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

Mark Johnson

joint work with Tom Griffiths and Sharon Goldwater

November, 2007

Outline

Why *computational* linguistics?

Grammars (finite descriptions of languages)

Learning morphology with adaptor grammars

Word segmentation using adaptor grammars

Conclusions

Technical details

Why is there a field of *computational* linguistics?

- Language is a symbolic system (involves manipulation of *meaning-bearing entities*)
  $\Rightarrow$ linguistic processes are *computational* processes
- Linguistic processes have a computational dimension (alongside formal, psychological, neurological, developmental, etc.)
- Empirical properties of linguistic processes motivating this work:
  - speakers/hearers can produce and comprehend sentences (parsing, generation)
  - children, starting from the same initial state, can learn any human language (acquisition)
  - these processes are faced with an astronomically large number of different possible sentences

Linguistic processing as inference

Comprehension	**Acquisition**
sentence	sentences
"grammar"	"universal grammar"
meaning (parse)	grammar

- Research agenda: *What information* is used in these processes?

Bayesian learning

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

- A Bayesian model *integrates information from multiple sources*
  - *Likelihood* reflects how well grammar fits input data
  - *Prior* encodes a priori preferences for particular grammars
- The *prior is as much a linguistic issue as the grammar*
  - Priors can be sensitive to linguistic structure (e.g., words should contain vowels)
  - Priors can encode *linguistic universals* and *markedness preferences*
- Priors can prefer *smaller grammars* (Occam's razor, MDL)
- A Bayesian model is *not* an implementation or algorithm

Outline

Why *computational* linguistics?

Grammars (finite descriptions of languages)

Learning morphology with adaptor grammars

Word segmentation using adaptor grammars

Conclusions

Technical details

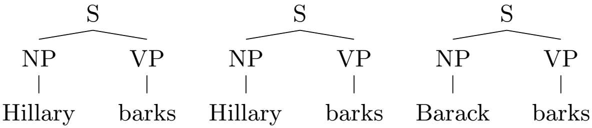
Probabilistic context-free grammars

- *Context-Free Grammars* (CFGs) provide rules (building blocks) for constructing phrases and sentences
- In a *Probabilistic CFG* (PCFG), each rule has a probability (cost)
- Probability of a tree is the *product of the probabilities of the rules* used to construct it

Rule r	$\mathrm{P}(\theta_r)$	Rule r	$\mathrm{P}(\theta_r)$
$\mathrm{S} \rightarrow \mathrm{NP\ VP}$	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

$$\mathrm{P}\left(\begin{array}{c} \mathrm{S} \\ \begin{array}{cc} \mathrm{NP} & \mathrm{VP} \\ | & | \\ \mathrm{Hillary} & \mathrm{barks} \end{array} \end{array}\right) = 0.45 \qquad \mathrm{P}\left(\begin{array}{c} \mathrm{S} \\ \begin{array}{cc} \mathrm{NP} & \mathrm{VP} \\ | & | \\ \mathrm{Barack} & \mathrm{snores} \end{array} \end{array}\right) = 0.1$$

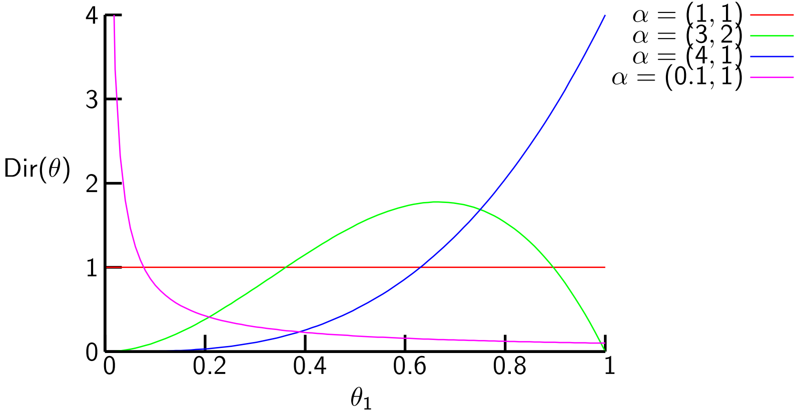
Estimating PCFG rule probabilities from trees



- Prior over rule probabilities: product of Dirichlet distributions with parameters α_r for each rule r
- Conjugacy $\Rightarrow$ posterior is also product of Dirichlets, with parameters $\alpha_r + n_r$, where n_r is number of times r occurs in trees

Rule r	α_r	n_r	$\alpha_r + n_r$	Sample θ_r	Sample θ_r
$\mathrm{S} \rightarrow \mathrm{NP\ VP}$	1	3	4	1	1
$\mathrm{NP} \rightarrow \mathrm{Hillary}$	1	2	3	0.61	0.51
$\mathrm{NP} \rightarrow \mathrm{Barack}$	1	1	2	0.39	0.49
$\mathrm{VP} \rightarrow \mathrm{barks}$	1	3	4	0.93	0.72
$\mathrm{VP} \rightarrow \mathrm{snores}$	1	0	1	0.07	0.28

The Dirichlet distribution



$$\mathrm{Dir}(\theta|\alpha) = \frac{\Gamma(\sum_i \alpha_i)}{\prod_i \Gamma(\alpha_i)} \prod_i \theta_i^{\alpha_i - 1}$$

- Increasingly concentrated when $\alpha_i \gg 1$ or $\alpha_i \ll 1$
- When $\alpha_i \ll 1$, $\mathrm{P}(\theta_i)$ is concentrated around 0
  $\Rightarrow$ *prior prefers not to use rule*

Estimating rule probabilities from strings alone

Hillary barks
Barack barks
Barack barks

- No closed-form solution, but various *Markov Chain Monte Carlo sampling algorithms* and *Variational Bayes approximations* have been developed
- Guess initial production probabilities
- Repeat:
  - produce sample parses for strings in training corpus
  - count rules in sampled parse trees
  - sample production probabilities from rule counts as before
- Repeat this long enough, converges to samples from posterior
- (It is possible to *integrate out* the rule probabilities)

Estimating rule probabilities for toy grammars

Initial rule probs

rule	prob
...	...
VP → V	0.2
VP → V NP	0.2
VP → NP V	0.2
VP → V NP NP	0.2
VP → NP NP V	0.2
...	...
Det → the	0.1
N → the	0.1
V → the	0.1

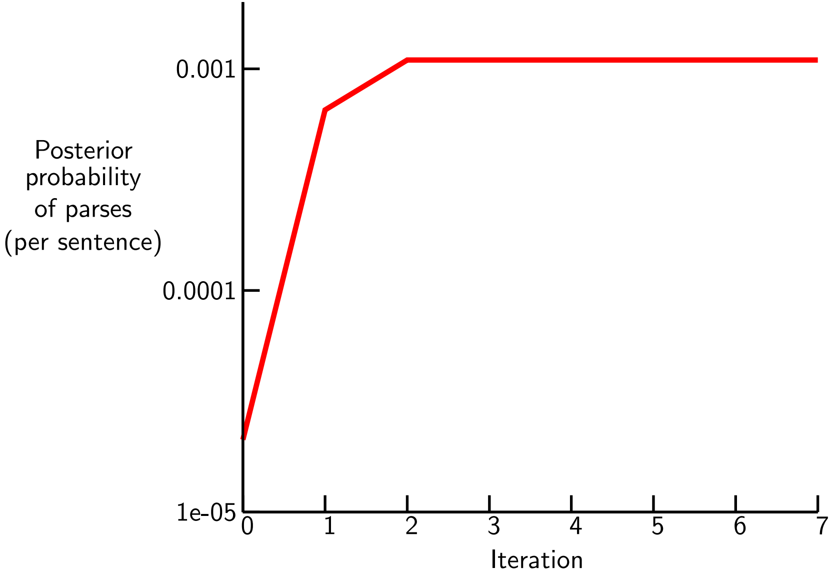
"English" input (50 sentences)

the dog bites
the dog bites a man
a man gives the dog a bone
...

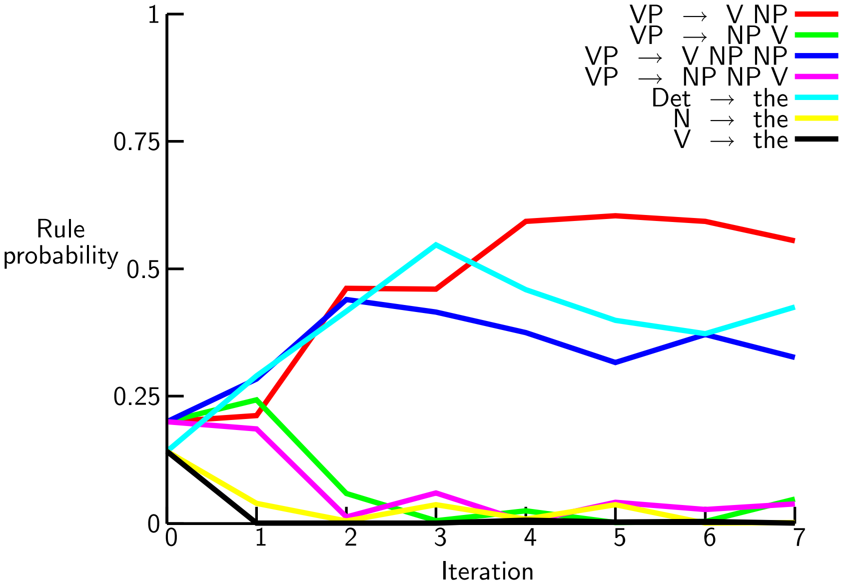
"pseudo-Japanese" input (50 sentenc

the dog bites
the dog a man bites
a man the dog a bone gives
...

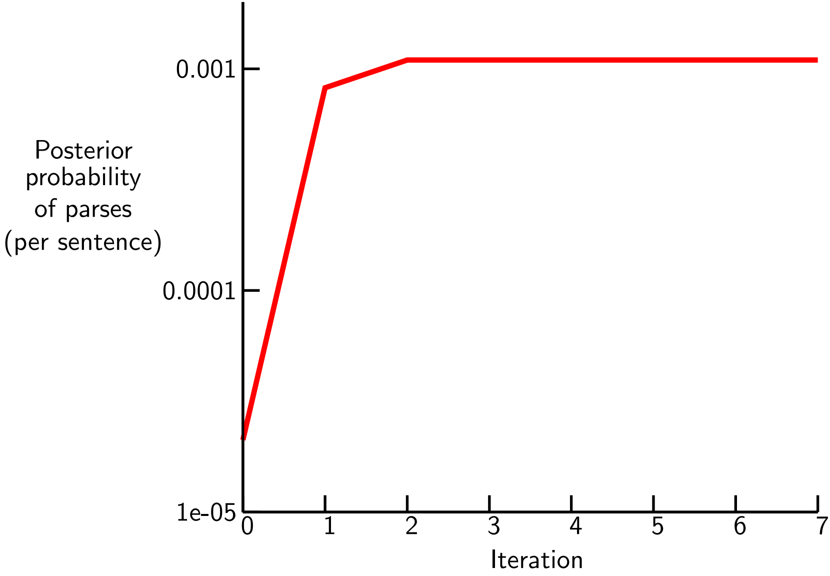
Probability of “English”



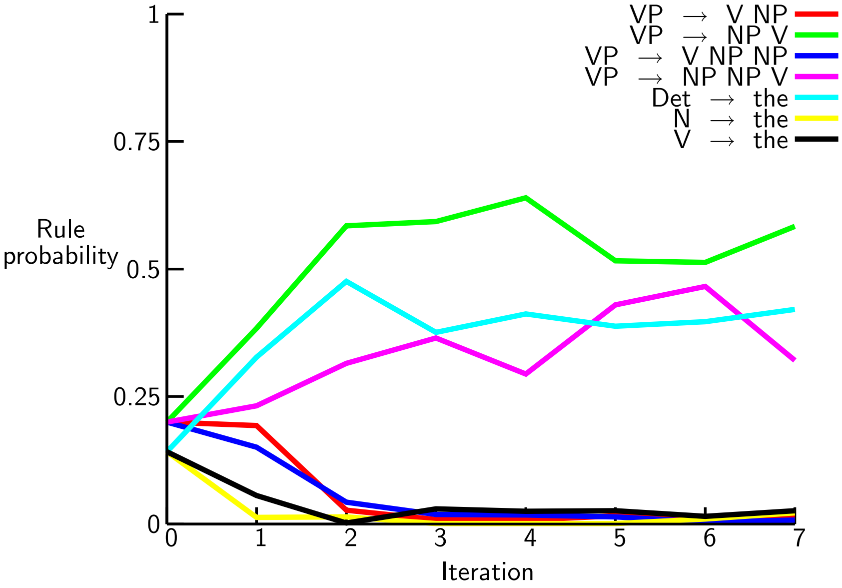
Rule probabilities from “English”



Probability of "Japanese"



Rule probabilities from "Japanese"



Summary so far

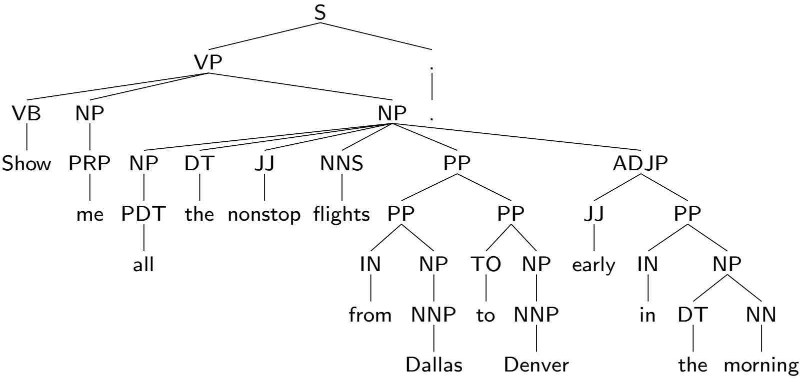
+ Simple algorithm for learning rule probabilities:
  learn from your current "best guesses"
  - requires learner to parse the input sentences
+ "Glass box" models: learner's prior knowledge and learnt generalizations are *explicitly represented*
- We've seen how to estimate the rule probabilities
  Where do the rules come from?

Where do the rules come from?

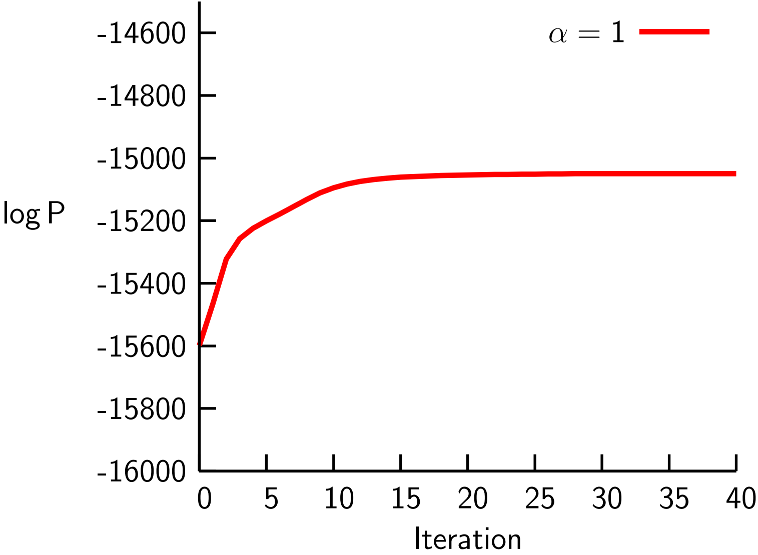
- Maybe they're all innate?
- Common approach: *generate and prune*
  - generate a large "superset" grammar (from where?)
  - use a "sparse" prior that prefers rules have zero probability
  - estimate rule probabilities
  - discard low probability rules

Estimation from real input

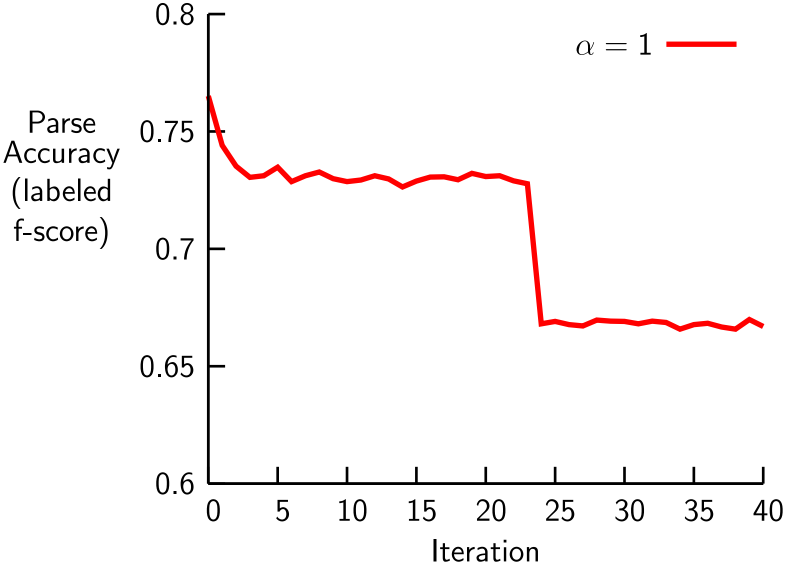
- ATIS treebank consists of 1,300 hand-constructed parse trees
- input consists of POS tags rather than words
- about 1,000 PCFG rules are needed to build these trees



Probability of training strings



Accuracy of parses of training strings



The PCFG model isn't a good model of syntax

- Parse accuracy drops as likelihood increases
  - higher likelihood $\not\Rightarrow$ better parses
  - *the statistical model is wrong*
- Initialized estimator with correct parse trees
  - started with true rules and their probabilities

  $\Rightarrow$ poor performance not due to search error
- Evaluated on training data
  - poor performance not due to over-learning

Why didn't it learn the right grammar?

- Higher likelihood $\not\Rightarrow$ better parse accuracy
  $\Rightarrow$ model is wrong
- What could be wrong?
  - Wrong grammar (Klein and Manning, Smith and Eisner)
  - Wrong training data (Yang)
  - Grammar *ignores semantics* (Zettlemoyer and Collins)

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

$\Rightarrow$ Study simpler problems

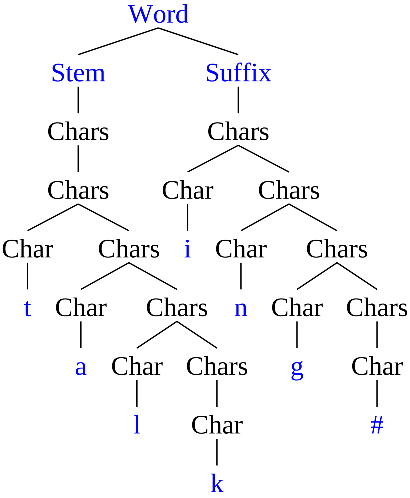
Outline

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., $\# \; \text{t a l k i n g} \; \#$
- Where we're going:
  - CFGs are good ways of generating potentially useful structures
  - but *PCFGs are not good at describing the probability of structures*

A CFG for stem-suffix morphology

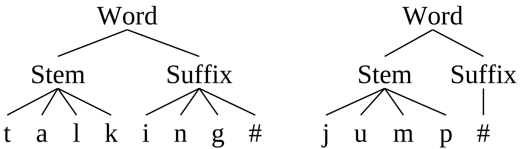
Word	→	Stem Suffix	Chars	→	Char
Stem	→	Chars	Chars	→	Char Chars
Suffix	→	Chars	Char	→	a | b | c | …



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

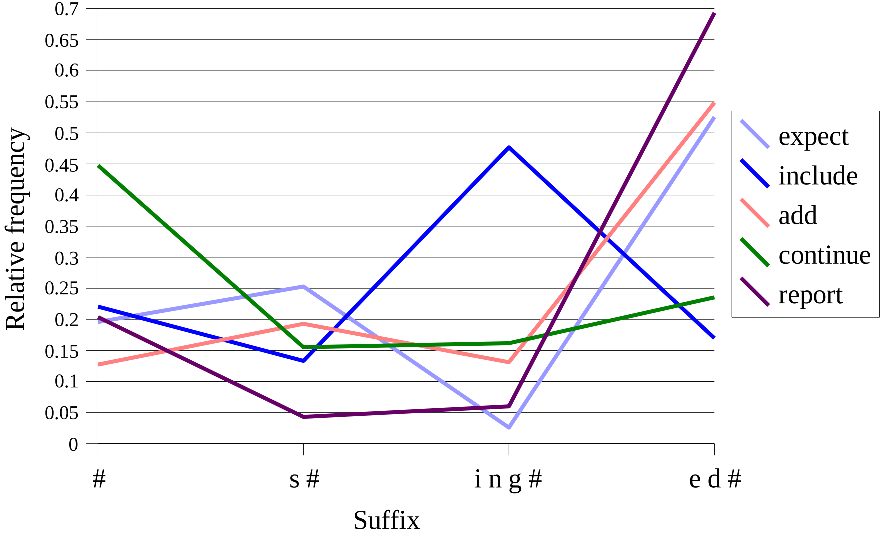
A “CFG” with one rule per possible morpheme

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



- A rule for each morpheme
  $\Rightarrow$ “PCFG” can represent probability of each morpheme
- Unbounded number of rules (but only a finite number can be used in any finite training data set)
- *Assumes* $\mathsf{P}(\text{Word}) = \mathsf{P}(\text{Stem})\mathsf{P}(\text{Suffix})$, which is false …

Relative frequencies of inflected verb forms



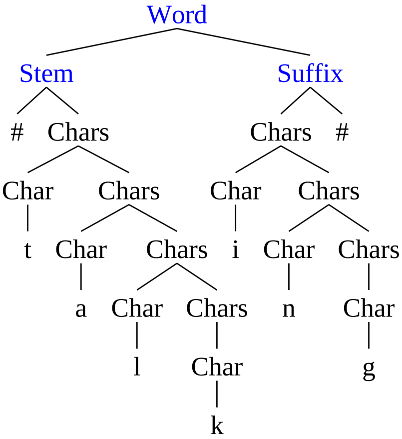
Adaptor grammars: informal description

- 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 $\alpha_A \theta_{A \rightarrow \beta}$

Adaptor grammar morphology example



Word	$\rightarrow$	Stem Suffix
Stem	$\rightarrow$	# Chars
Suffix	$\rightarrow$	#
Suffix	$\rightarrow$	Chars #
Chars	$\rightarrow$	Char
Chars	$\rightarrow$	Char Chars
Char	$\rightarrow$	$a \mid \ldots \mid z$

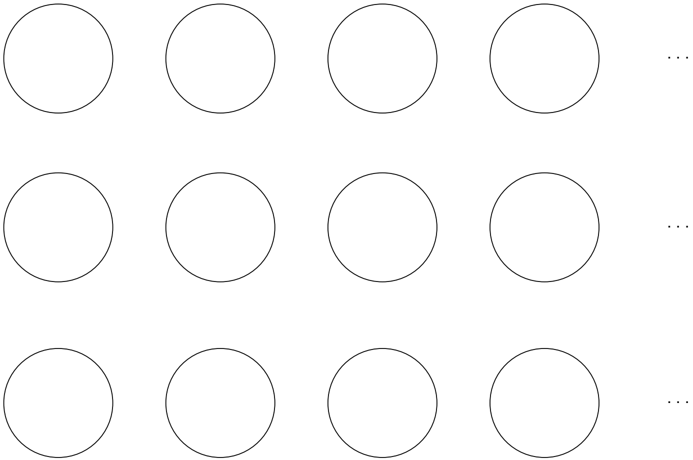
- Stem and Suffix rules generate all possible stems and suffixes
- Adapt Word, Stem and Suffix nonterminals
- Sampler uses *"Chinese restaurant" processes*

Morphology adaptor grammar (0)

Word restaurant
Word → Stem Suffix

Stem restaurant
Stem → #
Stem → # Chars

Suffix restaurant
Suffix → #
Suffix → Chars #



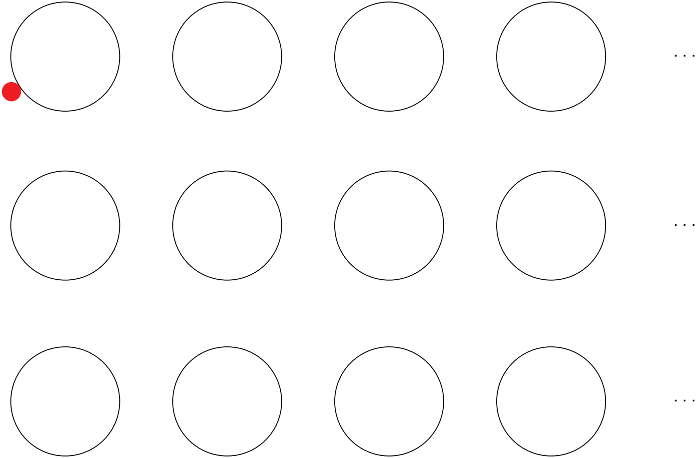
Chars factory
Chars → Char
Chars → Char Chars
Char → a . . . z

Morphology adaptor grammar (1a)

Word restaurant
Word → Stem Suffix

Stem restaurant
Stem → #
Stem → # Chars

Suffix restaurant
Suffix → #
Suffix → Chars #



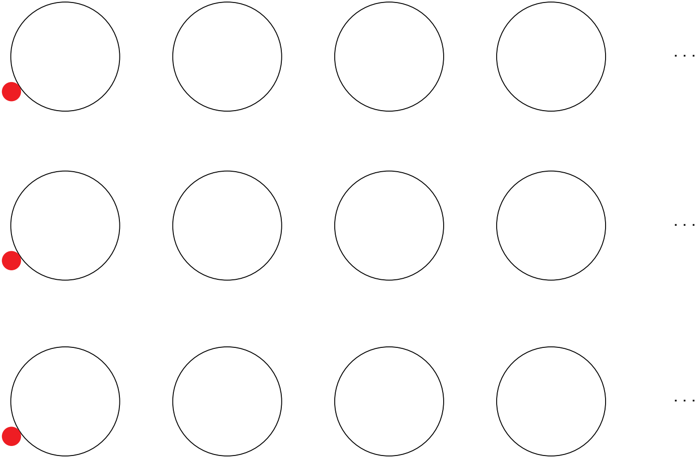
Chars factory
Chars → Char
Chars → Char Chars
Char → a . . . z

Morphology adaptor grammar (1b)

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

Stem restaurant
$\mathrm{Stem} \rightarrow \#$
$\mathrm{Stem} \rightarrow \#\ \mathrm{Chars}$

Suffix restaurant
$\mathrm{Suffix} \rightarrow \#$
$\mathrm{Suffix} \rightarrow \mathrm{Chars}\ \#$



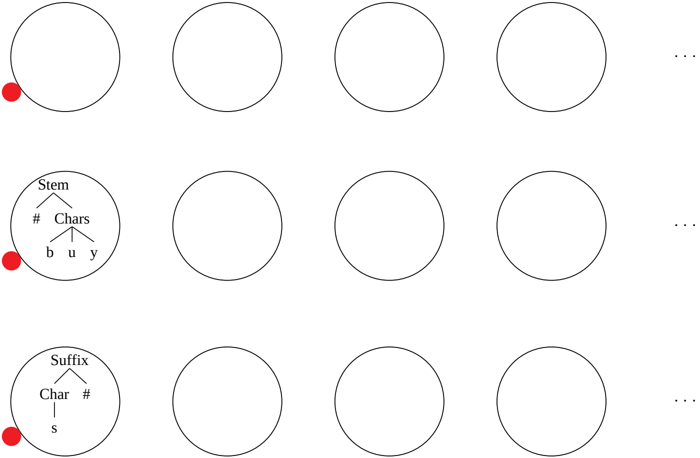
Chars factory
$\mathrm{Chars} \rightarrow \mathrm{Char}$
$\mathrm{Chars} \rightarrow \mathrm{Char}\ \mathrm{Chars}$
$\mathrm{Char} \rightarrow \mathrm{a} \ldots \mathrm{z}$

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

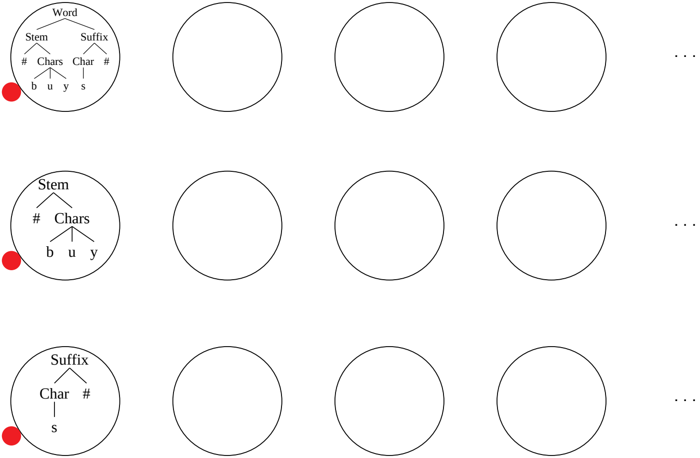
Morphology adaptor grammar (1d)

Word restaurant
Word $\rightarrow$ 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

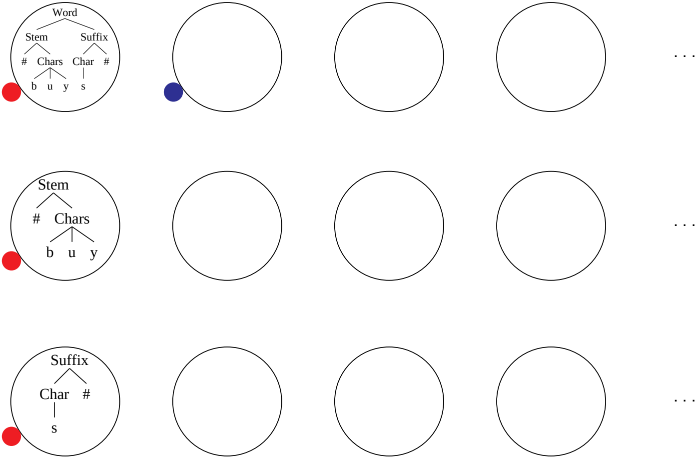


Morphology adaptor grammar (2a)

Word restaurant
Word → Stem Suffix

Stem restaurant
Stem → #
Stem → # Chars

Suffix restaurant
Suffix → #
Suffix → Chars #



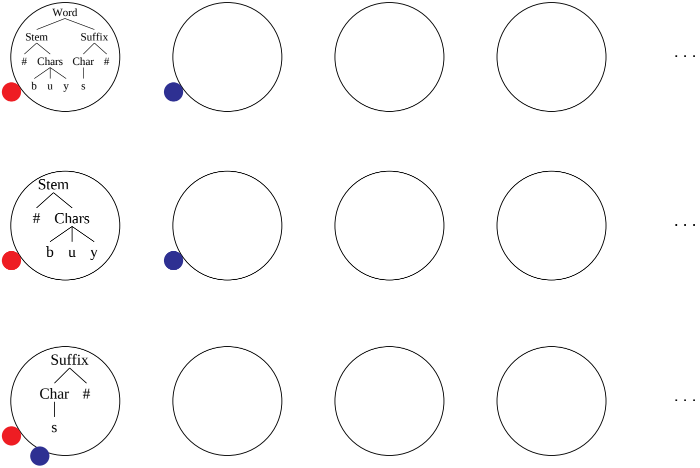
Chars factory
Chars → Char
Chars → Char Chars
Char → a . . . z

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

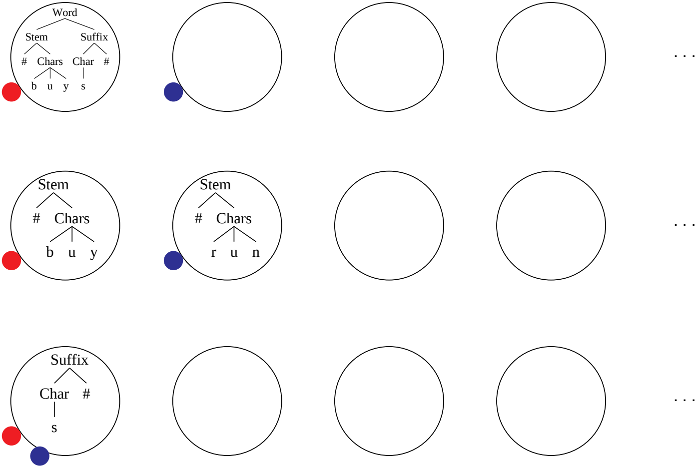
Morphology adaptor grammar (2c)

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

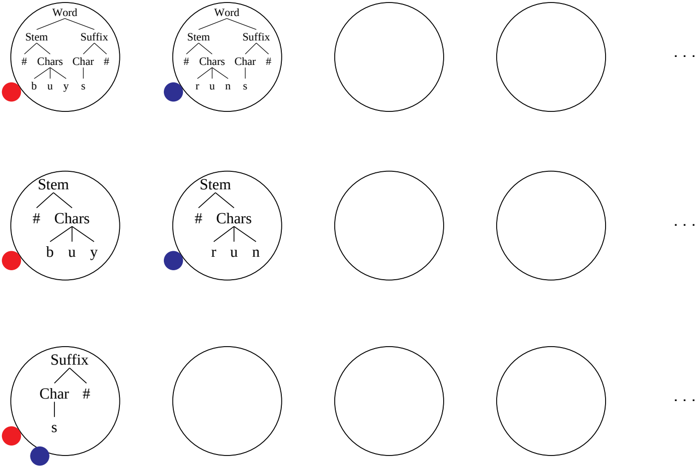


Morphology adaptor grammar (2d)

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

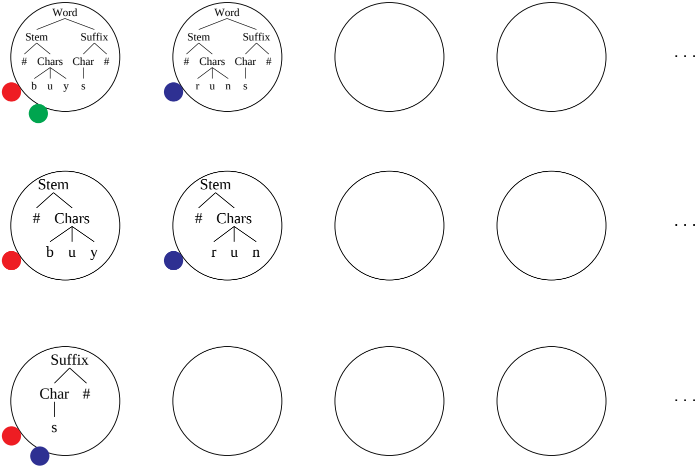
Morphology adaptor grammar (3)

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

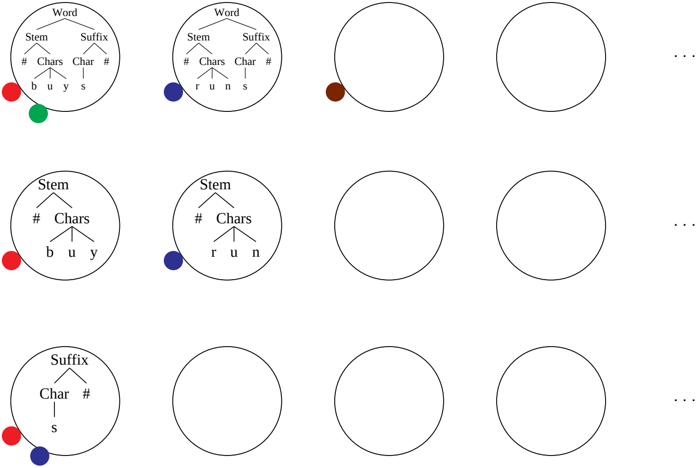


Morphology adaptor grammar (4a)

Word restaurant
Word $\to$ Stem Suffix

Stem restaurant
Stem $\to$ #
Stem $\to$ # Chars

Suffix restaurant
Suffix $\to$ #
Suffix $\to$ Chars #



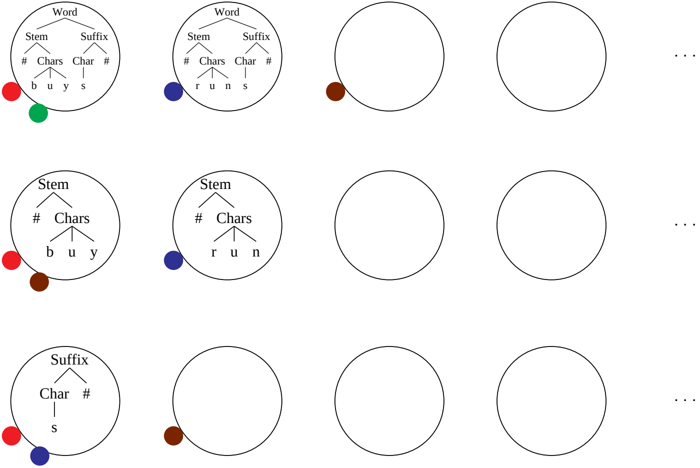
Chars factory
Chars $\to$ Char
Chars $\to$ Char Chars
Char $\to$ a . . . z

Morphology adaptor grammar (4b)

Word restaurant
Word → Stem Suffix

Stem restaurant
Stem → #
Stem → # Chars

Suffix restaurant
Suffix → #
Suffix → Chars #



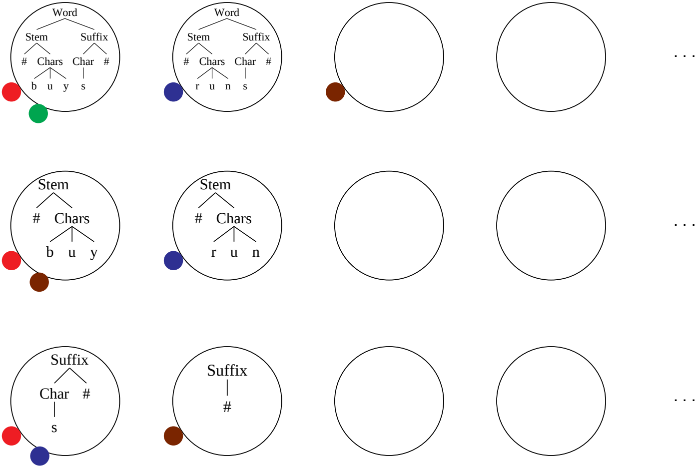
Chars factory
Chars → Char
Chars → Char Chars
Char → a . . . z

Morphology adaptor grammar (4c)

Word restaurant
Word → Stem Suffix

Stem restaurant
Stem → #
Stem → # Chars

Suffix restaurant
Suffix → #
Suffix → Chars #



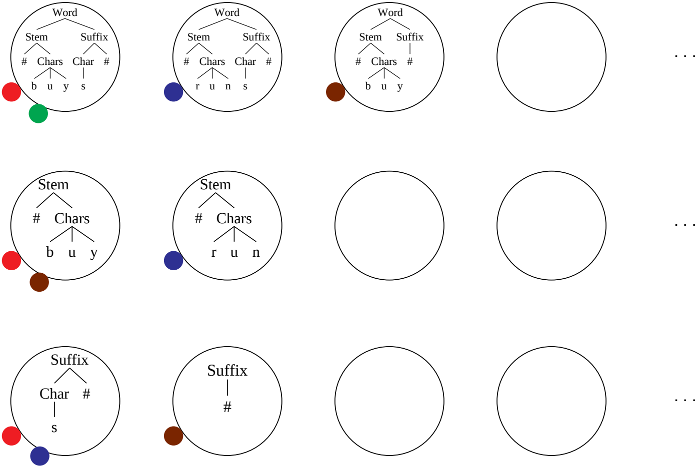
Chars factory
Chars → Char
Chars → Char Chars
Char → a . . . z

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

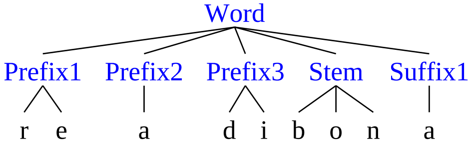
$\Rightarrow$ Useful compound structures can be *more probable than their parts*

- 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"



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

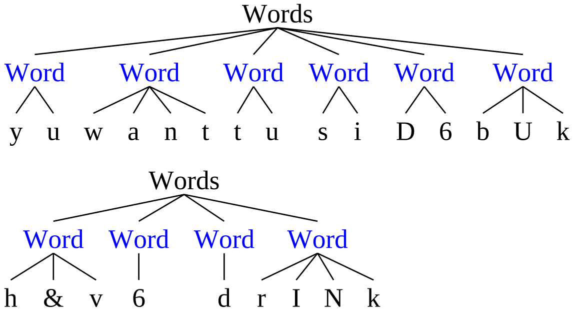
- Sesotho is a Bantu language with complex morphology, not much phonology
- Demuth's Sesotho corpus contains morphological parses for 2,283 distinct verb types
- An adaptor grammar finds morphological analyses for these verbs
  - 62% f-score (morpheme accuracy)
  - 41% words completely correct

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

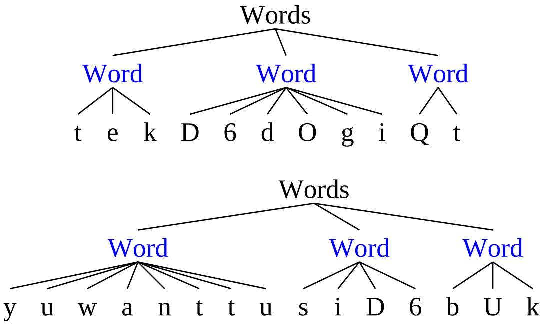
$$\text{Words} \rightarrow \text{Word}^{+}$$
$$\text{Word} \rightarrow \text{Char}^{+}$$



- Unigram word segmentation on Brent corpus: 54% token f-score, 59% type f-score

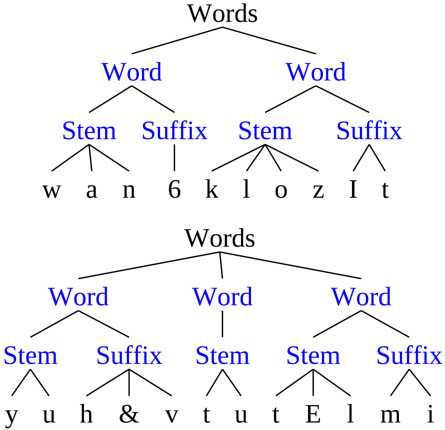
Unigram model often finds collocations

- 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



Combining morphology and word segmentation

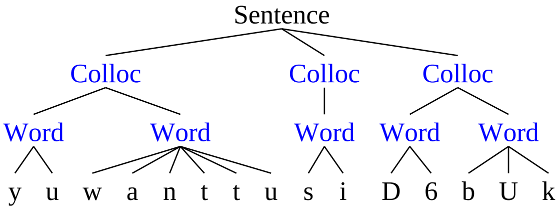
$$\begin{aligned}
\text{Words} &\rightarrow \text{Word}^{+} \\
\text{Word} &\rightarrow \text{Stem Suffix} \\
\text{Word} &\rightarrow \text{Stem} \\
\text{Stem} &\rightarrow \text{Char}^{+} \\
\text{Suffix} &\rightarrow \text{Char}^{+}
\end{aligned}$$



- Adaptors for Word, Stem and Suffix terminals
- Doesn't do a good job of learning morphology, but does find interesting collocations

Modeling collocations improves segmentation

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



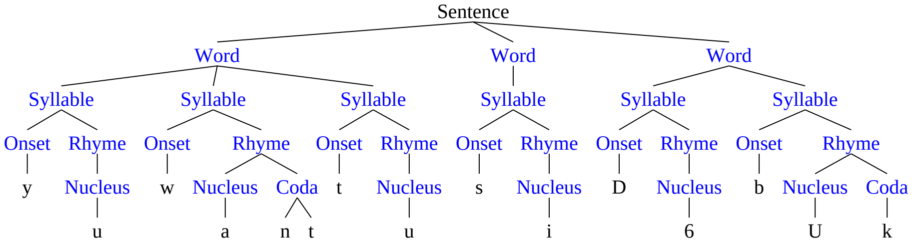
- A collocation consists of one or more words
- Both words and collocations are adapted
- Significantly improves word segmentation accuracy over unigram model (64% token f-score)

Simultaneously learning word segmentation and syllable structure

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

- Word, Syllable, Onset, Nucleus and Coda are all adapted
- Seems to do a fairly good job of identifying syllable boundaries
- Doesn't do as well at segmentation as unigram model (46% token f-score)
- but I haven't tried tweaking the prior, or sampling longer ...

Simultaneous word segmentation and syllable structure



Outline

Summary and future work

- Adaptor grammars “adapt” their distribution to the strings they have generated
- They learn the subtrees of an adapted nonterminal 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

Why *computational* linguistics?

Grammars (finite descriptions of languages)

Learning morphology with adaptor grammars

Word segmentation using adaptor grammars

Conclusions

Technical details

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* $\mathsf{DP}(\alpha, \mathsf{P}_G)$
- $\mathsf{DP}(\alpha, \mathsf{P}_G)$ is called a *Dirichlet process* with *concentration parameter* α and *base distribution* P_G
- Distributions in $\mathsf{DP}(\alpha, \mathsf{P}_G)$ are *discrete* (w.p. 1) even if the base distribution P_G is continuous

PCFGs as recursive mixtures

The distributions over strings induced by a PCFG in *Chomsky-normal form* (i.e., all productions are of the form $A \rightarrow B\ C$ or $A \rightarrow w$, where $A, B, C \in N$ and $w \in T$) is G_S where:

$$G_A = \sum_{A \rightarrow B\ C \in R_A} \theta_{A \rightarrow B\ C} G_B \bullet G_C + \sum_{A \rightarrow w \in R_A} \theta_{A \rightarrow w} \delta_w$$

$$\begin{aligned} (P \bullet Q)(z) &= \sum_{xy=z} P(x) Q(y) \\ \delta_w(x) &= 1 \text{ if } w = x \text{ and } 0 \text{ otherwise} \end{aligned}$$

In fact, $G_A(x) = \mathrm{P}(A \Rightarrow^\star x | \theta)$, the sum of the probability of all trees with root node A and yield x

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.

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

The probabilistic language defined by the grammar is 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