Beyond Chomsky normal form: Extending strong learning algorithms for PCFGs

Alexander Clark ALEXSCLARK@GMAIL.COM *Department of Philosophy, King's College London*

Editors: Jane Chandlee, Rémi Eyraud, Jeffrey Heinz, Adam Jardine, and Menno van Zaanen

Abstract

We extend a recent consistent strong learning algorithm for a subclass of probabilistic context-free grammars in Chomsky normal form, (Clark and Fijalkow, 2020) to a much larger class of grammars by weakening the normal form constraints to allow for a richer class of derivation structures that are not necessarily binary branching, including among other possibilities unary branching trees which introduce some technical difficulties. By modifying the structural conditions appropriately we obtain an algorithm which is computationally efficient, and a consistent estimator for the class of grammars defined.

Keywords: context-free grammars; probabilistic learning; consistent estimator; strong learning.

1. Introduction

Recently Clark and Fijalkow (2020), (CF after this) presented an algorithm for strong learning of Probabilistic Context-Free Grammars (PCFGs). In this learning framework, which was initiated by Horning (1969), we have some class of PCFGs and the learner receives a sample of strings generated by some unknown PCFG in the class, and must return a PCFG where the underlying CFG is isomorphic to the target PCFG and where the parameters are close to the original, using a standard ϵ,δ convergence. This is *strong learning* where we have to learn the correct structure of the grammar and distribution over trees, not *weak learning* where all we want is the correct distribution over strings.

The class of PCFGs studied by CF consists of all PCFGs where the underlying CFG is in Chomsky Normal Form and which satisfy three structural conditions: being anchored, strictly upward monotonic (SUM) and locally unambiguous (LUA), which conditions we define below. Here we keep the three structural conditions more or less unchanged and weaken the CNF condition, allowing CFGs that have non binary branching productions, and treating the start symbol differently. This requires a weakening of the SUM condition.

There are several motivations for this approach: firstly while CNF is a normal form for the whole class of CFGs, in the sense that every context-free language can be defined by a grammar in CNF, in the strong learning setting the change in rule formats changes the classes of languages in significant respects, so this modification greatly enlarges the classes of grammars *and* languages that can be learned. Secondly, adding unary rules allow one to have grammars which have a hierarchy of nonterminals that can represent, partially, sets of feature structures which are essential for modeling syntax (ϕ-features in syntactic terms). Finally, while CFGs have a fairly simple type of rules, and CNF is a natural normal form, more linguistically adequate formalisms like Multiple Context-Free Grammars (Seki et al., 1991) have much more complex families of rules and it's crucial to understand how

to manipulate the sets of allowable rule types while maintaining learnability, and this is best started with a more tractable formalism. For example, Tree-Adjoining Grammars, considered as tree grammars (Kepser and Rogers, 2011) typically have productions whose right hand sides are quite large, and therefore a "minimal" normal form may be inadequate for the purposes of strong learning (Clark, 2021).

In the context of syntax, the binary branching restriction is fairly standard (Kayne, 1984), but CFGs are widely used in other problem domains with hierarchical structure, and in any event there are a number of phenomena in natural language which naturally require non binary branching structures.

2. Technical Preliminaries

A CFG is as usual a tuple $\langle \Sigma, V, S, P \rangle$ where Σ is a nonempty finite set of symbols, called the terminal symbols, V is a nonempty finite set of nonterminal symbols, disjoint from Σ, S is a distinguished element of V called the start symbol, and $P \subseteq V \times (V \cup \Sigma)^*$ is a finite set of productions, where a production π is written as $A \to \alpha$ for $A \in V$, $\alpha \in (V \cup \Sigma)^*$. Throughout we will use upper case letters $A, B, C, \ldots$ for nonterminals and lower case letters for terminals, and α, β, γ for strings of nonterminals and terminals.

A grammar is in Chomsky Normal Form (CNF) if the productions are a subset of $V \times (\Sigma \cup (V \setminus \{S\})^2)$. We exclude epsilon productions in this paper.

We will weaken the CNF condition later but in this paper we will maintain the restriction that productions either have a nonempty string of nonterminals or a single terminal on their right hand side, and that S occurs only on the left hand side of a production. So the productions are a subset of $V \times (\Sigma \cup (V \setminus \{S\})^+)$.

As is standard we will define $\Rightarrow_G$ as a string rewriting relation between strings over $(V \cup \Sigma)^*$, where $\beta A \gamma \Rightarrow_G \beta \alpha \gamma$ if $A \to \alpha \in P$, and $\Rightarrow^*$ as the reflexive transive closure of $\Rightarrow$. Using this semantics the language defined by the grammar is $\{w \in \Sigma^* \mid S \stackrel{*}{\Rightarrow} w\}$. A language here is just a subset of Σ^*, the set of all finite length strings of elements from Σ.

We will also give a tree based semantics for the CFG formalism, since PCFGs define a distribution over trees. We will treat the set of productions as a ranked alphabet where the rank of each production is the number of nonterminals on the right hand side of the production. We also add a set of context symbols $\{\Box_B \mid B \in V\}$, which have rank 0. The sort of a production $B \to \alpha$ and a context symbol $\Box_B$ is B. The sort of a tree is the sort of the label of the root of the tree. We define $\Omega(G)$ to be the set of trees over this ranked alphabet, including the context symbols, which satisfy the obvious compatibility constraint that if a production π labels a node, then the ith nonterminal on the right hand side of π is the sort of the ith child of that node. We can also write trees using prefix notation so $\pi(\tau_1, \tau_2)$ for example refers to the tree whose root is labeled with π a production of rank 2, with two child subtrees τ_1 and τ_2, and we will lift this to sets in the natural way: so $\pi(\Omega_1, \Omega_2) = \{\pi(\tau_1, \tau_2) \mid \tau_1 \in \Omega_1, \tau_2 \in \Omega_2\}$.

We define the yield of a tree, $y(\tau)$, to be the left to right sequence of leaves of τ, interpreting the context symbols $\Box_B$ as a distinguished gap symbol, $\Box$, not in Σ.

$$\begin{aligned} y(\pi(\tau_1, \ldots, \tau_k)) &= y(\tau_1) \cdot \cdots \cdot y(\tau_k) \text{ if rank of } \pi > 0 \\ y(\pi) &= a \text{ if } \pi = A \to a \\ y(\Box_B) &= \Box \end{aligned}$$

A context $l\Box r$ can be combined with a string u using the operation $\odot$ defined as $l\Box r \odot u = lur$. We need to count things in trees so we write $f(\pi; \tau)$ for the number of times the production π occurs in τ. The set $\Omega(G, B, Y)$ is the set of trees in $\Omega(G)$ of sort B where the yield is in Y for some set of strings Y. We define $\mathcal{L}(G, A) = \{y(\tau) \mid \tau \in \Omega(G, A, \Sigma^*)\}$, and the language defined by G is $\mathcal{L}(G, S)$.

The set $\Xi(G, B)$ is the set of derivation contexts of B. These are trees of sort S which have exactly one occurrence of the symbol $\Box_B$ in and no other occurrences of context symbols. These will have yields in $\Sigma^*\Box\Sigma^*$. We define $\mathcal{C}(G, A) = \{y(\tau) \mid \tau \in \Xi(G, A)\}$.

We can combine a context $\xi \in \Xi(G, B)$ with a tree $\tau \in \Omega(G, B)$ by replacing the occurrence of $\Box_B$ with τ, to get a tree in $\Omega(G, S)$, we write this as $\xi \oplus \tau$. Note that $y(\xi \oplus \tau) = y(\xi) \odot y(\tau)$.

2.1. Weighted CFGs

A stochastic language is a function $\mathbb{P}$ from $\Sigma^* \to [0, 1]$ such that $\sum_{w \in \Sigma^*} \mathbb{P}(w) = 1$. Given such a language the expected number of times a string $u \in \Sigma^*$ occurs is defined to be $\mathbb{E}(u) = \sum_{l,r \in \Sigma^*} \mathbb{P}(lur)$. We also can define the pointwise mutual information of a string $w = a_1 \ldots a_k$ to be

$$\text{PMI}(w) = \log \frac{\mathbb{E}(w)}{\prod_i \mathbb{E}(a_i)}$$

A Weighted CFG (WCFG) $G; \theta$ is a CFG together with a function θ from the set of productions to positive real numbers. The weight of a tree is defined as

$$w(\tau; \theta) = \prod_{\pi \in P} \theta(\pi)^{f(\pi; \tau)}$$

Note that for matching contexts ξ and trees τ, $w(\xi \oplus \tau) = w(\xi)w(\tau)$. The weight of a set of trees is the sum of the weights of the trees in the set. This may be infinite if the set is infinite. For a nonterminal A we define the inside and outside normalisation factors to be:

$$\begin{aligned} I(A) &= w(\Omega(G, A, \Sigma^*)) \\ O(A) &= w(\Xi(G, A)) \end{aligned}$$

If $I(S) = 1$, the grammar defines a probability distribution over $\Omega(G, S, \Sigma^*)$ and via that a stochastic language whose support is equal to $\mathcal{L}(G)$, where $\mathbb{P}(u) = w(\Omega(G, S, \{u\})$. With respect to this distribution we can define expectations of productions and nonterminals:

$$\mathbb{E}(\pi) = \sum_{\tau} w(\tau) f(\pi; \tau)$$

The expectation of a nonterminal is just the natural sum: $\mathbb{E}(A) = \sum_\alpha \mathbb{E}(A \to \alpha)$. The following identities follow:

$$\begin{aligned}\mathbb{E}(A) &= O(A)I(A)\\ \mathbb{E}(A \to B_1 \dots B_k) &= O(A)\theta(A \to B_1 \dots B_k)\prod_i I(B_i)\\ \mathbb{E}(A \to a) &= O(A)\theta(A \to a)\end{aligned}$$

There are as CF observe, two natural parameterisations: one is the standard PCFG model where $I(A) = 1$ for all nonterminals A, and the other is the bottom up model, where $O(A) = 1$ and $I(A) = \mathbb{E}(A)$ under which assumptions the parameters take the values:

$$\begin{aligned}\theta(A \to B_1 \dots B_k) &= \frac{\mathbb{E}(A \to B_1 \dots B_k)}{\prod_i \mathbb{E}(B_i)}\\ \theta(A \to a) &= \mathbb{E}(A \to a).\end{aligned}$$

We can convert between the two formalisms by numerically solving various systems of equations (Nederhof and Satta, 2009).

The advantage of the bottom up model, in the context of learning, is that this gives a well defined probability distribution over the contexts, rather than the yields; as we shall see, the learning method we use depends on modeling exactly these context distributions.

3. Distributional learning

We use a family of algorithms broadly called distributional learning, as surveyed in Clark and Yoshinaka (2016), which operate by modeling the Galois connection between derivation contexts and derivation yields: in the case of CFGs, the contexts of a nonterminal A are $l\Box r$ and the yields are simple substrings: note that by the context-free property of the grammar: $\mathcal{C}(G, A) \odot \mathcal{L}(G, A) \subseteq L$ for any nonterminal A. We can learn therefore by locating suitable sets of contexts and strings that satisfy this condition. For a set of strings W we define its distribution in a language to be:

$$W^\triangleright = \{l\Box r \mid l\Box r \odot W \subseteq L\}$$

In the case of a single string w we write $w^\triangleright$ for $\{w\}^\triangleright$. For a set of contexts C we define:

$$C^\triangleleft = \{w \mid C \odot w \subseteq L\}$$

Together the function $(\cdot)^{\triangleright\triangleleft}$ form a closure operator on the set of strings Σ^*. Note that L is always a closed set of strings in that $L = L^{\triangleright\triangleleft}$.

Various learning algorithms have been proposed using this lattice directly or indirectly: here we are concerned with so-called *primal* algorithms, where a nonterminal is defined by a string or set of strings: given a string w, we define a nonterminal that can or should generate all strings in $w^{\triangleright\triangleleft}$.

Given a stochastic language whose support is L, a string u with $\mathbb{E}(u) > 0$ defines a probability distribution over its contexts which we denote $\boldsymbol{u}$, whose support is $u^\triangleright$, where for $l\Box r$,

$$\boldsymbol{u}(l\Box r) = \frac{\mathbb{P}(lur)}{\mathbb{E}(u)}.$$

To measure similarity between two such distributions we need to use a generalization of the familiar Kullback-Liebler divergence called the Rényi divergence with $\alpha = \infty$, which is defined as:

$$\mathcal{R}_\infty(\boldsymbol{u}\|\boldsymbol{v}) = \sup_{l\Box r \in u^\triangleright} \log \frac{\boldsymbol{u}(l\Box r)}{\boldsymbol{v}(l\Box r)}$$

The need for this divergence rather than some other measure of distributional similarity is explained below in Subsection 4.1.

4. Structural Conditions and Grammar Format

We can now define the class of grammars that we will consider; we start by defining the normal form that we use: for any $r \geq 2$ the class of grammars $\mathcal{G}_r$ consists of all CFGs with productions of the type:

- $A \to B_1, \ldots B_k$ where $1 \leq k \leq r$, where $B_i \neq S$
- $A \to a$, where a is a terminal symbol, and $A \neq S$

For a grammar G we define $P(G, r)$ to be the (finite) set of all *possible* rules using the nonterminals and terminals in G that are in the class $\mathcal{G}_r$; the actual set of productions P is a subset of this. It's clear that (modulo the empty string) for any r each of these classes can define all context-free languages. $\mathcal{G}_2$ differs from CNF in two respects: we allow unary rules, and there are no lexical productions with the start symbol on the left hand side.

We now define three structural conditions: the most fundamental is *anchoring* (Stratos et al., 2016).

Definition 1 *A grammar is anchored if for every nonterminal A, apart from S, there is some terminal a such that $A \to a$ is in the grammar and no other production using a is in the grammar. In this case we will say that a is an anchor for the nonterminal A.*

If this assumption holds, then distributionally the terminal a will behave like the nonterminal A that it anchors in the sense that $a^\triangleright = \mathcal{C}(G, A)$. We can therefore infer something about the parameters of productions involving A from the distributional properties of a.

This definition differs from CF only in that we don't require the start symbol to be anchored: indeed it can't be anchored in this formalism since we cannot have a production like $S \to a$. However we will define a terminal symbol s not in Σ, and stipulate that $s^\triangleleft = \{\Box\}$; $\boldsymbol{s}$ is then the distribution that assigns probability 1 to this empty context. This allows us to treat the start and non-start symbols uniformly. If the terminal symbol a is an anchor we will write $[[a]]$ for the nonterminal symbol anchored by a. Note the important distinction between a production like $[[a]] \to b$, which is a rank 0 production with a terminal on the right hand side, and $[[a]] \to [[b]]$ which is a rank 1 production with a nonterminal on the right hand side. Conversely, an anchoring of a grammar is a function $\phi : V \to \Sigma \cup \{s\}$ such that $\phi(S) = s$ and $\phi(A)$ is an anchor for a for all other nonterminals; in other words it just replaces the nonterminal symbols with their anchors. We extend this to a function from $V \cup \Sigma$ by saying that ϕ is the identity on Σ, and then to $(V \cup \Sigma)^* \to \Sigma^*$ homomorphically.

Definition 2 *A production π in a grammar G, defining a language L, is locally unambiguous (*LUA*) if*

- *when π is $A \to a$, there is a string $w \in L$ that decomposes as $w = lar$ such that*

$$\Omega(G, S, \{w\}) = \Xi(G, A, \{l\square r\}) \oplus \pi$$

- *when π is of the form $A \to B_1, \ldots, B_k$ then there is a string $w \in L$ that decomposes as $w = lu_1 \ldots u_k r$ such that if we write $\Omega_i = \Omega(G, B_i, \{u_i\})$ then*

$$\Omega(G, S, \{w\}) = \Xi(G, A, \{l\square r\}) \oplus \pi(\Omega_1, \ldots, \Omega_k)$$

A grammar is LUA *iff every one of its productions is* LUA.

This condition is a weak bound on the degree of ambiguity of a grammar; it is satisfied if for example every production is used in the tree of an unambiguous sentence, or if the grammar is unambiguous, but also allows many ambiguous grammars, as long as they are, informally, not too ambiguous. This condition ensures that in the computation of the Rényi divergence between two context distributions there is a context which attains a minimal value which correctly isolates a particular production.

Now clearly we have a problem with the combination of the LUA condition and this grammar format. Suppose we have productions $A \to BC$ and $C \to DE$ and $A \to BDE$. Any occurrence of the production $A \to BDE$ can be replaced with occurrences of the other two: so $A \to BDE$ cannot be LUA. The problem here is that the production $A \to BDE$ is *redundant*, which we make precise below.

Definition 3 *Given an anchored grammar in $\mathcal{G}_r$ with anchoring ϕ, defining a language L, a production $A \to \alpha \in P(G, r)$ is valid if $\phi(A)^\triangleright \subseteq \phi(\alpha)^\triangleright$.*

CF use a condition *strict upward monotonicity* (SUM) which says that adding any new production to the grammar must increase the set of strings defined by the grammar. This is equivalent, in an anchored grammar, to saying that the grammar contains all valid rules. The identity production $A \to A$ is of course always valid and we need to exclude this explicitly either at the grammar format level or elsewhere.

Definition 4 *A production $A \to \alpha$ is redundant if there are two valid non-identity productions, $A \to \beta B \gamma$ and $B \to \delta$ such that $\alpha = \beta\delta\gamma$.*

We say that a grammar G is complete in a $\mathcal{G}_r$ if P is the set of all productions in $P(G, r)$ that are valid and not redundant.

Similar conditions are defined in Clark (2015). In such a grammar, every valid production is either in the grammar, or can be derived using other productions in the grammar. This replaces the SUM condition; it coincides on the class of CNF grammars, since adding any invalid production will generate additional strings; and there are no redundant productions in the CNF rule format.

Definition 5 *An anchored grammar in $\mathcal{G}_r$ is minimal if there is no other anchored grammar in $\mathcal{G}_r$ with fewer nonterminals that defines the same language.*

We can now define the classes of grammars that will be learnable with the algorithm we present in section 5.

Definition 6 *Let $\mathfrak{G}_r$ be the set of all CFGs in $\mathcal{G}_r$ which are anchored, minimal, complete and* LUA*. Let $\mathfrak{P}_r$ be the class of all PCFGs where the underlying CFG is in $\mathfrak{G}_r$ and which have finite expected length, and $\mathfrak{W}_r$ be the same class of WCFGs where the parameters are in bottom up form.*

We now prove some lemmas that will lead up to theorem 10. We start by showing that the anchors will be defined by a simple language theoretic criterion: this guarantees that all grammars in the class for the same language will have equivalent sets of nonterminals.

Definition 7 *A terminal a is* essential *in a language if there is a context $l\Box r \in a^{\triangleright}$ such that there is no terminal b such that $b^{\triangleright} \subset a^{\triangleright}$ and $l\Box r \in b^{\triangleright}$.*

Lemma 8 *If $G \in \mathfrak{G}_r$, then if a is essential then it is an anchor.*

Proof Let $R(a)$ be the set of anchors for all nonterminals such that $A \to a$ is in the grammar.

So $a^{\triangleright} = \bigcup_{b \in R(a)} b^{\triangleright}$. But a is essential so there is some $l\Box r \in a^{\triangleright}$ that is not in the distribution of any terminal with smaller distribution. And there is some $b \in R(a)$ such that $l\Box r \in b^{\triangleright}$. So $b^{\triangleright} = a^{\triangleright}$ and b is an anchor, so a is an anchor. ■

Lemma 9 *If $G \in \mathfrak{G}_r$, and G is minimal, then if a is an anchor it is essential.*

Proof We show that if a is an anchor for A and not essential then A would appear only on the right hand sides of unary rules: in which case we can remove it, by adding productions $B_i \to \alpha_j$ for all $B_i \to A$ and $A \to \alpha_j$; the resulting grammar is still anchored and has one less nonterminal.

Suppose we have a production $\pi = B \to \alpha A \beta$. By LUA there is a string w where every derivation of w is of the form $\Xi(G, A, \{l\Box r\}) \oplus \Omega(G, A, u)$, and where the bottom production of the context is π. Let $\tau = \xi_A \oplus \tau_A$ be one such derivation. Since a is not essential there is there is some b_i such that $l\Box r \in b_i^{\triangleright}$. Pick some nonterminal $[[c]]$ such that $[[c]] \stackrel{*}{\Rightarrow} b_i$ and $l\Box r \in c^{\triangleright}$. Let $\xi_c \in \Xi(G, [[c]], l\Box r)$; since $c^{\triangleright} \supseteq a^{\triangleright}$, we must have $C \stackrel{*}{\Rightarrow} A$, so let $\pi_1(\pi_2 \dots (\pi_k(\Box_A)))$ be some derivation of $C \stackrel{*}{\Rightarrow} A$. Then $\xi_b \oplus \pi_1(\pi_2 \dots (\pi_k(\tau_A)))$ is a derivation of w. This will violate LUA unless $\pi = \pi_k$, so π is unary. ■

Theorem 10 *Suppose G and G' are in $\mathfrak{G}_r$ and $\mathcal{L}(G) = \mathcal{L}(G')$: then they are isomorphic.*

Proof By previous lemmas, the anchors in both grammars must correspond to the set of essential terminals. Therefore there is a natural bijection ψ between the nonterminals given by $\psi(A) = A'$ iff $\mathcal{C}(G, A) = \mathcal{C}(G', A')$. Given this the sets of productions must be exactly the valid productions that are not redundant, and so the two grammars are isomorphic. ■

4.1. Parameters

It is straightforward to extend the parameter identities of CF to this more general case; the proofs are exactly the same as those in CF. In this section we use θ to refer to the bottom up parameters of a WCFG. Suppose we have a production $S \to A_1, \ldots A_k$. Let a_i be an anchor for A_i. Then the string $w = a_1 \ldots a_k$ is unambiguous, by LUA.

$$\mathbb{P}(w) = \theta(S \to A_1 \ldots A_k) \prod_i \theta(A \to a_i)$$

therefore $\log \theta(S \to A_1 \ldots A_k) = \log \mathbb{P}(w) - \sum_i \log \mathbb{E}(a_i)$, or using the s symbol, we can write this as

$$\log \theta(S \to A_1 \ldots A_k) = \mathrm{PMI}(w) - \mathcal{R}_\infty \left(\boldsymbol{s} \| \boldsymbol{w}\right)$$

Considering now productions with a non start nonterminal on the left hand side; for a lexical production the parameter is:

$$\log \theta(A \to b) = \log \mathbb{E}(b) - \mathcal{R}_\infty \left(\boldsymbol{a} \| \boldsymbol{b}\right)$$

For productions of the form $A \to B_1 \ldots B_k$, $k \geq 1$ assuming anchors $a, b_1, \ldots, b_k$, we have by the same process:

$$\log \theta(A \to B_1 \ldots B_k) = \mathrm{PMI}(b_1 \ldots b_k) - \mathcal{R}_\infty \left(\boldsymbol{a} \| \boldsymbol{b_1 \ldots b_k}\right)$$

The special case of unary rules $A \to B$, anchored by a and b respectively, reduces to the trivial equation

$$\log \theta(A \to B) = -\mathcal{R}_\infty \left(\boldsymbol{a} \| \boldsymbol{b}\right). \tag{1}$$

Since this sort of rule does not occur in the CF model, we will give a proof of this identity. First note that for any context $l \Box r$

$$\Omega(G, S, \{lbr\}) \supseteq \Xi(G, A, \{l \Box r\}) \oplus (A \to B) \oplus (B \to b)$$

Therefore

$$\mathbb{P}(lbr) \geq w(\Xi(G, A, \{l \Box r\}) \theta(A \to B) \theta(B \to b) \tag{2}$$

By the properties of the bottom up parameterisation, $\theta(B \to b) = \mathbb{E}(B \to b) = \mathbb{E}(b)$ since b is an anchor, and

$$w(\Xi(G, A, \{l \Box r\}) = \frac{\mathbb{P}(lar)}{\mathbb{E}(a)}$$

Rearranging eq. (2) using these two identities gives:

$$\theta(A \to B) \leq \frac{\mathbb{P}(lbr)\mathbb{E}(a)}{\mathbb{E}(b)\mathbb{P}(lar)} = \frac{\boldsymbol{b}(l \Box r)}{\boldsymbol{a}(l \Box r)}.$$

Since this is true for any context $l \Box r$, we can minimize over the contexts giving:

$$\theta(A \to B) \leq \min_{l \Box r} \frac{\boldsymbol{b}(l \Box r)}{\boldsymbol{a}(l \Box r)} = \exp(-\mathcal{R}_\infty \left(\boldsymbol{a} \| \boldsymbol{b}\right)).$$

The LUA condition then guarantees that this minimum is attained and so the inequality is in fact an equality giving eq. (1). The justification for the use of the Rényi divergence rather than any other measure of distributional similarity is given by these equations: other divergences such as the KL divergence will not satisfy these identities.

These identities all define the parameters of the grammar in terms of the properties of the distribution over strings. Therefore if two grammars in $\mathfrak{W}_r$ define the same distribution over strings, they must be isomorphic by theorem 10, and the parameters must also be the same. Therefore the same holds for $\mathfrak{P}_r$, since there is a bijection between $\mathfrak{P}_r$ and $\mathfrak{W}_r$.

5. Algorithm

Identifiability from strings says that $\mathfrak{P}_r$ is learnable from strings in a very abstract sense: we also need to show that it can be learned using an explicit and efficient computation. The algorithm takes as input a sample of N strings $w_1, \ldots, w_N$, each sampled from the distribution of strings defined by a PCFG, and a hyperparameter r that bounds the length of right hand sides. We write $n(w)$ for the number of times that $w_i = w$. As in CF we use the naive maximum likelihood estimators of $\mathbb{E}(u)$ and $\mathbb{P}(u)$ which we write as $\hat{\mathbb{E}}_N(u)$ and $\hat{\mathbb{P}}_N(u)$. We also define for a string $w = a_1 \ldots a_k$, the quantity $\widehat{\mathrm{PMI}}_N(w) = \log \frac{\hat{\mathbb{E}}(w)}{\prod_i \hat{\mathbb{E}}(a_i)}$ as the estimate of $\mathrm{PMI}(w)$ and

$$\hat{\mathcal{R}}_N\left(u \| v\right) = \log \frac{\hat{\mathbb{E}}(v)}{\hat{\mathbb{E}}(u)} \max_{r : n(lar) \geq \sqrt{N}} \frac{n(lur)}{n(lvr)}$$

as the estimate of the divergence.

In the initial phase we only want to estimate whether $\mathcal{R}_\infty\left(\boldsymbol{a} \| \boldsymbol{b}\right)$ is infinite or not, which in the finite sample case reduces to whether there is a $l \square r$ in the support of $a^\triangleright$ and not in $b^\triangleright$. We also need to test, given a finite set of $b_1, \ldots, b_k$ as we shall see whether there is a $l \square r$ in $a^\triangleright$ and not in $\bigcup_i (b_i^\triangleright)$. So we define the following boolean estimators $\hat{\mathcal{B}}_N\left(u \| v\right)$ which returns false if there is some $l \square r$ such that $n(lur) \geq \sqrt{N}$ and $n(lvr) = 0$, and is true otherwise. Intuitively this is true if $[[u]] \to [[v]]$ is a possible rule. If $u = s$, then this is equivalent to $n(v) > 0$. We also define for a set of strings V, $\hat{\mathcal{B}}_N\left(u \| V\right)$ which returns false if there is some $l \square r$ such that $n(lur) \geq \sqrt{N}$ and $n(lvr) = 0$ for all $v \in V$, and is true otherwise. This is used to test whether u is essential; with the right choice of V, this will be false iff u is essential.

Algorithmically we need two changes to the CF algorithm. First we need to identify a set of essential terminals that can serve as anchors: pseudocode is presented in algorithm 1. Secondly we need to construct the grammar excluding redundant nonterminals, presented in algorithm 2. The full algorithm then takes the input data; extracts the observed set of terminals; applies algorithm 1 to get a set of anchors and unary productions. Then it applies algorithm 2, which outputs a WCFG. We then convert this to a PCFG by using one iteration of the inside-outside algorithm (Eisner, 2016) on the input data. If the WCFG is divergent ($I(S) = \infty$) then the algorithm outputs an empty grammar.

The key difference from CF is in algorithm 1; we will prove this lemma:

Data: Terminals Σ, samples $w_1, \ldots, w_N$
Result: Set of anchors V, set of lexical productions P_L, unary productions P_U
$V = \{s\}$, $H = \Sigma$, $P_L = \emptyset$, $P_U = \emptyset$
while $H \neq \emptyset$ **do**
  `/* Find the terminals of minimal distribution in` H `*/`
  Construct the set of terminals $a \in H$ with no terminal $b \in H$ such that $\hat{\mathcal{B}}_N(b\|a)$ but not $\hat{\mathcal{B}}_N(a\|b)$
  Divide these into $\sim$-equivalence classes where $a \sim b$ if $\hat{\mathcal{B}}_N(a\|b)$ and $\hat{\mathcal{B}}_N(b\|a)$.
  for *each equivalence class* E **do**
    `/* These are equivalence classes of anchors */`
    Let a be the the most frequent terminal in E;
    Add a to V;
    for *each* $b \in E$ **do**
      add $[[a]] \to b$ to P_L and remove b from H.
    end
    for *each* $b \in V$ **do**
      if $\hat{\mathcal{B}}_N(b\|a)$ **then**
        Add $[[b]] \to [[a]]$ to P_U
      end
    end
  end
  for *each* a *in* H **do**
    let V_a be the set of all terminals $b \in V$ such that $\hat{\mathcal{B}}_N(b\|a)$.
    if $\hat{\mathcal{B}}_N(a\|V_a)$ **then**
      `/*` a `is not essential as` $a^{\triangleright} = \bigcup_{b \in V_a} b^{\triangleright}$ `*/`
      remove a from H
      for *each* $b \in V_a$ **do**
        add $[[b]] \to a$ to P_L.
      end
    end
  end
end
return V *and* P_U *and* P_L

Algorithm 1: Selecting anchors, unary productions and lexical productions.

Lemma 11 *For every grammar* $G_*; \theta_*$ *in* $\mathfrak{P}_r$*, for all* $\delta > 0$*, there is an* N *such that if algorithm 1 receives a sample of* $N > n$ *strings drawn independently from the target PCFG, then with probability at least* $1 - \delta$ *it will output a correct set of anchors for* G_**.*

Proof We write "n is large enough for X" to mean that n is big enough that if $N > n$, with probability at least $1 - \delta/3$, X holds. First let n be large enough that all terminals occur in the sample. Secondly let n be large enough that for all anchors a and for all other terminals b, and $\hat{\mathcal{B}}_N(b\|a)$ are both correct.

Write V_a for the set of all anchors b such that $b^{\triangleright} \subset a^{\triangleright}$. Finally, let n be large enough that for all anchors a with V_a nonempty, $\hat{\mathcal{B}}_N(a\|V_a)$ is correct. Given these assumptions, consider

the main while loop in algorithm 1. In the first iteration, the set constructed will consist of anchors of nonterminals that are not on the right hand side of any unary rules. These are all of course essential; since $\hat{\mathcal{B}}_N(a\|b)$ is correct for all of these, $a \sim b$ iff $a^{\triangleright} = b^{\triangleright}$. So at the end of the first iteration we will have a correct set of anchors for these nonterminals of minimal distribution. In subsequent iterations, we maintain the invariant that V always consists of essential terminals, no two of which are distributionally identical, and that H consist only of terminals a which have a production $A \to a$ for some A not in V. When this algorithm terminates therefore, V will consist of a set of essential terminals in bijection with the nonterminals of the target grammar. The chance that one of the assumptions fails is less than δ. ■

Formally the convergence of the algorithm is stated in the following theorem.

Theorem 12 *This algorithm has the property that for all grammars $G_*;\theta_*$ in $\mathfrak{P}_r$, for all $\epsilon, \delta > 0$ there is an n such that if the algorithm receives a sample of $N > n$ strings drawn independently from the target PCFG and hyperparameter r, then with probability at least $1-\delta$ it will output a grammar $G_N;\theta_N$ such that G_N is isomorphic to G_* and for all π under this isomorphism, $|\theta_*(\pi) = \theta_N(\pi)| < \epsilon$.*

Proof (Sketch) Given the correct set of nonterminals, for sufficiently large N with high probability all of the $\hat{\mathcal{R}}_N(a\|b_1 \dots b_k)$ will be finite iff $\mathcal{R}_\infty(\boldsymbol{a}\|\boldsymbol{b_1 \dots b_k})$ is finite. This suffices to establish that with high probability the output grammar will be isomorphic to the target grammar by theorem 10. Convergence of the parameters then follows by the parameter identities in section 4.1, and the consistency of the Rényi estimators. This establishes that the WCFG output by algorithm 2 converges to the WCFG equivalent of the target PCFG. To convert the parameters from the bottom up form to the top down form used in the PCFG, we use one iteration of the EM algorithm on the training data, and return the result; this is guaranteed to converge to the correct PCFG. ■

6. The classes of languages.

Some simple examples of grammars/languages that are in $\mathfrak{G}_3$ but not in $\mathfrak{G}_2$ are languages like $\{a^n b a^n \mid n \geq 0\}$, which has the natural ternary branching grammar: $S \to B$, $B \to ABA$, $A \to a$, $B \to b$. Similarly the languages $L_k = \{a^{kn} b a^{kn} \mid n \geq 0\}$, for any $k \geq 1$ are definable by $\mathfrak{G}_{2k+1}$ but not in any smaller class.

It's natural to ask whether all grammars in $\mathfrak{G}_r$ are also in $\mathfrak{G}_{r'}$ when $r < r'$. Clearly the anchoring condition is satisfied automatically, but the grammar may not be complete in the larger class as there may be valid productions of rank r' that are not redundant and if we add those productions we complete the grammar but the additional productions may then violate the LUA property. So for example consider the grammar G_2 with productions $S \to EG, S \to AF$, $E \to BC$ $G \to DF$ and anchoring rules $A \to a$, $\dots$, $G \to g$. This grammar is in $\mathfrak{G}_2$ and generates the finite language $\{af, eg, edf, bcg, bcdf\}$. The rank 3 production $A \to BCD$ is valid, but we cannot derive $A \stackrel{*}{\Rightarrow} BCD$. Within the class $\mathfrak{G}_3$ then, to make the grammar complete, we must add this production to get the grammar G_3. However in this case the grammar is no longer LUA, since that production is only used in

Data: Set of anchors V, set of unary productions P_U, and lexical productions P_L
Data: Hyperparameter r, sample $w_1, \ldots, w_N$
Construct a set of nonterminals $N = \{[[a]] \mid a \in V\}$
$P = \emptyset$
for *for every* $\pi = [[a]] \to [[b]]$ *in* P_U **do**
  if π *is not redundant in* P_U **then**
    Add π to P with parameter $\theta(\pi) = \exp(-\hat{\mathcal{R}}_N\left(a \| b\right))$.
  end
end
for *for every* $\pi = [[a]] \to b$ *in* P_L **do**
  if π *is not redundant in* $P_L \cup P_U$ **then**
    Add π to P with parameter $\theta(\pi) = \hat{\mathbb{E}}_N(b) \exp(-\hat{\mathcal{R}}_N\left(a \| b\right))$.
  end
end
for $k = 2$ *up to* r **do**
  for *every* $a \in V$ **do**
    for *every* $b_1, \ldots, b_k \in V \setminus \{s\}$ **do**
      $\pi = [[a]] \to [[b_1]] \ldots [[b_k]]$
      if π *is not redundant in* P **then**
        $\alpha = \hat{\mathcal{R}}_N\left(a \| b_1 \ldots b_k\right)$
        if $\alpha < \infty$ **then**
          Add π to P with parameter $\theta(\pi) = \exp(\widehat{\mathrm{PMI}}_N(b_1 \ldots b_r) - \alpha)$.
        end
      end
    end
  end
end
return $\langle \Sigma, N, [[s]], p \rangle; \theta$

Algorithm 2: Constructing grammar from a given set of anchors and unary and lexical rules. We assume that s, a special terminal symbol, occurs in V. A production is redundant in a set of productions if there are two other productions that can be combined to form it.

the derivation of *bcdf* which also has a derivation via $S \to EG$. Indeed naively by counting parameters, the distribution over the 4 strings in the language has 3 free parameters, which is the same as number of the free parameters in the grammar G_2, but G_3 has one more, and is no longer identifiable; there are therefore different parameter values — indeed an infinite number —which give the same distribution over strings. Thus the additional expressive power of the higher rank classes comes at a price.

This class does not contain all finite languages. Consider the language $\{ac, bc, bd\}$. Since $a^{\triangleright} \subset b^{\triangleright}$ and $d^{\triangleright} \subset c^{\triangleright}$ the complete grammar for this language is $S \to AC, S \to BD$, $A \to B$, $D \to C$ and $A \to a, B \to b, C \to c, D \to d$. The string bc is then ambiguous, with two trees which we can write, abusing notation, as $[S[A[B[b]]][C[c]]]$ and $[S[B[b]][D[C[c]]]]$. This grammar then is not LUA, since the two unary productions are not LUA. Finally we note that the class of grammars learned by CF's algorithm is a proper subclass of $\mathfrak{G}_2$.

For two grammars that are in $\mathfrak{G}_r$ we can efficiently compute strong equivalence which by theorem 10 is the same as weak equivalence. However, it is natural to ask, as one reviewer does, whether it is possible to compute for an arbitrary PCFG, whether there is a grammar in $\mathfrak{P}_r$ for some r that defines the same distribution. This seems very hard since it is even difficult to determine whether a given CFG lies in $\mathfrak{G}_r$ for a given r; while it is easy to test whether a grammar is anchored, the other criteria depend on properties that seem likely to be undecidable in general, though we do not have a proof.

7. Conclusion

There are alternative strategies to the minimal anchoring condition: for example one could make every terminal an anchor, in which case we would have a potentially much larger set of nonterminals that correspond to congruence classes of terminals. Weakening the anchoring condition seems to require slightly different techniques. Obviously the syntactic concepts $a^{\triangleright\triangleleft}$ are *prime* in the sense of Clark (2015), so it is natural to look for primes of the form $\{w\}^{\triangleright\triangleleft}$ where $|w| > 1$, but algorithmically that seems complicated, without very strong constraints on the grammars. The general principles applied here though seem quite general: to identify a minimal set of nonterminals, and then a maximal set of productions using those nonterminals, subject to redundancy constraints. Though we have left the anchoring condition intact, of course for $r > 2$, we can apply a binarisation algorithm to the learned grammars, and the resulting grammars will no longer be anchored. So indirectly this approach already defines a learning algorithm for nonanchored grammars. Indeed, other grammar transformations are possible: it seems therefore that it is enough for a sufficiently large fraction of the nonterminals to be anchored, and for the non-anchored nonterminals to be inferred using the anchors as scaffolding.

Many of the ideas in this paper have close technical analogues in the Eastern European and Soviet traditions of structuralist linguistics (Marcus, 1967): in particular Dobrushin domination is exactly the property we define as $a^{\triangleright} \supseteq b^{\triangleright}$.

Acknowledgments

I am grateful to Nathanaël Fijalkow for his contributions to the paper which led to this and many discussions, and to the reviewers for helpful comments. Any remaining errors are my own.

References

Alexander Clark. Canonical context-free grammars and strong learning: Two approaches. In *Proceedings of the 14th Meeting on the Mathematics of Language (MOL 2015)*, 2015. URL https://www.aclweb.org/anthology/W15-2309.pdf.

Alexander Clark. Strong learning of probabilistic tree adjoining grammars. *Proceedings of the Society for Computation in Linguistics*, 4(48), 2021. URL https://scholarworks.umass.edu/scil/vol4/iss1/48.

Alexander Clark and Nathanaël Fijalkow. Consistent unsupervised estimators for anchored PCFGs. *Transactions of the Association for Computational Linguistics*, 8:409–422, 2020. URL https://www.aclweb.org/anthology/2020.tacl-1.27.pdf.

Alexander Clark and Ryo Yoshinaka. Distributional learning of context-free and multiple context-free grammars. In Jeffrey Heinz and M. José Sempere, editors, *Topics in Grammatical Inference*, pages 143–172. Springer Berlin Heidelberg, Berlin, Heidelberg, 2016.

Jason Eisner. Inside-outside and forward-backward algorithms are just backprop (tutorial paper). In *Proceedings of the Workshop on Structured Prediction for NLP*, pages 1–17, 2016.

James Jay Horning. *A study of grammatical inference*. PhD thesis, Computer Science Department, Stanford University, 1969.

Richard S Kayne. *Connectedness and binary branching*, volume 16. Walter de Gruyter GmbH & Co KG, 1984.

Stephan Kepser and Jim Rogers. The equivalence of tree adjoining grammars and monadic linear context-free tree grammars. *Journal of Logic, Language and Information*, 20(3): 361–384, 2011.

Solomon Marcus. *Algebraic Linguistics; Analytical Models*. Academic Press, N. Y., 1967.

Mark-Jan Nederhof and Giorgio Satta. Computing partition functions of PCFGs. *Research on Language and Computation*, 2009.

H. Seki, T. Matsumura, M. Fujii, and T. Kasami. On multiple context-free grammars. *Theoretical Computer Science*, 88(2):229, 1991.

Karl Stratos, Michael Collins, and Daniel Hsu. Unsupervised part-of-speech tagging with anchor hidden markov models. *Transactions of the Association for Computational Linguistics*, 4:245–257, 2016.