On the Derivational Entropy of Left-to-Right Probabilistic Finite-State Automata and Hidden Markov Models

Joan Andreu Sánchez*
Universitat Politècnica de València

Martha Alicia Rocha**
Instituto Tecnológico de León

Verónica Romero*
Universitat Politècnica de València

Mauricio Villegas†
SearchInk

Probabilistic finite-state automata are a formalism that is widely used in many problems of automatic speech recognition and natural language processing. Probabilistic finite-state automata are closely related to other finite-state models as weighted finite-state automata, word lattices, and hidden Markov models. Therefore, they share many similar properties and problems. Entropy measures of finite-state models have been investigated in the past in order to study the information capacity of these models. The derivational entropy quantifies the uncertainty that the model has about the probability distribution it represents. The derivational entropy in a finite-state automaton is computed from the probability that is accumulated in all of its individual state sequences. The computation of the entropy from a weighted finite-state automaton requires a normalized model. This article studies an efficient computation of the derivational entropy of left-to-right probabilistic finite-state automata, and it introduces an efficient algorithm for normalizing weighted finite-state automata. The efficient computation of the derivational entropy is also extended to continuous hidden Markov models.

* Camí de Vera, s/n, 46022 València, Spain. E-mail: {jandreu, vromero}@prhlt.upv.es.
** mrocha@dsic.upv.es.
† mauricio@searchink.com.

Submission received: 11 May 2016, revised version received: 15 July 2017, accepted for publication: 28 August 2017.

doi:10.1162/COLI_a_00306

1. Introduction

Probabilistic Finite-State Automata (PFA) and hidden Markov models (HMM) are well-known formalisms that have been widely used in automatic speech recognition (Ortmanns and Ney 1997), machine translation (Ueffing, Och, and Ney 2002), natural language processing (Mohri, Pereira, and Riley 2002), and, more recently, in handwritten text recognition (Romero, Toselli, and Vidal 2012). PFA and HMM can be considered special cases of weighted finite-state automata (WFA) (Mohri, Pereira, and Riley 2002; Dupont, Denis, and Esposito 2005). PFA and HMM were extensively researched in Vidal et al. (2005) and interesting probabilistic properties were demonstrated. In formal language theory, automata are considered as string acceptors, but PFA may be considered as generative processes (see Section 2.2 in Vidal et al. [2005]). We have followed the point of view of Vidal et al. (2005) in this article about this issue. PFA are closely related to word lattices (WL) (Tomita 1986), which are currently a fundamental tool for many applications because WL convey most of the hypotheses produced by a decoder. WL have also been used for parsing (Tomita 1986), for computing confidence measures in speech recognition (Kemp and Schaaf 1997; Ortmanns and Ney 1997; Sanchis, Juan, and Vidal 2012), machine translation (Ueffing, Och, and Ney 2002), and handwritten text recognition (Puigcerver, Toselli, and Vidal 2014) for interactive transcription (Toselli, Vidal, and Casacuberta 2011) and term detection (Can and Saraçlar 2011). All of these applications require the WL to be correctly defined, and therefore, knowing the stochastic properties of WL becomes very important.

This article deals with entropy-based measures that are computed for PFA and HMM. Entropy measures give account of their expressiveness, and, therefore, the computation of measures of this type is a fundamental problem related to these models. Entropy measures are fundamental to studying the uncertainty that a model has about the distribution it represents. The concepts of sequence entropy (also known as **sentential entropy**) and derivational entropy for formal grammars were introduced in Grenander (1967). Given a finite-state automaton $\mathcal{A}$, the sequence entropy[1] of the model is defined as:[2]

$$H(\mathcal{A}) = -\sum_{w \in L(\mathcal{A})} p_{\mathcal{A}}(w) \log p_{\mathcal{A}}(w) \tag{1}$$

where $L(\mathcal{A})$ is the set of all word sequences generated by $\mathcal{A}$. Note that this expression requires that the condition $\sum_{w \in L(\mathcal{A})} p_{\mathcal{A}}(w) = 1.0$ must be fulfilled. The concept of sequence entropy of the model is different from the concept of the entropy given an observation sequence w, which is defined in Hernando, Crespi, and Cybenko (2005):

$$H_{\mathcal{A}}(\theta|w) = -\sum_{\theta \in \Theta_{\mathcal{A}}(w)} p_{\mathcal{A}}(\theta|w) \log p_{\mathcal{A}}(\theta|w) \tag{2}$$

1 Although the concrete formal notation used in this article will be introduced in Section 2, in this introduction we assume that the reader is familiar with some concepts related to finite-state automata theory.

2 Throughout this article, we assume that $0 \log 0 = 0$. In addition, logarithm to base 2 is used in this article.

where $\Theta_{\mathcal{A}}(w)$ is the set of all state sequences in the model $\mathcal{A}$ starting in an initial state and reaching a final state such that each state sequence θ accounts for the string w. Note that Equation (2) requires that the condition $\sum_{\theta \in \Theta_{\mathcal{A}}(w)} p_{\mathcal{A}}(\theta|w) = 1.0$ must be fulfilled.

The derivational entropy is defined as in Grenander (1967):

$$H_d(\mathcal{A}) = -\sum_{\theta \in \Theta(\mathcal{A})} p_{\mathcal{A}}(\theta) \log p_{\mathcal{A}}(\theta) \tag{3}$$

where $\Theta(\mathcal{A})$ is the set of all state sequences in the model $\mathcal{A}$ starting in an initial state and reaching a final state. Note that this expression requires that the condition $\sum_{\theta \in \Theta(\mathcal{A})} p_{\mathcal{A}}(\theta) = 1.0$ must be fulfilled. If the model is a PFA, then the derivational entropy is computed from the probability that is accumulated in all the state sequences of the automaton rather than the probability that is accumulated in the strings generated by the automaton (Soule 1974). Figure 1 shows an example of a simple PFA with initial state labeled as 0 and final state labeled as 4. In the case that the automaton is a WL, it also has time information about when each word starts and ends. In this PFA $\mathcal{A}$, the sequence entropy is defined as:

$$\begin{aligned} H(\mathcal{A}) &= -p_{\mathcal{A}}(aa) \log p_{\mathcal{A}}(aa) - p_{\mathcal{A}}(bb) \log p_{\mathcal{A}}(bb) = -0.8 \log 0.8 - 0.2 \log 0.2 \\ &= 0.72 \text{ bits} \end{aligned}$$

The sequence entropy given the string *aa* is:

$$\begin{aligned} H_{\mathcal{A}}(\theta|aa) &= -p_{\mathcal{A}}(034|aa) \log p_{\mathcal{A}}(034|aa) - p_{\mathcal{A}}(014|aa) \log p_{\mathcal{A}}(014|aa) \\ &= -\frac{0.5}{0.8} \log \frac{0.5}{0.8} - \frac{0.3}{0.8} \log \frac{0.3}{0.8} = 0.95 \text{ bits} \end{aligned}$$

Finally, the derivational entropy is:

$$\begin{aligned} H_d(\mathcal{A}) &= -p_{\mathcal{A}}(034) \log p_{\mathcal{A}}(034) - p_{\mathcal{A}}(014) \log p_{\mathcal{A}}(014) - p_{\mathcal{A}}(024) \log p_{\mathcal{A}}(024) \\ &= -0.5 \log 0.5 - 0.3 \log 0.3 - 0.2 \log 0.2 = 1.49 \text{ bits} \end{aligned}$$

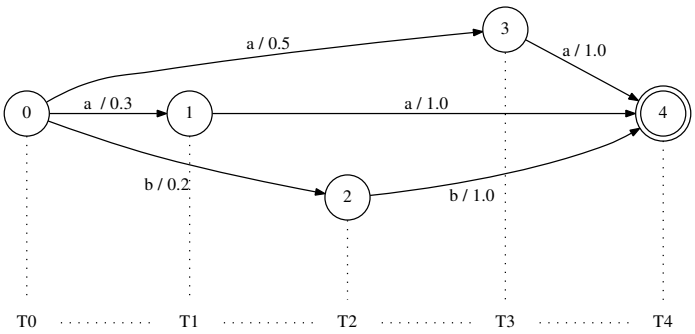

Figure 1
Example of a PFA. The dotted lines represent time information if we consider this model as a WL.

Note that if we consider the graph in Figure 1 as a WL produced by a decoder, the lower the sequence entropy is, the better, because it means that the decoder has low uncertainty about the output. A similar reasoning can be applied to the opposite—that is, the lower the uncertainty about the output, the lower the sequence entropy is. In the case of the derivational entropy, the lower it is, the lower the decoder's uncertainty about the output. However, the opposite is not necessarily true. Unfortunately, the computation of Equation (1) is an open problem. Nevertheless, the sequence entropy (1) is upper-bounded by the derivational entropy (Soule 1974), and, therefore, the lower the derivational entropy, the lower the sequence entropy. This makes the computation of the derivational entropy a very interesting problem. The computation of Equation (2) was studied in Hernando, Crespi, and Cybenko (2005). In this article, we will focus on the efficient computation of Equation (3) for PFA and HMM. If the finite-state model is considered as a WL, then it has to be normalized because the WL produced by a decoder is not necessarily normalized.

An algorithm that is based on a matrix inversion was introduced in Grenander (1967) for computing Equation (3), and, therefore, the time complexity is cubic with the number of states. Note that in the case of WL, the number of states can be thousands in practical situations, and therefore this solution is not interesting in practice. There is also interesting research related to the efficient computation of Equation (2). As mentioned, an efficient computation with HMM was proposed in Hernando, Crespi, and Cybenko (2005), and a generalized version was explored in Ilic (2011). When the word sequence is partially known, Mann and McCallum (2007) provide an algorithm for computing the entropy with conditional random fields. The derivational entropy of a probabilistic context-free grammar was investigated in Corazza and Satta (2007), and it was demonstrated that the derivational entropy coincides with the cross-entropy when the cross-entropy is used as objective function for probabilistic estimation of the probabilities of the rules. The computation of the derivational entropy for finite-state models was studied in Nederhof and Satta (2008), but an approximate solution was stated as the solution of a linear system of equations. Note that, in Hernando, Crespi, and Cybenko (2005), Mann and McCallum (2007), and Ilic (2011), the computation of Equation (2) is studied, whereas in this article the computation of Equation (3) is studied, as in Grenander (1967).

Cubic algorithms with respect to the number of states have been proposed for computing the derivational entropy of PFA without restrictions (Corazza and Satta 2007; Nederhof and Satta 2008). This article presents an algorithm that is asymptotically more efficient for left-to-right PFA (and therefore for WL), that is, linear with respect to the number of edges. This algorithm is then extended to left-to-right HMM. If the PFA is obtained from a WFA, like a WL, then it has to be adequately normalized for computing the derivational entropy. In this article, we adopt the normalization described in Thompson (1974), and we improve its computation for left-to-right PFA. The proposed normalization guarantees that the relative weights of the state sequences after the normalization are preserved. Normalization of WFA and probabilistic grammars have been studied in different articles. Thompson (1974) proposed the normalization of probabilistic grammars, on which our normalization technique is based. Normalization of WFA has also been studied in Mohri (2009), who proposed the *weight pushing* algorithm. Grammar normalization is also investigated in Chi (1999) and Abney, McAllester, and Pereira (1999). The normalization in WL is also a necessary process in order to compute word confidence measures at the frame level for many purposes (Wessel et al. 2001).

This article is organized as follows: Section 2 specifies the notation related to PFA, and the computation of like-forward and like-backward probabilities from these

models. These forward and backward probabilities are not the classical probabilities and they are necessary for subsequent computations. In the case of models that may not be normalized, such as WL produced by a decoder, they need to be normalized before computing the derivational entropy. This normalization is explained in Section 3. At the end of Section 3, we include a discussion on related normalization techniques. Section 4 explains the main contribution of this article—namely, the efficient computation of the derivational entropy. Section 5 extends the computation of the derivational entropy to continuous HMM.

2. Left-to-Right Probabilistic Finite-State Automata

We introduce the notation related to PFA that will be used in this article following Vidal et al. (2005).

Definition 1. A *PFA* is a tuple $\mathcal{A} = \langle Q, \Sigma, \delta, I, F, P \rangle$, where: Q is a finite set of states; Σ is the alphabet; $\delta \subseteq Q \times \Sigma \times Q$ is a set of transitions; $I : Q \to \mathbb{R}^{\geq 0}$ is the probability function of a state being an initial state; $P : \delta \to \mathbb{R}^{\geq 0}$ is a probability function of transition between states; and $F : Q \to \mathbb{R}^{\geq 0}$ is the probability function of a state being a final state. I, P, and F are functions such that:

$$\sum_{i \in Q} I(i) = 1 \tag{4}$$

$$\forall i \in Q, F(i) + \sum_{v \in \Sigma, j \in Q} P(i, v, j) = 1 \tag{5}$$

For the sake of notation simplification, P is assumed to be extended with $P(i, v, j) = 0$ for all $(i, v, j) \notin \delta$. An automaton is said to be **proper** if it satisfies Equation (5).

In this article, we will assume that all states are nominated by integers from 0 to $|Q| - 1$. For simplicity, we assume that the PFA have only one initial state, named 0, and, therefore, the sum in Equation (4) has only one term. We assume without loss of generality that the PFA has only one final state, named $|Q| - 1$, without loops. These assumptions greatly simplify the notation. For practical reasons, and also for simplifying the notation, we assume that the empty string is not in $L(\mathcal{A})$. This last assumption implies that PFA with only one state are not allowed.

In order to deal with the probability of the finite-length strings generated by a PFA, we introduce the following notation. Let $\theta_{\mathcal{A}} = (i_0, v_1, i_1, v_2, i_2, \ldots, i_{k-1}, v_k, i_k)$ be a path in $\mathcal{A}$ for the string w whose length is k; that is, there is a sequence of transitions $(i_0, v_1, i_1), (i_1, v_2, i_2), \ldots, (i_{k-1}, v_k, i_k) \in \delta$ such that $w = v_1 v_2 \ldots v_k$. We do not consider empty paths and therefore $k \geq 1$. The probability of generating such a path is:

$$p_{\mathcal{A}}(\theta_{\mathcal{A}}) = \left(\prod_{j=1}^{k} P(i_{j-1}, v_j, i_j) \right) = \prod_{(i,v,j) \in \delta} P(i, v, j)^{\mathrm{N}((i,v,j), \theta_{\mathcal{A}})} \tag{6}$$

where $\mathrm{N}((i, v, j), \theta_{\mathcal{A}})$ is the number of times that the transition (i, v, j) has been used in $\theta_{\mathcal{A}}$.

Definition 2. A **valid path** in a PFA $\mathcal{A}$ is a path $\theta_{\mathcal{A}} = (i_0 = 0, v_1, i_1, v_2, i_2, \ldots, i_{k-1}, v_k, i_k = |Q| - 1)$. for some $w = v_1 v_2 \ldots v_k$. The set of valid paths in $\mathcal{A}$ will be denoted as $\Theta_{\mathcal{A}}$.

The probability of generating w with $\mathcal{A}$ is $p_{\mathcal{A}}(w) = \sum_{\theta \in \Theta_{\mathcal{A}}(w)} p_{\mathcal{A}}(\theta)$, where $\Theta_{\mathcal{A}}(w)$ is the set of all valid paths in $\mathcal{A}$ for a given w. The language generated by a PFA $\mathcal{A}$ is $L(\mathcal{A}) = \{w : p_{\mathcal{A}}(w) > 0\}$. A PFA is said to be consistent if $\sum_{w \in L(\mathcal{A})} p_{\mathcal{A}}(w) = 1$. A PFA is consistent if its states are useful, that is, all states appear in at least one valid path (Vidal et al. 2005).

Definition 3. A **left-to-right PFA**[3] is defined as a PFA such that each state can have loops but it has no cycles, and if the loops are removed then a directed acyclic automaton is obtained. The final state has no loops.

A left-to-right PFA has an equivalent *left-to-right* HMM[4] representation and vice versa, and they share the characteristic that once a state i has been reached, then it is not possible to go back to state j such that $j < i$. If the loops are removed in a left-to-right PFA, then a topological order might be induced on the resulting PFA. In the induced loop-free PFA, we call this order *pseudo*-topological order.[5] From now on, we consider left-to-right PFA.

From Definition 2, we define $\Theta_{\mathcal{A}}(i, j, l)$ as the set of all paths starting in i, and ending in j with l ($l \geq 1$) different states (i and j inclusive). Note that l is not the length of the path, but the number of states that are different in each path from i to j. For example, in the PFA in Figure 2, the two paths $(0, a, 1, a, 2)$ and $(0, a, 0, a, 1, a, 1, b, 2)$ belong to $\Theta_{\mathcal{A}}(0, 2, 3)$, because both start in state 0, arrive to state 2, and use three different states: $0, 1$, and 2. But their lengths are different. The generated string when traversing a path is not relevant for the computations involved in this article, so, to simplify, the symbol information is omitted in subsequent definitions. We define the following probability:

$$p_{\mathcal{A}}(\Theta_{\mathcal{A}}(0, i, l)) = \sum_{\theta \in \Theta_{\mathcal{A}}(0,i,l)} p_{\mathcal{A}}(\theta) \tag{7}$$

as the probability that is accumulated in all prefixes generated by $\mathcal{A}$ going through l different states and ending in state i. This value is necessary for normalizing a WFA, as we describe subsequently. A similar expression to Equation (7) can be defined for suffixes:

$$p_{\mathcal{A}}(\Theta_{\mathcal{A}}(i, |Q| - 1, l)) = \sum_{\theta \in \Theta_{\mathcal{A}}(i,|Q|-1,l)} p_{\mathcal{A}}(\theta) \tag{8}$$

is the probability that is accumulated in all suffixes generated by $\mathcal{A}$ traversing l different states and starting in state i. This expression is also necessary for normalizing a WFA. Note that Equation (8) makes sense for $l \geq 2$ because for $l = 1$ we have that $\Theta_{\mathcal{A}}(|Q| - 1, |Q| - 1, 1)$ is empty, since we assumed only one final state without loops. When $l = |Q|$ then $p_{\mathcal{A}}(\Theta_{\mathcal{A}}(0, |Q| - 1, |Q|)) = 1.0$.

For subsequent computations, the computation of the probability of all paths (substrings) that can be composed from the loops of a single state is needed. If the state has just one loop, then the computation is trivial. Given a state of a left-to-right PFA that has two loops, each of which is labeled with a different symbol (like state 2 in Figure 2)

3 PFA of this type are also known as Bakis models (Bakis 1976).
4 HMM will be defined formally in Section 5.
5 Note that different *pseudo*-topological orders can exist, but this is not relevant in this article. We will consider just one of them.

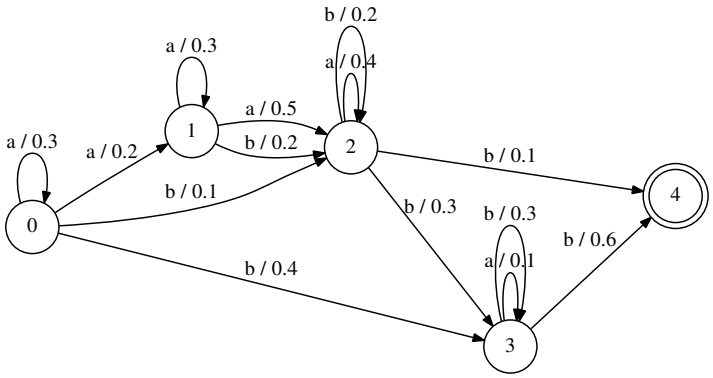

Figure 2
Example of a left-to-right PFA.

the substrings that can be composed using the loops in state 2 in increasing length and their probability is:

$$p(a) + p(b) = 0.4 + 0.2$$
$$p(aa) + p(ab) + p(ba) + p(bb) = 0.4^2 + 2 \cdot 0.4 \cdot 0.2 + 0.2^2$$
$$p(aaa) + p(aab) + p(aba) + p(abb) + p(baa) + p(bab) + p(bba) + p(bbb) = 0.4^3 + 3 \cdot 0.4^2 \cdot 0.2 + 3 \cdot 0.4 \cdot 0.2^2 + 0.4^3$$
$$\dots$$

Note that the previous terms are related to binomial coefficients and the product of a polynomial with two terms (there are two terms because there are two loops, each of which has a different symbol). If we add all these terms for all possible lengths, including the empty string, and we call r, ($r < 1$), the addition of all loop probabilities in the same state, then we obtain:

$$\sum_{n=0}^{+\infty} (P(2,a,2) + P(2,b,2))^n = \sum_{n=0}^{+\infty} r^n = \frac{1}{1-r}$$

In the previous expression, when $n = 0$, no loop in state is used and the probability of paths that go through the state is multiplied by 1 so its value is not changed.

If there are three loops with different symbols (e.g. $\{a, b, c\}$), with probabilities $a_1 = P(i,a,i)$, $b_1 = P(i,b,i)$, and $c_1 = P(i,c,i)$, respectively, then the result is analogous:

$$p(a) + p(b) + p(c) = a_1 + b_1 + c_1$$
$$p(aa) + p(ab) + p(ac) + p(ba) + p(bb) + p(bc) + p(ca) + p(cb) + p(cc) = a_1 + b_1 + c_1 + 2a_1b_1 + 2a_1c_1 + 2b_1c_1$$
$$\dots$$

In general, we have that the addition of the probabilities of all paths using the loops of a single state i, $0 \leq i < |Q| - 1$, is:

$$\sum_{n=0}^{+\infty} \left(\sum_{\substack{v \in \Sigma: \\ P(i,v,i) \neq 0}} P(i,v,i) \right)^n = \sum_{n=0}^{+\infty} r^n = \frac{1}{1-r}$$

Proposition 1. For any i, $0 \leq i < |Q|$, it is fulfilled that $\sum_{l=2}^{|Q|-i} p_{\mathcal{A}}(\Theta_{\mathcal{A}}(i, |Q| - 1, l)) = 1$.

Proof. Note that i affects the number of different states in the paths from i to $|Q| - 1$, and this is the reason why the previous sum ranges from $l = 2$ to $|Q| - 1 - i + 1 = |Q| - i$.

The proof is by induction on l and following the nodes in the reverse of the *pseudo*-topological order. Note that we start the proof for $i = |Q| - 2$ because the final state has no loops. In this case, the sum ranges from $l = 2$ to $|Q| - (|Q| - 2)) = 2$:

$$\sum_{l=2}^{2} p_{\mathcal{A}}(\Theta_{\mathcal{A}}(|Q| - 2, |Q| - 1, l)) = \sum_{\theta \in \Theta_{\mathcal{A}}(|Q|-2, |Q|-1, 2)} p_{\mathcal{A}}(\theta) = \frac{1}{1 - r_{|Q|-2}} (1 - r_{|Q|-2}) = 1$$

The last product is the probability of all paths generated through the loops in state $|Q| - 2$ times the transitions to the final state.

For i in the *pseudo*-topological order, such that $0 \leq i < |Q| - 2$,

$$\begin{aligned}
&\sum_{l=2}^{|Q|-i} p_{\mathcal{A}}(\Theta_{\mathcal{A}}(i, |Q| - 1, l)) = \sum_{l=2}^{|Q|-i} \sum_{\theta \in \Theta_{\mathcal{A}}(i, |Q|-1, l)} p_{\mathcal{A}}(\theta) \\
&= \sum_{\substack{i < j \leq |Q|-1 \\ v \in \Sigma}} \frac{1}{1 - r_i} P(i, v, j) \sum_{l=2}^{|Q|-j} \sum_{\theta \in \Theta_{\mathcal{A}}(j, |Q|-1, l)} p_{\mathcal{A}}(\theta) \\
&= \sum_{\substack{i < j \leq |Q|-1 \\ v \in \Sigma}} \frac{1}{1 - r_i} P(i, v, j) \underbrace{\sum_{l=2}^{|Q|-j} p_{\mathcal{A}}(\Theta_{\mathcal{A}}(j, |Q| - 1, l))}_{1} = \frac{1}{1 - r_i} (1 - r_i) = 1
\end{aligned}$$

□

The proposition establishes that given a consistent PFA, then any PFA induced from a node and the nodes reachable from it and the corresponding edges is also consistent.

We now describe how Equation (7) can be computed. This computation is similar to the classical *forward* computation proposed for HMM (Vidal et al. 2005). We define $\widehat{\alpha}_{\mathcal{A}}(i, l)$, $0 \leq i \leq |Q| - 1$ and $1 \leq l \leq |Q|$ as the probability accumulated in all prefixes with paths each of which has l different states and reaching state

i: $\widehat{\alpha}_{\mathcal{A}}(i,l) = p_{\mathcal{A}}(\Theta_{\mathcal{A}}(0,i,l))$. The computation of $\widehat{\alpha}_{\mathcal{A}}(\cdot,\cdot)$ can be performed with this new *forward* algorithm:

$$\widehat{\alpha}_{\mathcal{A}}(0,1) = \frac{1}{1-r_0}$$

$$\widehat{\alpha}_{\mathcal{A}}(i,l) = \sum_{\theta \in \Theta_{\mathcal{A}}(0,i,l)} p_{\mathcal{A}}(\theta) = \sum_{\substack{v \in \Sigma, \\ 0 \le j < i}} \widehat{\alpha}_{\mathcal{A}}(j,l-1) P(j,v,i) \frac{1}{1-r_i}$$

$$= \frac{1}{1-r_i} \sum_{\substack{v \in \Sigma, \\ 0 \le j < i}} \widehat{\alpha}_{\mathcal{A}}(j,l-1) P(j,v,i) \qquad 1 < l \le |Q|, l-1 \le i \le |Q|-1$$

where r_i is the addition of all loop probabilities in state i. If $i = |Q|-1$, then $r_{|Q|-1} = 0$.

Proposition 2. $\sum_{l=1}^{|Q|} \widehat{\alpha}_{\mathcal{A}}(|Q|-1,l) = 1$. □

Similar results can be obtained for suffixes that are generated by $\mathcal{A}$ as can be seen in Equation (8). We define for $0 \le i \le |Q|-1$ and $2 \le l \le |Q|$,

$$\widehat{\beta}_{\mathcal{A}}(i,l) = p_{\mathcal{A}}(\Theta_{\mathcal{A}}(i,|Q|-1,l)) \tag{9}$$

as the probability accumulated in all suffixes with paths that have l different states, start in state i, and reach the final state $|Q|-1$. This expression can be computed with this *backward* algorithm:

$$\widehat{\beta}_{\mathcal{A}}(|Q|-1,1) = 1 \qquad \text{for completeness,}$$

$$\widehat{\beta}_{\mathcal{A}}(i,l) = \frac{1}{1-r_i} \sum_{\substack{v \in \Sigma, \\ i < j \le |Q|-1}} \widehat{\beta}_{\mathcal{A}}(j,l-1) P(i,v,j) \quad 1 < l \le |Q|, 0 \le i < |Q|-l\,.$$

Let us see an example of the computation of the $\widehat{\alpha}_{\mathcal{A}}(\cdot,\cdot)$ and $\widehat{\beta}_{\mathcal{A}}(\cdot,\cdot)$ values with the left-to-right PFA in Figure 2. First, we show the computation of the $\widehat{\alpha}_{\mathcal{A}}(\cdot,\cdot)$ values:

	1	2	3	4	5
0	1.429				
1		0.408			
2		0.357	0.714		
3		0.953	0.179	0.357	
4			0.608	0.179	0.214

Second, we show the computation of the $\widehat{\beta}_{\mathcal{A}}(\cdot,\cdot)$ values:

	1	2	3	4	5
0			0.607	0.179	0.214
1			0.250	0.750	
2		0.250	0.750		
3		1.000			
4	1.000				

Note that if we consider that each state is nominated with integers in increasing order according to the *pseudo*-topological order, then $\widehat{\alpha}_{\mathcal{A}}(\cdot,\cdot)$ have null values above the main diagonal. Something analogous can be demonstrated for $\widehat{\beta}_{\mathcal{A}}(\cdot,\cdot)$, but, in that case, the null values are below the other diagonal.

The following proposition can be demonstrated from Proposition 1.

Proposition 3. For all i, $0 \leq i \leq |Q| - 1$, we have:

$$\sum_{l=1}^{|Q|-i} \widehat{\beta}_{\mathcal{A}}(i,l) = 1 \,.$$

□

Values $\widehat{\alpha}_{\mathcal{A}}(\cdot,\cdot)$ and $\widehat{\beta}_{\mathcal{A}}(\cdot,\cdot)$ are related to values $in(\cdot)$ and $out(\cdot)$ in Nederhof and Satta (2008). These values can be efficiently computed with time complexity $O(|\delta|)$ following the *pseudo*-topological order defined on $\mathcal{A}$. Note that $|\delta|$ is at most $O(|Q|^2|\Sigma|)$.

3. Weighted Finite-State Automata Normalization

WFA are defined similarly to PFA, but the probability transition of PFA is substituted by a weight function.

Definition 4. A *WFA* is a tuple $\mathcal{A} = \langle Q, \Sigma, \delta, I_W, F_W, W \rangle$, where: Q is a finite set of states; Σ is the alphabet; $\delta \subseteq Q \times \Sigma \times Q$ is a set of transitions; $I_W : Q \to \mathbb{R}^{\geq 0}$ is the weight of a state being an initial state; $W : \delta \to \mathbb{R}^{\geq 0}$ is the weight associated with transitions between states; and $F_W : Q \to \mathbb{R}^{\geq 0}$ is the weight of a state being a final state.

As we previously did with PFA, we only consider WFA with one initial state and one final state. Each path through the WFA has an associated weight that is computed as the product of the weights of the edges of that path. In order to compute the derivational entropy as described in Section 4, it is necessary to guarantee that the automaton is consistent. Therefore, the weights of the edges in the WFA have to be normalized, because the normalization is a sufficient condition for the WFA to be consistent and to become a PFA if all states appear at least in one path.

It is a desirable property for the normalization process of a WFA to keep the relative weights of the paths unaffected once the WFA becomes a PFA. Note that this condition can be guaranteed only if the loops in each state give rise to an infinite addition that converges to some constant value. This is the case of WL, and, therefore, we only consider WFA for which this condition is fulfilled.

A similar normalization solution proposed in Thompson (1974) for probabilistic regular grammars is adopted in this article, and it is adapted for left-to-right finite-state automata.

The final vector ν (Definition 12 in Thompson [1974]) of a WFA is a $(|Q| - 1) \times 1$ column vector, where:

$$\nu(i) = \sum_{v \in \Sigma} W(i, v, |Q| - 1), \quad 0 \leq i < |Q| - 1$$

The following definition is related to a definition introduced in Thompson (1974) for probabilistic grammars.

Definition 5. The normalizing vector $\mathcal{N}$ of a WFA $\mathcal{A}$ is a $|Q| \times 1$ column vector where each term $\mathcal{N}(i), 0 \leq i < |Q| - 1$, is defined such that, if all transition weights $W(i, v, j) \in \delta$ are multiplied by $\mathcal{N}(j)/\mathcal{N}(i)$ with $\mathcal{N}(|Q| - 1) = 1$, then the WFA $\mathcal{A}$ is transformed into a proper PFA.

Definition 6. Given a WFA $\mathcal{A}$, we define the characteristic matrix M (Thompson 1974) of dimensions $|Q| \times |Q|$ as:

$$M(i, j) = \sum_{v \in \Sigma} W(i, v, j)$$

If $i = j$ then $M(i, i) = r_i$.

The following theorem for WFA is stated in Thompson (1974) for probabilistic regular grammars.

Theorem 1. Let $\mathcal{A}$ be a WFA and M, ν, and $\mathcal{N}$ be the corresponding characteristic matrix, final vector, and normalizing vector, where M and ν are known, and where the row and column of the characteristic matrix associated with state $|Q| - 1$ have been removed. Then $\mathcal{N}$ can be computed as:

$$\mathcal{N} = (I - M)^{-1}\nu \tag{10}$$

Proof. We have to demonstrate how to obtain Equation (10) and that the relative weights of the paths are unaffected. First, we demonstrate how to obtain Equation (10) as in Thompson (1974). By definition of $\mathcal{N}$, normalization takes place by changing the transition weights $W(i, v, j)$, $0 \leq i, j < |Q| - 1$, to $W(i, v, j)\mathcal{N}(j)/\mathcal{N}(i)$. When $j = |Q| - 1$, then $W(i, v, |Q| - 1)/\mathcal{N}(i)$. Then, the sum of the new weights have to add up to 1 for all i. That is:

$$\sum_{\substack{v \in \Sigma \\ 0 \leq j < |Q|-1}} W(i, v, j)\frac{\mathcal{N}(j)}{\mathcal{N}(i)} + \sum_{v \in \Sigma} W(i, v, |Q| - 1)\frac{1}{\mathcal{N}(i)} = 1$$

This yields

$$\sum_{\substack{v \in \Sigma \\ 0 \leq j < |Q|-1}} W(i, v, j)\mathcal{N}(j) + \sum_{v \in \Sigma} W(i, v, |Q| - 1) = \mathcal{N}(i)$$

$$\sum_{0 \leq j < |Q|-1} M(i, j)\mathcal{N}(j) + \nu(i) = \mathcal{N}(i)$$

that in matrix form is $M\mathcal{N} + \nu = \mathcal{N}$, which yields $\mathcal{N} = (I - M)^{-1}\nu$. Matrix $(I - M)^{-1}$ is upper triangular with non-null diagonal for left-to-right WFA.

Second, we demonstrate that the relative weights of the valid paths are unaffected. Suppose that $\mathcal{A}'$ is the normalized PFA obtained after normalizing a WFA $\mathcal{A}$. Then, for a given path θ in $\mathcal{A}'$ of size k:

$$p_{\mathcal{A}'}(\theta) = \prod_{i=1}^{k} P_{\mathcal{A}'}(s_{i-1}, v_i, s_i) = \prod_{i=1}^{k} \frac{W_{\mathcal{A}}(s_{i-1}, v_i, s_i)\mathcal{N}(s_i)}{\mathcal{N}(s_{i-1})}$$

$$= \frac{W_{\mathcal{A}}(0, v_1, s_1)\mathcal{N}(1)}{\mathcal{N}(0)} \cdots \frac{W_{\mathcal{A}}(s_{k-1}, v_k, |Q|-1)\overbrace{\mathcal{N}(|Q|-1)}^{1}}{\mathcal{N}(s_{k-1})}$$

$$= \frac{\prod_{i=1}^{k} W_{\mathcal{A}}(s_{i-1}, v_i, s_i)}{\mathcal{N}(0)} = \frac{\mathcal{W}_{\mathcal{A}}(\theta)}{\mathcal{W}_{\mathcal{A}}(\Theta_{\mathcal{A}}(0, |Q|-1, \cdot))}$$

The last equation is because $\mathcal{N}(0)$ is the weight accumulated in all paths in the WFA, $\mathcal{W}_{\mathcal{A}}(\Theta_{\mathcal{A}}(0, |Q|-1, \cdot))$ (as we will see below), and $\mathcal{W}_{\mathcal{A}}(\theta)$ is defined in a way similar to Equation (6). Therefore, the normalization keeps the relative weight distribution of the paths in the normalized PFA $\mathcal{A}'$. □

Note that $\mathcal{N}(|Q|-1) = 1$ according to Definition 5.

Expression (10) requires the computation of a matrix inversion. Note that this inverse matrix exists only if the condition about the convergence of the infinite additions in loops is satisfied. As we noted previously, this condition is satisfied for WL, which is the type of WFA that we are interested in. Let us see how $\mathcal{N}$ can be efficiently computed in a left-to-right automaton by avoiding the matrix inversion. Note that

$$(I - M)^{-1} = \sum_{k=0}^{\infty} M^k \tag{11}$$

Each $M^k(i, j)$ is in fact the addition of the weights of all paths from i to j ($i, j \neq |Q|-1$) with length k that have at most $k+1$ different states. Therefore, if $i = j$, $i \neq |Q|-1$, expression (11) becomes:

$$\sum_{k=0}^{\infty} M^k(i, i) = \sum_{k=0}^{\infty} r_i^k = \frac{1}{1 - r_i} \tag{12}$$

given that there are no cycles, only loops. If $i \neq j$, then Equation (11) becomes:

$$\sum_{k=0}^{\infty} M^k(i, j) = \sum_{k=1}^{|Q|-2} \mathcal{W}(\Theta_{\mathcal{A}}(i, j, k+1)) \tag{13}$$

In the summation in the right-hand side of the previous equation, $k \neq |Q|-1$ because, as stated in Theorem 1, the row and column of the characteristic matrix M associated with state $|Q|-1$ have been removed. Therefore:

$$\mathcal{N}(i) = \sum_{0 \leq j < |Q|-1} (I - M)^{-1}(i, j)\nu(j) = \frac{1}{1 - r_i}\nu(i) + \sum_{i < j < |Q|-1} \sum_{k=1}^{|Q|-2} \mathcal{W}_{\mathcal{A}}(\Theta_{\mathcal{A}}(i, j, k+1))\ \nu(j)$$

$$= \sum_{k=1}^{|Q|-1} \mathcal{W}_{\mathcal{A}}(\Theta_{\mathcal{A}}(i, |Q|-1, k+1)) \tag{14}$$

The inner part in Equation (14) is analogous to Equation (9), and it can be interpreted as the addition of the weights of all paths from i to $|Q|-1$. Note that $\mathcal{N}(0)$ represents the weight accumulated in all paths in the WFA $\mathcal{W}_{\mathcal{A}}(\Theta_{\mathcal{A}}(0,|Q|-1,\cdot))$ (as we mentioned in Theorem 1):

$$\mathcal{N}(0) = \sum_{k=1}^{|Q|-1} \mathcal{W}_{\mathcal{A}}(\Theta_{\mathcal{A}}(0,|Q|-1,k+1)) = \mathcal{W}_{\mathcal{A}}(\Theta_{\mathcal{A}}(0,|Q|-1,\cdot)) \tag{15}$$

As a final remark, note how the normalization takes place: Each transition weight $W(i,v,j)$ in the WFA $\mathcal{A}$ is changed inversely proportional to the weight accumulated in all the paths that start in i, that is, $\mathcal{N}(i)$, and it is changed directly proportional to the weight accumulated in all the paths that start in j, that is, $\mathcal{N}(j)$.

Expression (14) can be computed with an algorithm that is similar to the backward algorithm, and then summing up for all values in row i. In this way, the time complexity for obtaining the normalizing vector is less than the cubic time required by the matrix inversion.

Let us see an example of the normalization process with the acyclic WFA of Figure 3. This WFA is neither proper nor consistent.

After computations in Theorem 1 and given that $\mathcal{N}(|Q|-1) = 1.0$, we get: $\mathcal{N}^{\mathrm{T}} = (0.2154 \quad 0.2332 \quad 0.3571 \quad 0.5712 \quad 1.0)$. If we use the backward algorithm and we sum up for each row, then we obtain the same normalizing vector but more efficiently:

	1	2	3	4	5	$\mathcal{N}$
0			0.1637	0.0368	0.0149	0.2154
1			0.1042	0.119		0.2232
2		0.1667	0.1904			0.3571
3		0.5712				0.5712
4	1.0000					1.0

If this normalizing vector is applied to the WFA of Figure 3, then the PFA of Figure 4 is obtained. PFA of Figure 4 is proper and consistent.

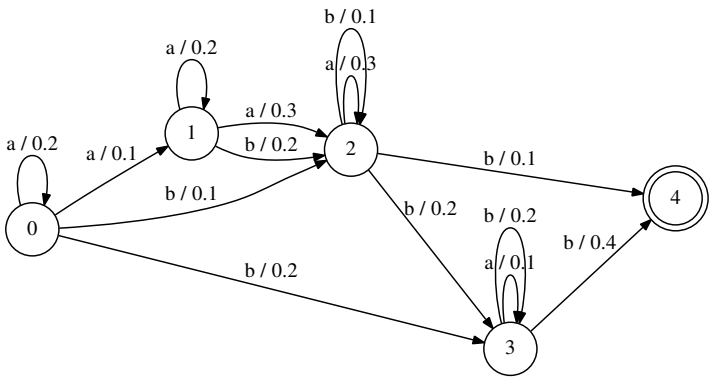

Figure 3
Example of a left-to-right WFA.

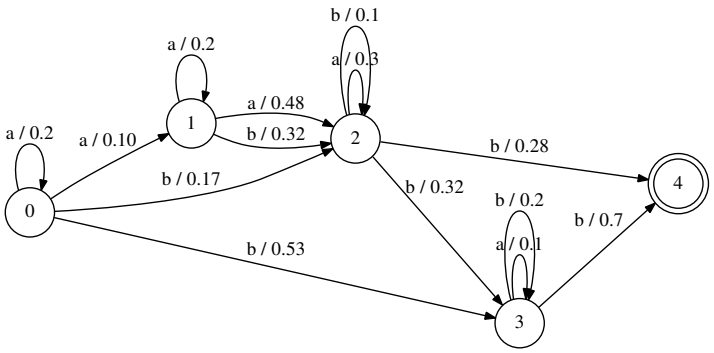

Figure 4
PFA obtained after the normalization has been applied.

This normalization technique is related to the weight pushing algorithm (Mohri 2009) as follows: The value $d[q]$ defined in expression (35) in Mohri (2009) is analogous to the value $\mathcal{N}(q)$ that is introduced in our Definition 5. Both values represent the accumulated weight in all state sequences from state q to the final state. The computations introduced in expressions (36)–(38) in Mohri (2009) are the same normalization that we explicitly introduce in the proof of Theorem 1 (see an interpretation of this normalization in the paragraph that follows our Equation (15)). The value $\mathcal{N}(q)$ is also related to the norm defined in Abney, McAllester, and Pereira (1999) and with the normalization described in Chi (1999; see Corollary 3).

One important contribution of our normalization algorithm with regard to the normalization algorithm presented in Mohri (2009) is that the time complexity for normalizing the graphs introduced in our article (left-to-right graphs with loops in the states) is $O(|\delta|)$, where δ is the set of edges of the automaton. Note that $|\delta| \leq |Q|^2$, where Q is the set of states in the automaton. In terms of the discussion in Mohri (2009), the semiring used in our article is $(R, +, \times, 0, 1)$ and it is a complete semiring. The normalization for semirings of this type is $O(|Q|^3)$ according to Mohri (2009). However, given the restricted graphs that we define, this complete semiring behaves as a k-closed semiring (see our Equations (12) and (13)) because the characteristic matrix M is an upper triangular matrix, and, therefore, the time complexity is $O(|\delta|)$.

The weight pushing algorithm is also related to the normalization algorithm described in Thompson (1974) as follows: The normalization in Thompson (1974) is applied to probabilistic regular grammars (see Theorem 6 in Thompson [1974]) and to probabilistic incontextual grammars in Chomsky Normal Form (see Theorem 7 in Thompson [1974]), but here we only focus on probabilistic regular grammars. One important issue in Thompson (1974) to be taken into account is that the initial probabilistic regular grammar may not be proper (see Definition 3 in Thompson [1974]), but consistent (see Definition 1 in Thompson [1974]). If the initial probabilistic regular grammar is not consistent, the normalization in Thompson (1974) may lead to a proper regular grammar that is not consistent. The main differences between our proposal for the normalization with respect to Thompson (1974) are the following:

1. Thompson (1974) proposes a simultaneous normalization of the model that requires the computation of the inverse of a matrix that is similar to our Equation (10) (see the final expression in Theorem 8 in Thompson

[1974]). Therefore, our proposal is more efficient for the given left-to-right automata that we are dealing with.

2. Thompson (1974) points out a problem when the start symbol of the grammar is not proper (see Condition 4, page 611), although he mentioned an "intermediation" operation to overcome this problem. As he mentioned: "The conditions restricting these problems are left as an open problem." This problem is not present in our proposal, as we demonstrated in Theorem 1.

The relation between Thompson (1974) and Mohri (2009) related to point 1 is that both normalization proposals are cubic for general regular models (grammars or automata). With regard to point 2, normalization in Mohri (2009) and our normalization do not have the problem of the consistency of the final model after normalization as may occur in Thompson (1974).

4. Derivational Entropy of a Left-to-Right PFA

The concept of **derivational entropy** was introduced in Grenander (1967) and was further studied in Soule (1974) for probabilistic grammars. Here we use a similar definition of **derivational entropy** for a Left-to-Right PFA.

Definition 7. The derivational entropy of a PFA $\mathcal{A}$ is defined as:

$$H_d(\mathcal{A}) = -\sum_{\theta \in \Theta_{\mathcal{A}}(0,|Q|-1,\cdot)} p_{\mathcal{A}}(\theta) \log p_{\mathcal{A}}(\theta)$$

The previous sum can have an infinite number of terms because of the loops in the states. We describe how the derivational entropy of a PFA can be efficiently computed.

We define the vector ξ, $0 \leq i < |Q| - 1$, as in Soule (1974):

$$\xi(i) = -\sum_{\substack{v \in \Sigma \\ 0 \leq j \leq |Q|-1}} P(i, v, j) \log P(i, v, j) \tag{16}$$

and $\xi(|Q| - 1) = 0$.

Theorem 2. (**Theorem 4.2 in Grenander [1967]**) If the largest eigenvalue of the characteristic matrix M is strictly less than 1, then the derivational entropy of the model can be computed as: $H_d(\mathcal{A}) = (1 - M)^{-1}\xi$. □

If the matrix M is obtained from a left-to-right PFA, then the largest eigenvalue of M is guaranteed to be strictly less than 1 (see Theorem 5 in Wetherell [1980]). The main idea in Theorem 2 is to compute the average number of times that state i has been used in all valid paths of $\mathcal{A}$ times the entropy introduced by transitions leaving from that state according to Equation (16). Because we are dealing with left-to-right PFA with only one initial state, only this initial state is relevant for computing the derivational entropy, which then becomes the following scalar:

$$H_d(\mathcal{A}) = \sum_{i=0}^{|Q|-2} ((I - M)^{-1})_{0i} \;\; \xi_i \tag{17}$$

According to Theorem 2, the derivational entropy can be computed with time complexity $O(|Q|^3)$ given the inverse matrix computation. An analogous result about the computation of the derivational entropy is stated in Nederhof and Satta (2008) (see Lemma 8).

We now describe how $H_d(\mathcal{A})$ can be efficiently computed for left-to-right PFA. Taking into account that we are considering PFA with only one initial state and one final state (the final state without loops), then from Definition 7, Equation (6), and following Nederhof and Satta (2008):

$$\begin{aligned} H_d(\mathcal{A}) &= -\sum_{2\leq l\leq |Q|} \sum_{\theta\in\Theta_{\mathcal{A}}(0,|Q|-1,l)} p_{\mathcal{A}}(\theta)\log \prod_{(i,v,j)\in\delta} P(i,v,j)^{\mathrm{N}((i,v,j),\theta)} \\ &= -\sum_{2\leq l\leq |Q|} \sum_{\theta\in\Theta_{\mathcal{A}}(0,|Q|-1,l)} p_{\mathcal{A}}(\theta) \sum_{(i,v,j)\in\delta} \log P(i,v,j)^{\mathrm{N}((i,v,j),\theta)} \\ &= -\sum_{(i,v,j)\in\delta} \log P(i,v,j) \sum_{2\leq l\leq |Q|} \sum_{\theta\in\Theta_{\mathcal{A}}(0,|Q|-1,l)} \mathrm{N}((i,v,j),\theta) p_{\mathcal{A}}(\theta) \end{aligned} \tag{18}$$

The two inner additions in Equation (18) can be interpreted as follows: Given a path $\theta \in \Theta_{\mathcal{A}}(0,|Q|-1,l)$, the probability $p_{\mathcal{A}}(\theta)$ is accumulated each time the transition (i,v,j) is used in that path. We distinguish three different cases for computing these two inner additions: i) edges between different states, ii) the loops in the initial state, and iii) all the other loops.

Case i corresponds to the addition of all paths that can be seen in Figure 5, where $i < j$. Therefore, taking into account the numeration of the states according to the *pseudo*-topological order, Equation (18) for these nodes becomes:

$$\begin{aligned} &-\sum_{\substack{v\in\Sigma \\ 0\leq i<j\leq |Q|-1}} \log P(i,v,j) \sum_{l=1}^{i+1} \sum_{m=1}^{|Q|-j} \widehat{\alpha}_{\mathcal{A}}(i,l) P(i,v,j) \widehat{\beta}_{\mathcal{A}}(j,m) \\ &= -\sum_{\substack{v\in\Sigma \\ 0\leq i<j\leq |Q|-1}} \log P(i,v,j) \sum_{l=1}^{i+1} \widehat{\alpha}_{\mathcal{A}}(i,l) P(i,v,j) \underbrace{\sum_{m=1}^{|Q|-j} \widehat{\beta}_{\mathcal{A}}(j,m)}_{1} \\ &= -\sum_{0\leq i<|Q|-2} \sum_{i<j\leq |Q|-1} \sum_{v\in\Sigma} \log P(i,v,j) \sum_{l=1}^{i+1} \widehat{\alpha}_{\mathcal{A}}(i,l) P(i,v,j) \\ &= -\sum_{0\leq i<|Q|-2} \sum_{i<j\leq |Q|-1} \sum_{v\in\Sigma} P(i,v,j) \log P(i,v,j) \sum_{l=1}^{i+1} \widehat{\alpha}_{\mathcal{A}}(i,l) \end{aligned} \tag{19}$$

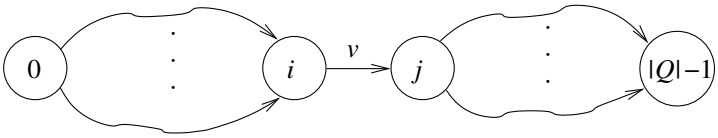

Figure 5
All paths that traverse an edge between two different states in a left-to-right PFA.

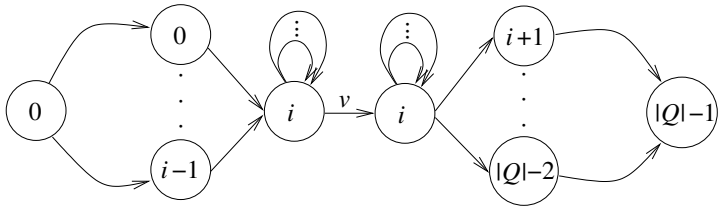

Figure 6
All paths that traverse a loop of a state in the left-to-right PFA.

Case iii in Equation (18) for a given i and v ($0 < i \leq |Q| - 2, v \in \Sigma$) corresponds to the addition of all paths that can be seen in Figure 6.

Thus, Equation (18) for these loops becomes:

$$
\begin{aligned}
& - \sum_{\substack{0 \leq i \leq |Q|-2 \\ v \in \Sigma}} \log P(i,v,i) \\
& \quad \sum_{\substack{0 \leq j < i \\ u \in \Sigma}} \sum_{l=1}^{j+1} \widehat{\alpha}_{\mathcal{A}}(j,l) P(j,u,i) \underbrace{\frac{P(i,v,i)}{(1-r_i)(1-r_i)}}_{(*)} \underbrace{\sum_{\substack{i < k \leq |Q|-1 \\ u' \in \Sigma}} P(i,u',k)}_{1-r_i} \underbrace{\sum_{m=1}^{|Q|-k} \widehat{\beta}_{\mathcal{A}}(k,m)}_{1} \\
& = - \sum_{\substack{0 < i \leq |Q|-2 \\ v \in \Sigma}} \log P(i,v,i) \sum_{\substack{0 \leq j < i \\ u \in \Sigma}} \sum_{l=1}^{j+1} \widehat{\alpha}_{\mathcal{A}}(j,l) P(j,u,i) \frac{P(i,v,i)}{1-r_i} \\
& = - \sum_{\substack{0 < i \leq |Q|-2 \\ v \in \Sigma}} P(i,v,i) \log P(i,v,i) \sum_{0 \leq j < i} \sum_{u \in \Sigma} \sum_{l=1}^{j+1} \widehat{\alpha}_{\mathcal{A}}(j,l) P(j,u,i) \frac{1}{1-r_i} \\
& = - \sum_{\substack{0 < i \leq |Q|-2 \\ v \in \Sigma}} P(i,v,i) \log P(i,v,i) \sum_{l=1}^{i+1} \widehat{\alpha}_{\mathcal{A}}(i,l) \\
& = - \sum_{0 < i \leq |Q|-2} \sum_{v \in \Sigma} P(i,v,i) \log P(i,v,i) \sum_{l=1}^{i+1} \widehat{\alpha}_{\mathcal{A}}(i,l) \qquad (20)
\end{aligned}
$$

The value $(*)$ represents the product of all strings that can be composed by traversing the loops in state i ($1/(1-r_i)$) multiplied by the probability of transition (i,v,i) multiplied again by the product of all strings that can be composed traversing the loops in state i.

Case ii in Equation (18) can be written as:

$$
\begin{aligned}
&-\sum_{v\in\Sigma}\log P(0,v,0)\frac{P(0,v,0)}{(1-r_0)(1-r_0)}\underbrace{\sum_{\substack{i<k\leq|Q|-1\\ u\in\Sigma}}P(i,v,k)}_{1-r_0}\underbrace{\sum_{m=1}^{|Q|-k}\widehat{\beta}_{\mathcal{A}}(k,m)}_{1}\\
&=-\sum_{v\in\Sigma}\log P(0,v,0)\frac{P(0,v,0)}{(1-r_0)}\\
&=-\sum_{v\in\Sigma}P(0,v,0)\log P(0,v,0)\widehat{\alpha}(0,1) \qquad (21)
\end{aligned}
$$

Finally, combining Equations (19), (20), and (21), Equation (18) becomes:

$$
\begin{aligned}
H_d(\mathcal{A}) &= -\sum_{0\leq i\leq|Q|-2}\sum_{i\leq j\leq|Q|-1}\sum_{v\in\Sigma}P(i,v,j)\log P(i,v,j)\sum_{l=1}^{i+1}\widehat{\alpha}_{\mathcal{A}}(i,l)\\
&=\sum_{0\leq i\leq|Q|-2}\xi_i\sum_{l=1}^{i+1}\widehat{\alpha}_{\mathcal{A}}(i,l) \qquad (22)
\end{aligned}
$$

The inner sum can be precomputed in $O(|\delta|)$; therefore, the time complexity of computing the derivational entropy with this method is $O(|\delta|)$. Note that this method is clearly better than the time complexity of the method proposed in Grenander (1967), which is cubic. Note that for dense graphs the time complexity of the new proposed method is at most quadratic with the number of nodes in the PFA.

It is interesting to see the relation between the inner addition in Equation (22) and the elements in the matrix of Equation (17):

$$\sum_{l=1}^{i+1}\widehat{\alpha}_{\mathcal{A}}(i,l)=((I-M)^{-1})_{0i}$$

For the left-to-right PFA in Section 2, $(I-M)^{-1}$ is:

$$\begin{pmatrix}1.43 & 0.41 & 1.07 & 1.49 & 1.0\\ 0.0 & 1.43 & 2.5 & 1.25 & 1.0\\ 0.0 & 0.0 & 2.5 & 1.25 & 1.0\\ 0.0 & 0.0 & 0.0 & 1.67 & 1.0\\ 0.0 & 0.0 & 0.0 & 0.0 & 1.0\end{pmatrix}$$

Each element in the first row coincides with the addition of each row of $\widehat{\alpha}(\cdot,\cdot)$ matrix.

5. Derivational Entropy of a Continuous Left-to-Right HMM

In this section, we extend the efficient computation of the derivational entropy to continuous left-to-right HMM. First, we introduce the notation that will be used for HMM following the notation in Romero, Toselli, and Vidal (2012).

Definition 8. A continuous HMM is defined as a tuple $\mathcal{H} = (Q, I, F, X, a, b)$ where Q is a finite set of states; I is the initial state, $I \in Q$; F is the final state with $F \in Q$; and X is a real d-dimensional space of observations: $X \subseteq \mathbb{R}^d$. To make the subsequent equations simpler, we will assume that X is a scalar (i.e., $X \subseteq \mathbb{R}$). The extension to multidimensional X is straightforward. a is the state-transition probability function, such that for all i, $0 \leq i < |Q| - 1$:

$$\sum_{i \leq j \leq |Q|-1} a(i,j) = 1$$

b is an emission probability distribution function such that for all i, $0 \leq i < |Q| - 1$

$$\int_{x \in X} b(i,x)\, dx = 1\ .$$

Note that we are considering continuous left-to-right HMM with just one initial state and just one final state. The concept of left-to-right is analogous to Definition 3. The emission probabilities are associated with states, not to transitions. For continuous HMM, $b(i,x)$ is a continuous probability density function (pdf).

Given a continuous left-to-right HMM $\mathcal{H}$, a similar algorithm for computing $\widehat{\alpha}_{\mathcal{H}}(\cdot,\cdot)$ can be defined. However, first, it is necessary to compute the probability of all real sequences that can be generated in a loop.

Let i be a state of a HMM $\mathcal{H}$ with r_i as the loop probability, and $b(i,x)$ as the associated pdf. To compute the probability of all real sequences in the loop, we start from a discrete case by sampling the pdf and then use limits to extend it to continuous:

$$\lim_{\substack{K\to\infty \\ y\to-\infty \\ \Delta x\to 0}} \lim_{N\to\infty} \sum_{n=0}^{N} \left(r_i \sum_{\substack{x_k = y + k\Delta x \\ k=\{1,\ldots,K\}}} b(i,x_k)\Delta x \right)^n = \lim_{N\to\infty} \sum_{n=0}^{N} \left(r_i \left(\lim_{\substack{K\to\infty \\ y\to-\infty \\ \Delta x\to 0}} \sum_{\substack{x_k = y + k\Delta x \\ k=\{1,\ldots,K\}}} b(i,x_k)\Delta x \right) \right)^n$$

$$= \lim_{N\to\infty} \sum_{n=0}^{N} \left(r_i \underbrace{\int_x b(i,x)dx}_{=1} \right)^n = \sum_{n=0}^{\infty} r_i^n = \frac{1}{1-r_i}$$

This result is the same as for PFA, and with this value, we can compute both the $\widehat{\alpha}_{\mathcal{H}}(\cdot,\cdot)$ and the $\widehat{\beta}_{\mathcal{H}}(\cdot,\cdot)$ probabilities as in Section 2.

Now, we obtain a similar expression to Equation (22):

$$H_d(\mathcal{H}) = \sum_{0 \leq i \leq |Q|-2} \sum_{l=1}^{i+1} \widehat{\alpha}_{\mathcal{H}}(i,l) \sum_{i \leq j \leq |Q|-1} \int_x p_{ij} f_i(x) \log p_{ij} f_i(x) dx$$

$$= \sum_{0 \leq i \leq |Q|-2} \sum_{l=1}^{i+1} \widehat{\alpha}_{\mathcal{H}}(i,l) \sum_{i \leq j \leq |Q|-1} p_{ij} \left(\log p_{ij} \underbrace{\int_x f_i(x)dx}_{=1} + \int_x f_i(x) \log f_i(x) dx \right)$$

$$= \sum_{0 \leq i \leq |Q|-2} \sum_{l=1}^{i+1} \widehat{\alpha}_{\mathcal{H}}(i,l) \sum_{i \leq j \leq |Q|-1} p_{ij} \left(\log p_{ij} + \int_x f_i(x) \log f_i(x) dx \right)$$

The last integral is the entropy of the distribution at state i, which for the case of Gaussian mixture models, which are commonly used in practice, the entropy can be approximated (Huber et al. 2008).

6. Conclusions

This article has studied the efficient computation of the derivational entropy for left-to-right PFA and HMM. This efficiency is mainly based on the left-to-right nature of the models. This efficient computation is based on algorithms that similar to the *forward* and *backward* algorithms defined for general finite state models. The algorithms that are introduced in this article are also necessary for normalizing the left-to-right WFA in order to guarantee that the derivational entropy is correctly computed.

Acknowledgments

This work has been partially supported through the European Union's H2020 grant READ (Recognition and Enrichment of Archival Documents) (Ref: 674943) and the MINECO/FEDER-UE project TIN2015-70924-C2-1-R. The second author was supported by the "División de Estudios de Posgrado e Investigación" of Instituto Tecnológico de León.

References

Abney, S., D. McAllester, and F. Pereira. 1999. Relating probabilistic grammars and automata. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 542–549, College Park, MD.

Bakis, R. 1976. Continuous speech word recognition via centisecond acoustic states. *Journal of the Acoustical Society of America*, 59:S97.

Can, D. and M. Saraçlar. 2011. Lattice indexing for spoken term detection. *IEEE Transactions on Audio, Speech and Language Processing*, 19(8):2338–2347.

Chi, Z. 1999. Statistical properties of probabilistic context-free grammar. *Computational Linguistics*, 25(1):131–160.

Corazza, A. and G. Satta. 2007. Probabilistic context-free grammars estimated from infinite distributions. *Transactions on Pattern Analysis and Machine Intelligence*, 29(8):1379–1393.

Dupont, P., F. Denis, and Y. Esposito. 2005. Links between probabilistic automata and hidden Markov models: probability distributions, learning models and induction algorithms. *Pattern Recognition*, 38: 1349–1371.

Grenander, U. 1967. Syntax controlled probabilities. Technical report. Brown University, Division of Applied Mathematics.

Hernando, D., V. Crespi, and G. Cybenko. 2005. Efficient computation of the hidden Markov model entropy for a given observation sequence. *IEEE Transactions of Information Theory*, 51(7):2681–2685.

Huber, M. F., T. Bailey, H. Durrant-Whyte, and U. D. Hanebeck. 2008. On entropy approximation for Gaussian mixture random vectors. In *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pages 181–188, Seoul.

Ilic, V. M. 2011. Entropy semiring forward-backward algorithm for HMM entropy computation. *CoRR.*, abs/1108.0347.

Kemp, T. and T. Schaaf. 1997. Estimating confidence using word lattices. *Eurospeech*, pages 827–830, Rhodes.

Mann, G. S. and A. McCallum. 2007. Efficient computation of entropy gradient for semi-supervised conditional random fields. In *Proceedings of HLT-NAACL, Companion Volume, Short Papers*, pages 109–112.

Mohri, M. 2009. Weighted automata algorithms. *Handbook of Weighted Automata, Monographs in Theoretical Computer Science*, Springer-Verlag.

Mohri, M., F. C. N. Pereira, and M. Riley. 2002. Weighted finite-state transducers in speech recognition. *Computer Speech and Language*, 16:69–88.

Nederhof, M. J. and G. Satta. 2008. Computation of distances for regular and context-free probabilistic languages. *Theoretical Computer Science*, 395:235–254.

Ortmanns, S. and H. Ney. 1997. A word graph algorithm for large vocabulary continuous speech recognition. *Computer Speech and Language*, 11:43–72.

Puigcerver, J., A. H. Toselli, and E. Vidal. 2014. Word-graph and character-lattice

combination for KWS in handwritten documents. In *International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 181–186, Crete.

Romero, V., A. H. Toselli, and E. Vidal. 2012. *Multimodal Interactive Handwritten Text Recognition*, volume 80 of Machine Perception and Artificial Intelligence. Word Scientific.

Sanchis, A., A. Juan, and E. Vidal. 2012. A word-based naïve Bayes classifier for confidence estimation in speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(2):565–574.

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

Thompson, R. A. 1974. Determination of probabilistic grammars for funtionally specified probability-measure languages. *IEEE Transactions of Computers*, c-23(6):603–614.

Tomita, M. 1986. An efficient word lattice parsing algorithm for continuous speech recognition. In *Proceedings of ICASSP*, pages 1569–1572, Tokyo.

Toselli, A. H., E. Vidal, and F. Casacuberta, editors. 2011. *Multimodal Interactive Pattern Recognition and Applications*, 1st edition. Springer.

Ueffing, N., F. J. Och, and H. Ney. 2002. Generation of word graphs in statistical machine translation. In *Proceedings on Empirical Method for Natural Language Processing*, pages 156–163, Philadelphia, PA.

Vidal, E., F. Thollard, C. de la Higuera, F. Casacuberta, and R. C. Carrasco. 2005. Probabilistic finite-state machines - Part I. *IEEE Transactions on Pattern Analysis Machine Intelligence*, 27(7):1013–1039.

Wessel, F., R. Schlüter, K. Macherey, and H. Ney. 2001. Confidence measures for large vocabulary continuous speech recognition. *IEEE Transactions on Speech and Audio Processing*, 9(3):288–298.

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