Synergies in learning syllables and words

or

Adaptor grammars: a class of nonparametric Bayesian models

Mark Johnson
Brown University

Joint work with Sharon Goldwater and Tom Griffiths

NECPHON, November, 2008

Research goals

- Most learning methods learn values of fixed set of parameters
  Can we learn units of generalization (rules) as well?
  - non-parametric Bayesian inference
  - Adaptor grammars
- Word segmentation and lexical acquisition (Brent 1996, 1999)

  Example: *y u w a n t t u s i D 6 b u k*

  Things we might want to learn: words, syllables, collocations
- *What regularities are useful for learning words and syllables?*
  - Learning words, collocations and syllables simultaneously is better than learning them separately

  $\Rightarrow$ there are powerful *synergies in acquisition*

Brief survey of related work

- Segmenting words and morphemes at conditional probability minima (Harris 1955, Saffran et al 1996)
- Bayesian unigram model of word segmentation (Brent 1996, 1999)
- Bigram model of word segmentation (Goldwater et al 2006)
- Syllables as basis for segmentation (Swingley 2005; Yang 2004)
- Using phonotactic cues for word segmentation (Blanchard et al 2008; Fleck 2008)
- Modelling syllable structure with PCFGs (Müller 2002, Goldwater et al 2005)

Outline

Adaptor grammars and nonparametric Bayesian models of learning

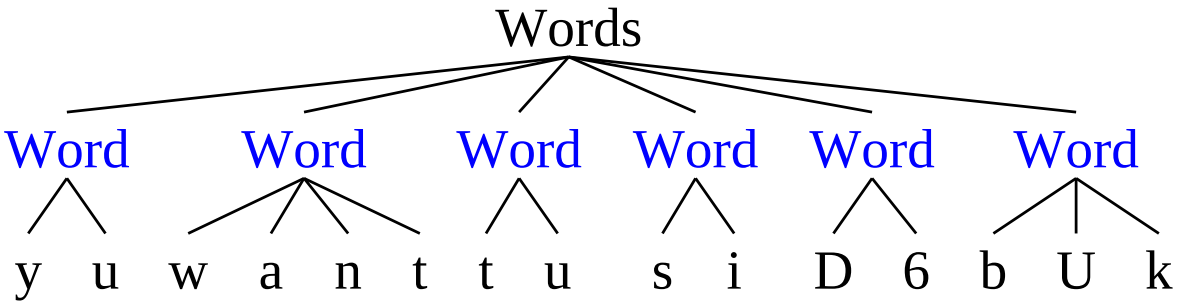
Learning syllables, words and collocations

Learning syllabification with adaptor grammars

Conclusions and future work

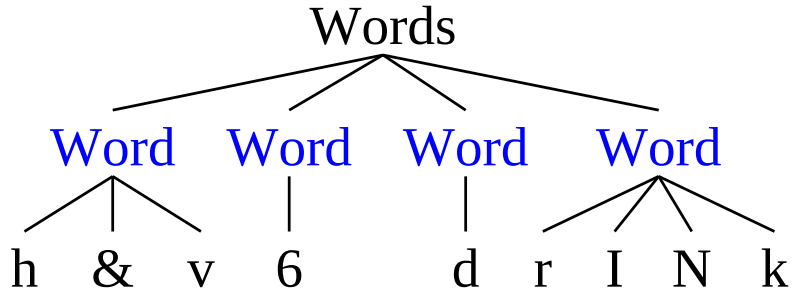
Unigram word segmentation adaptor grammar

- Input is *unsegmented broad phonemic transcription*
  Example: y u w a n t t u s i D 6 b u k
- Word is adapted ⇒ reuses previously generated words



"You want to see the book"

Words → Word$^+$
Word → Phoneme$^+$



"Have a drink"

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

Adaptor grammars: informal description

- *Adaptor grammars learn the units of generalization*
- An adaptor grammar has a set of CFG rules
- These determine the possible tree 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)
- Potential generalizations are all possible subtrees of adapted nonterminals, but *only those actually used are learned*

Adaptor grammars as generative processes

- An *unadapted nonterminal* A expands using $A \rightarrow \beta$ with probability $\theta_{A \rightarrow \beta}$
- An *adapted nonterminal* A expands:
  - to a subtree τ rooted in A with probability proportional to the number of times τ was previously generated
  - using $A \rightarrow \beta$ with probability proportional to $\alpha_A \theta_{A \rightarrow \beta}$
- Zipfian "rich-get-richer" power law dynamics
- Full disclosure:
  - also learn base grammar PCFG rule probabilities $\theta_{A \rightarrow \beta}$
  - use Pitman-Yor adaptors (which discount frequency of adapted structures)
  - learn the parameters (e.g., α_A) associated with adaptors

The basic learning algorithm is simple

- Integrated parsing/learning algorithm:
  - Certain structures (words, syllables) are *adapted* or memorized
  - Algorithm counts how often each adapted structure appears in previous parses
  - Chooses parse for next sentence with probability proportional to parse's probability
  - Probability of an adapted structure is proportional to:
    - *number of times structure was generated before*
    - plus α times probability of generating structure from *base distribution* (PCFG rules)
- *Why does this work?*

  (cool math about Bayesian inference)

Adaptor grammar learnt from Brent corpus

- *Initial grammar*

1	Sentence → Word Sentence	1	Sentence → Word
100	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
100	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))))		

Non-parametric Bayesian inference

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

- Parametric model $\Rightarrow$ finite, prespecified parameter vector
- *Non-parametric* model $\Rightarrow$ parameters chosen based on data
- *Bayesian inference* relies on Bayes rule:

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

- Likelihood measures how well grammar describes data
- Prior expresses knowledge of grammar before data is seen
  - base PCFG specifies prior in adaptor grammars
- Posterior is *distribution* over grammars
  - expresses uncertainty about which grammar is correct
  - *sampling* is a natural way to characterize posterior

Algorithms for learning adaptor grammars

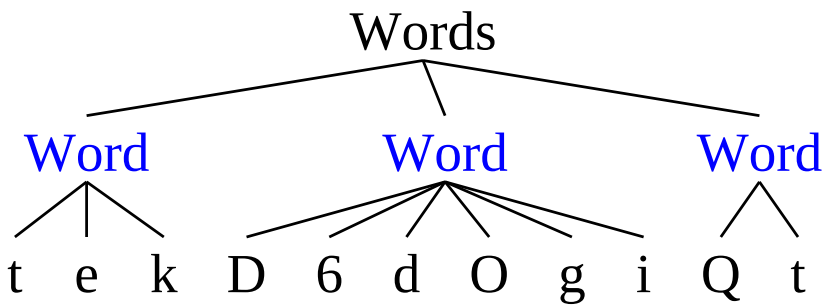
- *Naive integrated parsing/learning algorithm*:
  - *sample* a parse for next sentence
  - *count* how often each adapted structure appears in parse
- Sampling parses addresses *exploration/exploitation dilemma*
- First few sentences receive random segmentations
  $\Rightarrow$ this algorithm does *not* optimally learn from data
- *Gibbs sampler* batch learning algorithm
  - assign every sentence a (random) parse
  - repeatedly cycle through training sentences:
    - withdraw parse (decrement counts) for sentence
    - sample parse for current sentence and update counts
- *Particle filter* online learning algorithm
  - Learn different versions ("particles") of grammar at once
  - For each particle sample a parse of next sentence
  - Keep/replicate particles with high probability parses

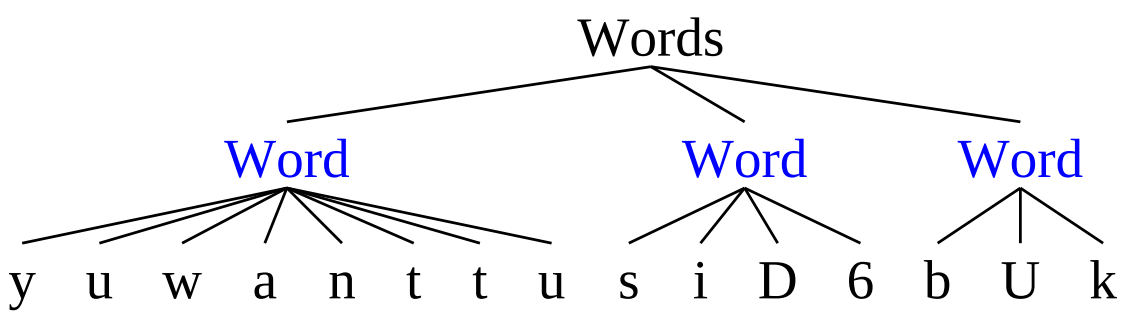
Outline

Unigram model often finds collocations

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

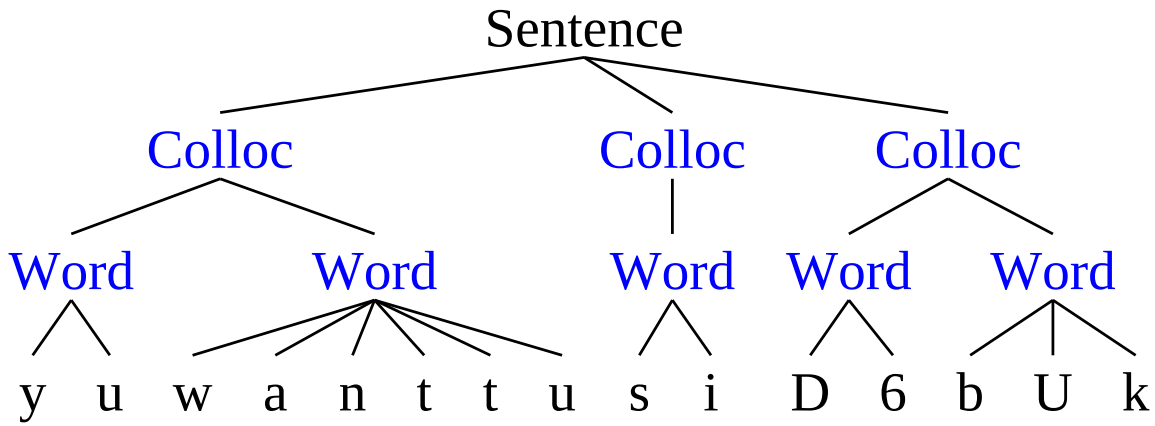
- Unigram word segmentation model assumes each word is generated independently
- But there are strong inter-word dependencies (collocations)
- Unigram model can only capture such dependencies by analyzing collocations as words (Goldwater 2006)





Modelling collocations reduces undersegmentation

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

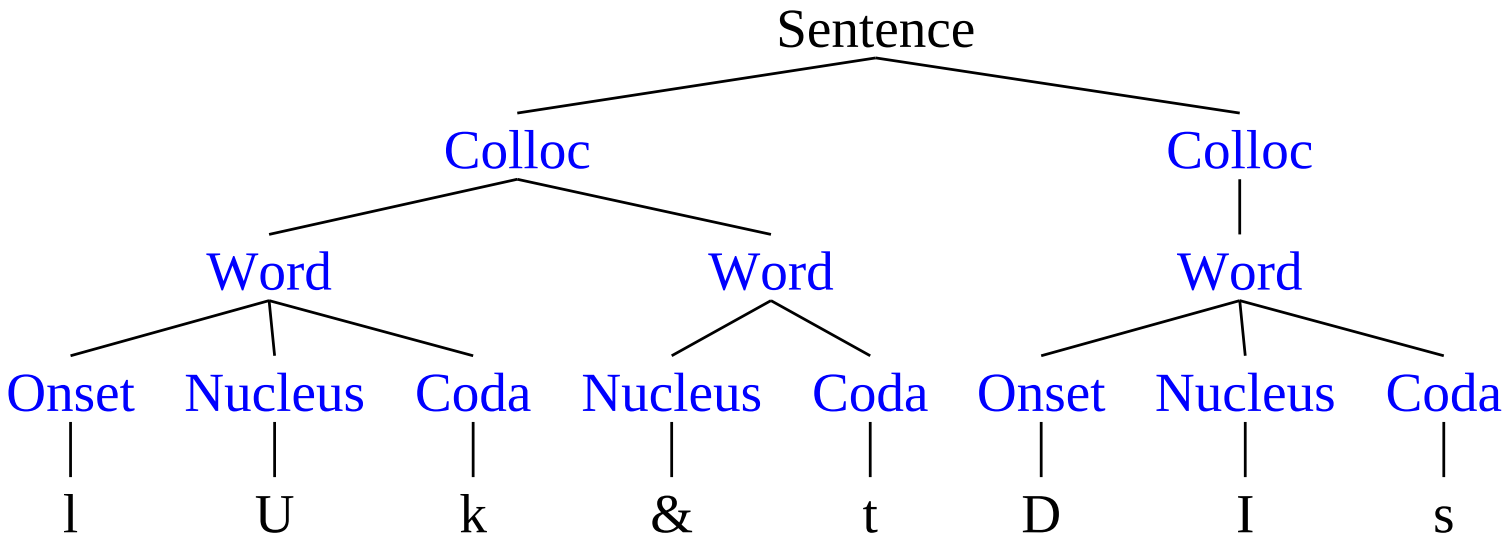


- A Colloc(ation) consists of one or more words
  - poor approximation to syntactic/semantic dependencies
- Both Words and Collocs are adapted (learnt)
  - learns collocations *without being told what the words are*
- Significantly improves word segmentation accuracy over unigram model (75% f-score; $\approx$ Goldwater's bigram model)
- Two levels of Collocations improves slightly (76%)

Syllables + Collocations + Word segmentation

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

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

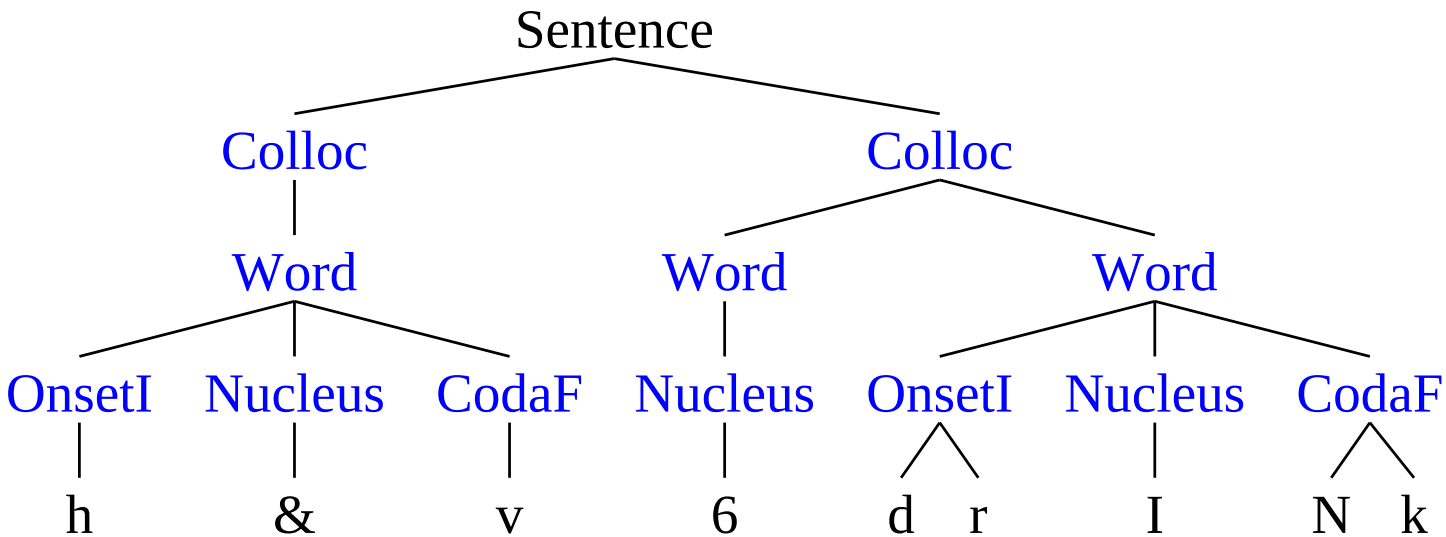


- With no supra-word generalizations, f-score = 68%
- With 2 Collocation levels, f-score = 82%

Distinguishing internal onsets/codas helps

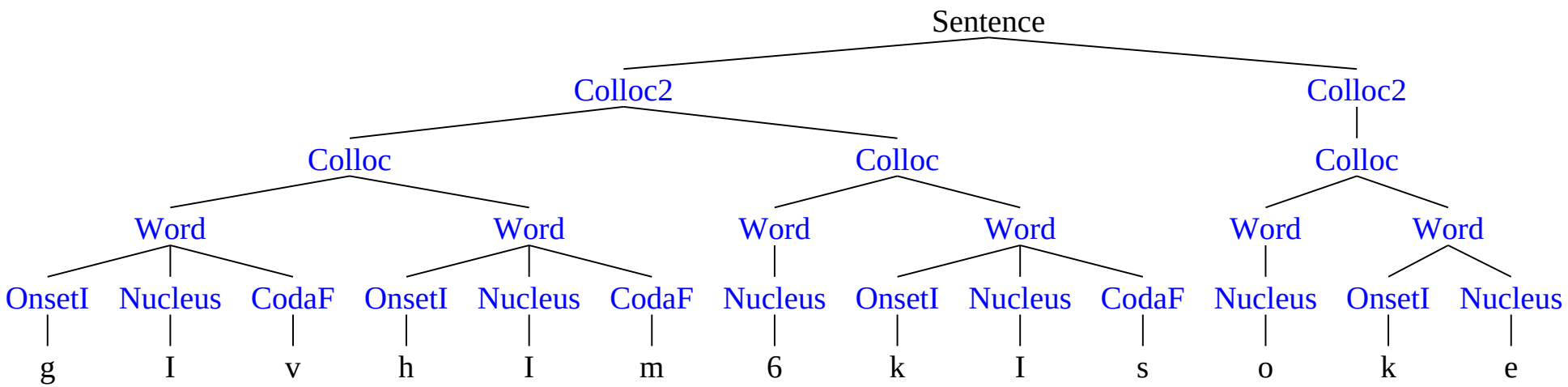
Sentence → Colloc$^+$
Word → SyllableIF
Word → SyllableI Syllable SyllableF
OnsetI → Consonant$^+$
Nucleus → Vowel$^+$

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%

Syllables + 2-level Collocations + Word segmentation



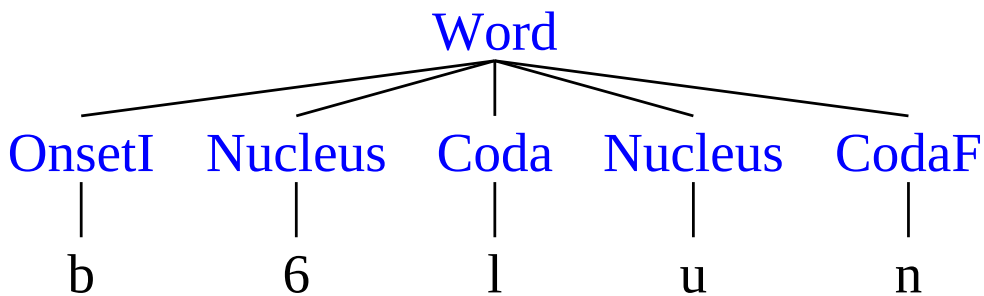
Outline

Syllabification learnt by adaptor grammars

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

1 Syllable → Onset Rhyme 1 Syllable → Rhyme

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

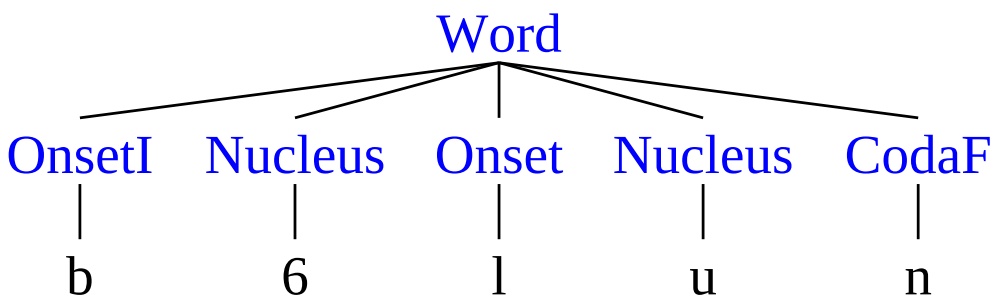


- Syllabification accuracy is relatively poor
  Syllabification given true word boundaries: f-score = 83%
  Syllabification learning word boundaries: f-score = 74%

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 ⇒ 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

$\text{Onset} \rightarrow \text{Onset}_{\text{Stop}}$ $\quad$ $\text{Onset} \rightarrow \text{Onset}_{\text{Fricative}}$

$\text{Onset}_{\text{Stop}} \rightarrow \text{Stop}$ $\quad$ $\text{Onset}_{\text{Stop}} \rightarrow \text{Stop}\,\text{Onset}_{\text{Fricative}}$

$\text{Stop} \rightarrow \text{p}$ $\quad$ $\text{Stop} \rightarrow \text{t}$

- 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%

Outline

Conclusions

- Adaptor grammars learn *an unbounded number of reusable structures*
- The learning algorithms are fairly simple
  - even if their mathematical justification is really cool . . .
- Different adaptor grammars can have different priors
  - preferring Onsets dramatically improves syllabification
- Different adaptor grammars learn different generalizations useful for studying *synergies in learning*
  - Learning interword dependencies improves word segmentation
  - Learning syllabification improves word segmentation
  - Learning word segmentation improves syllabification

$\Rightarrow$ Learning is easier if these are acquired together

- Data and software available from **http://cog.brown.edu/~mj**

Summary of adaptor grammars

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

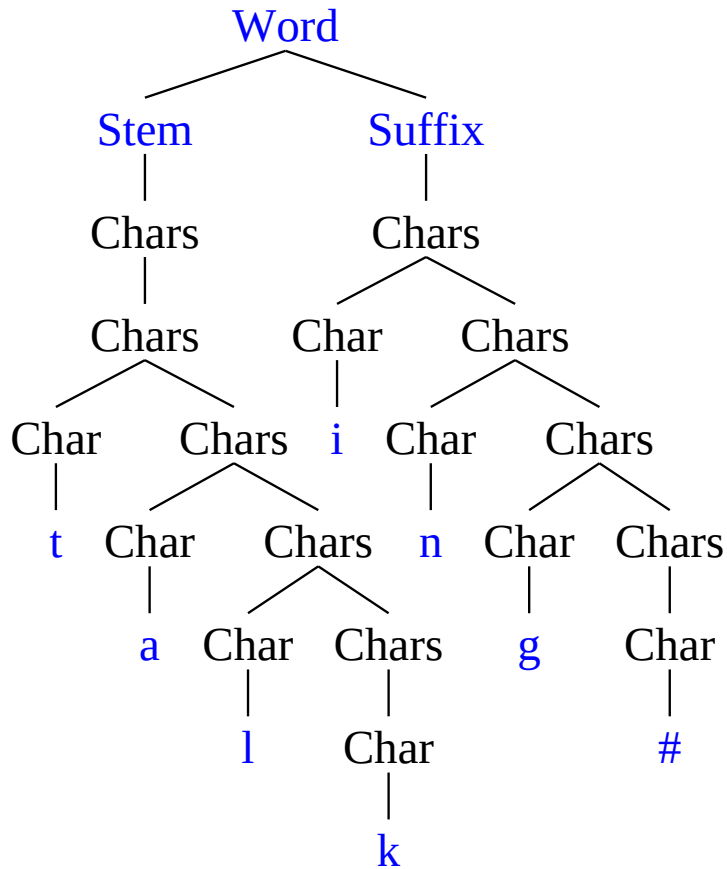
$\Rightarrow$ Frequent structures can be *more probable than their parts*

- Reusing cached structure doesn't increment base counts
  $\Rightarrow$ adaptor grammars learn from types, not tokens
- Trees generated by adaptor grammars are *not* independent
  - an adaptor grammar *learns* from its previous output

  but they are *exchangable*

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
  $\Rightarrow$ higher in Bayesian hierarchy



Chinese restaurant and Pitman-Yor processes

- Pitman-Yor processes (PYPs) are a generalization of Chinese Restaurant Processes (CRPs)
  - An adaptor grammar has one CRP or PYP for each adapted nonterminal
- CRPs and PYPs both map a *base distribution* B to a distribution over distributions with same support as B
  - In adaptor grammars, B is given by the PCFG rules
- Suppose we have generated $\boldsymbol{h} = (x_1, \ldots, x_n)$ so far:

CRP: $\mathsf{P}(X_{n+1} = x | \boldsymbol{h}, \alpha, B) \propto n(x) + \alpha B(x)$, where $n(x)$ is number of times x appears in $\boldsymbol{h}$

PYP: $\mathsf{P}(X_{n+1} = x | \boldsymbol{h}, a, b, B) \propto n(x) - a\, m(x) + (b + a\, m) B(x)$, where $m(x)$ is number of times x has been generated from B in $\boldsymbol{h}$ (i.e., number of "tables" labelled x) and $m = \sum_x m(x)$.