The Infinite PCFG using Hierarchical Dirichlet Processes

Liang, Petrov, Jordan & Klein

Presented by: Will Allen

November 8, 2011

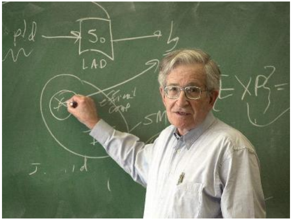
Overview

1. Overview
2. (Very) Brief History of Context Free Grammars
3. Probabilistic Context Free Grammars (PCFG)
4. HDP-PCFG Model
5. HDP-PCFG for grammar refinement (HDP-PCFG-GR)
6. HDP-PCFG Variational Inference
7. Experimental Results

Overview

- Goal: To understanding the latent rules generating the recursive structure of phrases and sentences in natural language.
- Not just for NLP: PCFGs also used in bioinformatics (RNA structure prediction), vision (geometric grammars), and probably other places.

(Very) Brief History of Context Free Grammars

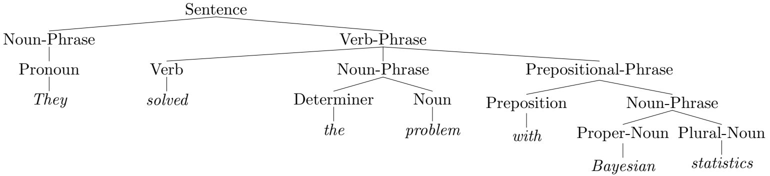


From: `http://lawanddisorder.org/`

- 4th century BC: first description by Pāṇini of a *grammar*, a set of rules dictating the order in which clauses and words appear.
- Grammars are tree-structured to model recursive structure of natural language.
- 1950s: Noam Chomsky invents *context free-grammar* formally describing how to generate these tree structures.

(Very) Brief History of Context Free Grammars

A parse tree:



From: Liang, P., Jordan, M.I., Klein, D. “Probabilistic Grammars and Hierarchical Dirichlet Processes.” (2009)

Book chapter: The Handbook of Applied Bayesian Analysis.

Probabilistic Context Free Grammars

Set of rules for generating parse trees.
A PCFG consists of:

- A set of *terminal symbols* Σ (e.g. actual words)
- A set of *nonterminal symbols* S (e.g. word types)
- A *root nonterminal symbol* $\textsc{Root} \in S$
- *Rule probabilities* $\phi = (\phi_s(\gamma) : s \in S, \gamma \in \Sigma \cup (S \times S)$ where $\phi_s(\gamma) \geq 0$ and $\sum_\gamma \phi_s(\gamma) = 1$ (produce terminal symbols or pairs of nonterminal symbols)

Context Free Grammars

Chomsky Normal Form is used in this paper:

- $A \to BC,\ A, B, C \in S$ (binary production)
- $A \to \alpha,\ \alpha \in \Sigma$ (emission)
- $\textsc{Root} \to \epsilon$ (empty string)

where $A \to BC$ occurs with probability $\phi_A((B, C))$.

HDP-PCFG Model

Previous work for learning PCFGs:

- Models have a fixed number symbols.
- Infer maximum-likelihood symbol transition and emission probabilities by Expectation Maximization algorithm.
- Use pseudocounts for smoothing: may only have a few training examples of each transition.

HDP-PCFG Model

- **Goal**: Learn how many grammar symbols to allocate given data. Use these symbols to learn transition and emission probabilities.
- **Method**: Use HDP to model syntactic tree structures. Nonterminal nodes are symbols.
- **Bonus**: Develop model for *grammar refinement*: given a coarse supervised annotation of tree structures, infer a richer model by learning how many subsymbols to split from existing symbols.

HDP-PCFG Model

- Using Chemosky Normal Form grammar, so only has emissions or binary productions.
- Each grammar symbol is a mixture component. Use DP prior to let number of grammar symbols $\rightarrow \infty$.

HDP-PCFG Model

HDP-PCFG

$\boldsymbol{\beta} \sim \text{GEM}(\alpha)$ [draw top-level symbol weights]

For each grammar symbol $z \in \{1, 2, \dots\}$:

$\phi_z^T \sim \text{Dirichlet}(\alpha^T)$ [draw rule type parameters]

$\phi_z^E \sim \text{Dirichlet}(\alpha^E)$ [draw emission parameters]

$\phi_z^B \sim \text{DP}(\alpha^B, \boldsymbol{\beta}\boldsymbol{\beta}^T)$ [draw binary production parameters]

For each node i in the parse tree:

$t_i \sim \text{Multinomial}(\phi_{z_i}^T)$ [choose rule type]

If $t_i = \text{EMISSION}$:

$x_i \sim \text{Multinomial}(\phi_{z_i}^E)$ [emit terminal symbol]

If $t_i = \text{BINARY-PRODUCTION}$:

$(z_{L(i)}, z_{R(i)}) \sim \text{Multinomial}(\phi_{z_i}^B)$ [generate children symbols]

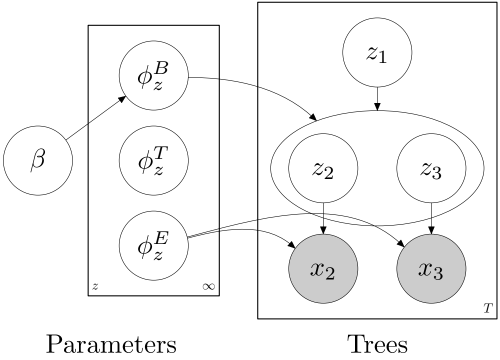
HDP-PCFG Model

Key points:

- Symbols are derived from global stick-breaking prior β
- $\mathrm{DP}(\alpha^B, \beta\beta^T)$ gives a distribution over pairs of symbols for each symbol.
- Unlike in HDP-HMM, either binary production or emission chosen. ϕ_z^T is distribution over type of rule to apply (2 types for CNF).
- Although use Dirichlet/Multinomial for emission distribution for NLP, could use more general base measure to get different emission distribution.

HDP-PCFG Model

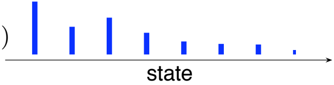
Graphical model of fixed tree
(not showing hyperparameters $\alpha, \alpha^T, \alpha^E, \alpha^B$):



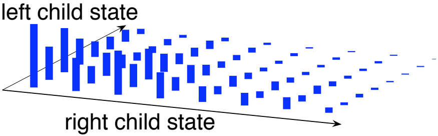
HDP-PCFG Model

Distribution over pairs of child symbols:

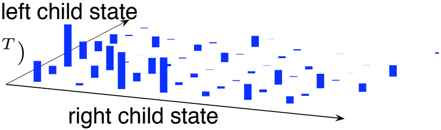
$\boldsymbol{\beta} \sim GEM(\alpha)$



$\boldsymbol{\beta}\boldsymbol{\beta}^T$



$\phi_z^B \sim \mathrm{DP}(\boldsymbol{\beta}\boldsymbol{\beta}^T)$



HDP-PCFG for Grammar Refinement

- Want to refine existing, human-created grammar.
- Are given a set of symbols. Want to allocate some number of subsymbols for each symbol.
- Idea is to better capture subtleties in types of grammatical objects (e.g. different types of noun phrases)

HDP-PCFG for Grammar Refinement

HDP-PCFG for grammar refinement (HDP-PCFG-GR)

For each symbol $s \in S$:
- $\boldsymbol{\beta}_s \sim \mathrm{GEM}(\alpha)$ [draw subsymbol weights]
- For each subsymbol $z \in \{1, 2, \dots\}$:
  - $\phi^T_{sz} \sim \mathrm{Dirichlet}(\alpha^T)$ [draw rule type parameters]
  - $\phi^E_{sz} \sim \mathrm{Dirichlet}(\alpha^E(s))$ [draw emission parameters]
  - $\phi^u_{sz} \sim \mathrm{Dirichlet}(\alpha^u)$ [unary symbol productions]
  - $\phi^b_{sz} \sim \mathrm{Dirichlet}(\alpha^b)$ [binary symbol productions]
  - For each child symbol $s' \in S$:
    - $\phi^U_{szs'} \sim \mathrm{DP}(\alpha^U, \boldsymbol{\beta}_{s'})$ [unary subsymbol prod.]
  - For each pair of children symbols $(s', s'') \in S \times S$:
    - $\phi^B_{szs's''} \sim \mathrm{DP}(\alpha^B, \boldsymbol{\beta}_{s'} \boldsymbol{\beta}^T_{s''})$ [binary subsymbol]

For each node i in the parse tree:
- $t_i \sim \mathrm{Multinomial}(\phi^T_{s_i z_i})$ [choose rule type]
- If $t_i = \textsc{Emission}$:
  - $x_i \sim \mathrm{Multinomial}(\phi^E_{s_i z_i})$ [emit terminal symbol]
- If $t_i = \textsc{Unary-Production}$:
  - $s_{L(i)} \sim \mathrm{Multinomial}(\phi^u_{s_i z_i})$ [generate child symbol]
  - $z_{L(i)} \sim \mathrm{Multinomial}(\phi^U_{s_i z_i s_{L(i)}})$ [child subsymbol]
- If $t_i = \textsc{Binary-Production}$:
  - $(s_{L(i)}, s_{R(i)}) \sim \mathrm{Mult}(\phi_{s_i z_i})$ [children symbols]
  - $(z_{L(i)}, z_{R(i)}) \sim \mathrm{Mult}(\phi^B_{s_i z_i s_{L(i)} s_{R(i)}})$ [subsymbols]

HDP-PCFG for Grammar Refinement

Key points:

1. Similar to previous model, but for each symbol $s \in S$. Creates distribution over symbol/subsymbol pairs (s_i, z_i).
2. Included unary productions (equivalent of state transition in HMM).
3. Since annotated symbols have child symbols already, have to have distribution over child symbols and subsymbols.

HDP-PCFG Variational Inference

- The authors chose to use *variational inference* to avoid having to deal with covergence and sample aggregation.
- Adapts existing efficient EM algorithm for PCFG refinement and induction.
- EM algorithm uses Markov structure of parse tree to do dynamic programming in E-step.

HDP-PCFG Variational Inference

- Recall: variational methods approximate posterior $p(\theta, \mathbf{z}|\mathbf{x})$ with
$$q^* = \arg\min_{q \in Q} \mathbb{KL}(q(\theta, \mathbf{z}) || p(\theta, \mathbf{z}|\mathbf{x}))$$
- In this case: $\theta = (\beta, \phi)$
  - $\beta =$ top-level symbol probabilities
  - $\phi =$ rule probabilities
  - $\mathbf{z} =$ training parse trees
  - $\mathbf{x} =$ observed sentences

HDP-PCFG Variational Inference

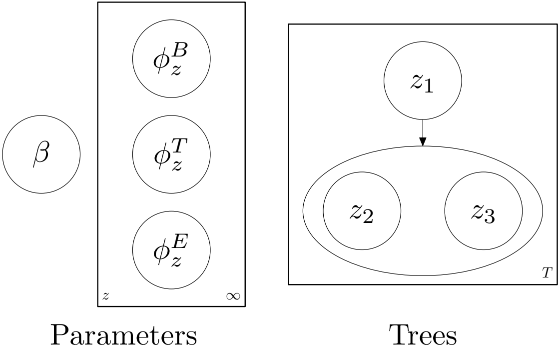
They use a *structured mean-field approxmation*. I.e. only look at distributions of the form

$$Q \equiv \left\{ q : q(\mathbf{z}) q(\beta) \prod_{z=1}^{K} q(\phi_z^T) q(\phi_z^E) q(\phi_z^B) \right\}$$

where $q(\phi_z^T), q(\phi_z^E), q(\phi_z^B)$ are Dirichlet, $q(\mathbf{z})$ is multinomial, $q(\beta)$ is a degenerate distribution truncated at K ($\beta_z = 0$ if $z > K$).

HDP-PCFG Variational Inference

Factorized model $q = q(\beta)q(\phi)q(\mathbf{z})$:



HDP-PCFG Variational Inference

- Optimization of q^* is intractable, but can use coordinate-ascent algorithm similar to EM.
- Optimize one factor at a time while keeping other factors constant

HDP-PCFG Variational Inference

Parse trees $q(\mathbf{z})$

- Uses inside-out algorithm with unnormalized rule weights $W(r)$: dynamic programming algorithm similar to forward-backward for HMMs
- Then computes expected sufficient statistics, rule counts $C(r)$: of binary productions $C(z \to z_l z_r)$ and emissions $C(z \to x)$.

HDP-PCFG Variational Inference

Rule probabilities $q(\phi)$

- Update Dirichlet posteriors: $C(r)$ + pseudocounts
- Compute rule weights: Compute multinomial weights

$$\begin{aligned} W_z^B(z_l, z_r) &= \exp \mathbf{E}_q[\log \phi_z^B(z_l, z_r)] = \frac{e^{\Psi(C(z \to z_l z_r) + \alpha^B \beta_{z_l} \beta_{z_r})}}{e^{\Psi(C(z \to **) + \alpha^B)}} \\ &= \frac{e^{\Psi(\text{prior}(r) + C(r))}}{e^{\Psi(\sum_{r'} \text{prior}(r') + C(r''))}} \end{aligned}$$

  where $\exp \Psi(\cdot)$ increases the weight of large counts and decrease the weight of small counts (as in DP).

- Similar for emission distributions.

HDP-PCFG Variational Inference

Top-level symbol probabilities $q(\beta)$:

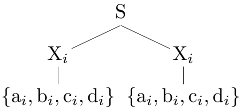
- Truncate at level K. $q(\beta) = \delta_{\beta^*}(\beta)$ so trying to find single best β^*. Use gradient projection method to find:

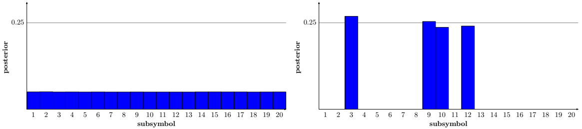
$$\begin{aligned}\arg\max_{\beta^*} L(\beta^*) &= \log \mathrm{GEM}(\beta^*; \alpha) \\ &+ \sum_{z=1}^{K} \mathbf{E}_q[\log \mathrm{Dirichlet}(\phi_z^B; \alpha^B \beta^* {\beta^*}^T)].\end{aligned}$$

Results

Recovering synthetic grammar:

$$
\begin{aligned}
S &\to X_1X_1 \mid X_2X_2 \mid X_3X_3 \mid X_4X_4 \\
X_1 &\to a_1 \mid b_1 \mid c_1 \mid d_1 \\
X_2 &\to a_2 \mid b_2 \mid c_2 \mid d_2 \\
X_3 &\to a_3 \mid b_3 \mid c_3 \mid d_3 \\
X_4 &\to a_4 \mid b_4 \mid c_4 \mid d_4
\end{aligned}
$$





standard PCFG HDP-PCFG

Generate 2000 trees, with terminal symbols having same i, then replace X_i with X.

Results

- Empirical results measured by $F_1 = 2\frac{\text{precision}*\text{recall}}{\text{precision}+\text{recall}}$.
- Uses labeled brackets to represent the tree: $LB(s) = \{(s_{[i,j]}, [i,j]) : s_{[i,j]} \neq \textsc{Non-node}, 1 \leq i \leq j \leq n\}$
- $\text{precision}(s, s') = \frac{\#\text{ correct}}{\#\text{ returned}} = \frac{|LB(s) \cap LB(s')|}{|LB(s')|}$
- $\text{recall}(s, s') = \frac{\#\text{ correct}}{\#\text{ should have returned}} = \frac{|LB(s) \cap LB(s')|}{|LB(s)|}$
- s is true parse tree, s' is predicted.

Results

Applied to one section (WSJ) of Penn Treebank (corpus of parsed sentences), preprocessed so fit CNF:

K	PCFG		PCFG (smoothed)		HDP-PCFG	
	F_1	Size	F_1	Size	F_1	Size
1	60.47	2558	60.36	2597	60.5	2557
2	69.53	3788	69.38	4614	71.08	4264
4	75.98	3141	77.11	12436	77.17	9710
8	74.32	4262	79.26	120598	79.15	50629
12	70.99	7297	78.8	160403	78.94	86386
16	66.99	19616	79.2	261444	78.24	131377
20	64.44	27593	79.27	369699	77.81	202767

Recap

Main contributions:

- Used HDP prior to allow Chomsky Normal Form PCFG to learn the number of symbols in a grammar while also learning the rule transition and emission probabilities.
- Developed an efficient variational methods for inference, similar to existing EM algorithms for PCFG.
- Can be extended to model other kinds of context free grammars.

Possible problems:

- Variational methods only finds local maxima?
- Anything else?