Strong Learning of Probabilistic Tree Adjoining Grammars

Strong Learning of Probabilistic Tree Adjoining Grammars

Alexander Clark
alexsclark@gmail.com
Department of Philosophy,
King's College London

1 Introduction

In this abstract we outline some theoretical work on the probabilistic learning of a representative mildly context-sensitive grammar formalism from positive examples only. In a recent paper, Clark and Fijalkow (2020) (CF from now on) present a consistent unsupervised learning algorithm for probabilistic context-free grammars (PCFGs) satisfying certain structural conditions: it converges to the correct grammar and parameter values, taking as input only a sample of *strings* generated by the PCFG. Here we extend this to the problem of learning tree grammars from derived trees, and show that under analogous conditions, we can learn a probabilistic tree grammar, of a type that is equivalent to Tree Adjoining Grammars (TAGs) (Vijay-Shankar and Joshi, 1985). In this learning model, we have a probabilistic tree grammar which generates a probability distribution over trees; given a sample of these trees, the learner must converge to a grammar that has the same structure as the original grammar and the same parameters.

This work is motivated ultimately by the problem of first language acquisition, and in particular the acquisition of syntactic structure. The derivation trees of these grammars are richer than those of context-free grammars and naturally account for limited forms of syntactic movement (Rogers, 2003), and so the issue of how these structural descriptions can be learned is of great theoretical importance.

However, an important limitation of this approach is that the input consists of trees not strings; the hope then is that this approach can be combined with the CF approach of learning trees from strings, to get a more plausible model, though there are various technical problems to be overcome.

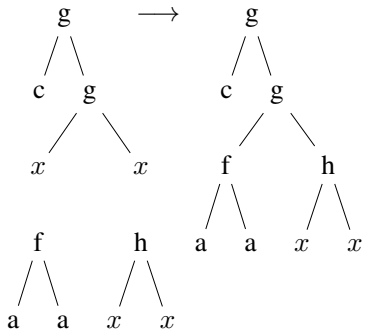


Figure 1: The three trees on the left are combined to form the one on the right, by substituting the bottom two for the two variables in the top; one tree is a 0-stub and the other 3 are all 2-stubs, as they contain two xs.

2 Definitions

We will give a technical presentation in the appendix, and in the body of the abstract just give some ostensive definitions with examples to give the fundamental intuitions. The observed objects we use are binary trees over a ranked alphabet Σ, where each node has either 0 or 2 children; the set of all such trees is called $\mathbb{T}_\Sigma$. The set of terminals of rank k is written Σ^k.

The grammars we use are a sort of simple context-free tree grammar in a variant of Chomsky Normal Form (CNF), that build up a tree using various combinatorial operations that manipulate fragments of trees. We have two sorts of fragments – complete trees which we call 0-stubs and trees where one node is missing its two children, which we call a 2-stub, the gaps being indicated by the symbol x of rank 0. Given a 2-stub and two other stubs , we can combine them to form a larger tree, by replacing the two variables in the first 2-stub with the two other stubs as shown in Figure 1.

A context-free tree grammar of the sort we consider here, is like a context-free grammar except

Proceedings of the Society for Computation in Linguistics (SCiL) 2021, pages 406-414.
Held on-line February 14-19, 2021

Unary	Branching
$\pi_a : A \to a$	$\pi_S : S \to X(D, D)$
$\pi_b : B \to b$	
$\pi_d : D \to d$	
$\pi_{a'} : A' \to a'(x, x)$	$\pi_A : X \to M(A, A'(x, x))$
$\pi_{b'} : B' \to b'(x, x)$	$\pi_{A'} : A' \to X(M(x, x), A)$
$\pi_c : X \to c(x, x)$	$\pi_B : X \to M(B, B'(x, x))$
$\pi_m : M \to m(x, x)$	$\pi_{B'} : B' \to X(M(x, x), B)$

Table 1: Sample grammar which generates a set of trees with noncontext-free string yield.

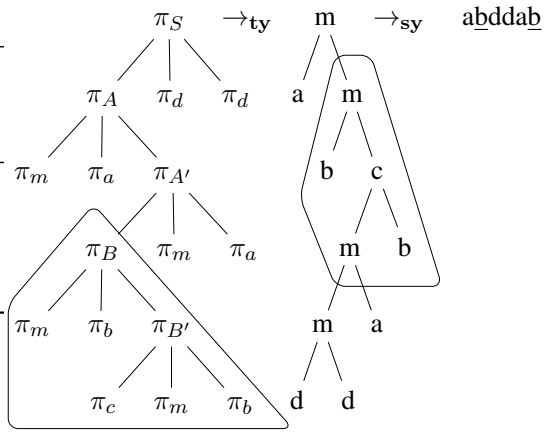
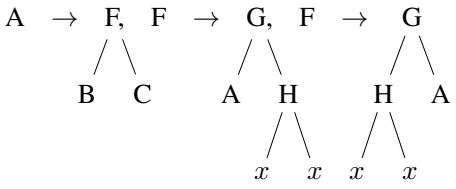


Figure 2: Example derivation tree, derived tree and string yield for the example grammar. Note that the unbounded dependencies between the as (and the bs) which can be arbitrarily distant from each other in the derived tree are local in the derivation tree. The boxed subtree in the derivation tree has tree yield which is the boxed 2-stub in the derived tree, which has discontinuous string yield of the two underlined bs in the string.

we combine trees using these operations, rather than strings using the concatenation operation. So a CFG in CNF, manipulates string fragments (which are just strings) and combines them using productions like $A \to BC$, after introducing the minimal fragments with productions like $A \to a$; here and throughout we use the convention that upper case letters are nonterminals and lower case letters are terminals.

Our tree grammars then have two sets of nonterminals of rank 0 and 2, written N^0 and $N^{(2)}$, and combining productions of these three types:



where A, B, C are of rank 0 (in $N^{0)}$) and F, G, H of rank 2, (in $N^{2)}$). As is normal we have a start symbol S, which must be of rank 0. It's helpful to save space by writing these productions in a flat format as $A \to B(C, D)$ instead. We also have productions that introduce individual terminals of rank 0 and 2, that look like $A \to a$ and $F \to g(x, x)$.

Consider the following simple example grammar G_1 where $\Sigma^{(0)} = \{a, b, d\}$, $\Sigma^{(2)} = \{a', b', c, m\}$ the start symbol is S, $N^{(0)} = \{S, A, B, D\}$,$N^{(2)} = \{A', B', M, X\}$. The productions are shown in Table 1 and a sample derivation, which we will explain shortly, in Figure 2. The string language defined by this grammar is

$$\{w(a|b|\lambda)ddw \mid w \in \{a, b\}^*\},$$

which is clearly not context-free.

A derivation tree of a CFG in CNF is binary branching, but here we have ternary branching derivation trees because we have three nonterminals on the right hand side; and we label each node in the tree with a production, which we write as π with some subscript. See Figure 2 for an example. Crucially, in this setting we observe the binary derived trees, but not the ternary derivation trees which are what we want to learn, in the same way that when doing unsupervised learning of CFGs we observe the derived strings, but not the derivation trees.

A weighted grammar is a grammar where each production has a nonzero parameter associated with it; we can get the probability of a derivation tree by multiplying the parameter of every production used, of course requiring that the sum of the probability of every derivation tree is equal to 1. Rather than the standard parameterization of this, we follow CF and use a bottom-up parameterisation (see appendix for details).

We also want to define the notion of an environment: this plays the role of the context of a fragment in a whole tree. We assume we have two distinguished symbols $\#_2$ and $\#_0$ of rank 2 and 0 respectively. A k-environment is a tree t with a single occurrence of this "gap" symbol. Write the set of all such as $\mathbb{E}^k_\Sigma$. We can combine a k-environment and a k-stub using the operator $\odot_k$ which will give us a complete tree. This is the analogue of combining a context and a string $l\Box r$ with a string u to form a string lur; see Figure 3 for an example. The distribution of a k-stub in a

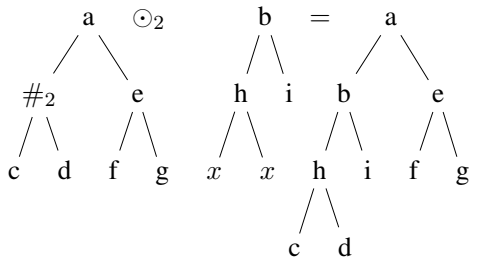


Figure 3: A 2-environment and a 2-stub being combined, via substitution of the symbol $\#_2$ to form a whole tree.

tree language L is just the set of environments in which it can occur. This again is precisely analogous to the structuralist idea of the distribution of a string in a corpus.

3 Structural conditions

We now define three conditions on the grammars that are parallel to those for CFGs defined by CF.

Definition 3.1. *A* FCFTG *is anchored if for every nonterminal A of rank k there is a terminal, say a, of rank k that is derived only from that nonterminal; in other words where $A \to a(x^k)$ is the only production that uses a in the grammar. We call such a terminal an anchor for A.*

Suppose this assumption holds and nonterminals F, G, H of rank 2, are anchored by terminals f, g, h, and A, B of rank 0 by a, b. Then if there was a production of the form $F \to G(BH(x, x))$; we would expect the distribution of the two 2-stubs $f(x, x)$ and $g(b, h(x, x))$ to be similar in the sense that they tend to occur in the same environments.

The main result is that for any such productions (and *mutatis mutandis* for the other various combining productions)

$$\log \theta(F \to G(BH(x,x))) = \mathrm{PMI}(g(b, h(x,x))) \\ -\rho(f(x,x), g(b, h(x,x)))$$

where θ is the parameter of the production in the bottom-up parameterisation, $\mathrm{PMI}(t)$ is a measure of association, approximately the pointwise mutual information, which measures the extent to which the fragment t occurs more often than expected by chance, and $\rho(t, t')$ is a particular asymmetric measure of the distributional similarity of the two 2-stubs, a type of Rényi divergence. Again see the appendix for the definitions. Note that the right hand side of this equation depends only on the distribution over trees that we observe, and can as a result be estimated from a sample of trees, and the left hand side is a parameter of the grammar.

For the productions that introduce fragments it is even simpler; for rank 0 A and c we can show that:

$$\log \theta(A \to c) = \log \mathbb{E}[c] - \rho(a, c)$$

Here $\mathbb{E}[t]$ is just the expected frequency of the fragment t.

We need two more fairly natural conditions to make this work:

Definition 3.2. *We say that a grammar is strictly upward monotonic (*SUM*) if adding a production increases the set of strings generated by the language.*

This implies that if a production is not in the grammar then the divergence term, $\rho(t, t')$, will be infinite, which will have the effect of setting the parameter to zero. Finally we have a fairly weak condition bounding the ambiguity of the model, which is called local unambiguity (LUA), which informally states that for every production there is a tree in which that production is always used in an unambiguous position; this guarantees that the divergence has the correct value. This is satisfied for example if every production is used to derived an unambiguous tree.

We can show that the class of all of these grammars that satisfy these conditions can be learned from a sufficiently large sample of trees by a computationally quite trivial learner. To be explicit about the learning model, we define an algorithm that takes as input a sample of derived trees generated by some probabilistic tree grammar. For every probabilistic grammar in the class defined above, that satisfies the three conditions — anchoring, SUM and LUA– and for any $\epsilon, \delta > 0$ there is some N such that with probability at least $1 - \delta$, given at least N trees, the learner outputs a grammar that is first of all isomorphic to the original grammar, and secondly, such that all of the parameters are within ϵ of the true values. This is therefore a *strong* learning algorithm in that it converges to the correct grammar, not just to some grammar that generates the same set of derived trees.

4 Discussion

There are very few algorithms for learning tree grammars beyond regular tree grammars, which generate only context-free string languages, beyond various extensions of distributional learning (Kasprzik and Yoshinaka, 2011), and they are all only weak learners; this paper shows that one can learn the derivation trees from derived trees, in a realistic learning model; contrast with the negative results of Florêncio (2012) in a non probabilistic learning paradigm.

This is really the first algorithm for strong learning of a representative mildly context-sensitive formalism; this is the least powerful formalism that is not clearly inadequate for natural language syntax (Vijay-Shanker and Weir, 1994). TAGs do not explicitly make a distinction between nonterminals and terminals; though the tree grammar formalism we use does, the fact that the nonterminals are each anchored by a terminal does blur the distinction significantly, and suggest a more direct connection to the TAG formalism than explored here.

A combination of this algorithm with CF suggests that these rich derivation trees can be learned just from surface strings, though it is far from straightforward to directly combine the two, as the modeling assumptions are to a certain extend incompatible. However it is simplistic to consider the entire process of language acquisition to be captured by a single model, and perhaps better to consider it as a sequence of processes, one learning simple constituent structure and another of learning movement within that structure (see for example Hamburger and Wexler (1975)). While the anchoring assumption is not implausible in the case of PCFGs, with these tree grammars it isn't clear what precisely the terminal symbols should be, and thus how realistic this assumption is.

As Hao (2019) argues, weak learning is probably too powerful to explain typological generalizations of interest to syntacticians; the structural constraints we require for strong learning seem to be potentially more fruitful in this regard: these are structurally sensitive constraints on movement, in the case of rank 2 productions, that we derive from learnability considerations. However, we are still at a distance from being able to explain, for example, island constraints; the conditions here are sufficient but clearly not necessary, and the learnability is asymptotic and not a finite sample result.

Acknowledgments

I would like to thank Ryo Yoshinaka; and the reviewers for comments on the paper of which this is an extended abstract.

References

Alexander Clark and Nathanaël Fijalkow. 2020. Consistent unsupervised estimators for anchored PCFGs. *Transactions of the Association for Computational Linguistics*, 8:409–422.

Christophe Costa Florêncio. 2012. Learning tree adjoining grammars from structures and strings. In *Proceedings of the Eleventh International Conference on Grammatical Inference*, volume 21 of *Proceedings of Machine Learning Research*, pages 129–132, University of Maryland, College Park, MD, USA. PMLR.

Henry Hamburger and Kenneth Wexler. 1975. A mathematical theory of learning transformational grammar. *Journal of Mathematical Psychology*, 12(2):137 – 177.

Yiding Hao. 2019. Learnability and overgeneration in computational syntax. *Proceedings of the Society for Computation in Linguistics*, 2(1):124–134.

Anna Kasprzik and Ryo Yoshinaka. 2011. Distributional learning of simple context-free tree grammars. In Jyrki Kivinen, Csaba Szepesvári, Esko Ukkonen, and Thomas Zeugmann, editors, *Algorithmic Learning Theory*, volume 6925 of *Lecture Notes in Computer Science*, pages 398–412. Springer Berlin Heidelberg.

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

James Rogers. 2003. Syntactic structures as multidimensional trees. *Research on Language and Computation*, 1(3-4):265–305.

Yves Schabes. 1992. Stochastic lexicalized tree-adjoining grammars. In *COLING 1992 Volume 2: The 15th International Conference on Computational Linguistics*.

K Vijay-Shankar and Aravind K Joshi. 1985. Some computational properties of tree adjoining grammars. In *Proceedings of the 23rd annual meeting on Association for Computational Linguistics*, pages 82–93. Association for Computational Linguistics.

K. Vijay-Shanker and David J. Weir. 1994. The equivalence of four extensions of context-free grammars. *Mathematical Systems Theory*, 27(6):511–546.

Appendix

We will give in this technical appendix a more formal definition of the ideas we have used here.

We will start by giving some standard definitions of trees and context-free tree grammars; following the notation of Kasprzik and Yoshinaka (2011) among others.

4.1 Trees and tree languages

Let Σ be a ranked alphabet. We define $\Sigma^{(k)}$ to be the set of elements of rank k, then the set of trees over this alphabet is written as $\mathbb{T}_\Sigma$. We can count the number of times an element of Σ occurs in a tree t using the notation $n(a;t)$. A tree language is a subset of $\mathbb{T}_\Sigma$.

We assume that Σ may contain a distinguished symbol λ of rank 0 which is interpreted as an empty string. We define a string yield operation $\mathbf{sy} : \mathbb{T}_\Sigma \to (\Sigma^{(0)} \setminus \{\lambda\})^*$ in the standard way.

We have a distinguished symbol x of rank 0.[1] We define $\mathbb{S}^k_\Sigma$, the set of k-stubs over a ranked alphabet Σ, to be the set of trees t over $\Sigma \cup \{x\}$, where x occurs exactly k times, $n(x;t) = k$. When $a \in \Sigma$ is of rank k, we will write $a(x^k)$ for the k-stub where a has k children all labeled x; k may be zero. Occasionally to reduce clutter we will write this as just a.

We can replace a single occurrence of a symbol of rank k in a tree, by a k-stub, in the standard way, attaching the children of the symbol to the occurrences of x in the k-stub. We denote this infix substitution by $\leftarrow_k$.

We also want to define the notion of an environment: this plays the role of a context of a k-stub. We assume we have distinguished symbols $\#_k$ of rank k.

A k-environment is a tree t over $\Sigma \cup \{\#_k\}$ where $n(\#_k;t) = 1$. Write the set of all such as $\mathbb{E}^k_\Sigma$. We can combine a k-environment and a k-stub using the operator $\odot_k$ which will give us a complete tree which we define; for $e \in \mathbb{E}^k_\Sigma$ and $s \in \mathbb{S}^k_\Sigma$,

$$e \odot_k s = e[\#_k \leftarrow_k s]$$

The notation here is well defined since there is only one occurrence of $\#_k$ in e. This is the analogue of combining a context and a string $l\Box r$ with a string u to form a string lur.

Now we want to define the idea of the distribution of a k-stub in a tree language L: For a $s \in \mathbb{S}^k_\Sigma$ and a tree language L we define

$$s^{|L\rangle} = \{e \in \mathbb{E}^k_\Sigma \mid e \odot_k s \in L\}$$

4.2 Stochastic languages

We are interested in probabilistic learning: so we will consider the probabilistic analogue of a distribution which will be a probability distribution over $\mathbb{E}^k_\Sigma$.

A stochastic tree language over $\mathbb{T}_\Sigma$ is a function $\mathbb{P}$ from $\mathbb{T}_\Sigma$ into non negative real numbers such that $\sum_{t \in \mathbb{T}_\Sigma} \mathbb{P}(t) = 1$.

For any element of $a \in \Sigma$ we can define the expected number of times that a occurs in a tree sampled according to $\mathbb{P}$; which we define as $\mathbb{E}(a) = \sum_{t \in \mathbb{T}_\Sigma} \mathbb{P}(t) n(a;t)$.

We are interested in some sense in unusually frequent substructures; and we can measure this using a generalisation of a familiar association measure:

Definition 4.1. *For a k-stub t, define the pointwise mutual information of t to be:*

$$\mathrm{PMI}(t) = \log \frac{\mathbb{E}(t)}{\prod_{a \in \Sigma} \mathbb{E}(a)^{n(a;t))}}$$

If a stochastic tree language over $\mathbb{T}_\Sigma$ has support[2] L, then a tree $t \in \mathbb{S}^k_\Sigma$ defines a distribution over environments, $\mathcal{D}(t)$, whose support is $t^{|L\rangle}$ defined as

$$\mathcal{D}(t)[e] = \frac{\mathbb{P}(e \odot t)}{\mathbb{E}(t)}$$

where $\mathbb{E}(t)$ is a normalisation constant which is the expected number of times we see the k-stub t in a tree sampled according to the stochastic language.

$$\mathbb{E}(t) = \sum_{e \in \mathbb{E}^k_\Sigma} \mathbb{P}(e \odot t)$$

This coincides with the earlier definition. Given these environment distributions we will want to define a type of distance measure between them and we will use a Renyi divergence, which is not strictly a distance since it is asymmetric. Suppose $t, t' \in \mathbb{S}^k_\Sigma$, then we define

$$\rho(t,t') = \mathcal{R}_\infty\left(\mathcal{D}(t) \middle\| \mathcal{D}(t')\right)$$

Where the Renyi divergence of order ∞ over two discrete distributions over a set $\mathcal{X}$ is defined to be

$$\mathcal{R}_\infty\left(D_1 \| D_2\right) = \sup_{x \in X} \log \frac{D_1(x)}{D_2(x)}$$

[1] Because of the restricted class, we can make do with 1 variable symbol rather than countably many of the form x_i.

[2] The support of a distribution is the set of elements with nonzero probability.

This takes a value in $\mathbb{R}^{\geq 0} \cup \{\infty\}$. with it being zero iff $D_1 = D_2$. It also satisfies the triangle inequality but is asymmetric. Note that

$$e^{-\rho(t,t')} = \inf_{x \in \mathbb{E}^k_\Sigma} \frac{\mathbb{P}(x \odot_k t)\mathbb{E}(t')}{\mathbb{P}(x \odot_k t')\mathbb{E}(t)}$$

which is a value in $[0, 1]$.

4.3 Tree grammars

We now define the special type of tree grammar that we are interested in: footed simple context free tree grammars in a simple normal form that only generate binary trees.

Definition 4.2. *A footed simple context free tree grammar of order 2 (*FCFTG*) is a tuple* (N, Σ, P, S) *where*

- N *is a finite ranked alphabet of nonterminals, all of rank 0 or 2.*
- Σ *is a finite ranked alphabet of terminals, all of rank 0 or 2, which may include* λ.
- S *is a nonterminal of rank 0, which occurs only on the left hand side of a production.*
- P *is a finite set of productions, each of which is in* $N^{(k)} \times \mathbb{S}^k_{\Sigma \cup N}$ *for* $k \in \{0, 2\}$*, written* $A \to t$*, where* A *is the left hand side (lhs), and* t *the righthand side (rhs), of the following types:*
  - *If* A, C, D *are nonterminals of rank 0,* a *a terminal of rank 0,* B *a nonterminal of rank 2,*
    - $A \to a$
    - $A \to B(CD)$
  - *If* A, B, D *are nonterminals of rank 2,* a *a terminal of rank 2, and* C *a nonterminal of rank 0:*
    - $A \to a(x, x)$
    - $A \to B(C, D(x, x))$
    - $A \to B(D(x, x), C)$

All of the production are such that the rank of the nonterminal is equal to the number of occurrences of x on the right hand side. We can put this more succinctly as the following two rule schemas, where

- $A \to a(x^k)$, where rank of A is k
- $A \to B(C(x^i), D(x^j))$, where rank of A is $i + j$.

These are a subset of footed tree grammars (Kepser and Rogers, 2011), which are weakly TAG-equivalent, so these generate a subset of the tree languages of TAGs. The string languages generated are the same as those of TAGs.

4.4 Derivation trees and contexts

We can consider the set P of productions of a grammar G, as a ranked alphabet, where the rank of each production is the number of nonterminals on the right hand side of the production, which in this case is always 0 or 3. A derivation tree for G is a tree in $\mathbb{T}_P$ where each node is labeled with a production such that that the preorder list of nonterminals on the rhs of the production matches the list of lhs of labels of the children. To reduce confusion between derived and derivation trees we will use Greek letters for notations involving the more abstract derivation tree, and Roman letters for the derived trees and related concepts. These derivations are really 3-dimensional trees (Rogers, 2003) but we flatten them out to make them 2d trees for presentational purposes. Note that the derivation trees (which we will denote by τ) are ternary and the derived trees (which we denote by t) are binary. Let $\Omega(G, A)$ be the set of all derivation trees where the root of the tree is labeled with a production whose lhs is A.

The tree yield of a derivation tree τ of a nonterminal of rank k is a k-stub. We can define tree yield, $t = \mathbf{ty}(\tau)$ recursively bottom up in the derivation tree:

- the tree yield of a derivation tree whose root is of rank 0 is just the right hand side of that production labeling the root.
- the tree yield of a tree whose root is of rank 3 is formed by substituting the tree yields of the three child derivation trees into the right hand side of the production labeling the root. For example if the root is labeled with $\pi_0 = A \to B(C, D)$ and the children are τ_1, τ_2, τ_3 then $\mathbf{ty}(\pi_0(\tau_1, \tau_2, \tau_3))$ will be $B(C, D$ with the three substitutions of $B \leftarrow_2 \mathbf{ty}(\tau_1)$, $C \leftarrow_0 \mathbf{ty}(\tau_2)$, and $D \leftarrow_0 \mathbf{ty}(\tau_3)$.

The tree language defined by the grammar is then just the set of tree yields of $\Omega(S)$:

$$\mathcal{L}(G) = \{\mathbf{ty}(\tau) \mid \tau \in \Omega(S)\}$$

We also need the notion of a derivation context of a nonterminal A. We assume that we have

some gap symbols, one for each nonterminal A, of rank 0, written $\Box_A$. The set of derivation contexts of A, written $\Xi(G, A)$, is a tree whose root is labeled with a production with S, over the ranked alphabet $P \cup \{\Box_A\}$ with a single occurrence of the gap symbol. These are just elements of $\Omega(G, S)$ where a single subtree in $\Omega(G, A)$ has been replaced with the gap symbol. We can fill the gap by replacing the gap symbol with some element of $\Omega(G, A)$ and get a whole tree in $\Omega(G, S)$; defining the operation $\oplus$ to be this substitution operation then if $\xi \in \Xi(G, A)$ and $\tau \in \Omega(G, A)$ then $\xi \oplus \tau \in \Omega(G, S)$. We can lift this to sets of trees and contexts in the standard way.

The yield ($\mathbf{cy}$) of a derivation context ξ is an environment, defined as with the tree yield, but where the gap symbol $\Box_A$, has yield $\#_k$ where rank of A is k.

We will often omit the grammar symbol where clear and given a set X of k-stubs or k-environments, use $\Omega(A, X)$ or $\Xi(A, X)$ to refer to the set of derivation trees/contexts of sort A whose yield is in X.

5 Probabilistic grammars

We want to model a distribution over the set of derivation trees and via that over the set of derived trees; this is more straightforward than it is with TAGs (Schabes, 1992). A weighted FCFTG is a grammar together with a function θ from the set of parameters into the nonnegative reals. We define the score or weight of a derivation tree as the product of the parameters of productions used in the derivation tree.

$$s(\tau) = \prod_{\pi \in P} \theta(\pi)^{n(\pi;\tau)} \tag{1}$$

For a nonterminal A we can define two normalisation constants, the inside and outside values of the nonterminal:

$$O(A) = s(\Xi(G, A))$$
$$I(A) = s(\Omega(G, A))$$

We assume that $I(S) = 1$ so we have a well-defined probability distribution. Since S only occurs on the lhs of a production, $O(S) = 1$. The expected number of times we see a production of sort A in a tree τ is

$$\mathbb{E}(A) = O(A)I(A)$$

There are two natural ways of parameterising these models: one is to set $I(A) = 1$ for all nonterminals, which gives a standard top-down probabilistic model. The other is to stipulate that $O(A) = 1$; and so $I(A) = \mathbb{E}(A)$ for all A. These is the same as the *bottom-up* parameterization of WCFGs from CF. For a production $\pi : A \to t$, the expected number of times it is used satisfies:

$$\mathbb{E}(\pi) = O(A)\theta(\pi) \prod_B I(B)^{n(B;t)}$$

Therefore if $O(A) = 1$, for a production $\pi : A \to t$ the parameter will be:

$$\theta(\pi) = \frac{\mathbb{E}(\pi)}{\prod_B \mathbb{E}(B)^{n(B;t)}}$$

Here the parameters are, roughly speaking, being normalised with respect to the right hand side of the productions, rather than by the left hand side in the more normal top down parameterisation. If π has no nonterminals on the left hand side then $\theta(\pi) = \mathbb{E}(\pi)$.

6 Structural Conditions

Definition 6.1. *A* FCFTG *is anchored if for every nonterminal A of rank k there is a terminal of rank k that is derived only from that nonterminal; we call such a terminal an anchor for A.*

If a grammar is anchored then an anchoring is a function $\phi : N \to \Sigma$ that maps each nonterminal to an anchor; we can extend this naturally to k-stubs over N.

Definition 6.2. *We say that a grammar (N, Σ, P, S) is strictly upward monotonic (*SUM*) if for every set of productions Q such that $Q \supsetneq P$, it is the case that: $\mathcal{L}(N, \Sigma, Q, S) \supsetneq \mathcal{L}(N, \Sigma, P, S)$.*

Definition 6.3. *Let A be a nonterminal of rank k. A production $\pi : A \to t$ is locally unambiguous if there is a $r \in \mathcal{L}(G)$ which can be split into $e \in \mathbb{E}^k_\Sigma$ and $s \in \mathbb{S}^k_\Sigma$ such that $e \odot s = r$ and where*

$$\Omega(G, S, r) = \Xi(G, A, \{e\}) \oplus \Omega(G, A, \{s\})$$

and where all the elements of $\Omega(G, A, \{s\})$ have the root labeled π.

*A grammar is locally unambiguous (*LUA*) if every production is locally unambiguous.*

Lemma 6.1. *Let G be an anchored grammar,* SUM, LUA *and ϕ an anchoring, Then if $A \to t$ is a production in G then:*

$$\log \theta(A \to t) = \mathrm{PMI}(\phi(t)) - \rho(\phi(A), \phi(t))$$

The proof follows exactly the method used in CF.

Lemma 6.2. *If $A \to t$ is not in the grammar, then $\rho(\phi(A), \phi(t)) = \infty$.*

The proof here uses SUM.

6.1 Estimators

In order to translate this into a learning algorithm we need some estimators; we use a natural naive plugin estimator for all quantities we use below; these are all consistent so we assume standard ϵ, δ convergence. These are trivial except for the ρ divergences which need a little more work (see CF for details)

We assume we have N samples of derived trees drawn i.i.d. from a fixed distribution. We use the following estimators for $\mathbb{E}$, $\hat{\mathrm{PMI}}_N$, and $\rho(t, t')$. Writing $n(t)$ for the number of times that a $t \in \mathbb{T}_\Sigma$ occurs in the data, so $n(t)/N$ is an unbiased estimate of $\mathbb{P}(t)$, we define

$$\hat{\mathbb{E}}_N(t) = \frac{1}{N} \sum_{e \in \mathbb{E}^k_\Sigma} n(e \odot t)$$

$$\hat{\mathrm{PMI}}_N(t) = \log \frac{\hat{\mathbb{E}}_N(t)}{\prod_{a \in \Sigma} \hat{\mathbb{E}}_N(a)^{n(a;t)}}$$

$$\hat{\rho}_N(t \to t') = \log \frac{\hat{\mathbb{E}}_N(t')}{\hat{\mathbb{E}}_N(t)} \max_{e : n(e \odot t) > \sqrt{N}} \frac{n(e \odot t)}{n(e \odot t')}$$

For any t and t', these naive estimators will converge to the true values, albeit extremely slowly. Here we will only consider cases where t is a single terminal $a(x^k)$ and where t' is either a single terminal or a small k-stub $b(c(x^i), d(x^j))$ where $i, j, i + j$ are in $\{0, 2\}$.

7 Algorithm

Algorithm A on the next page gives pseudocode. Informally, the algorithm receives a sample of N derived trees. By inspection we obtain a ranked alphabet, which will be the alphabet of the learned grammar; we compute $\hat{\rho}_N(a \to b)$ for all pairs of terminals of the same rank. For each rank k, we identify terminals that cannot be anchors: if we have terminals a, b where $\hat{\rho}_N(a \to b) = \infty$ and $\hat{\rho}_N(b \to a) < \infty$ then a is not a possible anchor. We may have multiple anchors a, b for the same nonterminal, in which case $\rho(a, b)$ should be zero; anchors for different nonterminals should have $\rho(a, b) = \infty$. For sufficiently large N we can divide these into equivalence classes and pick one anchor from each class. These will give us a set of representatives for each of the two sets of nonterminals. We then try all possible legal productions using these representatives. If nonterminal A is represented by a, then for each terminal b of the same rank, we compute $\hat{\mathbb{E}}_N(b)$, and estimate the log parameter of $A \to b$ as $\log \hat{\mathbb{E}}_N(b) - \hat{\rho}_N(a \to b)$.

Similarly if we have nonterminals B, C, D anchored by b, c, d; we compute $\hat{\mathrm{PMI}}_N(b(c, d))$ and $\hat{\rho}_N(a \to b(c, d))$ and set the log parameter of $A \to B(C, D)$ to $\hat{\mathrm{PMI}}_N(b(c, d)) - \hat{\rho}_N(a \to b(c, d))$, and likewise for the remaining production types. The start symbol is picked trivially: this will be anchored by a terminal of rank 0 that can occur as a whole tree in the data. We discard the productions where the log parameter is $-\infty$ as these are productions with zero probability. This will give us a weighted grammar in bottom-up form which can then be converted to a probabilistic grammar using the techniques described in CF.

Input: A sequence of trees $D = t_1, t_2, \ldots, t_N$
Output: A FCFTG G and a parameter function θ
$\Sigma \leftarrow$ all the ranked terminal symbols in D;
for $k \in \{0, 2\}$ **do**
  compute $\hat{\rho}_N(a \to b)$ for all $a, b \in \Sigma^{(k)}$;
  $\Gamma^{(k)} \leftarrow \{a \in \Sigma^{(k)} \mid \forall b \in \Sigma^{(k)},\ \hat{\rho}_N(a \to b) < \infty \vee \hat{\rho}_N(b \to a) = \infty\}$;
  Define the equivalence relation on $\Gamma^{(k)}$ given by $a \sim^{(k)} b$ iff $\hat{\rho}_N(a \to b) < \infty$ and $\hat{\rho}_N(b \to a) < \infty$. Let $\Delta^{(k)}$ be the set formed by picking the terminal a with maximal $m(a)$ from each equivalence class in $\Gamma^{(k)} / \sim^{(k)}$;
  $V^{(k)} \leftarrow \{\mathbf{a} \mid a \in \Delta^{(k)}\}$;
end
$s \leftarrow \arg\max\{n(a) \mid a \in \Delta^{(0)}\}$;
$P_L \leftarrow \emptyset$;
$P_B \leftarrow \emptyset$;
for $k \in \{0, 2\}$ **do**
  for $a \in \Delta^{(k)}$ **do**
    $P_L \leftarrow P_L \cup \{\mathbf{a} \to b \mid a \in \Delta^{(k)}, b \in \Sigma^{(k)}\}$;
    if $k = 0$ **then**
      for $b \in \Delta^{(2)}, c, d \in \Delta^{(0)}\}$ **do**
        $P_B \leftarrow P_B \cup \{\mathbf{a} \to \mathbf{b}(\mathbf{c}, \mathbf{d})\}$;
      end
    end
    if $k = 2$ **then**
      for $b, c \in \Delta^{(2)}, d \in \Delta^{(0)}\}$ **do**
        $P_B \leftarrow P_B \cup \{\mathbf{a} \to \mathbf{b}(\mathbf{c}, \mathbf{d}(x, y))\}$ $P_B \leftarrow P_B \cup \{\mathbf{a} \to \mathbf{b}(\mathbf{d}(x, y), \mathbf{c})\}$;
      end
    end
  end
end
for $\mathbf{a} \to b \in P_L$ **do**
  $\theta(\mathbf{a} \to b) \leftarrow \hat{\mathbb{E}}(b) \exp(-\hat{\rho}_N(a \to b))$. ;
end
for $\mathbf{a} \to t \in P_B$ **do**
  $\theta(\mathbf{a} \to t) \leftarrow \exp(-\hat{\rho}_N(a \to t)\hat{\mathrm{PMI}}_N(t))$.;
end
$G \leftarrow \langle \Sigma, V, \mathbf{s}, \{a \to t \in P_L \cup P_B \mid \theta(a \to t) > 0\}\rangle$;
return $G; \theta$

Algorithm A: Basic primal learning algorithm A. We use the notation $\mathbf{a}$ for a nonterminal symbol defined by a.