Probabilistic Grammars and Hierarchical Dirichlet Processes (Liang et. al 2009)

Sean Massung & Gourab Kundu

CS 598jhm

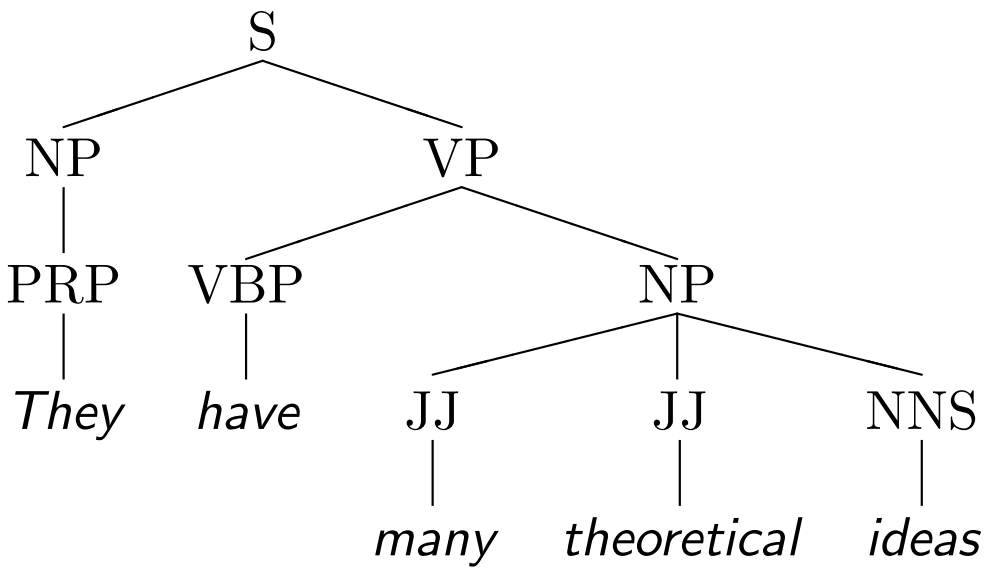
April 9th 2013

Background

This paper (chapter of a book) describes a Bayesian approach to the problem of syntactic parsing and the underlying problems of *grammar induction* and *grammar refinement*.

- **Grammar induction**: estimating grammars based on raw sentences alone, without any other type of supervision
  - Original approaches had poor performance due to the coarse-grained nature of the syntactic categories
- **Grammar refinement**: "splitting" coarse-grained syntactic categories into finer, more accurate and descriptive labels
  - *e.g.* parent annotation (syntactic), lexicalization (semantic)

PCFG Example



s		γ	$\phi_s(\gamma)$
S	$\rightarrow$	NP VP	0.9
S	$\rightarrow$	S CONJ S	0.1
NP	$\rightarrow$	JJ JJ NNS	0.5
NP	$\rightarrow$	PRP	0.5
VP	$\rightarrow$	VP NP	0.4
VP	$\rightarrow$	VBP NP	0.3
VP	$\rightarrow$	VBG NP	0.3

Mathematical Definition

Formally, a PCFG is specified by the following:

- Σ, a set of terminal symbols (the words in the sentence)
- S, a set of nonterminal symbols (the syntactic categories)
- $\textsc{Root} \in S$, a designated nonterminal starting symbol
- ϕ, rule probabilities: $\phi = (\phi_s(\gamma) : s \in S, \gamma \in \Sigma \cup (S \times S))$, such that $\phi_s(\gamma) \geq 0$ and $\sum_\gamma \phi_s(\gamma) = 1$

Note the restriction on γ, that $\gamma \in \Sigma$ or $\gamma \in (S \times S)$. Such transitions make a PCFG in Chomsky normal form.

Mathematical Definition II

A parse tree has a set of nonterminal nodes N along with the corresponding symbols $s = (s_i \in S, i \in N)$. Now, let

- N_E denote nodes having one terminal child,
- N_B denote nodes having two nonterminal children

The tree structure is represented by

- $c = (c_j(i) : i \in N_B, j = 1, 2)$ for nonterminal nodes
- $x = (x_i : i \in N_E)$ for terminal nodes (the "yield")

The joint probability of a parse tree $z = (N, s, c)$ and x is then

$$p(x, z|\phi) = \prod_{i \in N_B} \phi_{s_i}(s_{c_1(i)}, s_{c_2(i)}) \prod_{i \in N_E} \phi_{s_i}(x_i)$$

HDP-PCFG: Generating the parse tree and its yield

So, given rule probabilities ϕ, for each syntactic category z consisting of ϕ_z^T (rule **t**ype parameters), ϕ_z^E (**e**mission parameters), and ϕ_z^B (**b**inary productions), we can generate a tree and its parse in the following way:

For each node i in the parse tree:

- $t_i \sim Mult(\phi_{z_i}^T)$
- if $t_i = \textsc{Emission}$, $x_i \sim Mult(\phi_{z_i}^E)$
- if $t_i = \textsc{BinaryProduction}$, $(z_{c_1(i)}, z_{c_2(i)}) \sim Mult(\phi_{z_i}^B)$

This Paper's Focus

- Traditionally, PCFGs are defined with a fixed, finite S and the parameters ϕ are fit using smoothed maximum likelihood
- This paper develops a nonparametric version of the PCFG that allows S to be countably infinite
- The model then performs posterior inference over S and the set of parse trees to find ϕ
- This model is called a Hierarchical Dirichlet Process PCFG (HDP-PCFG), and is described in the next section

HDP-PCFG: Generating the grammar

$\beta \sim GEM(\alpha)$

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

- $\phi_z^T \sim Dir(\alpha^T)$
- $\phi_z^E \sim Dir(\alpha^E)$
- $\phi_z^B \sim DP(\alpha^B, \beta\beta^\top)$

What do $\beta, \phi_z^{\{T,E,B\}}$, and $\beta\beta^\top$ look like?

HDP-PCFG: The whole process

$\beta \sim GEM(\alpha)$

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

- $\phi_z^T \sim Dir(\alpha^T)$
- $\phi_z^E \sim Dir(\alpha^E)$
- $\phi_z^B \sim DP(\alpha^B, \beta\beta^\top)$

For each node i in the parse tree:

- $t_i \sim Mult(\phi_{z_i}^T)$
- if $t_i = \textsc{Emission}$, $x_i \sim Mult(\phi_{z_i}^E)$
- if $t_i = \textsc{BinaryProduction}$, $(z_{c_1(i)}, z_{c_2(i)}) \sim Mult(\phi_{z_i}^B)$

Why is an HDP model advantageous?

- Allows the complexity of the grammar to grow as more training data is available; a DP prior penalizes the use of more symbols than are supported in the training data
- . . . which in turn means the level of sophistication of the grammar can adequately match the corpus
- Can you think of any disadvantages?

Hierarchical Dirichlet Process

- How is this a *Hierarchical* DP?
- How is this related to the HDP-HMM from Thursday?
- Why not a simpler model: for each symbol z, draw a distribution separately over left children $l_z \sim DP(\beta)$ and right children $r_z \sim DP(\beta)$?

Bayesian Inference for HDP-PCFG

- The authors chose to use structured mean-field approximation (variational inference with KL-divergence as a dissimilarity function)
- The random variables of interest are the parameters $\theta = (\beta, \phi)$, the parse tree z, and the observed yield x
- Thus the goal is to approximate the posterior $p(\theta, z|x)$. We want to find a $q(\theta, z)$ such that

$$\underset{q \in \mathcal{Q}}{\operatorname{argmin}}\ KL(q(\theta, z)||p(\theta, z|x))$$

where $\mathcal{Q}$ is a tractable subset of distributions.

Bayesian Inference for HDP-PCFG

The set of approximate distributions $\mathcal{Q}$ are defined to be those that factor as follows:

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

Additionally, other constraints are introduced:

- $q(\beta)$ is degenerate and truncated
- $q(\phi_z^{\{T,E,B\}})$ are Dirichlet distributions
- $q(z)$ is any multinomial distribution

Note that we have a fixed K. How does this affect the approximation?

Coordinate Ascent

- The optimization problem to find the best q is non-convex
- They use a coordinate ascent algorithm to find a local optimum
- Iteratively,
  1. Optimize $q(z)$, keeping $q(\phi)$ and $q(\beta)$ fixed
  2. Optimize $q(\phi)$, keeping $q(z)$ and $q(\beta)$ fixed
  3. Optimize $q(\beta)$, keeping $q(z)$ and $q(\phi)$ fixed

Prediction

We want to parse a new sentence with the induced grammar.

The prediction is given by

$$z^*_{new} = \underset{z_{new}}{\text{argmax}}\ E_{p(\theta,z|x)} p(z_{new}|\theta, x_{new})$$

Synthetic Experiment

- From a grammar of 15 rules, 1000 sentences were sampled
- With latent symbols, PCFG recovered a grammar of 150 active rules
- HDP-PCFG yielded a grammar of 25 active rules

Sample Grammar Inductions

True Grammar

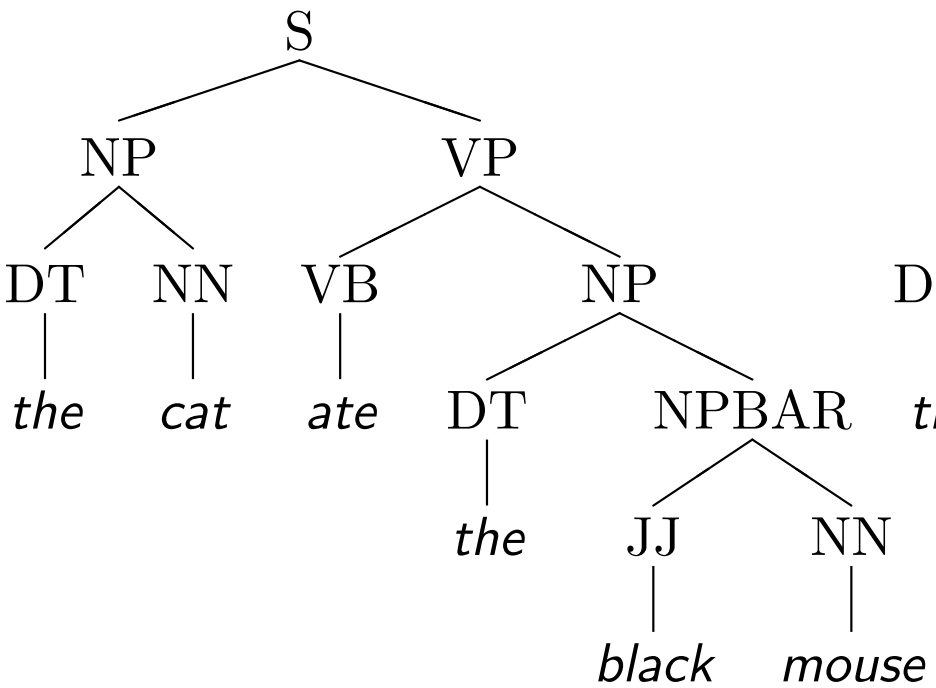
S	→	NP VP	1.0
NP	→	DT NN	0.5
NP	→	DT NPBAR	0.5
NPBAR	→	JJ NN	0.5
NPBAR	→	JJ NBPAR	0.5
VP	→	VB NP	1.0
DT	→	the	0.5
DT	→	a	0.5
JJ	→	big	0.5
JJ	→	black	0.5
NN	→	mouse	0.33
NN	→	cat	0.33
NN	→	dog	0.33
VB	→	chased	0.5
VB	→	ate	0.5

Induced Grammar

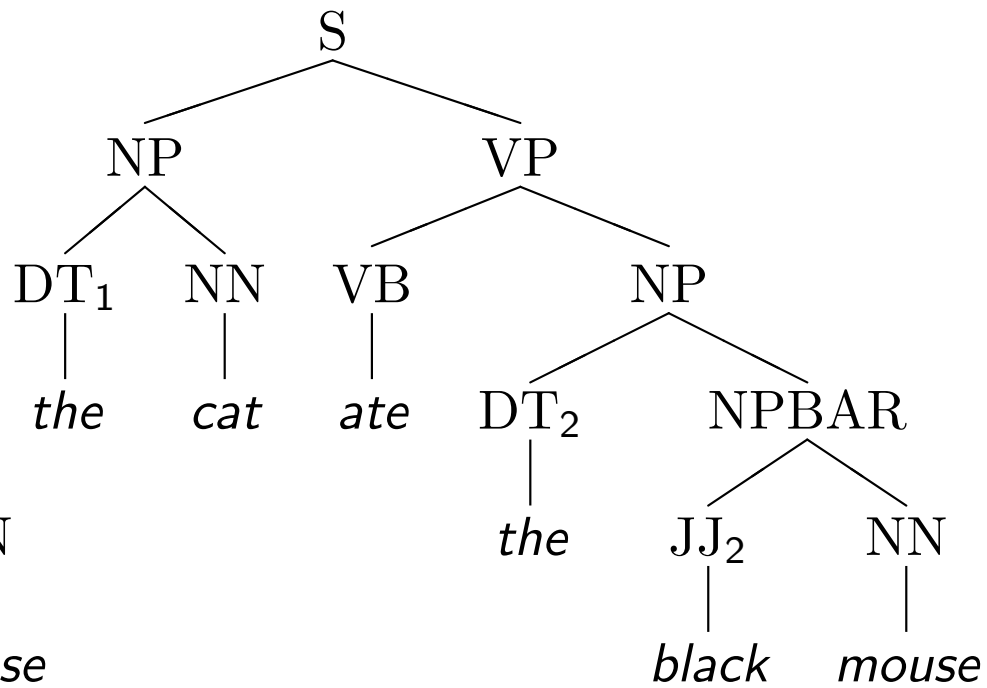
S	→	NP VP	1.0
NP	→	DT NN	0.5
NP	→	DT NPBAR	0.5
VP	→	VB NP	1.0
NPBAR	→	JJ_2 NPBAR	0.11
NPBAR	→	JJ_1 NPBAR	0.36
NPBAR	→	JJ-big NN	0.07
NPBAR	→	JJ_2 NN	0.42
NPBAR	→	JJ_2 NN-$\{cat, dog\}$	0.02
DT	→	a	0.5
DT	→	the	0.5
JJ_1	→	big	0.52
JJ_1	→	black	0.48
JJ-big	→	big	1.0
JJ_2	→	black	0.59
JJ_2	→	big	0.41
NN	→	mouse	0.35
NN	→	dog	0.32
NN	→	cat	0.33
NN-$\{cat, dog\}$	→	cat	0.47
NN-$\{cat, dog\}$	→	dog	0.53
VB	→	chased	0.49
VB	→	ate	0.51

Sample Parse Trees

True Parse



Induced Parse



HDP-PCFG-GR

- We want to use treebanks as labeled data
- Treebank symbols (node types) are coarse
  - Noun phrase can be subject or object and have a different behavior
- We need to model subsymbols
  - NP_{subj}, NP_{obj}, etc
- Each node is now a combination of a symbol and a subsymbol

The Generative Process

for each symbol $s \in S$

$\beta_s \sim GEM(\alpha)$

for each subsymbol $z \in \{1, 2, ...\}$

$\phi^T_{sz} \sim Dir(\alpha^T)$
$\phi^E_{sz} \sim Dir(\alpha^E(s))$
$\phi^u_{sz} \sim Dir(\alpha^u(s))$
for each child symbol $s' \in S$

$\phi^U_{szs'} \sim DP(\alpha^U, \beta_{s'})$

$\phi^b_{sz} \sim Dir(\alpha^b(s))$
for each pair of child symbols $(s', s'') \in S \times S$

$\phi^B_{szs's''} \sim DP(\alpha^B, \beta_{s'}, \beta^T_{s''})$

The Generative Process II

for each node i in the parse tree

$$t_i \sim Mult(\phi^T_{s_i,z_i})$$

if $t_i = \textsc{Emission}$

$$x_i \sim Mult(\phi^E_{s_i,z_i})$$

if $t_i = \textsc{UnaryProduction}$

$$s_{c_1(i)} \sim Mult(\phi^u_{s_i,z_i})$$
$$z_{c_1(i)} \sim Mult(\phi^U_{s_i,z_i,s_{c_1(i)}})$$

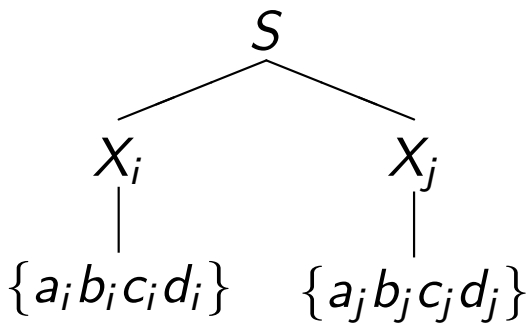
if $t_i = \textsc{BinaryProduction}$

$$(s_{c_1(i)}, s_{c_2(i)}) \sim Mult(\phi_{s_i,z_i})$$
$$(z_{c_1(i)}, z_{c_2(i)}) \sim Mult(\phi^B_{s_i,z_i,s_{c_2(i)}})$$

Synthetic Refinement Experiment

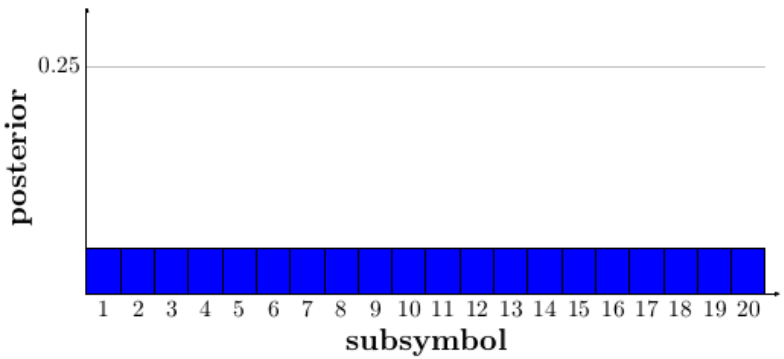
- 2000 trees were constructed and X_i was replaced by X
- 20 subsymbols were used for both S and X

$$S \to X_1X_1|X_2X_2|X_3X_3|X_4X_4$$
$$X_1 \to a_1|b_1|c_1|d_1$$
$$X_2 \to a_2|b_2|c_2|d_2$$
$$X_3 \to a_3|b_3|c_3|d_3$$
$$X_4 \to a_4|b_4|c_4|d_4$$

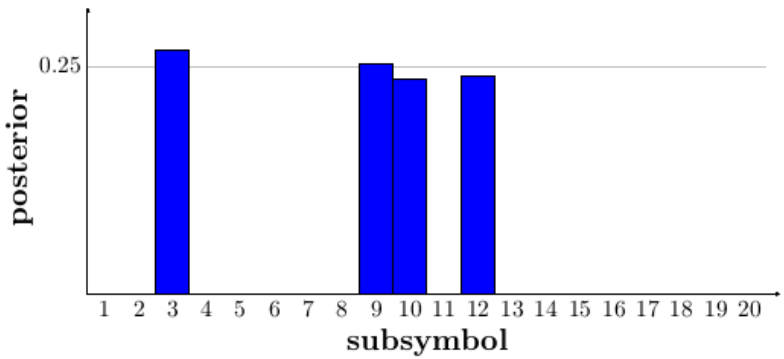


Results

- PCFG used all 20 subsymbols of both S and X
- HDP-PCFG-GR used only 4 subsymbols of X and one from S



(a) PCFG-GR



(b) HDP-PCFG-GR

Real Dataset Experiment I

HDP-PCFG-GR trained on section 2 and tested on section 22 of the Penn Treebank

K	PCFG-GR		PCFG-GR (smoothed)		HDP-PCFG-GR	
	F_1	Size	F_1	Size	F_1	Size
1	60.47	2558	60.36	2597	60.50	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	78.80	160403	78.94	86386
16	66.99	19616	79.20	261444	78.24	131377
20	64.44	27593	79.27	369699	77.81	202767

Real Dataset Experiment II

They experimented with the standard parsing setting by training on sections 2-21. HDP-PCFG-GR gave comparable performance with the smaller grammar

K	PCFG-GR		HDP-PCFG-GR	
	F_1	Size	F_1	Size
16	88.36	706157	87.08	428375