Kullback-Leibler Distance between Probabilistic Context-Free Grammars and Probabilistic Finite Automata

Mark-Jan Nederhof
Faculty of Arts
University of Groningen
P.O. Box 716
NL-9700 AS Groningen, The Netherlands
markjan@let.rug.nl

Giorgio Satta
Department of Information Engineering
University of Padua
via Gradenigo, 6/A
I-35131 Padova, Italy
satta@dei.unipd.it

Abstract

We consider the problem of computing the Kullback-Leibler distance, also called the relative entropy, between a probabilistic context-free grammar and a probabilistic finite automaton. We show that there is a closed-form (analytical) solution for one part of the Kullback-Leibler distance, viz. the cross-entropy. We discuss several applications of the result to the problem of distributional approximation of probabilistic context-free grammars by means of probabilistic finite automata.

1 Introduction

Among the many formalisms used for description and analysis of syntactic structure of natural language, the class of context-free grammars (CFGs) is by far the best understood and most widely used. Many formalisms with greater generative power, in particular the different types of unification grammars, are ultimately based on CFGs.

Regular expressions, with their procedural counter-part of finite automata (FAs), are not able to describe hierarchical, tree-shaped structure, and thereby seem less suitable than CFGs for full analysis of syntactic structure. However, there are many applications where only partial or approximated analysis of structure is needed, and where full context-free processing could be prohibitively expensive. Such applications can for example be found in real-time speech recognition systems: of the many hypotheses returned by a speech recognizer, shallow syntactic analysis may be used to select a small subset of those that seem most promising for full syntactic processing in a next phase, thereby avoiding further computational costs for the less promising hypotheses.

As FAs cannot describe structure as such, it is impractical to write the automata required for such applications by hand, and even difficult to derive them automatically by training. For this reason, the used FAs are often derived from CFGs, by means of some form of approximation. An overview of different methods of approximating CFGs by FAs, along with an experimental comparison, was given by (Nederhof, 2000).

The next step is to assign probabilities to the transitions of the approximating FA, as the application outlined above requires a qualitative distinction between hypotheses rather than the purely boolean distinction of language membership. Under certain circumstances, this may be done by carrying over the probabilities from an input probabilistic CFG (PCFG), as shown for the special case of n-grams by (Rimon and Herz, 1991; Stolcke and Segal, 1994), or by training of the FA on a corpus generated by the PCFG (Jurafsky et al., 1994). See also (Mohri and Nederhof, 2001) for discussion of related ideas.

An obvious question to ask is then how well the resulting PFA approximates the input PCFG, possibly for different methods of determining an FA and different ways of attaching probabilities to the transitions. Until now, any direct way of measuring the distance between a PCFG and a PFA has been lacking. As we will argue in this paper, the natural distance measure between probability distributions, the Kullback-Leibler (KL) distance, is difficult to compute. (The KL distance is also called relative entropy.) We can however derive a closed-form (analytical) solution for the cross entropy of a PCFG and a PFA, provided the FA underlying the PFA is deterministic. The difference between the cross-entropy and the KL distance is the entropy of the PCFG, which does not rely on the PFA. This means that if we are interested in the relative quality of different approximating PFAs with respect to a single input PCFG, the cross-entropy may be used instead of the KL distance. The constraint of determinism is not a problem in practice, as any FA can be determinized, and FAs derived by approximation algorithms are normally determinized (and minimized).

As a second possible application, we now look more closely into the matter of determinization of finite-state models. Not all PFAs can be determinized, as discussed by (Mohri, 1997). This is unfortunate, as deterministic (P)FAs process input with time and space costs independent of the size of the automaton, whereas these costs are linear in the size of the automaton in the nondeterministic case, which may be too high for some real-time applications. Instead of distribution-preserving determinization, we may therefore approximate a nondeterministic PFA by a deterministic PFA whose probability distribution is close to, but not necessarily identical to, that of the first PFA. Again, an important question is how close the two models are to each other. It was argued before by (Juang and Rabiner, 1985; Falkhausen et al., 1995; Vihola et al., 2002) that the KL distance between finite-state models is difficult to compute in general. The theory developed in this paper shows however that the cross-entropy between the input PFA and the approximating deterministic PFA can be expressed in closed form, relying on the fact that a PFA can be seen as a special case of a PCFG. Thereby, different approximating deterministic PFAs can be compared for closeness to the input PFA. We can even compute the KL distance between two unambiguous PFAs, in closed form. (It is not difficult to see that ambiguity is a decidable property for FAs.)

The structure of this paper is as follows. We provide some preliminary definitions in Section 2. Section 3 discusses the expected frequency of a rule in derivations allowed by a PCFG, and explains how such values can be effectively computed. The KL distance between a PCFG and a PFA is closely related to the entropy of the PCFG, which we discuss in Section 4. Essential to our approach is the intersection of PCFGs and PFAs, to be discussed in Section 5. As we show in Section 6, the part of the KL distance expressing the cross-entropy can be computed in closed form, based on this intersection. Section 7 concludes this paper.

2 Preliminaries

Throughout the paper we use mostly standard formal language notation, as for instance in (Hopcroft and Ullman, 1979; Booth and Thompson, 1973), which we summarize below.

A *context-free grammar* (CFG) is a 4-tuple $G = (\Sigma, N, S, R)$ where Σ and N are finite disjoint sets of *terminals* and *nonterminals*, respectively, $S \in N$ is the *start symbol* and R is a finite set of *rules*. Each rule has the form $A \rightarrow \alpha$, where $A \in N$ and $\alpha \in (\Sigma \cup N)^*$.

The 'derives' relation $\Rightarrow$ associated with G is defined on triples consisting of two strings $\alpha, \beta \in (\Sigma \cup N)^*$ and a rule $\pi \in R$. We write $\alpha \overset{\pi}{\Rightarrow} \beta$ if and only if α is of the form $uA\delta$ and β is of the form $u\gamma\delta$, for some $u \in \Sigma^*$, $\delta \in (\Sigma \cup N)^*$, and $\pi = (A \rightarrow \gamma)$. A *left-most derivation* (for G) is a string $d = \pi_1 \cdots \pi_m$, $m \geq 0$, such that $\alpha_0 \overset{\pi_1}{\Rightarrow} \alpha_1 \overset{\pi_2}{\Rightarrow} \cdots \overset{\pi_m}{\Rightarrow} \alpha_m$, for some $\alpha_0, \ldots, \alpha_m \in (\Sigma \cup N)^*$; $d = \epsilon$ (where ϵ denotes the empty string) is also a left-most derivation. In the remainder of this paper, we will let the term 'derivation' refer to 'left-most derivation', unless specified otherwise. If $\alpha_0 \overset{\pi_1}{\Rightarrow} \cdots \overset{\pi_m}{\Rightarrow} \alpha_m$ for some $\alpha_0, \ldots, \alpha_m \in (\Sigma \cup N)^*$, then we say that $d = \pi_1 \cdots \pi_m$ *derives* α_m from α_0 and we write $\alpha_0 \overset{d}{\Rightarrow} \alpha_m$; $d = \epsilon$ derives any $\alpha_0 \in (\Sigma \cup N)^*$ from itself.

A (left-most) derivation d such that $S \overset{d}{\Rightarrow} w$, $w \in \Sigma^*$, is called a *complete* derivation. If d is a complete derivation, we write $y(d)$ to denote the (unique) string $w \in \Sigma^*$ such that $S \overset{d}{\Rightarrow} w$. The language generated by G is the set of all strings $y(d)$ derived by complete derivations, i.e., $L(G) = \{w \,|\, S \overset{d}{\Rightarrow} w,\ d \in R^*,\ w \in \Sigma^*\}$. It is well-known that there is a one-to-one correspondence between complete derivations and parse trees for strings in $L(G)$.

A *probabilistic* CFG (PCFG) is a pair $G_p = (G, p_G)$, where G is a CFG and p_G is a function from R to real numbers in the interval $[0, 1]$. A PCFG is *proper* if $\sum_{\pi=(A\rightarrow\alpha)} p_G(\pi) = 1$ for all $A \in N$. Function p_G can be used to associate probabilities to derivations of the underlying CFG G, in the following way. For $d = \pi_1 \cdots \pi_m \in R^*$, $m \geq 0$, we define $p_G(d) = \prod_{i=1}^{m} p_G(\pi_i)$ if $S \overset{d}{\Rightarrow} w$ for some $w \in \Sigma^*$, and $p_G(d) = 0$ otherwise. The probability of a string $w \in \Sigma^*$ is defined as $p_G(w) = \sum_{d:y(d)=w} p_G(d)$. A PCFG is *consistent* if $\sum_w p_G(w) = 1$. Consistency implies that the PCFG defines a probability distribution on the set of terminal strings as well as on the set of grammar derivations. If a PCFG is proper, then consistency means that no probability mass is lost in 'infinite' derivations.

A *finite automaton* (FA) is a 5-tuple $M = (\Sigma, Q, q_0, Q_f, T)$, where Σ and Q are two finite sets of *terminals* and *states*, respectively, q_0 is the *initial* state, $Q_f \subseteq Q$ is the set of *final* states, and T is a finite set of *transitions*, each of the form $s \overset{a}{\mapsto} t$, where $s, t \in Q$ and $a \in \Sigma$. A *probabilistic finite automaton* (PFA) is a pair $M_p = (M, p_M)$, where M is an FA and p_M is a function from T to real numbers in the interval $[0, 1]$.[1]

For a fixed (P)FA M, we define a *configuration* to be an element of $Q \times \Sigma^*$, and we define the relation $\vdash$ on triples consisting of two configurations and a transition $\tau \in T$ by: $(s, w) \overset{\tau}{\vdash} (t, w')$ if and only if w is of the form aw', for some $a \in \Sigma$, and $\tau = (s \overset{a}{\mapsto} t)$. A *complete computation* is a string $c = \tau_1 \cdots \tau_m$, $m \geq 0$, such that $(s_0, w_0) \overset{\tau_1}{\vdash} (s_1, w_1) \overset{\tau_2}{\vdash} \cdots \overset{\tau_m}{\vdash} (s_m, w_m)$, for some $(s_0, w_0), \ldots, (s_m, w_m) \in Q \times \Sigma^*$, with $s_0 = q_0$, $s_m \in Q_f$ and $w_m = \epsilon$, and we write $(s_0, w_0) \overset{c}{\vdash} (s_m, w_m)$. The language accepted by M is $L(M) = \{w \in \Sigma^* \,|\, (q_0, w) \overset{c}{\vdash} (s, \epsilon), c \in T^*, s \in Q_f\}$.

For a PFA $M_p = (M, p_M)$, and $c = \tau_1 \cdots \tau_m \in T^*$, $m \geq 0$, we define $p_M(c) = \prod_{i=1}^{m} p_M(\tau_i)$ if c is a complete computation, and $p_M(c) = 0$ otherwise. A PFA is *consistent* if $\sum_c p_M(c) = 1$.

We say M is *unambiguous* if for each $w \in \Sigma^*$, $\exists_{s \in Q_f}[(q_0, w) \overset{c}{\vdash} (s, \epsilon)]$ for at most one $c \in T^*$. We say M is *deterministic* if for each s and a, there is at most one transition $s \overset{a}{\mapsto} t$. Determinism implies unambiguity. It can be more readily checked whether an FA is deterministic than whether it is unambiguous. Furthermore, any FA can be effectively turned into a deterministic FA accepting the same language. Therefore, this paper will assume that FAs are deterministic, although technically, unambiguity is sufficient for our constructions to apply.

3 Expectation of rule frequency

Here we discuss how we can compute the expectation of the frequency of a rule or a nonterminal over all derivations of a probabilistic context-free grammar. These quantities will be used later by our algorithms.

Let $(A \to \alpha) \in R$ be a rule of PCFG G_p, and let $d \in R^*$ be a complete derivation in G_p. We define $f(A \to \alpha; d)$ as the number of occurrences, or *frequency*, of $A \to \alpha$ in d. Similarly, the frequency of nonterminal A in d is defined as $f(A; d) = \sum_\alpha f(A \to \alpha; d)$. We consider the following related quantities

$$\begin{aligned} E_{p_G}\ f(A \to \alpha; d) &= \sum_d p_G(d) \cdot f(A \to \alpha; d), \\ E_{p_G}\ f(A; d) &= \sum_d p_G(d) \cdot f(A; d) \\ &= \sum_\alpha E_{p_G}\ f(A \to \alpha; d). \end{aligned}$$

A method for the computation of these quantities is reported in (Hutchins, 1972), based on the so-called *momentum matrix*. We propose an alternative method here, based on an idea related to the inside-outside algorithm (Baker, 1979; Lari and Young, 1990; Lari and Young, 1991). We observe that we can factorize a derivation d at each occurrence of rule $A \to \alpha$ into an 'innermost' part d_2 and two 'outermost' parts d_1 and d_3. We can then write

$$E_{p_G}\ f(A \to \alpha; d) = \sum_{\substack{d=\pi_1\cdots\pi_m, m_1, m_2, w, \beta, v, x: \\ S \overset{d_1}{\Rightarrow} wA\beta, \text{ with } d_1=\pi_1\cdots\pi_{m_1-1}, \\ (A\to\alpha)=\pi_{m_1}, \\ \alpha \overset{d_2}{\Rightarrow} v, \text{ with } d_2=\pi_{m_1+1}\cdots\pi_{m_2}, \\ \beta \overset{d_3}{\Rightarrow} x, \text{ with } d_3=\pi_{m_2+1}\cdots\pi_m}} \prod_{i=1}^{m} p_G(\pi_i).$$

Next we group together all of the innermost and all of the outermost derivations and write

$$E_{p_G}\ f(A \to \alpha; d) = out_{G_p}(A) \cdot p_G(A \to \alpha) \cdot in_{G_p}(\alpha)$$

where

$$out_{G_p}(A) = \sum_{\substack{d=\pi_1\cdots\pi_m, d'=\pi'_1\cdots\pi'_{m'}, w, \beta, x: \\ S \overset{d}{\Rightarrow} wA\beta,\ \beta \overset{d'}{\Rightarrow} x}} \prod_{i=1}^{m} p_G(\pi_i) \cdot \prod_{i=1}^{m'} p_G(\pi'_i)$$

and

$$in_{G_p}(\alpha) = \sum_{\substack{d=\pi_1\cdots\pi_m, v: \\ \alpha \overset{d}{\Rightarrow} v}} \prod_{i=1}^{m} p_G(\pi_i).$$

Both $out_{G_p}(A)$ and $in_{G_p}(\alpha)$ can be described in terms of recursive equations, of which the least

[1]Our definition of PFAs amounts to a slight loss of generality with respect to standard definitions, in that there are no epsilon transitions and no probability function on states being final. We want to avoid these concepts as they would cause some technical complications later in this article. There is no loss of generality however if we may assume an end-of-sentence marker, which is often the case in practice.

fixed-points are the required values. If G_p is proper and consistent, then $in_{G_p}(\alpha) = 1$ for each $\alpha \in (\Sigma \cup N)^*$. Quantities $out_{G_p}(A)$ for every A can all be (exactly) calculated by solving a linear system, requiring an amount of time proportional to the cube of the size of G_p; see for instance (Corazza et al., 1991).

On the basis of all the above quantities, a number of useful statistical properties of G_p can be easily computed, such as the expected length of derivations, denoted $\mathrm{EDL}(G_p)$ and the expected length of sentences, denoted $\mathrm{EWL}(G_p)$, discussed before by (Wetherell, 1980). These quantities satisfy the relations

$$\mathrm{EDL}(G_p) = E_{p_G}\ |d| = \sum_{A\to\alpha} out_{G_p}(A) \cdot p_G(A\to\alpha) \cdot in_{G_p}(\alpha),$$

$$\mathrm{EWL}(G_p) = E_{p_G}\ |y(d)| = \sum_{A\to\alpha} out_{G_p}(A) \cdot p_G(A\to\alpha) \cdot in_{G_p}(\alpha) \cdot |\alpha|_\Sigma\,,$$

where for a string $\gamma \in (N \cup \Sigma)^*$ we write $|\gamma|_\Sigma$ to denote the number of occurrences of terminal symbols in γ.

4 Entropy of PCFGs

In this section we introduce the notion of derivational entropy of a PCFG, and discuss an algorithm for its computation.

Let $G_p = (G, p_G)$ be a PCFG. For a nonterminal A of G, let us define the entropy of A as the entropy of the distribution p_G on all rules of the form $A \to \alpha$, i.e.,

$$\begin{aligned} H(A) &= E_{p_G}\ \log\frac{1}{p_G(A\to\alpha)} \\ &= \sum_\alpha p_G(A\to\alpha)\cdot\log\frac{1}{p_G(A\to\alpha)}. \end{aligned}$$

The *derivational entropy* of G_p is defined as the expectation of the information of the complete derivations generated by G_p, i.e.,

$$\begin{aligned} H_d(G_p) &= E_{p_G}\ \log\frac{1}{p_G(d)} \\ &= \sum_d p_G(d)\cdot\log\frac{1}{p_G(d)}. \qquad (1) \end{aligned}$$

We now characterize derivational entropy using expected rule frequencies as

$$\begin{aligned} &H_d(G_p) = \\ &\sum_d p_G(d)\cdot\log\frac{1}{p_G(d)} = \\ &\sum_d p_G(d)\cdot\log\prod_{A\to\alpha}\left(\frac{1}{p_G(A\to\alpha)}\right)^{f(A\to\alpha;d)} = \\ &\sum_d p_G(d)\cdot\sum_{A\to\alpha} f(A\to\alpha;d)\cdot\log\frac{1}{p_G(A\to\alpha)} = \\ &\sum_{A\to\alpha}\log\frac{1}{p_G(A\to\alpha)}\cdot\sum_d p_G(d)\cdot f(A\to\alpha;d) = \\ &\sum_{A\to\alpha}\log\frac{1}{p_G(A\to\alpha)}\cdot E_{p_G}\ f(A\to\alpha;d) = \\ &\sum_A\sum_\alpha\log\frac{1}{p_G(A\to\alpha)}\cdot out_{G_p}(A)\cdot p_G(A\to\alpha)\cdot \\ &\qquad in_{G_p}(\alpha) = \\ &\sum_A out_{G_p}(A)\cdot\sum_\alpha p_G(A\to\alpha)\cdot\log\frac{1}{p_G(A\to\alpha)}\cdot \\ &\qquad in_{G_p}(\alpha). \end{aligned}$$

As already discussed, under the assumption that G_p is proper and consistent we have $in_{G_p}(\alpha) = 1$ for every α. Thus we can write

$$H_d(G_p) = \sum_A out_{G_p}(A)\cdot H(A). \qquad (2)$$

The computation of $out_{G_p}(A)$ was discussed in Section 3, and also $H(A)$ can easily be calculated.

Under the restrictive assumption that a PCFG is proper and consistent, the characterization in (2) was already known from (Grenander, 1976, Theorem 10.7, pp. 90–92). The proof reported in that work is different from ours and uses a momentum matrix (Section 3). Our characterization above is more general and uses simpler notation than the one in (Grenander, 1976).

The *sentential entropy*, or *entropy* for short, of G_p is defined as the expectation of the information of the strings generated by G_p, i.e.,

$$\begin{aligned} H(G_p) &= E_{p_G}\ \log\frac{1}{p_G(w)} \\ &= \sum_w p_G(w)\cdot\log\frac{1}{p_G(w)}, \qquad (3) \end{aligned}$$

assuming $0\cdot\log\frac{1}{0} = 0$, for strings w not generated by G_p. It is not difficult to see that $H(G_p) \leq H_d(G_p)$ and equality holds if and only if G is unambiguous (Soule, 1974, Theorem 2.2). As ambiguity of CFGs is undecidable, it follows that we cannot hope to obtain a closed-form solution for $H(G_p)$ for which equality to (2) is decidable. We will return to this issue in Section 6.

5 Weighted intersection

In order to compute the cross-entropy defined in the next section, we need to derive a single probabilistic model that simultaneously accounts for both the computations of an underlying FA and the derivations of an underlying PCFG. We start from a construction originally presented in (Bar-Hillel et al., 1964), that computes the intersection of a context-free language and a regular language. The input consists of a CFG $G = (\Sigma, N, S, R)$ and an FA $M = (\Sigma, Q, q_0, Q_f, T)$; note that we assume, without loss of generality, that G and M share the same set of terminals Σ.

The output of the construction is CFG $G_\cap = (\Sigma, N_\cap, S_\cap, R_\cap)$, where $N_\cap = Q \times (\Sigma \cup N) \times Q \cup \{S_\cap\}$, and $R_\cap$ consists of the set of rules that is obtained as follows.

- For each $s \in Q_f$, let $S_\cap \to (q_0, S, s)$ be a rule of $G_\cap$.
- For each rule $A \to X_1 \cdots X_m$ of G and each sequence of states $s_0, \ldots, s_m$ of M, with $m \geq 0$, let $(s_0, A, s_m) \to (s_0, X_1, s_1) \cdots (s_{m-1}, X_m, s_m)$ be a rule of $G_\cap$; for $m = 0$, $G_\cap$ has a rule $(s_0, A, s_0) \to \epsilon$ for each state s_0.
- For each transition $s \overset{a}{\mapsto} t$ of M, let $(s, a, t) \to a$ be a rule of $G_\cap$.

Note that for each rule $(s_0, A, s_m) \to (s_0, X_1, s_1) \cdots (s_{m-1}, X_m, s_m)$ there is a unique rule $A \to X_1 \cdots X_m$ from which it has been constructed by the above. Similarly, each rule $(s, a, t) \to a$ uniquely identifies a transition $s \overset{a}{\mapsto} t$. This means that if we take a complete derivation $d_\cap$ in $G_\cap$, we can extract a sequence $h_1(d_\cap)$ of rules from G and a sequence $h_2(d_\cap)$ of transitions from M, where h_1 and h_2 are string homomorphisms that we define point-wise as

- $h_1(\pi_\cap) = \epsilon$, if $\pi_\cap$ is $S_\cap \to (q_0, S, s)$;
  $h_1(\pi_\cap) = \pi$, if $\pi_\cap$ is $(s_0, A, s_m) \to (s_0, X_1, s_1) \cdots (s_{m-1}, X_m, s_m)$ and π is $(A \to X_1 \cdots X_m)$;
  $h_1(\pi_\cap) = \epsilon$, if $\pi_\cap$ is $(s, a, t) \to a$;
- $h_2(\pi_\cap) = \epsilon$, if $\pi_\cap$ is $S_\cap \to (q_0, S, s)$;
  $h_2(\pi_\cap) = \tau$, if $\pi_\cap$ is $(s, a, t) \to a$ and τ is $s \overset{a}{\mapsto} t$;
  $h_2(\pi_\cap) = \epsilon$, if $\pi_\cap$ is $(s_0, A, s_m) \to (s_0, X_1, s_1) \cdots (s_{m-1}, X_m, s_m)$.

We define $h(d_\cap) = (h_1(d_\cap), h_2(d_\cap))$. It can be easily shown that if $S_\cap \overset{d_\cap}{\Rightarrow} w$ and $h(d_\cap) = (d, c)$, then for the same w we have $S \overset{d}{\Rightarrow} w$ and $(q_0, w) \overset{c}{\vdash} (s, \epsilon)$, some $s \in Q_f$. Conversely, if for some w, d and c we have $S \overset{d}{\Rightarrow} w$ and $(q_0, w) \overset{c}{\vdash} (s, \epsilon)$, some $s \in Q_f$, then there is precisely one derivation $d_\cap$ such that $h(d_\cap) = (d, c)$ and $S_\cap \overset{d_\cap}{\Rightarrow} w$.

As noted before by (Nederhof and Satta, 2003), this construction can be extended to apply to a PCFG $G_p = (G, p_G)$ and an FA M. The output is a PCFG $G_{\cap,p} = (G_\cap, p_{G_\cap})$, where $G_\cap$ is defined as above and $p_{G_\cap}$ is defined by:

- $p_{G_\cap}(S_\cap \to (q_0, S, s)) = 1$;
- $p_{G_\cap}((s_0, A, s_m) \to (s_0, X_1, s_1) \cdots (s_{m-1}, X_m, s_m)) = p_G(A \to X_1 \cdots X_m)$;
- $p_{G_\cap}((s, a, t) \to a) = 1$.

Note that $G_{\cap,p}$ is non-proper. More specifically, probabilities of rules with left-hand side $S_\cap$ or (s_0, A, s_m) might not sum to one. This is not a problem for the algorithms presented in this paper, as we have never assumed properness for our PCFGs. What is most important here is the following property of $G_{\cap,p}$. If $d_\cap$, d and c are such that $h(d_\cap) = (d, c)$, then $p_{G_\cap}(d_\cap) = p_G(d)$.

Let us now assume that M is deterministic. (In fact, the weaker condition of M being unambiguous is sufficient for our purposes, but unambiguity is not a very practical condition.) Given a string w and a transition $s \overset{a}{\mapsto} t$ of M we define $f(s \overset{a}{\mapsto} t; w)$ as the frequency (number of occurrences) of $s \overset{a}{\mapsto} t$ in the unique computation of M, if it exists, that accepts w; this frequency is 0 if w is not accepted by M. On the basis of the above construction of $G_{\cap,p}$ and of Section 3, we find

$$\begin{aligned}
&E_{p_G}\, f(s \overset{a}{\mapsto} t; y(d)) = \\
&\sum_d p_G(d) \cdot f(s \overset{a}{\mapsto} t; y(d)) = \\
&out_{G_{\cap,p}}((s,a,t)) \cdot p_{G_\cap}((s,a,t) \to a) \cdot in_{G_{\cap,p}}(a) = \\
&out_{G_{\cap,p}}((s,a,t))
\end{aligned} \tag{4}$$

6 Kullback-Leibler distance

In this section we consider the Kullback-Leibler distance between a PCFGs and a PFA, and present a method for its optimization under certain assumptions. Let $G_p = (G, p_G)$ be a consistent PCFG and let $M_p = (M, p_M)$ be a consistent PFA. We demand that M be deterministic (or more generally, unambiguous). Let us first assume that $L(G) \subseteq L(M)$; we will later drop this constraint.

The *cross-entropy* of G_p and M_p is defined as usual for probabilistic models, viz. as the expectation under distribution p_G of the information of the strings generated by M, i.e.,

$$\begin{aligned} H(G_p \,||\, M_p) &= E_{p_G} \log \frac{1}{p_M(w)} \\ &= \sum_w p_G(w) \cdot \log \frac{1}{p_M(w)}. \end{aligned}$$

The Kullback-Leibler distance of G_p and M_p is defined as

$$\begin{aligned} D(G_p \,||\, M_p) &= E_{p_G} \log \frac{p_G(w)}{p_M(w)} \\ &= \sum_w p_G(w) \cdot \log \frac{p_G(w)}{p_M(w)}. \end{aligned}$$

Quantity $D(G_p \,||\, M_p)$ can also be expressed as the difference between the cross-entropy of G_p and M_p and the entropy of G_p, i.e.,

$$D(G_p \,||\, M_p) = H(G_p \,||\, M_p) - H(G_p). \quad (5)$$

Let $G_{\cap,p}$ be the PCFG obtained by intersecting G_p with the non-probabilistic FA M underlying M_p, as in Section 5. Using (4) the cross-entropy of G_p and M_p can be expressed as

$$\begin{aligned} & H(G_p \,||\, M_p) = \\ & \sum_w p_G(w) \cdot \log \frac{1}{p_M(w)} = \\ & \sum_d p_G(d) \cdot \log \frac{1}{p_M(y(d))} = \\ & \sum_d p_G(d) \cdot \log \prod_{s \overset{a}{\mapsto} t} \left(\frac{1}{p_M(s \overset{a}{\mapsto} t)} \right)^{f(s \overset{a}{\mapsto} t; y(d))} = \\ & \sum_d p_G(d) \cdot \sum_{s \overset{a}{\mapsto} t} f(s \overset{a}{\mapsto} t; y(d)) \cdot \log \frac{1}{p_M(s \overset{a}{\mapsto} t)} = \\ & \sum_{s \overset{a}{\mapsto} t} \log \frac{1}{p_M(s \overset{a}{\mapsto} t)} \cdot \sum_d p_G(d) \cdot f(s \overset{a}{\mapsto} t; y(d)) = \\ & \sum_{s \overset{a}{\mapsto} t} \log \frac{1}{p_M(s \overset{a}{\mapsto} t)} \cdot E_{p_G} f(s \overset{a}{\mapsto} t; y(d)) = \\ & \sum_{s \overset{a}{\mapsto} t} \log \frac{1}{p_M(s \overset{a}{\mapsto} t)} \cdot out_{G_{\cap,p}}((s,a,t)). \end{aligned}$$

We can combine the above with (5) to obtain

$$\begin{aligned} & D(G_p \,||\, M_p) = \\ & \sum_{s \overset{a}{\mapsto} t} out_{G_{\cap,p}}((s,a,t)) \cdot \log \frac{1}{p_M(s \overset{a}{\mapsto} t)} - H(G_p). \end{aligned}$$

The values of $out_{G_{\cap,p}}$ can be calculated easily, as discussed in Section 3. Computation of $H(G_p)$ in closed-form is problematic, as already pointed out in Section 4. However, for many purposes computation of $H(G_p)$ is not needed.

For example, assume that the non-probabilistic FA M underlying M_p is given, and our goal is to measure the distance between G_p and M_p, for different choices of p_M. Then the choice that minimizes $H(G_p \,||\, M_p)$ determines the choice that minimizes $D(G_p \,||\, M_p)$, irrespective of $H(G_p)$. Formally, we can use the above characterization to compute

$$\begin{aligned} p_M^* &= \underset{p_M}{\operatorname{argmax}} \; D(G_p \,||\, M_p) \\ &= \underset{p_M}{\operatorname{argmax}} \, H(G_p \,||\, M_p). \end{aligned}$$

When $L(G) - L(M)$ is non-empty, both $D(G_p \,||\, M_p)$ and $H(G_p \,||\, M_p)$ are undefined, as their definitions imply a division by $p_M(w) = 0$ for $w \in L(G) - L(M)$. In cases where the non-probabilistic FA M is given, and our goal is to compare the relative distances between G_p and M_p for different choices of p_M, it makes sense to ignore strings in $L(G) - L(M)$, and define $D(G_p \,||\, M_p)$, $H(G_p \,||\, M_p)$ and $H(G_p)$ on the domain $L(G) \cap L(M)$. Our equations above then still hold. Note that strings in $L(M) - L(G)$ can be ignored since they do not contribute non-zero values to $D(G_p \,||\, M_p)$ and $H(G_p \,||\, M_p)$.

7 Conclusions

We have discussed the computation of the KL distance between PCFGs and deterministic PFAs. We have argued that exact computation is difficult in general, but for determining the relative qualities of different PFAs, with respect to their closeness to an input PCFG, it suffices to compute the cross-entropy. We have shown that the cross-entropy between a PCFG and a deterministic PFA can be computed exactly.

These results can also be used for comparing a pair of PFAs, one of which is deterministic. Generalization of PCFGs to probabilistic tree-adjoining grammars (PTAGs) is also possible, by means of the intersection of a PTAG and a PFA, along the lines of (Lang, 1994).

Acknowledgements

Helpful comments from Zhiyi Chi are gratefully acknowledged. The first author is supported by the PIONIER Project *Algorithms for Linguistic Processing*, funded by NWO (Dutch Organization for Scientific Research). The second author is partially supported by MIUR (Italian Ministry of Education) under project PRIN No. 2003091149_005.

References

J.K. Baker. 1979. Trainable grammars for speech recognition. In J.J. Wolf and D.H. Klatt, editors, *Speech Communication Papers Presented at the 97th Meeting of the Acoustical Society of America*, pages 547–550.

Y. Bar-Hillel, M. Perles, and E. Shamir. 1964. On formal properties of simple phrase structure grammars. In Y. Bar-Hillel, editor, *Language and Information: Selected Essays on their Theory and Application*, chapter 9, pages 116–150. Addison-Wesley.

T.L. Booth and R.A. Thompson. 1973. Applying probabilistic measures to abstract languages. *IEEE Transactions on Computers*, C-22(5):442–450, May.

A. Corazza, R. De Mori, R. Gretter, and G. Satta. 1991. Computation of probabilities for an island-driven parser. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(9):936–950.

M. Falkhausen, H. Reininger, and D. Wolf. 1995. Calculation of distance measures between Hidden Markov Models. In *Proceedings of Eurospeech '95*, pages 1487–1490, Madrid.

U. Grenander. 1976. *Lectures in Pattern Theory, Vol. I: Pattern Synthesis*. Springer-Verlag.

J.E. Hopcroft and J.D. Ullman. 1979. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley.

S.E. Hutchins. 1972. Moments of strings and derivation lengths of stochastic context-free ggrammars. *Information Sciences*, 4:179–191.

B.-H. Juang and L.R. Rabiner. 1985. A probabilistic distance measure for hidden Markov models. *AT&T Technical Journal*, 64(2):391–408.

D. Jurafsky, C. Wooters, G. Tajchman, J. Segal, A. Stolcke, E. Fosler, and N. Morgan. 1994. The Berkeley Restaurant Project. In *Proceedings of the International Conference on Spoken Language Processing (ICSLP-94)*, pages 2139–2142, Yokohama, Japan.

B. Lang. 1994. Recognition can be harder than parsing. *Computational Intelligence*, 10(4):486–494.

K. Lari and S.J. Young. 1990. The estimation of stochastic context-free grammars using the Inside-Outside algorithm. *Computer Speech and Language*, 4:35–56.

K. Lari and S.J. Young. 1991. Applications of stochastic context-free grammars using the Inside-Outside algorithm. *Computer Speech and Language*, 5:237–257.

M. Mohri and M.-J. Nederhof. 2001. Regular approximation of context-free grammars through transformation. In J.-C. Junqua and G. van Noord, editors, *Robustness in Language and Speech Technology*, pages 153–163. Kluwer Academic Publishers.

M. Mohri. 1997. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2):269–311.

M.-J. Nederhof and G. Satta. 2003. Probabilistic parsing as intersection. In *8th International Workshop on Parsing Technologies*, pages 137–148, LORIA, Nancy, France, April.

M.-J. Nederhof. 2000. Practical experiments with regular approximation of context-free languages. *Computational Linguistics*, 26(1):17–44.

M. Rimon and J. Herz. 1991. The recognition capacity of local syntactic constraints. In *Fifth Conference of the European Chapter of the Association for Computational Linguistics, Proceedings of the Conference*, pages 155–160, Berlin, Germany, April.

S. Soule. 1974. Entropies of probabilistic grammars. *Information and Control*, 25:57–74.

A. Stolcke and J. Segal. 1994. Precise *N*-gram probabilities from stochastic context-free grammars. In *32nd Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*, pages 74–79, Las Cruces, New Mexico, USA, June.

M. Vihola, M. Harju, P. Salmela, J. Suontausta, and J. Savela. 2002. Two dissimilarity measures for HMMs and their application in phoneme model clustering. In *ICASSP 2002*, volume I, pages 933–936.

C.S. Wetherell. 1980. Probabilistic languages: A review and some open questions. *Computing Surveys*, 12(4):361–379, December.