Count-Based State Merging for Probabilistic Regular Tree Grammars

Toni Dietze
Faculty of Computer Science
Technische Universität Dresden
01062 Dresden, Germany
toni.dietze@tu-dresden.de

Mark-Jan Nederhof
School of Computer Science
University of St Andrews
KY16 9SX, UK

Abstract

We present an approach to obtain language models from a tree corpus using probabilistic regular tree grammars (prtg). Starting with a prtg only generating trees from the corpus, the prtg is generalized step by step by merging nonterminals. We focus on bottom-up deterministic prtg to simplify the calculations.

1 Introduction

Constituent parsing plays an important role in natural language processing (nlp). One can easily read off a pcfg from a tree corpus and use it for parsing. This might work quite well (Charniak, 1996), but it can be even more fruitful to introduce a state behaviour that is not visible in the corpus (Klein and Manning, 2003). The Expectation-Maximization Algorithm (Dempster et al., 1977) can be used to train probabilities if the state behaviour is fixed (Matsuzaki et al., 2005). This can be improved by adapting the state behaviour automatically by cleverly splitting and merging states (Petrov et al., 2006).

More generally, finding a grammar by examining terminal objects is one of the problems investigated in the field of grammatical inference. There are many results for the string case, e.g., on how to learn deterministic stochastic finite (string) automata from text (Carrasco and Oncina, 1994; Carrasco and Oncina, 1999). For the tree case, there are, e.g., results for identifying function distinguishable regular tree languages from text (Fernau, 2002). There is also a generalization of n-grams to trees including smoothing techniques (Rico-Juan et al., 2000; Rico-Juan et al., 2002).

The mentioned results for deterministic stochastic finite (string) automata were generalized to an algorithm that learns stochastic deterministic tree automata from trees (Carrasco et al., 2001). Given a tree corpus, this approach yields a single grammar. The authors experimentally showed that if the corpus is too small, the grammar tends to be too general. In contrast, the split-merge approach of Petrov et al. (2006) produces a sequence of different grammars. One can use cross validation to select a grammar from that sequence that suitably abstracts away from the training corpus. Because of the intricate combination of splitting and merging however, the behavior is very difficult to analyse theoretically. Our approach is similar to the split-merge procedure in that a sequence of grammars is obtained. However, it differs by operating without splitting and relying on merging alone, thereby obtaining a simpler framework.

Our goal is to create a sequence of probabilistic regular tree grammars (prtg) from a corpus such that every prtg abstracts away from the corpus more than its predecessors in the sequence (cf. Algorithm 1). We start with a prtg that admits no more and no less than the trees in the corpus. The rules of the prtg are then changed step by step to make the grammar more general.

We generalize a prtg by *merging* nonterminals, which means we replace several nonterminals by a single new one. The weights of the resulting prtg are assigned by maximum likelihood estimation on the corpus.

To make the approach easier, we only consider bottom-up deterministic prtg. On the one hand, this simplifies our calculations, e.g., the maximum likelihood estimate; on the other hand, this simplifies the application of the prtg as language model, since the search for the most probable tree for a given yield does not have to consider several derivations for a single tree.

2 Preliminaries

Let X be a set. The *identity relation on* X is defined as $\{(x, x) \mid x \in X\}$. An *equivalence relation on* X is a reflexive, symmetric, and transitive relation $(\equiv) \subseteq X \times X$. We write $x \equiv y$ instead of $(x, y) \in (\equiv)$ for $x, y \in X$. Note that the identity relation is an equivalence relation. Let $x \in X$. The *equivalence class of x (induced by $(\equiv)$)*, denoted by $[x]_{\equiv}$ (or just $[x]$ if $(\equiv)$ is clear from the context), is defined as $\{y \in X \mid x \equiv y\}$. The *quotient set of X by $(\equiv)$*, denoted by $X/{\equiv}$, is defined as $\{[x] \mid x \in X\}$. Note that $X/{\equiv}$ is a partition of X and that for every partition P of X there is an equivalence relation $(\equiv')$ such that $P = X/{\equiv'}$. The set of equivalence relations over X forms a complete lattice ordered by set inclusion.

We denote the set of the natural numbers including 0 by $\mathbb{N}$. An *alphabet* (of *symbols*) is a finite, non-empty set. A *ranked alphabet* is an alphabet Σ where we associate a *rank* $\mathrm{rk}(\sigma) \in \mathbb{N}$ with every $\sigma \in \Sigma$. Let Σ be a ranked alphabet. The *set of trees over Σ*, denoted by T_Σ, is the smallest set T such that $\sigma(t_1, \ldots, t_{\mathrm{rk}(\sigma)}) \in T$ for every $\sigma \in \Sigma$ and $t_1, \ldots, t_{\mathrm{rk}(\sigma)} \in T$. Additionally, we define $\mathrm{T}_\emptyset = \emptyset$. Let $t = \sigma(t_1, \ldots, t_k) \in \mathrm{T}_\Sigma$. The *set of positions of t*, denoted by $\mathrm{pos}(t)$, is defined as $\{\varepsilon\} \cup \{iw \mid i \in \{1, \ldots, k\},\ w \in \mathrm{pos}(t_i)\}$. Let $w \in \mathrm{pos}(t)$. The *symbol of t at w*, denoted by $t(w)$, is defined as σ if $w = \varepsilon$, and by $t_i(w')$ if $w = iw'$. The *subtree of t at w*, denoted by $t|_w$, is defined as t if $w = \varepsilon$, and by $t_i|_{w'}$ if $w = iw'$.

Let X be a set and $f\colon X \to \mathbb{R}_{\geq 0}$ a mapping. The *support of f*, denoted by $\mathrm{supp}(f)$, is defined as $\{x \in X \mid f(x) \neq 0\}$. The *size of f*, denoted by $|f|$, is defined as $\sum_{x \in \mathrm{supp}(f)} f(x)$. We call f a *corpus (over X)* if $\mathrm{supp}(f)$ is finite and non-empty. We call f a *probability distribution (over X)* if $|f| = 1$. We denote the set of all probability distributions over X by $\mathrm{Pd}(X)$. Note that f may be both a corpus and a probability distribution. Sometimes we refer to the values of a corpus by the word *counts*.

Definition 1. A *regular tree grammar (rtg)* is a tuple (N, Σ, I, R) where

- N is an alphabet (of *nonterminals*),
- Σ is a ranked alphabet (of *terminals*) such that $N \cap \Sigma = \emptyset$,
- $I \subseteq N$ is a non-empty set (of *initial nonterminals*), and
- R is a finite set of *rules* of the form $A_0 \to \sigma(A_1, \ldots, A_{\mathrm{rk}(\sigma)})$ where $\sigma \in \Sigma$ and $A_0, \ldots, A_{\mathrm{rk}(\sigma)} \in N$. □

Our definition above corresponds to a normal form of rtg with regard to a more general definition, which was given by, for example, Gécseg and Steinby (1984, Chapter II, Section 3).

Let $G = (N, \Sigma, I, R)$ be an rtg. For a rule $r \in R$ of the form $A_0 \to \sigma(A_1, \ldots, A_{\mathrm{rk}(\sigma)})$ we define $\mathrm{lhs}(r) = A_0$ and $\mathrm{rhs}(r) = \sigma(A_1, \ldots, A_{\mathrm{rk}(\sigma)})$. We associate with r the rank $\mathrm{rk}(r) = \mathrm{rk}(\sigma)$, hence we may view R as a ranked alphabet if R is non-empty. We call G *bottom-up deterministic* if $\mathrm{rhs}(r_1) = \mathrm{rhs}(r_2)$ implies $r_1 = r_2$ for every $r_1, r_2 \in R$.

Let $t \in \mathrm{T}_\Sigma$. A *derivation tree of G for t* is a tree $d \in \mathrm{T}_R$ such that $\mathrm{pos}(d) = \mathrm{pos}(t)$, $\mathrm{lhs}(d(\varepsilon)) \in I$, and for every $w \in \mathrm{pos}(t)$ we have $\mathrm{rhs}(d(w)) = t(w)(\mathrm{lhs}(d(w1)), \ldots, \mathrm{lhs}(d(w\,\mathrm{rk}(t(w)))))$. The *set of all derivation trees of G for t* is denoted by $\mathrm{D}_G(t)$. The *language of trees generated by G*, denoted by $[\![G]\!]$, is defined as $\{t \in \mathrm{T}_\Sigma \mid \mathrm{D}_G(t) \neq \emptyset\}$. Note that if G is bottom-up deterministic, $\mathrm{D}_G(t)$ has at most one element. We denote this element, if it exists, by d^t_G.

Definition 2. A *probabilistic regular tree grammar (prtg)* is a tuple (G, ι, ρ) where

- $G = (N, \Sigma, I, R)$ is an rtg,
- $\iota\colon I \to [0, 1]$ is a mapping (*initial weights*), and
- $\rho\colon R \to [0, 1]$ is a mapping (*rule weights*). □

Let $P = (G, \iota, \rho)$ be a prtg. Terminology for rtg is carried over to prtg, e.g., P is bottom-up deterministic iff G is bottom-up deterministic. We call P *proper* if ι is a probability distribution and $\sum_{r \in R\colon \mathrm{lhs}(r) = A} \rho(r) = 1$ for every $A \in N$.

Let $t \in \mathrm{T}_\Sigma$ and $d \in \mathrm{D}_P(t)$. The *weight of d (induced by P)*, denoted by $[\![P]\!](d)$, is defined as $\iota(\mathrm{lhs}(d(\varepsilon))) \cdot \prod_{w \in \mathrm{pos}(t)} \rho(d(w))$. The *weight of t (induced by P)*, denoted by $[\![P]\!](t)$, is defined as $\sum_{d \in \mathrm{D}_P(t)} [\![P]\!](d)$. If $[\![P]\!]$ is a probability distribution over T_Σ, then P is called *consistent*.

Let c be a corpus over T_Σ. The *canonical rtg of c* and the *canonical prtg of c* are defined as $G = (N, \Sigma, I, R)$ and $P = (G, \iota, \rho)$, respectively, where

- $N = \{t|_w \mid t \in \mathrm{supp}(c),\ w \in \mathrm{pos}(t)\}$,
- $I = \mathrm{supp}(c)$,
- $R = \{t|_\varepsilon \to t(\varepsilon)(t|_1, \ldots, t|_{\mathrm{rk}(t(\varepsilon))}) \mid t \in N\}$,
- $\iota(t) = \frac{c(t)}{|c|}$ for every $t \in I$, and
- $\rho(r) = 1$ for every $r \in R$.

Note that every canonical prtg is bottom-up deterministic, proper, and consistent, and that $[\![P]\!](t) = \iota(t)$ for every $t \in \mathrm{supp}([\![P]\!])$; hence, $[\![P]\!]$ represents the relative frequencies of the trees in c.

Let $c\colon X \to \mathbb{R}_{\geq 0}$ be a corpus and $p \in \mathrm{Pd}(X)$.

The *likelihood of* c *given* p is defined as

$$\mathrm{L}(c \mid p) = \prod_{t \in \mathrm{supp}(c)} p(t)^{c(t)}.$$

Let $\mathcal{M} \subseteq \mathrm{Pd}(X)$. The *maximum likelihood estimate from* $\mathcal{M}$ *for* c, denoted by $\mathrm{mle}_{\mathcal{M}}(c)$, is defined as

$$\mathrm{mle}_{\mathcal{M}}(c) = \operatorname*{argmax}_{p \in \mathcal{M}} \mathrm{L}(c \mid p).$$

If $\mathcal{M} = \mathrm{Pd}(\mathrm{T}_\Sigma)$, then $\mathrm{mle}_{\mathcal{M}}(c)$ maps a tree to its relative frequency in c (Prescher, 2004, Theorem 1). Hence, the canonical prtg represents the corpus perfectly.

Let $G = (N, \Sigma, I, R)$ be an rtg, $\mathcal{M} = \{[\![P]\!] \mid P = (G, \iota, \rho)$ is a consistent prtg$\}$, and c a corpus over T_Σ. Then we also write $\mathrm{mle}_G(c)$ instead of $\mathrm{mle}_{\mathcal{M}}(c)$.

Let $G = (N, \Sigma, I, R)$ be a bottom-up deterministic rtg and c a corpus over T_Σ such that $\mathrm{supp}(c) \subseteq [\![G]\!]$. Note that there is exactly one derivation tree d_G^t of G for every tree $t \in \mathrm{supp}(c)$. Based on G, we derive three corpora from c:

$$c_G^{\mathrm{R}} \colon R \to \mathbb{R}_{\geq 0} \colon r \mapsto \sum_{t \in \mathrm{supp}(c)} c(t) \cdot |\{w \in \mathrm{pos}(t) \mid r = \mathrm{d}_G^t(w)\}|,$$

$$c_G^{\mathrm{N}} \colon N \to \mathbb{R}_{\geq 0} \colon A \mapsto \sum_{t \in \mathrm{supp}(c)} c(t) \cdot |\{w \in \mathrm{pos}(t) \mid A = \mathrm{lhs}(\mathrm{d}_G^t(w))\}|,$$

$$c_G^{\mathrm{I}} \colon N \to \mathbb{R}_{\geq 0} \colon A \mapsto \sum_{\substack{t \in \mathrm{supp}(c)\colon \\ A = \mathrm{lhs}(\mathrm{d}_G^t(\varepsilon))}} c(t).$$

Now $\mathrm{mle}_G(c) = [\![(G, \iota, \rho)]\!]$ where for every $A \in I$ and $r \in R$ (Prescher, 2004, Theorem 10):

$$\iota(A) = \frac{c_G^{\mathrm{I}}(A)}{|c|} \quad \text{and} \quad \rho(r) = \frac{c_G^{\mathrm{R}}(r)}{c_G^{\mathrm{N}}(\mathrm{lhs}(r))}.$$

Note that $c_G^{\mathrm{N}}(A) = \sum_{r \in R\colon A = \mathrm{lhs}(r)} c_G^{\mathrm{R}}(r)$.

3 Count-Based State Merging

Algorithm 1 summarizes the idea of our approach. We detail the used notions in what follows.

Let $G = (N, \Sigma, I, R)$ be an rtg and $(\equiv)$ an equivalence relation on N. The *G-merger w.r.t.* $(\equiv)$ is the overloaded expression $\pi_\equiv$ for

- nonterminals: $\pi_\equiv(A) = [A]$ for every $A \in N$,
- rules: $\forall \sigma \in \Sigma \colon \forall A_0, \ldots, A_{\mathrm{rk}(\sigma)} \in N \colon$
  $\pi_\equiv(A_0 \to \sigma(A_1, \ldots, A_{\mathrm{rk}(\sigma)}))$
  $= [A_0] \to \sigma([A_1], \ldots, [A_{\mathrm{rk}(\sigma)}])$,
- sets of nonterminals or rules by applying $\pi_\equiv$ elementwise, and
- rtg: $\pi_\equiv(G) = (N/{\equiv}, \Sigma, \pi_\equiv(I), \pi_\equiv(R))$.

Note that $[\![G]\!] \subseteq [\![\pi_\equiv(G)]\!]$, because by replacing each rule r in a derivation tree of G by $\pi_\equiv(r)$ we get a derivation tree of $\pi_\equiv(G)$. We call $\pi_\equiv$ *non-trivial*, if $(\equiv)$ is not the identity relation. We say $\pi_\equiv$ *merges* A_1 *and* A_2, iff $A_1 \equiv A_2$. We carry over the lattice structure of the set of equivalence relations over N to the set of G-mergers in order to identify minimal and least G-mergers with certain properties.

Algorithm 1 Count-Based State Merging

Input: corpus c over T_Σ
Output: sequence of bottom-up deterministic prtg $P_0, \ldots, P_n$, some $n \in \mathbb{N}$, such that $\mathrm{supp}([\![P_0]\!]) \subseteq \ldots \subseteq \mathrm{supp}([\![P_n]\!])$

```
P_0 = (G_0, ι_0, ρ_0) ← canonical prtg of c
i ← 0
while there exists a non-trivial G_i-merger do
    π ← BESTMERGER(G_i, c)
    i ← i + 1
    G_i ← π(G_{i−1})
    let P_i be prtg such that mle_{G_i}(c) = [[P_i]]
```

To deal with prtg, we fix a corpus c over T_Σ such that $\mathrm{supp}(c) \subseteq [\![G]\!]$. We repeatedly use the maximum likelihood estimate to assign weights to an rtg. That is, the weights are not manipulated during merging itself, but they are used to choose a G-merger: Let Π be a set of G-mergers. The *best G-merger from* Π *w.r.t.* c is defined as

$$\operatorname*{argmax}_{\pi \in \Pi} \mathrm{L}(c \mid [\![\mathrm{mle}_{\pi(G)}(c)]\!]). \tag{1}$$

So far, the presented notions are defined for general rtg. Now let $G = (N, \Sigma, I, R)$ be a bottom-up deterministic rtg. Assuming $0^0 = 1$, we then have the following, which is proven in Appendix A:

$$\mathrm{L}(c \mid [\![\mathrm{mle}_G(c)]\!]) = \frac{\prod_{A \in N} c_G^{\mathrm{I}}(A)^{c_G^{\mathrm{I}}(A)}}{|c|^{|c|}} \cdot \frac{\prod_{r \in R} c_G^{\mathrm{R}}(r)^{c_G^{\mathrm{R}}(r)}}{\prod_{A \in N} c_G^{\mathrm{N}}(A)^{c_G^{\mathrm{N}}(A)}}. \tag{2}$$

This can be used to make Expression 1 more manageable: Let $G = (N, \Sigma, I, R)$ be a bottom-up deterministic rtg and let Π be the set of all non-trivial G-mergers π such that $\pi(G)$ is bottom-up deterministic. Then Equation 2 gives a more direct way of computing the likelihood in Expression 1 without explicitly calculating $\mathrm{mle}_{\pi(G)}(c)$. This is the reason for calling our approach "count-based", and this is the idea of BESTMERGER in Algorithm 1.

Improving efficiency Let $\pi \in \Pi$ and $G' = (N', \Sigma, I', R') = \pi(G)$. We need $c^{\mathrm{R}}_{G'}$, $c^{\mathrm{N}}_{G'}$, and $c^{\mathrm{I}}_{G'}$ to calculate $\mathrm{L}(c \mid [\![\mathrm{mle}_{G'}(c)]\!])$. Bottom-up determinism allows us to derive $\mathrm{d}^t_{G'}$ for every $t \in \mathrm{supp}(c)$ by replacing every rule r in d^t_G by $\pi(r)$. Hence,

$$\forall x' \in X' \colon c^{\mathrm{X}}_{G'}(x') = \sum_{x \in X \colon x' = \pi(x)} c^{\mathrm{X}}_G(x), \quad (3)$$

where X is any of R, N, or I (with or without prime, italic or roman). So we can reuse the corpora related to G to calculate $\mathrm{L}(c \mid [\![\mathrm{mle}_{G'}(c)]\!])$.

We may rewrite Expression 1 by dividing the likelihood by $\mathrm{L}(c \mid [\![\mathrm{mle}_G(c)]\!])$. Then, for many instantiations of π many factors in the fraction cancel out. In detail: Let $(\equiv)$ be the equivalence relation underlying π, and

$$\overline{N} = \{A \in N \mid |[A]| > 1\}, \qquad \overline{N'} = \pi(\overline{N}),$$
$$\overline{R} = \{r \in R \mid |[r]| > 1\}, \text{ and } \quad \overline{R'} = \pi(\overline{R}),$$

where $(\equiv)$ is extended to rules such that $r_1 \equiv r_2$ iff $\pi(r_1) = \pi(r_2)$ for every $r_1, r_2 \in R$. Then, with Equation 2 and $G' = \pi(G)$:

$$\frac{\mathrm{L}(c \mid [\![\mathrm{mle}_{G'}(c)]\!])}{\mathrm{L}(c \mid [\![\mathrm{mle}_G(c)]\!])} = \frac{\prod_{A \in \overline{N'}} c^{\mathrm{I}}_{G'}(A)^{c^{\mathrm{I}}_{G'}(A)}}{\prod_{A \in \overline{N}} c^{\mathrm{I}}_G(A)^{c^{\mathrm{I}}_G(A)}} \cdot \frac{\prod_{r \in \overline{R'}} c^{\mathrm{R}}_{G'}(r)^{c^{\mathrm{R}}_{G'}(r)}}{\prod_{r \in \overline{R}} c^{\mathrm{R}}_G(r)^{c^{\mathrm{R}}_G(r)}} \cdot \frac{\prod_{A \in \overline{N}} c^{\mathrm{N}}_G(A)^{c^{\mathrm{N}}_G(A)}}{\prod_{A \in \overline{N'}} c^{\mathrm{N}}_{G'}(A)^{c^{\mathrm{N}}_{G'}(A)}}. \quad (4)$$

Yet, finding the maximum is still expensive.

Heuristics Assume $\overline{N} = \{A_1, A_2\}$, $\overline{R} = \emptyset$ and $\overline{N} \cap I = \emptyset$. Then, using Equation 3, the right-hand side of Equation 4 is equal to $f(c^{\mathrm{N}}_G(A_1), c^{\mathrm{N}}_G(A_2))$ where

$$f(x, y) = \frac{x^x \cdot y^y}{(x+y)^{x+y}}. \quad (5)$$

For positive x and y, $f(x, y)$ is monotonically decreasing (cf. Appendix B). Hence, with our assumption, it is best to merge nonterminals with the least counts w.r.t. c^{N}_G.

This may be used to guide the search for the best merger. Recall that we want to generalize the canonical (p)rtg step by step. We want to take the smallest steps possible w.r.t. loss of likelihood (cf. Expression 1), so we consider only minimal non-trivial mergers, i.e., mergers that merge exactly two nonterminals. Note that the application of larger mergers may be decomposed into a sequential application of several minimal non-trivial mergers. We can easily sort the minimal non-trivial mergers by the value of f for the counts of the merged nonterminals. Note that the merger which merges the nonterminals with the lowest counts comes first. We choose a beam width $n > 0$ and select the n first mergers for further investigation assuming that the best merger is among them.

Unfortunately, a minimal non-trivial merger $\pi_\equiv$ does not necessarily result in a bottom-up deterministic rtg, but there is a least (i.e. unique minimal) merger $\pi_{\equiv'}$ such that $(\equiv) \subseteq (\equiv')$ which does (cf. Lemma 2 in Appendix C). We use Algorithm 2 to calculate this merger for each of the n chosen mergers and determine the best merger from the results w.r.t. Equation 4.

To restrict the number of considered mergers even more, it may be useful to only merge nonterminals which produce the same terminal. Note that Algorithm 2 preserves this property.

Algorithm 2

Input: rtg $G = (N, \Sigma, I, R)$ and equivalence relation $(\equiv_0)$ on N
Output: the least G-merger $\pi_\equiv$ such that $\pi_\equiv(G)$ is bottom-up deterministic and $(\equiv_0) \subseteq (\equiv)$

```
1: (≡) ← (≡_0)
2: while π_≡(G) not bottom-up deterministic do
3:     find rules r_1 and r_2 in G such that
           rhs(π_≡(r_1)) = rhs(π_≡(r_2)), but
           lhs(π_≡(r_1)) ≠ lhs(π_≡(r_2))
4:     let A_1 = lhs(r_1) and A_2 = lhs(r_2)
5:     let (≡') equivalence relation s.t. N/≡'
           = N/≡ \ {[A_1], [A_2]} ∪ {[A_1] ∪ [A_2]}
6:     replace (≡) by (≡')
```

4 Practical results and outlook

So far, our implementation is not competitive. We are only able to train with small corpora. For example, training with 5 900 trees (consisting of 120 000 distinct subtrees) and a beam width of 1 000 takes about 8 days. For some inputs Algorithm 2 is very expensive. It remains to be seen whether we can avoid such inputs or reduce the effort by reusing results from previous iterations in Algorithm 1.

There is generally a very large number of nonterminals with count 1 in the canonical rtg. Our current heuristics yields the same value for every merger which considers two of such nonterminals. Especially in the first iterations of Algorithm 1 the number of mergers with the same (lowest) heuristic value far exceeds the beam width. This means it is arbitrary which mergers lie within the beam. We hope to improve the heuristics by comparing the trees which are produced by the merged nonterminals.

Using the generated grammars for parsing, we are currently only able to process sentences consisting of words seen in the training data. Even for this limited subset of sentences we are not able to improve precision or recall of brackets (Sekine and Collins, 1997) in comparison to the probabilistic context-free grammar straightforwardly obtained from the corpus. We hope this will improve with a better heuristics.

We have restricted our attention to bottom-up deterministic regular tree grammars. Thanks to this, the conceptual framework could remain relatively simple. What is unclear at this time is whether the bottom-up determinism *per se* restricts the potential accuracy of the models, in relation to the split-merge framework, which allows nondeterministic regular tree grammars.

References

Rafael C. Carrasco and Jose Oncina. 1994. Learning stochastic regular grammars by means of a state merging method. In Rafael C. Carrasco and Jose Oncina, editors, *Grammatical Inference and Applications*, volume 862 of *Lecture Notes in Computer Science*, page 139–152. Springer Berlin Heidelberg.

Rafael C. Carrasco and Jose Oncina. 1999. Learning deterministic regular grammars from stochastic samples in polynomial time. *RAIRO – Theoretical Informatics and Applications*, 33(1):1–19.

Rafael C. Carrasco, Jose Oncina, and Jorge Calera-Rubio. 2001. Stochastic inference of regular tree languages. *Machine Learning*, 44(1-2):185–197.

Eugene Charniak. 1996. Tree-bank grammars. In *Proc. of AAAI/IAAI 1996*, volume 2, pages 1031–1036.

A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38.

Henning Fernau. 2002. Learning tree languages from text. In Jyrki Kivinen and Robert H. Sloan, editors, *Computational Learning Theory*, volume 2375 of *Lecture Notes in Computer Science*, page 153–168. Springer Berlin Heidelberg.

Ferenc Gécseg and Magnus Steinby. 1984. *Tree Automata*. Akadémiai Kiadó, Budapest, Hungary.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proc. of ACL 2003*, volume 1, pages 423–430.

Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Probabilistic CFG with latent annotations. In *Proc. of ACL 2005*, pages 75–82.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proc. of COLING/ACL 2006*, pages 433–440.

Detlef Prescher. 2004. A tutorial on the expectation-maximization algorithm including maximum-likelihood estimation and EM training of probabilistic context-free grammars. *CoRR*, abs/cs/0412015.

Juan Ramón Rico-Juan, Jorge Calera-Rubio, and Rafael C. Carrasco. 2000. Probabilistic k-testable tree languages. In Arlindo L. Oliveira, editor, *Grammatical Inference: Algorithms and Applications*, volume 1891 of *Lecture Notes in Computer Science*, page 221–228. Springer Berlin Heidelberg.

Juan Ramón Rico-Juan, Jorge Calera-Rubio, and Rafael C. Carrasco. 2002. Stochastic k-testable tree languages and applications. In Pieter Adriaans, Henning Fernau, and Menno van Zaanen, editors, *Grammatical Inference: Algorithms and Applications*, volume 2484 of *Lecture Notes in Computer Science*, page 199–212. Springer Berlin Heidelberg.

Satoshi Sekine and Michael John Collins. 1997. Evalb. `http://nlp.cs.nyu.edu/evalb/`. Accessed 2015-03-19.

A The likelihood of the maximum likelihood estimate in the bottom-up deterministic case (Equation 2)

Let $G = (N, \Sigma, I, R)$ be a bottom-up deterministic rtg and c a corpus over T_Σ such that $\mathrm{supp}(c) \subseteq [\![G]\!]$. Let $P = (G, \iota, \rho)$ such that $[\![P]\!] = \mathrm{mle}_G(c)$. We can transform $\mathrm{L}(c \mid [\![P]\!])$ as follows, assuming $0^0 = 1$:

$$
\begin{aligned}
&\mathrm{L}(c \mid [\![P]\!]) \\
&= \prod_{t \in \mathrm{supp}(c)} [\![P]\!](t)^{c(t)} && \text{(def. of L)} \\
&= \prod_{t \in \mathrm{supp}(c)} \Big(\sum_{d \in \mathrm{D}_P(t)} [\![P]\!](d) \Big)^{c(t)} && \text{(def. of } [\![P]\!]\text{)} \\
&= \prod_{t \in \mathrm{supp}(c)} [\![P]\!](\mathrm{d}_G^t)^{c(t)} && \text{(assumptions for } G \text{ and } c\text{)} \\
&= \prod_{t \in \mathrm{supp}(c)} \Big(\iota(\mathrm{lhs}(\mathrm{d}_G^t(\varepsilon))) \cdot \prod_{w \in \mathrm{pos}(t)} \rho(\mathrm{d}_G^t(w)) \Big)^{c(t)} && \text{(def. of } [\![P]\!]\text{)} \\
&= \prod_{t \in \mathrm{supp}(c)} \iota(\mathrm{lhs}(\mathrm{d}_G^t(\varepsilon)))^{c(t)} \cdot \prod_{w \in \mathrm{pos}(t)} \rho(\mathrm{d}_G^t(w))^{c(t)} && \text{(distributivity)} \\
&= \Big(\prod_{t \in \mathrm{supp}(c)} \iota(\mathrm{lhs}(\mathrm{d}_G^t(\varepsilon)))^{c(t)} \Big) \cdot \prod_{t \in \mathrm{supp}(c)} \prod_{w \in \mathrm{pos}(t)} \rho(\mathrm{d}_G^t(w))^{c(t)} && \text{(commutativity)} \\
&= \Big(\prod_{A \in N} \prod_{\substack{t \in \mathrm{supp}(c): \\ A = \mathrm{lhs}(\mathrm{d}_G^t(\varepsilon))}} \iota(A)^{c(t)} \Big) \cdot \prod_{r \in R} \prod_{t \in \mathrm{supp}(c)} \prod_{\substack{w \in \mathrm{pos}(t): \\ r = \mathrm{d}_G^t(w)}} \rho(r)^{c(t)} && \text{(commutativity)} \\
&= \Big(\prod_{A \in N} \iota(A)^{c_G^{\mathrm{I}}(A)} \Big) \cdot \prod_{r \in R} \rho(r)^{\sum_{t \in \mathrm{supp}(c)} \sum_{w \in \mathrm{pos}(t):\, r = \mathrm{d}_G^t(w)} c(t)} && (b^c \cdot b^d = b^{c+d},\ 0^0 = 1, \text{ def. of } c_G^{\mathrm{I}}) \\
&= \Big(\prod_{A \in N} \iota(A)^{c_G^{\mathrm{I}}(A)} \Big) \cdot \prod_{r \in R} \rho(r)^{\sum_{t \in \mathrm{supp}(c)} c(t) \cdot |\{w \in \mathrm{pos}(t) \mid r = \mathrm{d}_G^t(w)\}|} && \text{(distributivity)} \\
&= \Big(\prod_{A \in N} \iota(A)^{c_G^{\mathrm{I}}(A)} \Big) \cdot \prod_{r \in R} \rho(r)^{c_G^{\mathrm{R}}(r)} && \text{(def. of } c_G^{\mathrm{R}}\text{)} \\
&= \frac{\prod_{A \in N} c_G^{\mathrm{I}}(A)^{c_G^{\mathrm{I}}(A)}}{\prod_{A \in N} |c|^{c_G^{\mathrm{I}}(A)}} \cdot \frac{\prod_{r \in R} c_G^{\mathrm{R}}(r)^{c_G^{\mathrm{R}}(r)}}{\prod_{r \in R} c_G^{\mathrm{N}}(\mathrm{lhs}(r))^{c_G^{\mathrm{R}}(r)}} && \text{(def. of } \iota \text{ and } \rho\text{, comm., distr.)} \\
&= \frac{\prod_{A \in N} c_G^{\mathrm{I}}(A)^{c_G^{\mathrm{I}}(A)}}{|c|^{|c|}} \cdot \frac{\prod_{r \in R} c_G^{\mathrm{R}}(r)^{c_G^{\mathrm{R}}(r)}}{\prod_{A \in N} c_G^{\mathrm{N}}(A)^{c_G^{\mathrm{N}}(A)}} && (b^c \cdot b^d = b^{c+d}, \text{ def. of } c_G^{\mathrm{I}} \text{ and } c_G^{\mathrm{N}}, \text{ comm.})
\end{aligned}
$$

This proves Equation 2.

B The function in Equation 5 is monotonically decreasing

We may transform Equation 5 as follows:

$$f(x, y) = \frac{x^x \cdot y^y}{(x+y)^{x+y}} = \Big(\frac{x}{x+y}\Big)^x \cdot \Big(\frac{y}{x+y}\Big)^y.$$

For positive arguments, the partial derivatives of f are

$$
\begin{aligned}
\frac{\partial f(x,y)}{\partial x} &= \ln\Big(\frac{x}{x+y}\Big) \cdot \Big(\frac{x}{x+y}\Big)^x \cdot \Big(\frac{y}{x+y}\Big)^y, \text{ and} \\
\frac{\partial f(x,y)}{\partial y} &= \ln\Big(\frac{y}{x+y}\Big) \cdot \Big(\frac{x}{x+y}\Big)^x \cdot \Big(\frac{y}{x+y}\Big)^y.
\end{aligned}
$$

For $x, y > 0$ the fractions are smaller than one. This means the logarithms are negative, hence the whole terms. So f is monotonically decreasing.

C Properties of mergers regarding bottom-up determinism

Lemma 1. *Let $G = (N, \Sigma, I, R)$ be an rtg, and let $(\equiv_1)$ and $(\equiv_2)$ be equivalence relations over N such that $\pi_{\equiv_1}(G)$ and $\pi_{\equiv_2}(G)$ are bottom-up deterministic. Let $(\equiv) = (\equiv_1) \cap (\equiv_2)$. Then also $\pi_{\equiv}(G)$ is bottom-up deterministic.*

Proof. Assume $\pi_{\equiv}(G)$ is *not* bottom-up deterministic. Then there are two rules $A_0 \to \sigma(A_1, \ldots, A_{\mathrm{rk}(\sigma)})$ and $B_0 \to \sigma(B_1, \ldots, B_{\mathrm{rk}(\sigma)})$ in R such that $A_i \equiv B_i$ for every $1 \leq i \leq \mathrm{rk}(\sigma)$, but $A_0 \not\equiv B_0$. Hence, $A_i \equiv_1 B_i$ and $A_i \equiv_2 B_i$ for every $1 \leq i \leq \mathrm{rk}(\sigma)$, and therefore $A_0 \equiv_1 B_0$ and $A_0 \equiv_2 B_0$. This implies $A_0 \equiv B_0$, which is a contradiction, so $\pi_{\equiv}(G)$ is bottom-up deterministic. q.e.d.

Lemma 2. *Let $G = (N, \Sigma, I, R)$ be an rtg, and let $(\equiv)$ be an equivalence relation over N. Then there is a least (i.e. unique minimal) $(\hat{\equiv})$ such that $(\equiv) \subseteq (\hat{\equiv})$ and $\pi_{\hat{\equiv}}(G)$ is bottom-up deterministic.*

Proof. Existence: Consider $(\equiv')$ such that $\forall A_1, A_2 \in N\colon A_1 \equiv' A_2$. Then $(\equiv')$ satisfies the conditions.

Uniqueness: Assume there is a minimal $(\equiv') \neq (\hat{\equiv})$ satisfying the conditions. Then, by Lemma 1, $(\equiv') \cap (\hat{\equiv})$ would also satisfy the conditions, which contradicts that $(\equiv')$ and $(\hat{\equiv})$ are minimal. Hence, $(\hat{\equiv})$ is unique. q.e.d.