviolable constraints and "soft" markedness preferences
  - bias learner toward universal tendancies, while permitting (high-frequency) exceptions
- Choice of prior ("universal grammar") is a *linguistic* question
- Potentially can *measure contribution* of prior to language learning
  - how much information do putative universals contain?

Statistical learning as parametric optimization

- Statistical learning is usually successful to the extent it can be reduced to a parameter optimization problem
  - model has finite number of adjustable parameters
  - adjust parameters to maximize model's fit to training data
  - (can be done on a scale far larger than anyone imagined, but most effective on *supervised* training data)
- Learning possible structures (or the *rules that generate them*) can be reduced to parameter estimation as follows:
  1. generate a set of possible rules somehow
  2. use a parameter estimator to estimate each rule's utility
  3. prune the useless rules, and repeat if desired
- Nonparametric Bayes offers a principled way of integrating rule generation and parameter estimation

Adaptor grammars

- "Nonparametric" means "not characterized by a *fixed number* of parameters"
- Adaptor grammars can be viewed as an extension of PCFGs that permit *an unbounded number of potential rules*
  - Any finite set of trees (e.g., sample parses for a corpus) can only use a finite number of them
    $\Rightarrow$ MCMC sampling algorithms for learning
  - c.f., iPCFGs, which extend PCFGs by permitting an unbounded number of nonterminals
- Adaptor grammars can express linguistically interesting nonparametric models
  - we'll look at several models of *word segmentation*
  - and show that those that simultaneously learn syllable structure do better (*synergy* in acquisition)

Outline

Probabilistic context-free grammars

- *Context-Free Grammars* (CFGs) provide rules (building blocks) for constructing phrases and sentences
- In a *Probabilistic CFG* (PCFG), each rule has a probability
- 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
$\mathrm{S} \rightarrow \mathrm{NP}\ \mathrm{VP}$	1.0		
$\mathrm{NP} \rightarrow \mathrm{Hillary}$	0.75	$\mathrm{NP} \rightarrow \mathrm{Barack}$	0.25
$\mathrm{VP} \rightarrow \mathrm{barks}$	0.6	$\mathrm{VP} \rightarrow \mathrm{snores}$	0.4

$$\mathsf{P}\left([_{\mathrm{S}}\ [_{\mathrm{NP}}\ \mathrm{Hillary}]\ [_{\mathrm{VP}}\ \mathrm{barks}]] \right) = 0.45 \qquad \mathsf{P}\left([_{\mathrm{S}}\ [_{\mathrm{NP}}\ \mathrm{Barack}]\ [_{\mathrm{VP}}\ \mathrm{snores}]] \right) = 0.1$$

Learning probabilistic context-free grammars

- Well-understood methods for statistical (Bayesian) estimation of PCFG rule probabilities
- These methods generalize to:
  - learning from words alone (unsupervised learning)
  - learning parametric grammars (e.g., X' grammars)
  - are efficient enough to learn from large amounts of data
- These learning procedures do really well on toy examples
- Unfortunately they do very poorly on real linguistic input

Unsupervised induction of PCFGs produces poor structures

- Learning procedures function by maximizing training data likelihood
- Higher likelihood $\not\Rightarrow$ more accurate parses
  $\Rightarrow$ model is wrong
- What could be wrong?
  - Wrong grammar (Klein and Manning, Smith and Eisner)
  - Ignoring useful information in input (Yang)
  - Grammar *ignores semantics* (Zettlemoyer and Collins)

$\Rightarrow$ Develop models of syntax/semantics mapping, e.g., from sentences to (non-linguistic) contexts

$\Rightarrow$ *Study simpler learning problems that we know humans solve and try to understand what goes wrong*

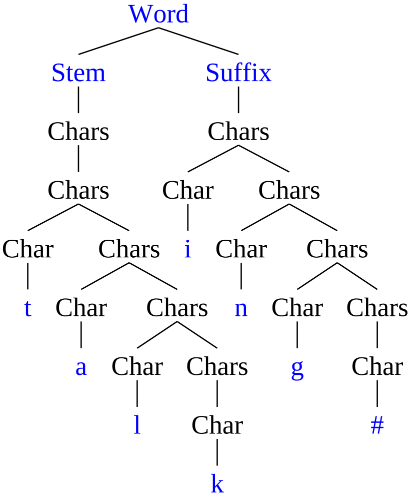
Outline

Learning agglutinative morphology

- Words consist of sequence of *morphemes*
  e.g., $\text{talk} + \text{ing}, \text{jump} + \text{s}$, etc.
- Given unanalyzed words as input training data,
  want to learn a grammar that:
  - generates words as a sequence of morphemes, and
  - correctly generates novel morphogical combinations not seen in training data
- Training data: sequences of characters, e.g., t a l k i n g
- Where we're going:
  - CFGs are good ways of generating potentially useful structures
  - but *PCFGs are not good models of a structure's probability*
  - Dependencies (generalizations) involve substructures, but we *don't know the relevant structures in advance*

A CFG for stem-suffix morphology

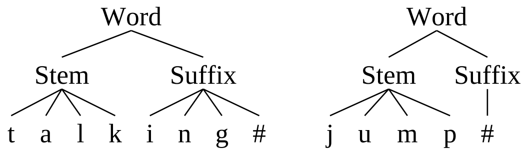
$$
\begin{array}{rcl@{\qquad}rcl}
\text{Word} & \rightarrow & \text{Stem Suffix} & \text{Chars} & \rightarrow & \text{Char} \\
\text{Stem} & \rightarrow & \text{Chars} & \text{Chars} & \rightarrow & \text{Char Chars} \\
\text{Suffix} & \rightarrow & \text{Chars} & \text{Char} & \rightarrow & \text{a}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



- Grammar generates acceptable structures
- But its *units of generalization (PCFG rules) are "too small" to learn morphemes*

A “CFG” with one rule per possible morpheme

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



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

Independence assumptions in PCFGs

- Context-free grammars are "context-free" because the possible expansions of each node do not depend on expansions of other nodes
- Probabilistic CFGs extend this by requiring each node expansion to be *statistically independent* (conditioned on the node's label)
- This is a very strong assumption, which is often false!
- Morphology grammar:

$$\mathrm{Word} \rightarrow \mathrm{Stem}\,\mathrm{Suffix}$$

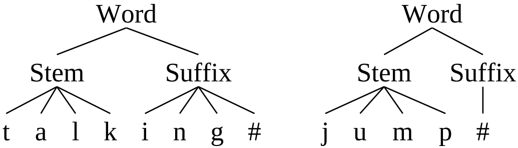
  Corresponding independence assumption:

$$\mathsf{P}(\mathrm{Word}) = \mathsf{P}(\mathrm{Stem})\mathsf{P}(\mathrm{Suffix})$$

  Causes PCFG model of morphology to fail

Learning English verbal morphology

Training data is a sequence of verbs, e.g.
$\mathcal{D} = (\#\text{talking}\#, \#\text{jump}\#, \ldots)$
Our goal is to infer trees such as:

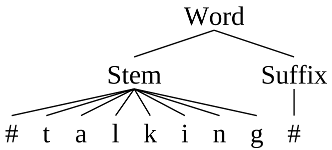


$$
\begin{array}{ll}
\text{Word} \rightarrow \text{Stem Suffix} & \\
\text{Stem} \rightarrow w & w \in \mathcal{T} \\
\text{Suffix} \rightarrow w & w \in \mathcal{F}
\end{array}
$$

where $\mathcal{T}$ is the set of all prefixes of words in $\mathcal{D}$ and $\mathcal{F}$ is the set of all suffixes of words in $\mathcal{D}$

Maximum likelihood estimate for θ is trivial

- Maximum likelihood selects θ that minimizes KL-divergence between model and data distributions
- *Saturated model* with $\theta_{\mathrm{Suffix} \rightarrow \#} = 1$ generates training data distribution $\mathcal{D}$ exactly
- Saturated model is maximum likelihood estimate
- Maximum likelihood estimate does not find any suffixes



Bayesian estimation

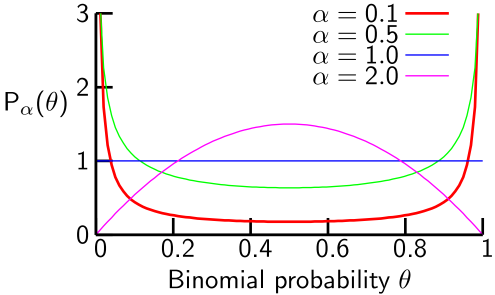
$$\underbrace{\mathsf{P}(\text{Hypothesis}|\text{Data})}_{\text{Posterior}} \propto \underbrace{\mathsf{P}(\text{Data}|\text{Hypothesis})}_{\text{Likelihood}} \underbrace{\mathsf{P}(\text{Hypothesis})}_{\text{Prior}}$$

- Priors can be sensitive to linguistic structure (e.g., a word should contain a vowel)
- Priors can encode linguistic universals and markedness preferences (e.g., complex clusters appear at word onsets)
- Priors can prefer *sparse solutions*
- The choice of the prior is as much a linguistic issue as the design of the grammar!

Dirichlet priors and sparse solutions

- The probabilities $\theta_{A \rightarrow \beta}$ of choosing productions $A \rightarrow \beta$ to expand nonterminal A define multinomial distributions
- Dirichlet distributions are the *conjugate priors* to multinomials

$$\mathrm{P}(\theta_{A \rightarrow \beta_1}, \ldots, \theta_{A \rightarrow \beta_n}) \quad \propto \quad \prod_{i=1}^{n} {\theta_{A \rightarrow \beta_i}}^{\alpha - 1} \qquad \alpha > 0$$



- There are MCMC algorithms for sampling from the posterior distribution of trees given strings $\mathcal{D}$

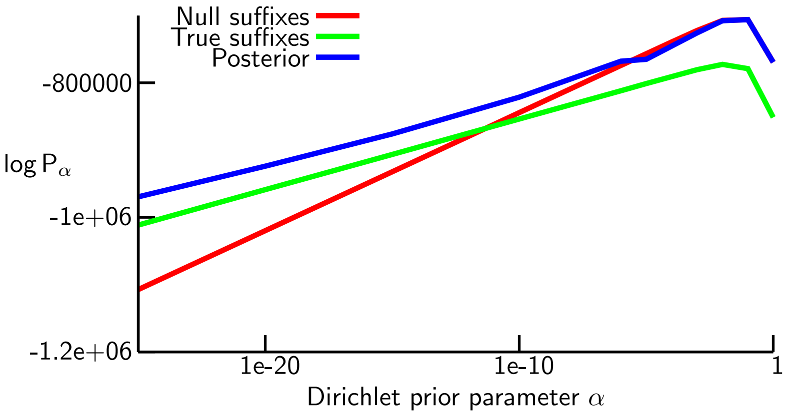
Morphological segmentation experiment

- Trained on orthographic verbs from U Penn. Wall Street Journal treebank
- Dirichlet prior prefers sparse solutions (sparser solutions as $\alpha \to 0$)
- MCMC Sampler used to sample from posterior distribution of parses
  - reanalyses each word based on a grammar estimated from the 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	

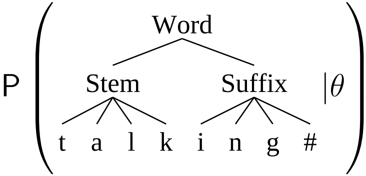
Log posterior of models on token data



- Correct solution is nowhere near as likely as posterior

⇒ model is wrong!

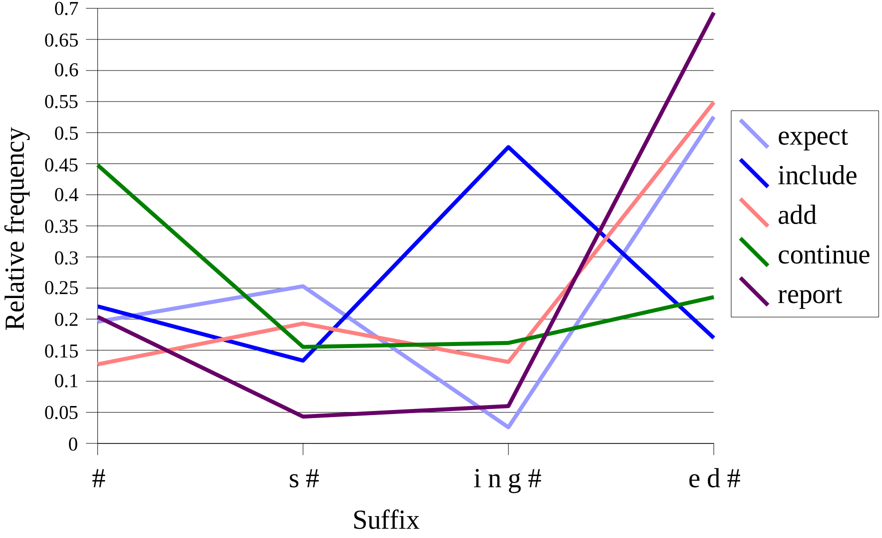
Independence assumption in PCFG model



$$= \theta_{\text{Word} \to \text{Stem Suffix}} \, \theta_{\text{Stem} \to \text{t a l k}} \, \theta_{\text{Suffix} \to \text{i n g \#}}$$

- Model assumes relative frequency of each suffix *to be the same for all stems*
- This turns out to be incorrect

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

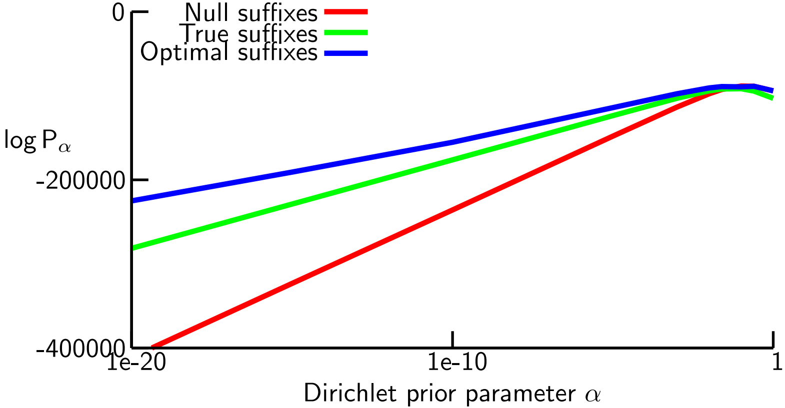
$$
\begin{aligned}
\text{Data} &= \text{“the cat chased the other cat”} \\
\text{Tokens} &= \text{“the”, “cat”, “chased”, “the”, “other”, “cat”} \\
\text{Types} &= \text{“the”, “cat”, “chased”, “other”}
\end{aligned}
$$

- Estimating θ from *word types* rather than word tokens eliminates (most) frequency variation
  - 4 common verb suffixes, so when estimating from verb types $\theta_{\text{Suffix} \rightarrow \text{ing\#}} \approx 0.25$
- Several psycholinguists believe that humans learn morphology from word types

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 $\alpha = 10^{-3}$

Outline

PCFGs and adaptor grammars

- PCFGs are good for describing possible structures, but *rules are too small a unit of generalization* for learning
- PCFGs assume the set of rules is fixed in advanced, but often we want to *learn the rules from data*, i.e., not assume a finite set of rules in advance
- PCFGs assume that each nonterminal expands independently, but often there are *probabilistic dependencies across expansions* that we need to learn

Adaptor grammars: informal description

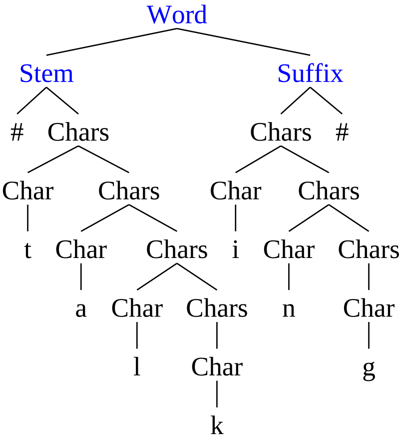
- An adaptor grammar has a set of PCFG rules
- These determine the possible structures as in a CFG
- A subset of the nonterminals are *adapted*
- *Unadapted nonterminals* expand by picking a rule and recursively expanding its children, as in a PCFG
- *Adapted nonterminals* can expand in two ways:
  - by picking a rule and recursively expanding its children, or
  - by generating a previously generated tree (with probability proportional to the number of times previously generated)
- Each adapted subtree behaves like a new rule added to the grammar
- The PCFG rules of the adapted nonterminals determine the *prior* over these trees

Adaptor grammars as generative processes

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

Outline

Adaptor grammar morphology example



$$
\begin{array}{rcl}
\text{Word} & \rightarrow & \text{Stem Suffix} \\
\text{Stem} & \rightarrow & \#\ \text{Chars} \\
\text{Suffix} & \rightarrow & \# \\
\text{Suffix} & \rightarrow & \text{Chars}\ \# \\
\text{Chars} & \rightarrow & \text{Char} \\
\text{Chars} & \rightarrow & \text{Char Chars} \\
\text{Char} & \rightarrow & a \mid \ldots \mid z
\end{array}
$$

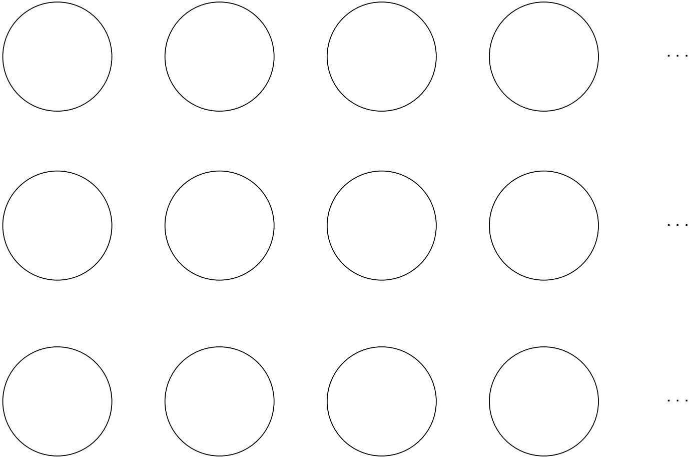
- Stem and Suffix rules generate all possible stems and suffixes
- Adapt Word, Stem and Suffix nonterminals
- Sampler uses *"Chinese restaurant" processes*

Morphology adaptor grammar (0)

Word restaurant
Word → Stem Suffix

Stem restaurant
Stem → #
Stem → # Chars

Suffix restaurant
Suffix → #
Suffix → Chars #



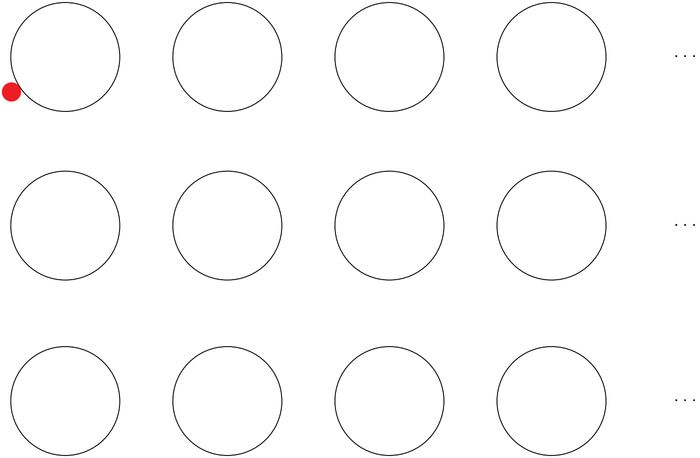
Chars factory
Chars → Char
Chars → Char Chars
Char → a . . . z

Morphology adaptor grammar (1a)

Word restaurant
Word → Stem Suffix

Stem restaurant
Stem → #
Stem → # Chars

Suffix restaurant
Suffix → #
Suffix → Chars #



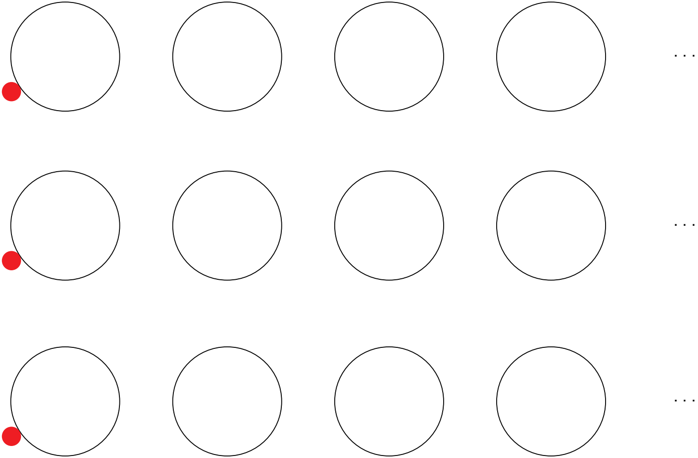
Chars factory
Chars → Char
Chars → Char Chars
Char → a . . . z

Morphology adaptor grammar (1b)

Word restaurant
Word $\rightarrow$ Stem Suffix

Stem restaurant
Stem $\rightarrow$ #
Stem $\rightarrow$ # Chars

Suffix restaurant
Suffix $\rightarrow$ #
Suffix $\rightarrow$ Chars #



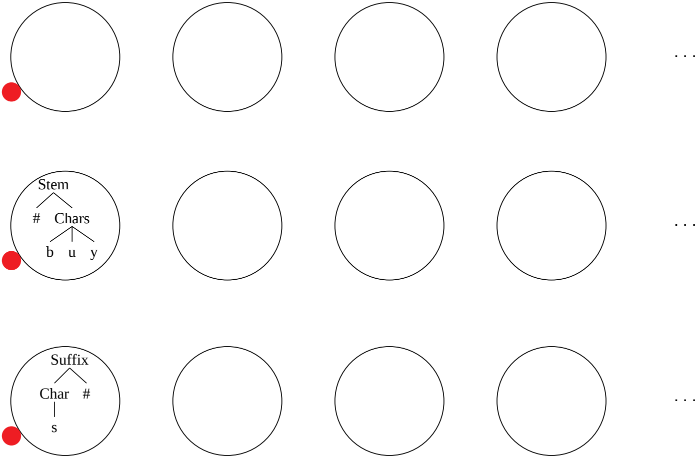
Chars factory
Chars $\rightarrow$ Char
Chars $\rightarrow$ Char Chars
Char $\rightarrow$ a ... z

Morphology adaptor grammar (1c)

Word restaurant
Word → Stem Suffix

Stem restaurant
Stem → #
Stem → # Chars

Suffix restaurant
Suffix → #
Suffix → Chars #



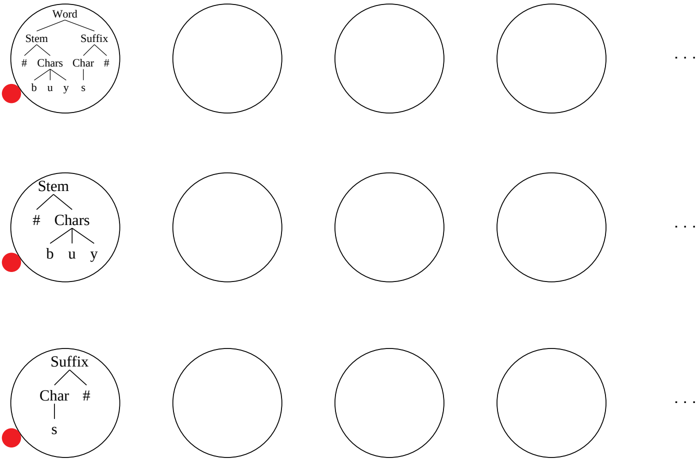
Chars factory
Chars → Char
Chars → Char Chars
Char → a . . . z

Morphology adaptor grammar (1d)

Word restaurant
Word → Stem Suffix

Stem restaurant
Stem → #
Stem → # Chars

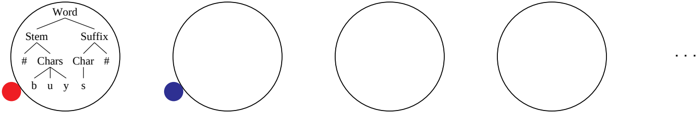
Suffix restaurant
Suffix → #
Suffix → Chars #



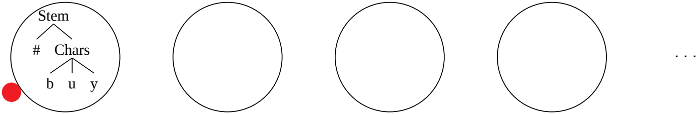
Chars factory
Chars → Char
Chars → Char Chars
Char → a . . . z

Morphology adaptor grammar (2a)

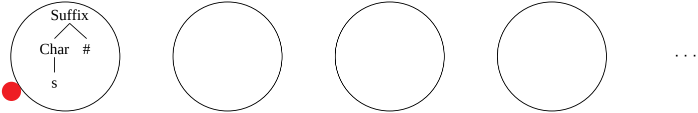
Word restaurant
Word → Stem Suffix



Stem restaurant
Stem → #
Stem → # Chars



Suffix restaurant
Suffix → #
Suffix → Chars #



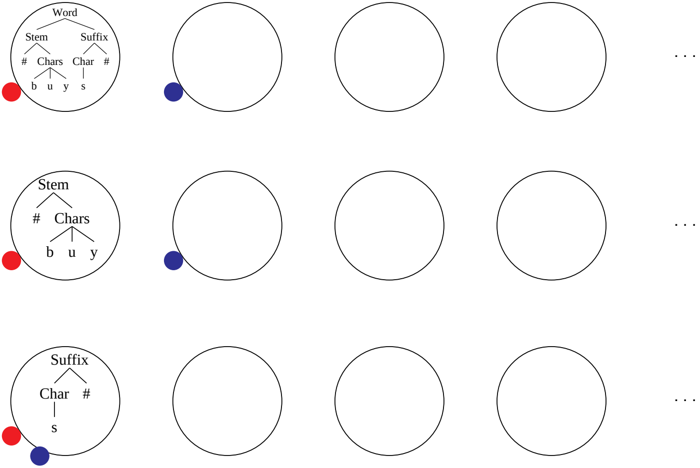
Chars factory
Chars → Char
Chars → Char Chars
Char → a . . . z

Morphology adaptor grammar (2b)

Word restaurant
Word → Stem Suffix

Stem restaurant
Stem → #
Stem → # Chars

Suffix restaurant
Suffix → #
Suffix → Chars #



Chars factory
Chars → Char
Chars → Char Chars
Char → a . . . z

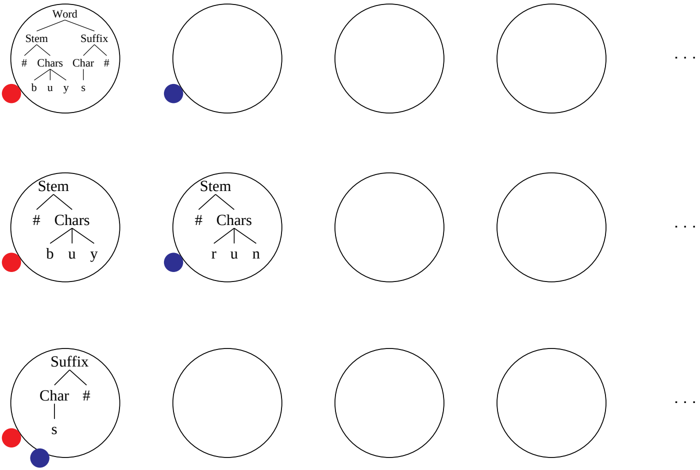
Morphology adaptor grammar (2c)

Word restaurant
Word → Stem Suffix

Stem restaurant
Stem → #
Stem → # Chars

Suffix restaurant
Suffix → #
Suffix → Chars #

Chars factory
Chars → Char
Chars → Char Chars
Char → a . . . z

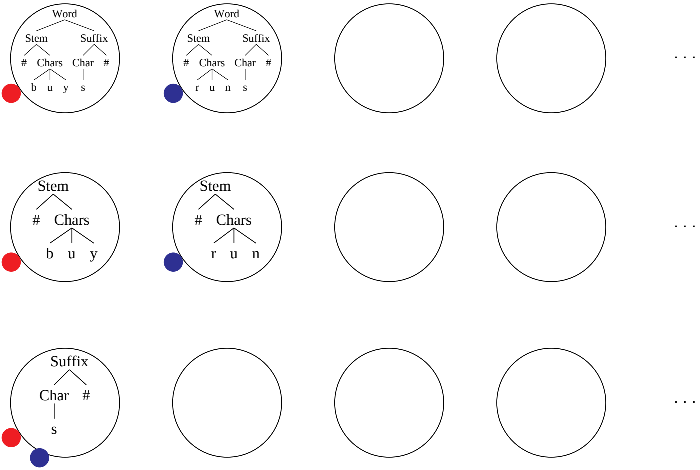


Morphology adaptor grammar (2d)

Word restaurant
Word → Stem Suffix

Stem restaurant
Stem → #
Stem → # Chars

Suffix restaurant
Suffix → #
Suffix → Chars #



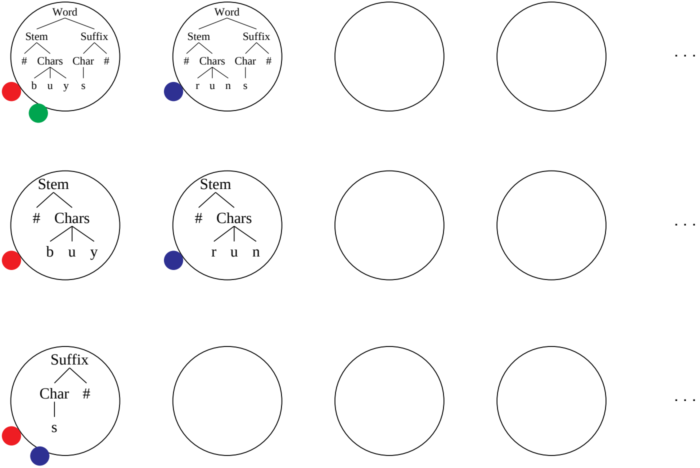
Chars factory
Chars → Char
Chars → Char Chars
Char → a . . . z

Morphology adaptor grammar (3)

Word restaurant
Word → Stem Suffix

Stem restaurant
Stem → #
Stem → # Chars

Suffix restaurant
Suffix → #
Suffix → Chars #



Chars factory
Chars → Char
Chars → Char Chars
Char → a . . . z

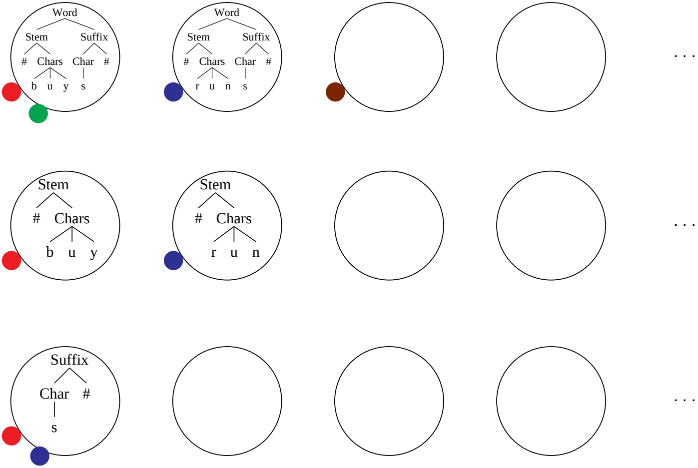
Morphology adaptor grammar (4a)

Word restaurant
Word → Stem Suffix

Stem restaurant
Stem → #
Stem → # Chars

Suffix restaurant
Suffix → #
Suffix → Chars #

Chars factory
Chars → Char
Chars → Char Chars
Char → a . . . z



Morphology adaptor grammar (4b)

Word restaurant

Word → Stem Suffix

Stem restaurant

Stem → #

Stem → # Chars

Suffix restaurant

Suffix → #

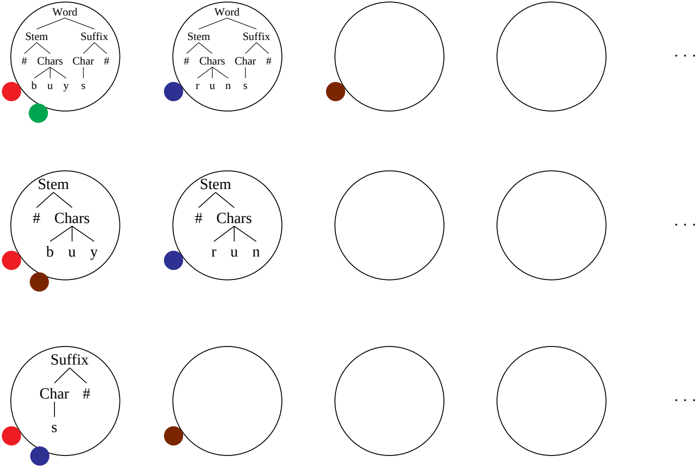
Suffix → Chars #

Chars factory

Chars → Char

Chars → Char Chars

Char → a . . . z

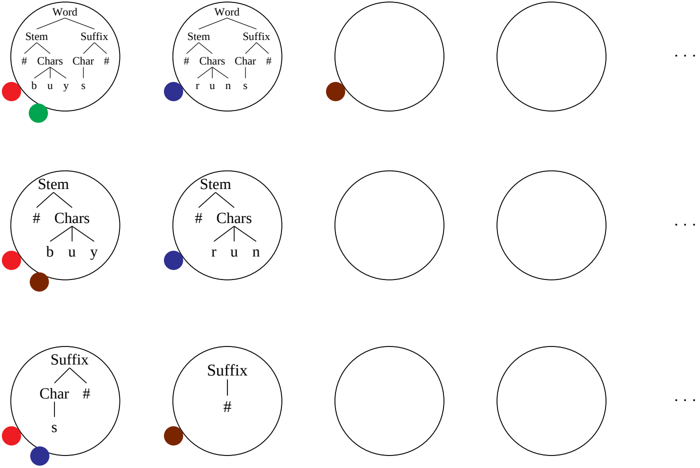


Morphology adaptor grammar (4c)

Word restaurant
Word → Stem Suffix

Stem restaurant
Stem → #
Stem → # Chars

Suffix restaurant
Suffix → #
Suffix → Chars #



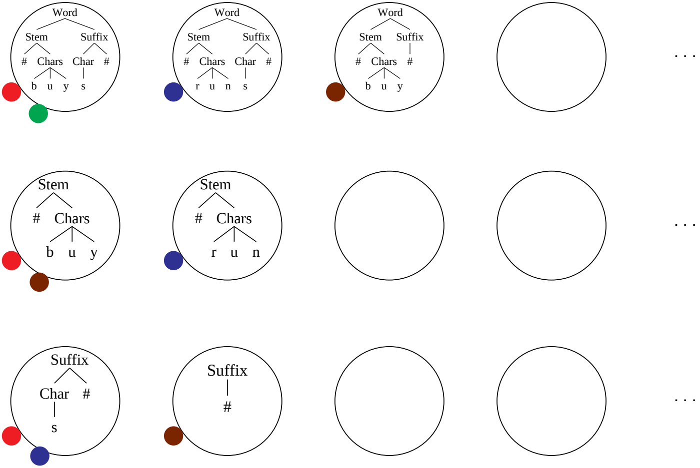
Chars factory
Chars → Char
Chars → Char Chars
Char → a . . . z

Morphology adaptor grammar (4d)

Word restaurant
Word → Stem Suffix

Stem restaurant
Stem → #
Stem → # Chars

Suffix restaurant
Suffix → #
Suffix → Chars #



Chars factory
Chars → Char
Chars → Char Chars
Char → a . . . z

Properties of adaptor grammars

- Possible trees generated by CFG rules
  but the probability of each adapted tree is estimated separately
- Probability of a tree is:

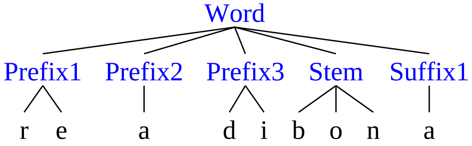
  proportional to the number of times seen before
  $\Rightarrow$ "rich get richer" dynamics (Zipf distributions)
  plus a constant times the probability of generating it via PCFG expansion

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

- PCFG rule probabilities estimated *from table labels*
  $\Rightarrow$ learns from types, not tokens
  $\Rightarrow$ dampens frequency variation

Learning Sesotho verbal morphology using an adaptor grammar

re a di bon a
SM T OM V M
"We see them"



$$\text{Word} \rightarrow (\text{Prefix1})\,(\text{Prefix2})\,(\text{Prefix3})\,\text{Stem}\,(\text{Suffix})$$

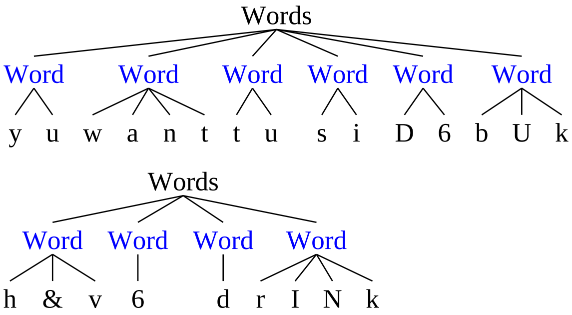
- Sesotho is a Bantu language with complex morphology, not much phonology
- Demuth's Sesotho corpus contains morphological parses for 2,283 distinct verb types
- An adaptor grammar finds morphological analyses for these verbs
  - 62% f-score (morpheme accuracy)
  - 41% words completely correct

Outline

Unigram model of word segmentation

- Unigram model: each word is generated independently
- Input is *unsegmented broad phonemic transcription* (Brent)
  Example: y u w a n t t u s i D 6 b u k
- Adaptor for Word non-terminal caches previously seen words

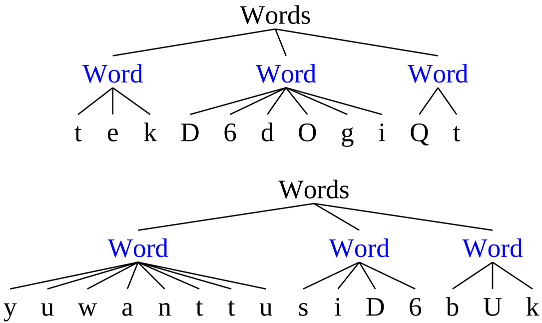
$$\text{Words} \rightarrow \text{Word}^{+}$$
$$\text{Word} \rightarrow \text{Phoneme}^{+}$$



- Unigram word segmentation on Brent corpus: 55% token f-score

Unigram model often finds collocations

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



Unigram word segmentation grammar learnt

- Based on the base grammar rules

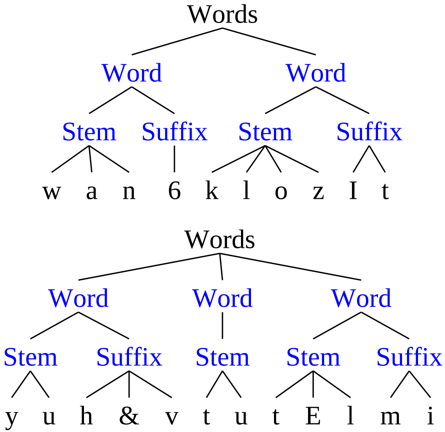
$$\begin{aligned} \text{Words} &\rightarrow \text{Word}^+ \\ \text{Word} &\rightarrow \text{Phoneme}^+ \end{aligned}$$

the adapted grammar contains 1,712 rules such as:

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

Combining morphology and word segmentation

$$
\begin{aligned}
\text{Words} &\to \text{Word}^+ \\
\text{Word} &\to \text{Stem}\ \text{Suffix} \\
\text{Word} &\to \text{Stem} \\
\text{Stem} &\to \text{Phoneme}^+ \\
\text{Suffix} &\to \text{Phoneme}^+
\end{aligned}
$$



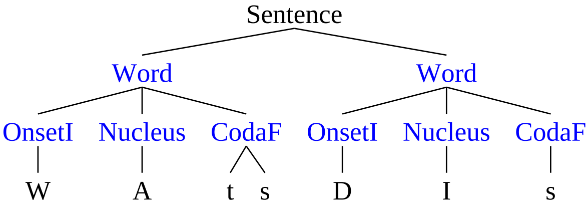
- Adaptors for Word, Stem and Suffix nonterminals
- Doesn't do a good job of learning morphology (which doesn't appear that much in corpus) or word segmentation (35% f-score), but does find interesting collocations!

Syllable structure and word segmentation

Sentence $\rightarrow$ Word$^+$	Word $\rightarrow$ SyllableIF
Word $\rightarrow$ SyllableI SyllableF	Word $\rightarrow$ SyllableI Syllable SyllableF
Syllable $\rightarrow$ (Onset) Rhyme	SyllableI $\rightarrow$ (OnsetI) Rhyme
SyllableF $\rightarrow$ (Onset) RhymeF	SyllableIF $\rightarrow$ (OnsetI) RhymeF
Rhyme $\rightarrow$ Nucleus (Coda)	RhymeF $\rightarrow$ Nucleus (CodaF)
Onset $\rightarrow$ Consonant$^+$	OnsetI $\rightarrow$ Consonant$^+$
Coda $\rightarrow$ Consonant$^+$	CodaF $\rightarrow$ Consonant$^+$
Nucleus $\rightarrow$ Vowel$^+$	

- Grammar distinguishes *initial* (I) and *final* (F) clusters (even though training data doesn't indicate which is which)

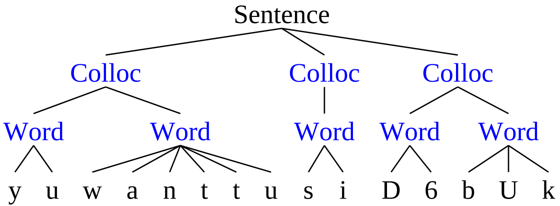
Analysis using syllable structure adaptor grammar



- Word, Onset, Nucleus and Coda are adapted (learnt)
  Syllable is not
- Performs word segmentation with 47% f-score (worse than unigram model)
- Strong tendancy to misanalyse function/content word collocations as single words

Modeling collocations improves segmentation

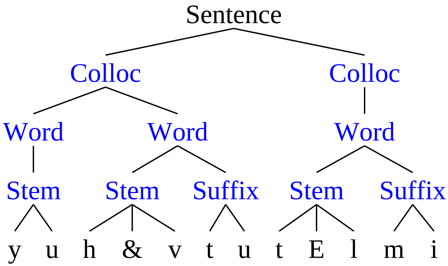
$$\begin{aligned} \text{Sentence} &\rightarrow \text{Colloc}^{+} \\ \text{Colloc} &\rightarrow \text{Word}^{+} \\ \text{Word} &\rightarrow \text{Phoneme}^{\star} \end{aligned}$$



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

Morphology + Collocations + Word segmentation

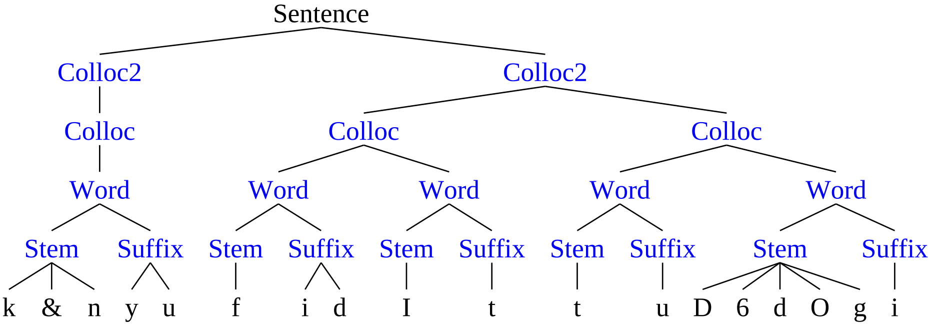
$$
\begin{array}{ll}
\text{Sentence} \rightarrow \text{Colloc}^{+} & \text{Colloc} \rightarrow \text{Word}^{+} \\
\text{Word} \rightarrow \text{Stem (Suffix)} & \text{Stem} \rightarrow \text{Phoneme}^{+} \\
\text{Suffix} \rightarrow \text{Phoneme}^{+} &
\end{array}
$$



- Word segmentation f-score = 59% (worse than collocations alone)

Morphology + 2 Collocation levels

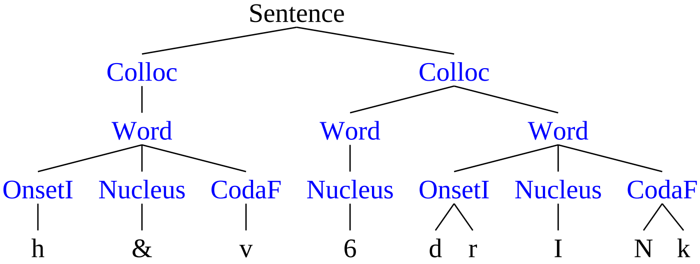
$$\text{Sentence} \rightarrow \text{Colloc2}^{+} \quad \text{Colloc2} \rightarrow \text{Colloc}^{+}$$
$$\text{Colloc} \rightarrow \text{Word}^{+} \quad \text{Word} \rightarrow \text{Stem (Suffix)}$$
$$\text{Stem} \rightarrow \text{Phoneme}^{+} \quad \text{Suffix} \rightarrow \text{Phoneme}^{+}$$



- But with two Collocation levels f-score = 79%

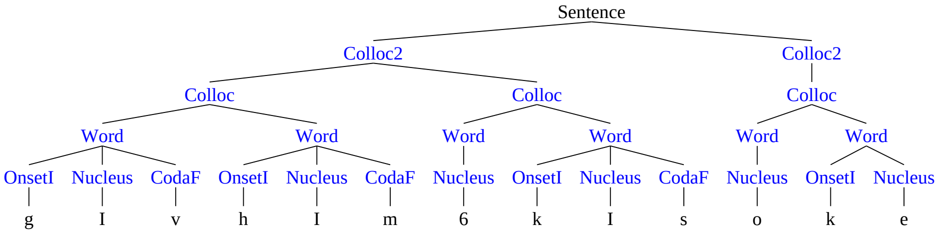
Syllables + Collocations + Word segmentation

Sentence → Colloc$^+$
Word → SyllableIF
Word → SyllableI Syllable SyllableF
Colloc → Word$^+$
Word → SyllableI SyllableF
Syllable → (as before)



- Word segmentation f-score = 68%
- With 2 Collocation levels f-score = 84% (better than Colloc+morphology)
- Without distinguishing initial/final clusters f-score = 82%

Syllables + 2-level Collocations + Word segmentation



Word segmentation results summary

	Collocation levels			
	0	1	2	3
word	0.55	0.73	0.75	0.72
morphology	0.35	0.55	0.79	0.73
syllable	0.32	0.69	0.82	0.79
syllableIF	0.46	0.68	**0.84**	**0.84**

- We can learning collocations and syllable structure together with word segmentation, even though we don't know where the word boundaries are
- Learning these together improves word segmentation accuracy
  - are there other examples of *synergistic interaction* in language learning?

Outline

Summary and future work

- Adaptor grammars "adapt" their distribution to the strings they have generated
- They learn the subtrees of the adapted nonterminals they generate
- This makes adaptor grammars *non-parametric*; the number of subtrees they track depends on the data
- A variety of different linguistic phenomena can be described with adaptor grammars
- Because they are grammars, they are easy to design and compose
- But they still have a "context-freeness" that makes it impossible to express e.g., Goldwater's bigram word segmentation model. Can we add context-sensitivity in a manageable way?
- The MCMC sampling algorithm used does not seem to scale well to large data or complicated grammars. Are there better estimators?

Outline

From Chinese restaurants to Dirichlet processes

- Labeled Chinese restaurant processes take a base distribution P_G and return a stream of samples from a different distribution with the same support
- The Chinese restaurant process is a sequential process, generating the next item conditioned on the previous ones
- We can get a different distribution each time we run a CRP (placing customers on tables and labeling tables are random)
- Abstracting away from sequential generation, a CRP maps P_G to a *distribution over distributions* $\mathrm{DP}(\alpha, \mathsf{P}_G)$
- $\mathrm{DP}(\alpha, \mathsf{P}_G)$ is called a *Dirichlet process* with *concentration parameter* α and *base distribution* P_G
- Distributions in $\mathrm{DP}(\alpha, \mathsf{P}_G)$ are *discrete* (w.p. 1) even if the base distribution P_G is continuous

PCFGs as recursive mixture processes

For simplicity assume all runs in CNF, i.e., all rules are of the form $A \to B\ C$ or $A \to w$, where $A, B, C \in N$ and $w \in T$.
Each nonterminal $A \in N$ generates a distribution G_A over trees rooted in A.

$$G_A = \sum_{A \to B\ C \in R_A} \theta_{A \to B\ C} \mathrm{T\small{REE}}_A(G_B, G_C) + \sum_{A \to w \in R_A} \theta_{A \to w} \mathrm{T\small{REE}}_A(w)$$

where $\mathrm{T\small{REE}}_A(w)$ puts all of its mass on the tree with child w and $\mathrm{T\small{REE}}_A(P, Q)$ is the distribution over trees rooted in A with children distributed according to P and Q respectively.

$$\mathrm{T\small{REE}}_A(P, Q) \left(\begin{array}{c} A \\ \diagup \diagdown \\ t_1 \quad t_2 \end{array} \right) = P(t_1)\, Q(t_2)$$

The tree language generated by the PCFG is G_S.

Adaptor grammars

An adaptor grammar (G, θ, α) is a PCFG (G, θ) together with a parameter vector α where for each $A \in N$, α_A is the parameter of the Dirichlet process associated with A.

$$
\begin{aligned}
G_A &\sim \mathrm{DP}(\alpha_A, H_A) \text{ if } \alpha_A > 0 \\
&= H_A \text{ if } \alpha_A = 0 \\
H_A &= \sum_{A \to B\, C \in R_A} \theta_{A \to B\, C} \, \mathrm{T{\small REE}}_A(G_B, G_C) + \sum_{A \to w \in R_A} \theta_{A \to w} \, \mathrm{T{\small REE}}_A(w)
\end{aligned}
$$

The grammar generates the distribution over trees G_S.
There is one Dirichlet Process for each non-terminal A where $\alpha_A > 0$.
Its base distribution H_A is a mixture of the language generated by the Dirichlet processes associated with other non-terminals.

Estimating adaptor grammars

- Need to estimate:
  - table labels and customer count for each table
  - (optional) probabilities of productions labeling tables
- Component-wise Metropolis-Hastings sampler
  - ith component is the parse tree for input string i
  - sample parse for input i using grammar estimated from parses for other inputs
- Sampling directly from conditional distribution of parses seems intractable
  - construct PCFG approximation on the fly
  - each table label corresponds to a production in PCFG approximation
  - Use accept/reject to convert stream of samples from PCFG approx to samples from adaptor grammar