Adaptor Grammars: A Framework for Specifying Compositional Nonparametric Bayesian Models

Mark Johnson and Thomas L. Griffiths

Presented by Yun Zhang

Some slices copied from Mark Johnson's talks of adaptor grammar

Motivation

- A Bayesian prior over grammars
  - Give all grammar rules some probabilities
  - Use a small value of α for Dirichlet prior and get a sparse solution
- A nonparametrics extension
  - Grammar symbols and rules can be unbounded, the numbers should be inferred by data

Pitman-Yor Process

- A natural extension of Chinese Restaurant Process in which atoms are weighted by draws from two-parameter Poisson-Dirichlet distributions

- PYP generates power-law distributions over data which makes it popular in NLP modeling

$$z_{n+1} | z_1, \ldots, z_n \sim \frac{ma + b}{n + b} \delta_{m+1} + \sum_{k=1}^{m} \frac{n_k - a}{n + b} \delta_k$$

Context Free Grammar (CFG)

- What is grammar?
  - A system of rules that defines the grammatical structural of a languange
- What is CFG?
  - Context Free Grammar is rules to interpret Context Free Language (CFL)
- What is CFL
  - Any language can be accepted by pushdown automata, which is a computational model stronger than FA but weaker than TM
- Why CFG?
  - Early linguistics researchers believe human language is CFL
  - Programming Languages are CFL

CFG cont.

- A context-free grammar (CFG) consists 4 tuples <N,W,R,S>:
  - A finite set N of nonterminals
  - A finite set W of terminals disjoint from N
  - A finite set R of rules A->β, where A belongs to N and β belongs to (N U W)*
  - a start symbol S belongs to N

- An example
  - <sentence> -> <noun><verb><noun>
  - <noun> -> <prep>
  - <prep> -> he
  - <noun> -> food
  - <verb> -> likes

- The grammar is then sufficient to parse “he likes food”

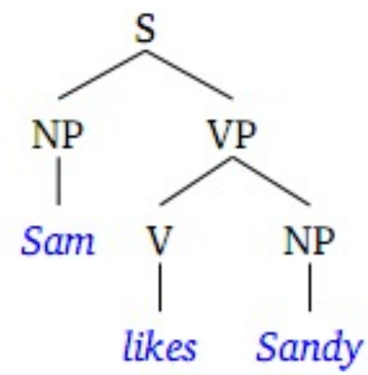
Probabilistic CFG

- A CGL can be interpreted by many CFG
- An ambiguous CFG can interpret one CFL string in many ways
- PCFG assign different probabilities to all possible interpretation
- A PCFG consists of 5 tuples <N,W,R,S,θ>, which specifies a multinomial distribution θ_A for each nonterminal A over the rules A->α expanding A, denoted as G_A
  - $\theta_{A\text{->}\alpha}$ is probability of A expanding to α

- Giving θ a prior distribution comes the Bayesian approach
- Using PYP to construct θ comes the BNP

PCFG cont.

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



P(Tree) = 1 x 0.7 x 1 x 0.8 x 0.3

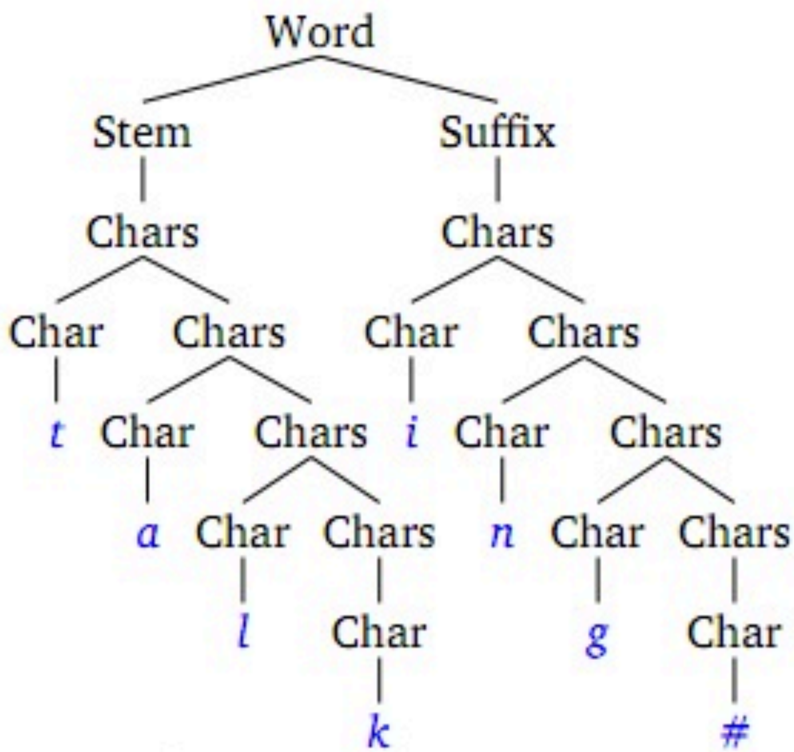
What is the problem?

- PCFG rules are "too small" to be effective units of generalization, e.g. although sentences in English are formed by word sequence, there're words that may have co-occurrence, "state of the art"

- Each rule in the sequence is selected independently at random, you don't always have to draw chars from a word like "Honorrificabilitudinitatibus"

A CFG for stem-suffix morphology

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



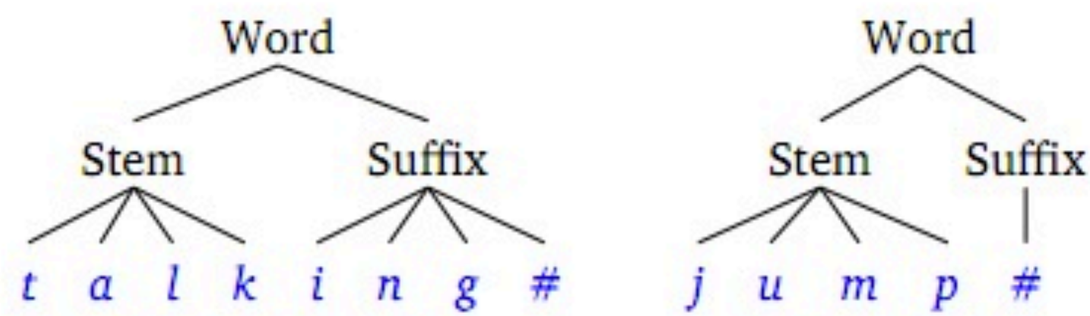
- Grammar's trees can represent any segmentation of words into stems and suffixes

⇒ 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

Pitman-Yor Adaptor Grammar(PYAG)

- Adaptor grammars consists of 6 tuples <N,W,R,S,θ,C>, where C is a function from distribution to a distribution over distribution with the same support as G
  - $C_A = (a_A, b_A, x_A, n_A)$
- For each nonterminal symbols A , rather than draw prob. from G_A in PCFG, we draw from $H_A \sim C_A (G_A)$
- When H is integrated over, we get a distribution for any specific choice of C.

From PCFGs to Adaptor grammars

- An adaptor grammar is a PCFG where a subset of the nonterminals are adapted

- Adaptor grammar generative process:
  - to expand an unadapted nonterminal B: (just as in PCFG)
    - select a rule B->β belongs to R with prob. $\theta_{B\text{->}\beta}$, and recursively expand nonterminals in β
  - to expand an adapted nonterminal B:
    - select a previously generated subtree T_B with prob. Propotional to number of times T_B was generated, or
    - select a rule B->β belongs to R with prob. $\theta_{B\text{->}\beta}$, and recursively expand nonterminals in β

Adaptor grammar for stem-suffix morphology

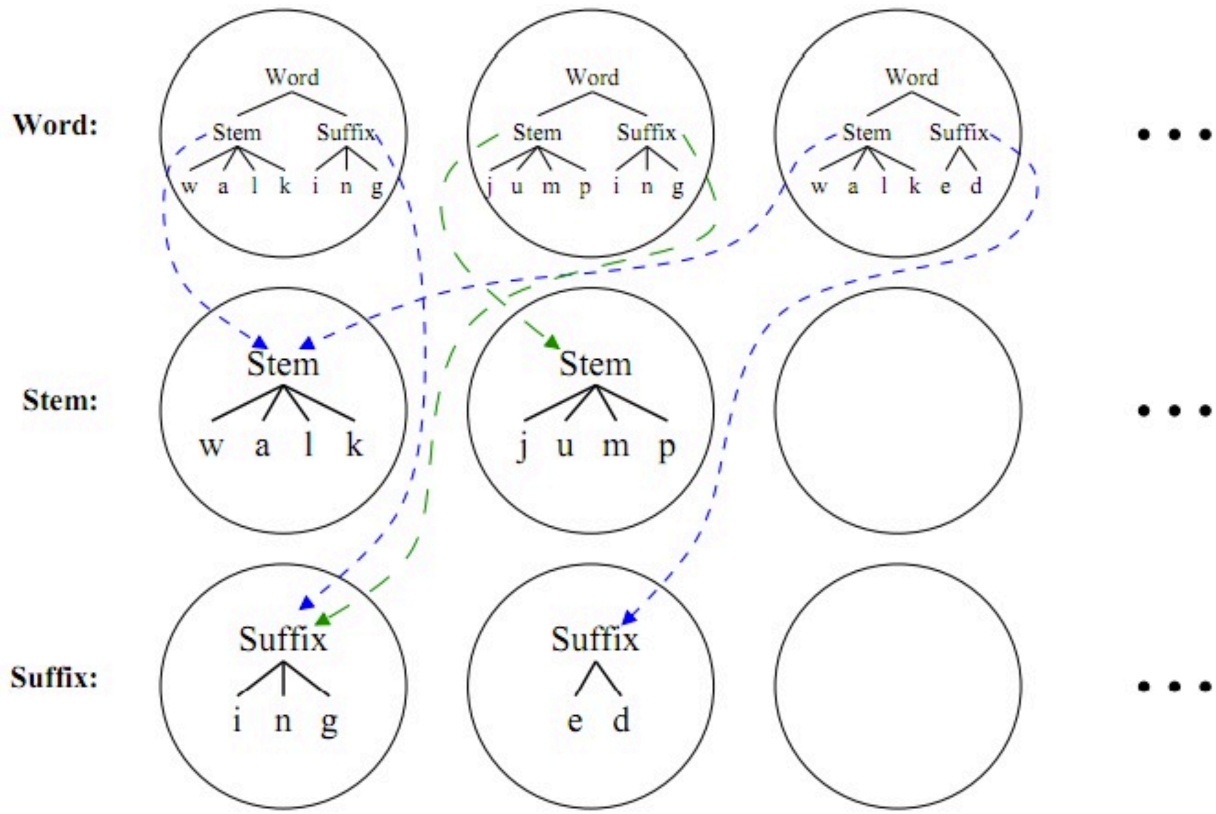


Figure 1: A depiction of a possible state of the Pitman-Yor adaptors in the adaptor grammar of Section 4.2 after generating *walking*, *jumping* and *walked*.

A Sampling method

$$P(\mathbf{u} \mid \alpha, \mathbf{a}, \mathbf{b}) = \prod_{A \in N} \frac{B(\alpha_A + \mathbf{f}_A(\mathbf{x}_A))}{B(\alpha_A)} \mathrm{PY}(\mathbf{n}_A(\mathbf{u}) | \mathbf{a}, \mathbf{b})$$

- The objective here is to maximize the posterior distribution over analyses $u=(u_1, u_2, \ldots, u_n)$ given string sequence $s=(s_1, s_2, \ldots, s_n)$

- u=(t,l) is a Pitman-Yor adaptor analysis. **t** is a potential parse tree rooted by starting symbol given a string sequence, **l** is the index function which is used to query the count of each subtree of **t** in **n**

- Problems**:**
  – The posterior is intractable so we need to do sampling.
  – Sampling is not efficient as PCFG no loner holds, we need to find an PCFG G'($\mathbf{u}_{-i}$) approximation of PYAG

PCFG approximation & MH algorithms

$$R' = R \cup \bigcup_{A \in N} \{A \rightarrow \textsc{Yield}(x) : x \in \mathbf{x}_A\}$$

$$\theta'_{A\rightarrow\beta} = \left(\frac{m_A a_A + b_A}{n_A + b_A}\right)\left(\frac{f_{A\rightarrow\beta}(\mathbf{x}_A) + \alpha_{A\rightarrow\beta}}{m_A + \sum_{A\rightarrow\beta\in R_A} \alpha_{A\rightarrow\beta}}\right) + \sum_{k:\textsc{Yield}(X_{A_k})=\beta} \left(\frac{n_{A_k} - a_A}{n_A + b_A}\right)$$

- Approximate the unbounded productions rules here with R', then make sure θ' sums up to 1
- The approximation G' is used as proposal in the Metroplis-Hasting algorithm
- The production probabilities θ can be computed from the u at convergence

$$A(u_i, u'_i) = \min\left\{1, \frac{\mathrm{P}(\mathbf{u}' \mid \alpha, \mathbf{a}, \mathbf{b})\,\mathrm{P}(u_i \mid s_i, G'(\mathbf{u}_{-i}))}{\mathrm{P}(\mathbf{u} \mid \alpha, \mathbf{a}, \mathbf{b})\,\mathrm{P}(u'_i \mid s_i, G'(\mathbf{u}_{-i}))}\right\}$$

Ideas to take

- Adaptor grammars weaken the independence assumptions made in PCFGs
- It allows both numbers of grammar symbols and rules goes to infinite as amount of observation increases
- The adaptor C constructed with PYP produces power law, it is possible to plugin other stochastic processes to model data with different structure

Discussion

- Choice of original grammar rules
- Cannot handle recursive grammar?
- How the approximation will affect the performance of the sampling algorithm?
- Other inference method