A Tree Sampler for Bounded Context-Free Languages

Anonymous Author(s)

Abstract

In the following paper, we present a simple method for sampling trees with and without replacement from bounded context-free languages (BCFLs). A BCFL is a context-free language (CFL) corresponding to finite-length strings with holes, which can be completed by valid terminals. We do so by defining an algebraic datatype that compactly represents candidate parse trees for an initial string. Once constructed, sampling trees is a straightforward matter of sampling integers uniformly without a replacement from a finite range.

1 Introduction

A CFG is a quadruple consisting of terminals (Σ), nonterminals (V), productions ($P\colon V \rightarrow (V \mid \Sigma)^*$), and a start symbol, ($S$). It is a well-known fact that every CFG is reducible to *Chomsky Normal Form*, $P'\colon V \rightarrow (V^2 \mid \Sigma)$, in which every production takes one of two forms, either $w \rightarrow xz$, or $w \rightarrow t$, where $w, x, z : V$ and $t : \Sigma$. For example, the CFG, $P := \{S \rightarrow SS \mid (S) \mid ()\}$, corresponds to the CNF:

$$P' = \left\{\ S \rightarrow QR \mid SS \mid LR, \quad L \rightarrow (, \quad R \rightarrow), \quad Q \rightarrow LS\ \right\}$$

Given a CFG, $\mathcal{G}' : \langle \Sigma, V, P, S\rangle$ in CNF, we can construct a recognizer $R : \mathcal{G}' \rightarrow \Sigma^n \rightarrow \mathbb{B}$ for strings $\sigma : \Sigma^n$ as follows. Let 2^V be our domain, 0 be $\varnothing$, $\oplus$ be $\cup$, and $\otimes$ be defined as:

$$X \otimes Z := \left\{\ w \mid \langle x, z\rangle \in X \times Z, (w \rightarrow xz) \in P\ \right\} \tag{1}$$

If we define $\hat{\sigma}_r := \{w \mid (w \rightarrow \sigma_r) \in P\}$, then initialize $M^0_{r+1=c}(\mathcal{G}', e) := \hat{\sigma}_r$ and solve for the fixpoint $M_\infty = M + M^2$,

$$M^0 := \begin{pmatrix} \varnothing & \hat{\sigma}_1 & \varnothing & \cdots & \varnothing \\ \vdots & & \ddots & & \vdots \\ \vdots & & & \ddots & \varnothing \\ \vdots & & & & \hat{\sigma}_n \\ \varnothing & \cdots & \cdots & \cdots & \varnothing \end{pmatrix} \Rightarrow M_\infty = \begin{pmatrix} \varnothing & \hat{\sigma}_1 & \Lambda & \cdots & \Lambda^*_\sigma \\ \vdots & & \ddots & & \vdots \\ \vdots & & & \ddots & \Lambda \\ \vdots & & & & \hat{\sigma}_n \\ \varnothing & \cdots & \cdots & \cdots & \varnothing \end{pmatrix}$$

we obtain the recognizer, $R(\mathcal{G}', \sigma) := S \in \Lambda^*_\sigma? \Leftrightarrow \sigma \in \mathcal{L}(\mathcal{G})?$ This procedure is essentially the standard CYK algorithm in a linear algebraic notation [**?**]. We are now ready to define the sampling problem as follows:

Definition 1.1 (Completion)**.** Let $\underline{\Sigma} := \Sigma \cup \{\_\}$, where _ represents a hole. We denote $\sqsubseteq\colon \Sigma^n \times \underline{\Sigma}^n$ as the relation $\{\langle \sigma', \sigma\rangle \mid \sigma_i : \Sigma \implies \sigma'_i = \sigma_i\}$ and the set $\{\sigma' : \Sigma^+ \mid \sigma' \sqsubseteq \sigma\}$ as $\text{H}(\sigma)$. Given an initial string $\sigma : \underline{\Sigma}^+$ we want to sample parse trees generated by $\mathcal{G}$ corresponding to $\sigma' : \text{H}(\sigma) \cap \ell$.

$\text{H}(\sigma) \cap \ell$ is often a large-cardinality set, so we want a procedure which samples trees uniformly without replacement, without enumerating the whole set, parsing and shuffling it.

2 Method

We define an algebraic data type $\mathbb{T}_3 = (V \cup \Sigma) \rightharpoonup \mathbb{T}_2$ where $\mathbb{T}_2 = (V \cup \Sigma) \times (\mathbb{N} \rightharpoonup \mathbb{T}_2 \times \mathbb{T}_2)$[1]. Morally, we can think of $\mathbb{T}_2$ as an implicit set of possible trees sharing the same root, and $\mathbb{T}_3$ as a dictionary of possible $\mathbb{T}_2$ values indexed by possible roots, given by a specific CFG under a finite-length porous string. We construct $\hat{\sigma}_r = \Lambda(\sigma_r)$ as follows:

$$\Lambda(s : \underline{\Sigma}) \mapsto \begin{cases} \bigoplus_{s \in \Sigma} \Lambda(s) & \text{if } s \text{ is a hole,} \\ \left\{\mathbb{T}_2\big(w, \left[\langle \mathbb{T}_2(s), \mathbb{T}_2(\varepsilon)\rangle\right]\big) \mid (w \rightarrow s) \in P\right\} & \text{otherwise.} \end{cases}$$

We then compute the fixpoint M_∞ by redefining the operations $\oplus, \otimes : \mathbb{T}_3 \times \mathbb{T}_3 \rightarrow \mathbb{T}_3$ as follows:

$$X \oplus Z \mapsto \bigcup_{k \in \pi_1(X \cup Z)} \left\{k \Rightarrow \mathbb{T}_2(k, x \cup z) \mid x \in \pi_2(X \circ k), z \in \pi_2(Z \circ k)\right\}$$

$$X \otimes Z \mapsto \bigoplus_{(w \rightarrow xz) \in P} \left\{\mathbb{T}_2\Big(w, \left[\langle X \circ x, Z \circ z\rangle\right]\Big) \mid x \in \pi_1(X), z \in \pi_1(Z)\right\}$$

These operators group subtrees trees by their root nonterminal, then aggregate their children. Each Λ now becomes a dictionary indexed by the root nonterminal, which can then be sampled by obtaining $(\Lambda^*_\sigma \circ S) : \mathbb{T}_2$, then recursively choosing twins as we describe in § 2.1, or without replacement via enumeration as described in § 2.2.

2.1 Sampling trees with replacement

Given a probabilistic CFG whose productions indexed by each nonterminal are decorated with a probability vector $\mathbf{p}$ (this may be uniform in the non-probabilistic case), we define a tree sampler $\Gamma : \mathbb{T}_2 \rightsquigarrow \mathbb{T}$ which recursively samples children according to a Multinoulli distribution:

$$\Gamma(T) \mapsto \begin{cases} \text{Multi}\big(\texttt{children}(T), \mathbf{p}\big) & \text{if } T \text{ is a } \texttt{root} \\ \big\langle \Gamma\big(\pi_1(T)\big), \Gamma\big(\pi_2(T)\big)\big\rangle & \text{if } T \text{ is a } \texttt{child} \end{cases}$$

This is closely related to the generating function for the ordinary Boltzmann sampler from analytic combinatorics,

$$\Gamma C(x) \mapsto \begin{cases} \text{Bern}\left(\frac{A(x)}{A(x)+B(x)}\right) \rightarrow \Gamma A(x) \mid \Gamma B(x) & \text{if } \mathcal{C} = \mathcal{A} + \mathcal{B} \\ \big\langle \Gamma A(x), \Gamma B(x)\big\rangle & \text{if } \mathcal{C} = \mathcal{A} \times \mathcal{B} \end{cases}$$

however unlike Duchon et al. [**?**], our work does not depend on rejection to guarantee exact-size sampling, as all trees contained in $\mathbb{T}_2$ will necessarily be the same width.

[1]Given a $T : \mathbb{T}_2$, we may also refer to $\pi_1(T), \pi_2(T)$ as $\texttt{root}(T)$ and $\texttt{children}(T)$ respectively, where children are pairs of conjoined twins.

2.2 Sampling without replacement

The type $\mathbb{T}_2$ of all possible trees that can be generated by a CFG in Chomksy Normal Form corresponds to the fixpoints of the following recurrence:

$$L(p) = 1 + pL(p) \qquad\qquad P(a) = V + aL\big(V^2 P(a)^2\big)$$

Given a $\sigma : \underline{\Sigma}^+$, we construct $\mathbb{T}_2$ from the bottom-up, and sample from the top-down. Illustrated below is a partial $\mathbb{T}_2$, where red nodes are `roots` and blue nodes are `children`:

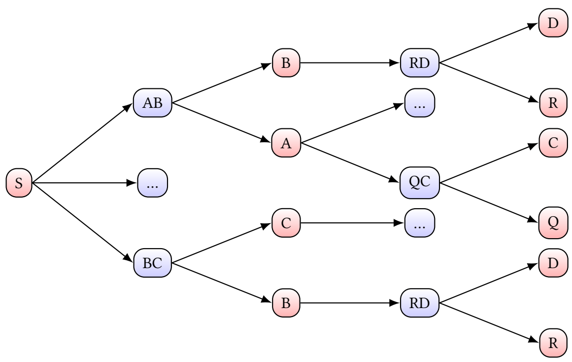

Figure 1. A partial $\mathbb{T}_2$ for the grammar with productions $P = \{S \to BC \mid \ldots \mid AB, B \to RD \mid \ldots, A \to QC \mid \ldots\}$.

The number of binary trees inhabiting a single instance of $\mathbb{T}_2$ is sensitive to the number of nonterminals and rule expansions in the grammar. To obtain the total number of trees with breadth n, we abstractly parse the initial string using the algebra defined in § 2, letting $T = \Lambda^*_{\underline{\sigma}} \circ S$, and compute the total number of trees using the recurrence:

$$|T : \mathbb{T}_2| \mapsto \begin{cases} 1 & \text{if } T \text{ is a leaf,} \\ \sum_{\langle T_1, T_2 \rangle \in \texttt{children}(T)} |T_1| \cdot |T_2| & \text{otherwise.} \end{cases}$$

To sample all trees in a given $T : \mathbb{T}_2$ uniformly without replacement, we then construct a modular pairing function $\varphi : \mathbb{T}_2 \to \mathbb{Z}_{|T|} \to \texttt{BTree}$, which is defined as follows:

$$\varphi(T : \mathbb{T}_2, i : \mathbb{Z}_{|T|}) \mapsto \begin{cases} \Big\langle \texttt{BTree}\big(\texttt{root}(T)\big), i \Big\rangle & \text{if } T \text{ is a leaf,} \\ \text{Let } b = |\texttt{children}(T)|, & \\ \quad q_1, r = \big\langle \lfloor \frac{i}{b} \rfloor, i \pmod b \big\rangle, & \\ \quad lb, rb = \texttt{children}[r], & \\ \quad T_1, q_2 = \varphi(lb, q_1), & \\ \quad T_2, q_3 = \varphi(rb, q_2) \text{ in} & \\ \Big\langle \texttt{BTree}\big(\texttt{root}(T), T_1, T_2\big), q_3 \Big\rangle & \text{otherwise.} \end{cases}$$

Then, instead of sampling trees, we can simply sample integers uniformly without replacement from $\mathbb{Z}_{|T|}$ using the construction defined in § 2, and lazily decode them into trees.

3 Prior work

Our work is closely related to Boltzmann sampling [**?**] in the case of sampling with replacement, but does not use rejection. Piantodosi [**?**] define a similar construction in the case of sampling without replacement, but it assumes the CFL is infinite and makes some additional assumptions about the ordering of productions. In the case when the initial string contains only holes, the problem of enumeration coincides with the Chomsky-Schützenberger enumeration theorem [**?**], which provides a constructive way to count finite-length words in unambiguous CFLs (i.e., $\ell \cap \Sigma^n$). Our construction is more general and works for any CFG and initial string, regardless of ambiguity, finitude, or production ordering.

4 Conclusion

We have presented a novel algorithm for sampling trees in bounded context-free languages with and without replacement. This technique has applications to code completion and program repair. In future work, we would like to construct a biased no-replacement rejection-free sampling procedure which prioritizes samples based on their likelihood according to a PCFG and extend our technique to the more general case of sampling from conjunctive languages.