Bayesian Inference for PCFGs via Markov chain Monte Carlo

Kevin Morenski

Outline

Bayesian Inference for PCFGs

Dirichlet Priors

Markov chain Monte Carlo

A Gibbs Sampler for $\mathrm{P}(\mathbf{t}, \theta | \mathbf{w}, \alpha)$

Introduction

Authors' Contributions

- A component-wise Gibbs sampler that: (*i*) draws parse trees conditioned on the current parameter values, and then (*ii*) samples the parameters conditioned on the current set of parse trees.

- A component-wise Hastings sampler that "collapses" the probalistic model and integrates over the rule probabilities of the PCFG, thereby facilitating a more rapid convergence.

Both algorithms use an efficient dynamic programming technique to sample parse trees[1].

Background

Context-Free Grammars

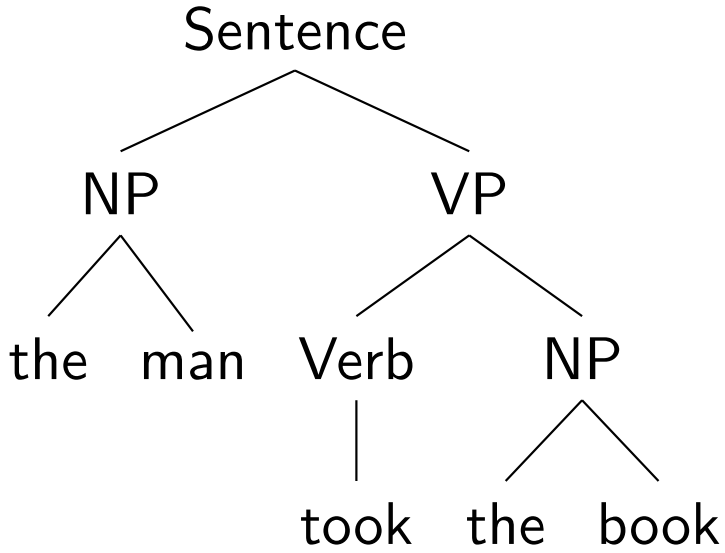


Figure 1: The first context-free grammar parse tree[2].

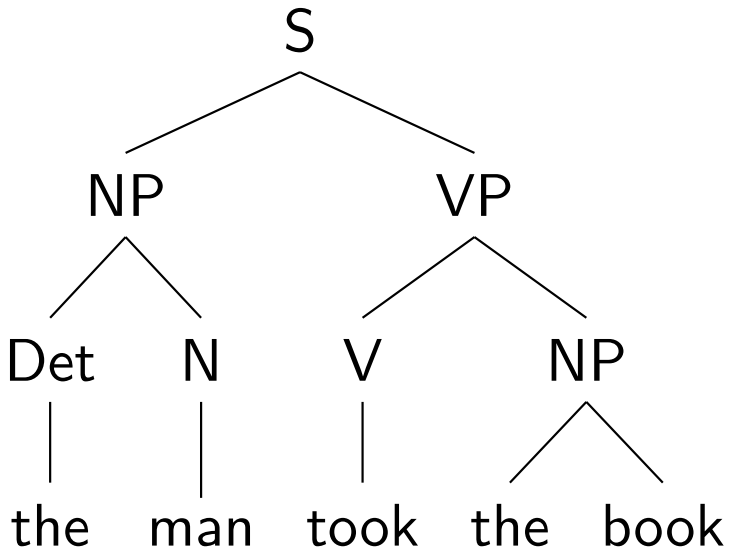
Definition

A context-free grammar consists of: (*i*) a set of linguistic **rules** that regulate the syntactically-valid groupings and orderings of a language's symbols, and (*ii*) a **lexicon** of words and symbols.

$S \rightarrow NP\ VP$	$V \rightarrow bought$
$NP \rightarrow Det\ N$	$V \rightarrow stole$
$VP \rightarrow V\ NP$	$P \rightarrow from$
$VP \rightarrow V\ PP$	$P \rightarrow to$
$PP \rightarrow P\ NP$	$P \rightarrow with$
$Det \rightarrow a$	$N \rightarrow man$
$Det \rightarrow the$	$N \rightarrow friend$
$V \rightarrow took$	$N \rightarrow store$

Table 1: The rules, which map **non-terminal** symbols to other non-terminal symbols (or combinations thereof), and the lexicon, which maps non-terminal symbols to **terminal** symbols (*i.e.*, words in the language).

Examples



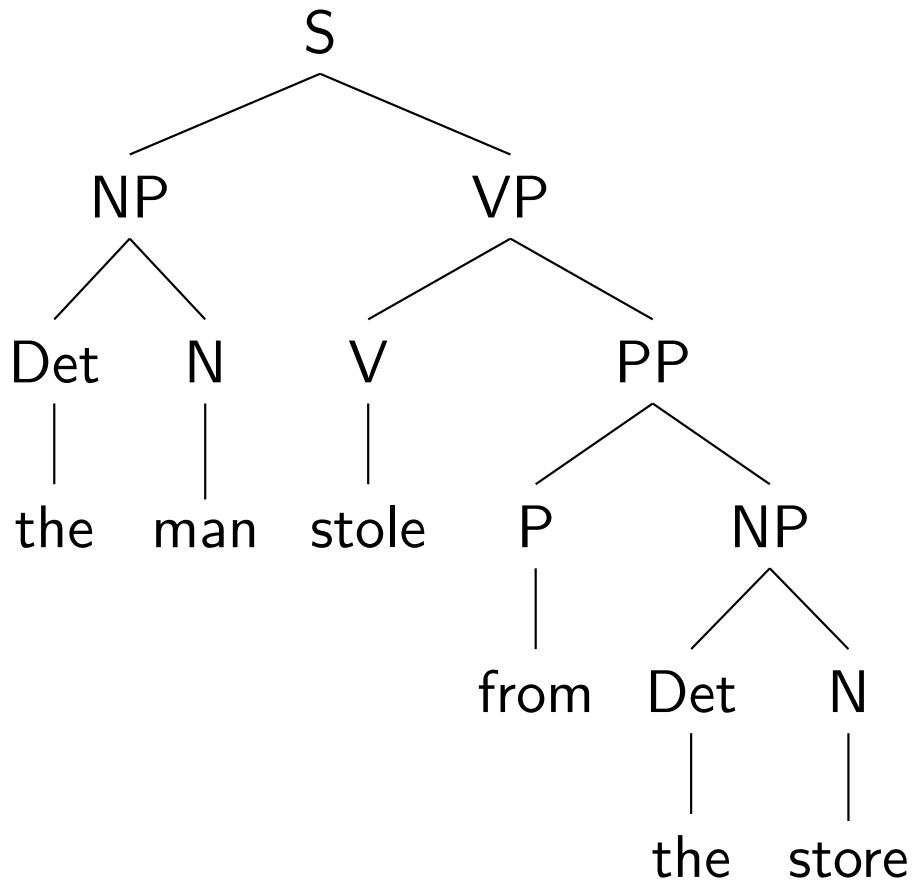


Figure 2: Syntactically and semantically valid.

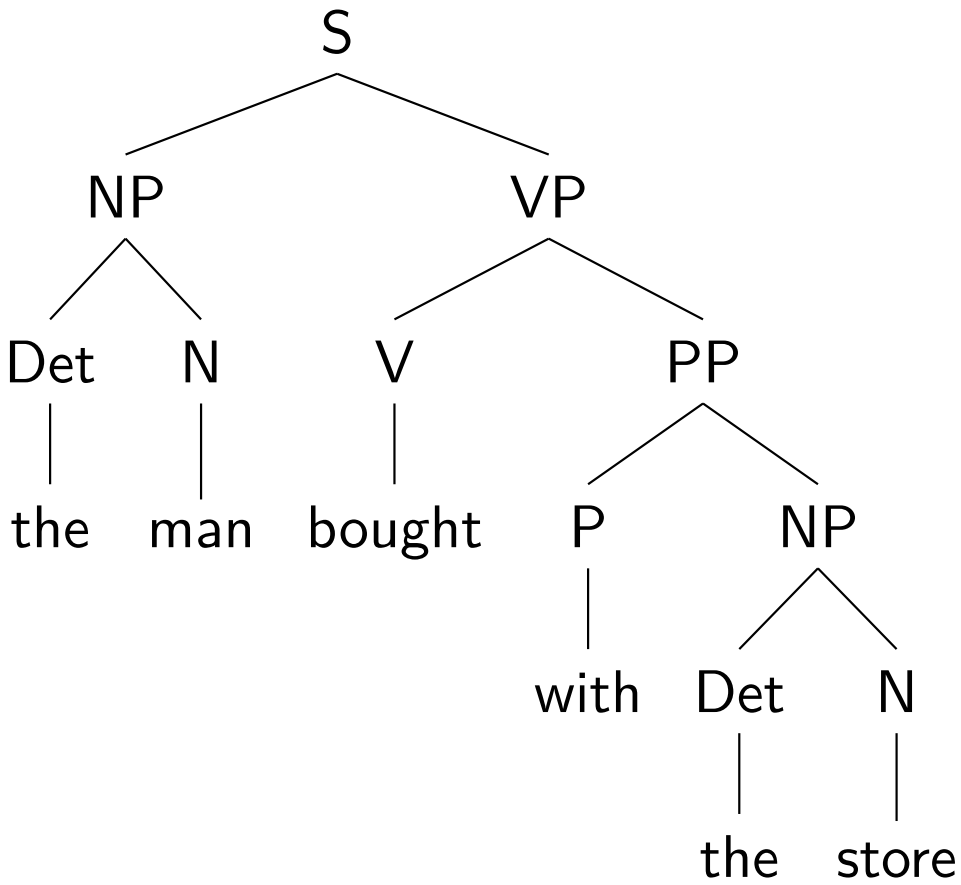


Figure 3: Syntactically, but not semantically, valid.

Thinking of CFGs

A CFG can be thought of in two ways[3]:

1. as a device for **generating** sentences (*viz.*, reading the "$\rightarrow$" as "rewrite the symbol on the left with the string of symbols on the right")

2. as a device for **assigning structure** to a given sentence

Parse trees represent derivations of the string of words generated by a CFG. The formal language defined by a CFG is the set of strings that are derivable from the designated **start symbol**, which each grammar must have.

Formal Definition

N	a set of **non-terminal symbols** (or **variables**)
Σ	a set of **terminal symbols** (disjoint from N)
R	a set of **rules** or productions, each of them in the form $A \to \beta$, where A is a non-terminal and β is a string of symbols from the infinite set of strings $(\Sigma \cup N)$.
S	a designated **start symbol**

Table 2: The four parameters that define a context-free grammar, G.

Probabilistic Context Free Grammars

A *Probabilistic Context Free Grammar* is a probalistic augmentation of context-free grammars, wherein each rule is associated with a probability[3]. The difference between a PCFG and CFG is that each rule R from Table 2 is augmented with a conditional probability: $A \rightarrow \beta[p]$, where p expresses the probability that the given non-terminal A will be expanded to the sequence β. That is to say, p is the conditional probability of a given expansion β given the left-hand-side (LHS) non-terminal A.

Thus, if we consider all possible expansions of a non-terminal, the sum of their probabilities must be 1:

$$\sum_{\beta} P(A \rightarrow \beta) = 1$$

Ways to Learn PCFG Rule Probabilities

The simplest way is to use a treebank (a corpus of sentences that have already been parsed and labeled). Maximum likelihood estimation can be used to compute the probability of each expansion of a non-terminal by counting the occurrences of that expansion and normalizing:

$$P(\alpha \to \beta|\alpha) = \frac{\text{Count}(\alpha \to \beta)}{\sum_{\gamma} \text{Count}(\alpha \to \gamma)} = \frac{\text{Count}(\alpha \to \beta)}{\text{Count}(\alpha \to \gamma)}$$

Since most sentences are ambiguous, which is to say have multiple syntactically-valid parses, we need to keep a separate count for each parse of a sentence and weight each of these partial counts with the probability of the parse in which it appears.

However, to get the parse probabilities necessary to weight the rules, we need to have a probabilistic parser. Intuitively, we can resolve this by incrementally improving our estimates in the following manner:

1. Begin with a parser with uniformly-distributed parse probabilities
2. Parse the sentence
3. Compute a probability for each parse
4. Use these probabilities to weight the counts
5. Re-estimate the rule probabilities
6. Repeat until the probabilities converge to those calculated via MLE or until the change in probabilities is less than or equal to some small value, ϵ.

The standard approach for computing this solution is called the **inside-outside** algortihm, which is a generalization of the **forward-backward** algorithm. Like the forward-backward algorithm, the inside-outside algorithm is a special case of the **expectation-maximization** algorithm[3].

The Expectation-Maximization Algorithm

EM iterates between taking an expectation and then maximizing in order estimate the MLE, $\hat{\theta}$.

Suppose we have data Y whose density $f(y;\theta)$ leads to a log-likelihood that's difficult to maximize. But suppose we can find another random variable, Z, such that $f(y;\theta) = \int f(y,z;\theta)dz$ and such that the likelihood based on $f(y,z;\theta)$ is easy to maximize. In other words, the model of interest is the marginal of a model with a simpler likelihood. In this case, we call Y the observed data and Z the hidden (or latent or missing) data.

Conceptually, EM works by filling in the missing data, maximizing the log-likelihood, and iterating.

The EM Algorithm:

1. Pick a starting value θ^0
2. For $j = 1, 2, 3, \ldots$, repeat the next steps until convergence:
3. (The E-step): Calculate

$$J(\theta|\theta^j) = \mathbb{E}_{\theta^j}\left(\log \frac{f(Y^n, Z^n; \theta)}{f(Y^n, Z^n; \theta^j)} \middle| Y^n = y^n\right)$$

   The expectation is over the missing data Z^n treating θ^i and the observed data Y^n as fixed.

4. Find θ^{j+1} to maximize $J(\theta|\theta^j)$.

How does the EM algorithm always increase the likelihood? In other words, how do we prove that $\mathcal{L}(\theta^{j+1}) \geq \mathcal{L}(\theta^j)$?

Note that

$$\begin{aligned} J(\theta^{j+1}|\theta) &= \mathbb{E}_{\theta^j}\left(\log \frac{f(Y^n, Z^n; \theta^{j+1})}{f(Y^n, Z^n; \theta^j)} \middle| Y^n = y^n\right) \\ &= \log \frac{f(y^n; \theta^{j+1})}{f(y^n; \theta^j)} + \mathbb{E}_{\theta^j}\left(\log \frac{f(Z^n|Y^n; \theta^{j+1})}{f(Z^n|Y^n; \theta^j)} \middle| Y^n = y^n\right) \end{aligned}$$

This leads to the conclusion that

$$\begin{aligned}\frac{\mathcal{L}(\theta^{j+1})}{\mathcal{L}(\theta^{j})} &= \log \frac{f(y^n;\theta^{j+1})}{f(y^n;\theta^{j})} \\ &= J(\theta^{j+1}|\theta j) - \mathbb{E}_{\theta^j}\left(\log \frac{f(Z^n|Y^n;\theta^{j+1})}{f(Z^n|Y^n;\theta^{j})}\middle| Y^n = y^n\right) \\ &= J(\theta^{j+1}|\theta j) + K(f_j, j_{j+1})\end{aligned}$$

where:

$$\begin{aligned}f_j &= f(y^n;\theta^j) \\ f_{j+1} &= f(y^n;\theta^{j+1})\end{aligned}$$

and $K(f, g) = \int \log(f(x)/g(x))f(x)dx$ is the Kullback-Leibler divergence. Now, θ^{j+1} was chosen to maximize $J(\theta|\theta^j)$, and therefore $J(\theta^{j+1}|\theta j) \geq J(\theta^j|\theta^j) = 0$. By the properties of KL divergence, $K(f_j, f_j + 1) \geq 0$. Therefore, $\mathcal{L}(\theta^{j+1}) \geq \mathcal{L}(\theta^j)$ as claimed[4].

Bayesian Inference for PCFGs

The goal is to infer the rule probabilities θ that best describe the corpus of strings $\mathbf{w} = (w_1, \ldots, w_n)$, where each w_i is a string of terminals generated by a known CFG, G. We can apply Bayes' theorem to obtain:

$$\mathrm{P}(\theta|\mathbf{w}) \propto \mathrm{P}_G(\mathbf{w}|\theta)\mathrm{P}(\theta)$$

$$\mathrm{P}(\mathbf{w}|\theta) = \prod_{i=1}^{n} \mathrm{P}_G(w_i|\theta)$$

Using $\mathbf{t}$ to denote a sequence of parse trees for $\mathbf{w}$, we can compute the joint posterior distribution over $\mathbf{t}$ and θ and marginalize over $\mathbf{t}$, with $\mathrm{P}(\theta|\mathbf{w}) = \sum_{\mathbf{t}} \mathrm{P}(\mathbf{t}, \theta|\mathbf{w})$. The joint posterior distribution on $\mathbf{t}$ is given by:

$$\begin{aligned}\mathrm{P}(\mathbf{t}, \theta|\mathbf{w}) &\propto \mathrm{P}(\mathbf{w}|\mathbf{t})\mathrm{P}(\mathbf{t}|\theta)\mathrm{P}(\theta) \\ &= \left(\prod_{i=1}^{n} \mathrm{P}(w_i|t_i)\mathrm{P}(t_i|\theta)\right)\mathrm{P}(\theta)\end{aligned}$$

with $\mathrm{P}(w_i|t_i) = 1$ if $y(t_i) = w_i$ and 0 otherwise.

Dirichlet Priors

Because we need a prior on θ before we can compute the posterior distribution, we take $\mathrm{P}(\theta)$ to be a product of Dirichlet distributions, with one distribution for each non-terminal $A \in N$.

The prior is parameterized by a positive real-valued vector α indexed by rules R such that each rule probability $\theta_{A \to \beta}$ has a corresponding Dirichlet parameter $\alpha_{A \to \beta}$.

The Dirichlet prior $\mathrm{P}(\theta|\alpha)$ is:

$$\mathrm{P}_D(\theta|\alpha) = \prod_{A \in N} \mathrm{P}_D(\theta_A|\alpha_A), \text{ where}$$

$$\mathrm{P}_D(\theta_A|\alpha_A) = \frac{1}{C(\alpha_A)} \prod_{r \in R_A} \theta_r^{\alpha_r - 1} \text{ and}$$

$$C(\alpha_A) = \frac{\prod_{r \in R_A} \Gamma(\alpha_r)}{\Gamma(\sum_{r \in R_A} \alpha_r)}$$

where Γ is the generalized factorial function and $C(\alpha)$ is the normalization constant that does not depend on θ_A.

Because Dirichlet priors are conjugate to the distribution over trees defined by a PCFG, the posterior distribution θ on a given set of parse trees, $\mathrm{P}(\theta|\mathbf{t}, \alpha)$, is also a Dirichlet distribution. Thus,

$$\begin{aligned}\mathrm{P}_G(\theta|\mathrm{t}, \alpha) &\propto \mathrm{P}_G(\mathbf{t}|\theta)\mathrm{P}_D(\theta|\alpha) \\ &\propto \left(\prod_{r\in R} \theta_r^{f_r\mathbf{t}}\right)\left(\prod_{r\in R} \theta_r^{\alpha_r - 1}\right) \\ &= \prod_{r\in R} \theta_r^{f_r(\mathbf{t})+\alpha_r-1}\end{aligned}$$

which is a Dirichlet distribution with parameters $f(\mathbf{t}) + \alpha$, where $f(\mathbf{t})$ is the vector of rule counts in $\mathbf{t}$ indexed by $r \in R$.

This allows us to write:

$$\mathrm{P}_G(\theta|\mathbf{t},\alpha) = \mathrm{P}_D(\theta|f(\mathbf{t})+\alpha)$$

which makes it clear that the production counts combine with the parameters of the prior.

Markov chain Monte Carlo

Computing even a single choice of $\mathbf{t}$ and θ is intractable, because evaluating the normalizing constant requires a summation over all possible parses for the entire corpus and all sets of rule probabilities. Thus, sampling algorithms are an appropriate response.

MCMC algorithms construct a Markov chain that samples states $s \in \mathcal{S}$. The approach is to begin in an initial state s_0, sample s_1 from $\mathrm{P}(s'|s_1)$, then sample s_2 from $\mathrm{P}(s'|s_1)$, and so on, with the probability that the Markov chain is in a particular state $\mathrm{P}(s_i)$ converging to $\pi(s_i)$ as $i \to \infty$. Here, $\mathrm{P}(s'|s)$ are transition probabilities guaranteed to converge to the desired distribution $\pi(s)$, *i.e.*, the posterior distribution:

$$\mathbb{E}_\pi[\theta] \approx \tfrac{1}{l} \sum_{i=1}^{l} \theta_i$$

A Gibbs Sampler for $\mathrm{P}(\mathbf{t}, \theta | \mathbf{w}, \alpha)$

Sampling each component of a Markov chain's state conditioned on the current value of all other variables is the basic method of computing transition probabilities using the Gibbs sampler. Here, this means alternating between sampling from two distributions:

$$\mathrm{P}(\mathbf{t}|\theta, \mathbf{w}, \alpha) = \prod_{i=1}^{n} \mathrm{P}(t_i|w_i, \theta), \text{ and}$$

$$\begin{aligned}\mathrm{P}(\theta|\mathbf{t}, \mathbf{w}, \alpha) &= \mathrm{P}_D(\theta|f(\mathbf{t}) + \alpha) \\ &= \prod_{A \in N} \mathrm{P}_D(\theta_A|f_A(\mathbf{t} + \alpha_A))\end{aligned}$$

This is reminiscent of the EM algorithm, with the E-step replaced by sampling $\mathbf{t}$ and the M-step replaced by sampling θ.

References

[1] *Bayesian Inference for PCFGs via Markov chain Monte Carlo*. Association for Computational Linguistics, 2007.

[2] N. Chomsky. Three models for the description of language. *IRE Transactions on Information Theory*, pages 113–124, 1956.

[3] D. Jurafsky and J. H. Martin. *Speech and Language Processing: An introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Pearson Education, Inc., 2009.

[4] L. Wasserman. *All of Statistics*. Springer, 2004.

Questions?