HAL
open science

Probabilistic finite-state machines - Part II.

Enrique Vidal, Franck Thollard, Colin de La Higuera, Francisco Casacuberta, R.C. Carrasco

To cite this version:

Enrique Vidal, Franck Thollard, Colin de La Higuera, Francisco Casacuberta, R.C. Carrasco. Probabilistic finite-state machines - Part II.. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2005, 27 (7), pp.1026-1039. 10.1109/TPAMI.2005.148 . ujm-00326250

HAL Id: ujm-00326250
https://ujm.hal.science/ujm-00326250v1

Submitted on 16 Oct 2008

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Probabilistic Finite-State Machines – Part II

E. Vidal, F. Thollard, C. de la Higuera, F. Casacuberta and R. C. Carrasco

Abstract

Probabilistic finite-state machines are used today in a variety of areas in pattern recognition, or in fields to which pattern recognition is linked. In part I of this paper, we surveyed these objects and studied their properties. In this part II, we study the relations between probabilistic finite-state automata and other well known devices that generate strings like hidden Markov models and n-grams, and provide theorems, algorithms and properties that represent a current state of the art of these objects.

Index Terms

Automata (F.1.1.a), Classes defined by grammars or automata (F.4.3.b), Machine learning (I.2.6.g), Language acquisition (I.2.6.h), Language models (I.2.7.c), Language parsing and understanding (I.2.7.d), Machine translation (I.2.7.f), Speech recognition and synthesis (I.2.7.g), Structural Pattern Recognition (I.5.1.f), Syntactic Pattern Recognition (I.5.1.g).

I. Introduction

In part one [1] of this survey we introduced probabilistic finite-state automata (PFA), their deterministic counterparts (DPFA) and the properties of the distributions these objects can generate; topology was also discussed, and so were consistency and equivalence issues.

In this second part we will describe additional features that are of use to those wishing to work with PFA or DPFA. As mentioned before there are many other finite-state machines that describe distributions. Section II is entirely devoted to compare them with PFA and DPFA. The comparison will be algorithmic: techniques (when existing) allowing to transform one model into another equivalent, in the sense that the same distribution is represented, will be

Dr. Vidal and Dr. Casacuberta are with Dto. Sistemas Informáticos y Computación and Instituto Tecnológico de Informática. Universitat Politècnica de València. Spain.

Dr. de la Higuera and Dr. Thollard are with EURISE and the Université Jean Monnet. France.

Dr. Carrasco is with Dto. de Lenguajes y Sistemas Informáticos. Universidad de Alicante. Spain.

provided. We will study n-grams along with stochastic local languages in section II-A and HMMs in section II-C. In addition, in Section II-B we will present a probabilistic extension of the classical *morphism theorem* that relates local languages with regular languages in general.

Once most of the issues concerning the task of dealing with existing PFA have been examined, we turn to the problem of building these models, presumably from samples. First, we address the case where the underlying automaton structure is known; then, we deal (section III-A) with the one of estimating the parameters of the model [2]–[7]. The case where the model structure is not known enters the field of machine learning and a variety of learning algorithms has been used. Their description, proofs and a comparison of their qualities and draw-backs would deserve a detailed survey in itself. We provide, in section III-B, an overview of the main methods, but we do not describe them throughly. We hope the bibliographical entries we provide, including the recent review which appears in [8], will be of use for the investigator who requires further reading in this subject. Smoothing [9]–[11] (in section III-C) is also becoming a standard issue.

A number of results do not fall into any of these main questions. Section IV will be a *pot-pourri*, presenting alternative results, open problems and new trends. Among these, more complex models such as stochastic transducers (in section IV-A), probabilistic context-free grammars [12] (in section IV-B), or probabilistic tree automata [13]–[15] (in section IV-C) are taking importance when coping with increasingly structured data.

The proofs of some of the propositions and theorems are left to the corresponding appendices.

As all surveys this one is incomplete. In our particular case the completeness is particularly difficult to achieve due to the enormous and increasing amount of very different fields where these objects have been used. In advance we would like to apologize to all those whose work on the subject we have not recalled.

II. Other finite-state models

Apart from the various types of PFA, a variety of alternative models has been proposed in the literature to generate or model probability distributions on the strings over an alphabet.

Many different definitions of probabilistic automata exist. Some assume probabilities on states, others on transitions. But the deep question is "which is the distinctive feature of the probability distribution defined?". All the models describe discrete probability distributions. Many of them aim at predicting the next symbol in a string, thereby describing probability distributions over each Σ^n, $\forall n > 0$. We will concentrate here on models where parsing will be done from left to right, one symbol at a time. As a consequence, the term *predicting history* will correspond to the amount of information one needs from the prefix to compute the next-symbol probability. Multiplying these next-symbol probabilities is called the *chain rule* which will be discussed in section II-A.

Among the models proposed so far, some are based on *acyclic automata* [1], [16]–[19]. Therefore, the corresponding probability distributions are defined on finite sets of strings, rather than on $\Sigma^\star$. In [18] automata that define probability distributions over Σ^n, for some fixed $n > 0$. This kind of models can be used to represent, for instance, logic circuits, where the value of n can be defined in advance. A main restriction of this model is that it cannot be used to compare probabilities of strings of different lengths. Ron *et al.* [19] define other probabilistic acyclic deterministic automata and apply them to optical character recognition.

Another kind of model describes a probability distribution over $\Sigma^\star$; that is, over an infinite number of strings. Stolcke and Omohundro [20] use other types of automata that are equivalent to our definition of DPFA. Many probabilistic automata, such as those discussed here, the HMM and the Markov chain (also known as the n-gram model), also belong to this class.

We give here an overview of some of the most relevant of these models. In all cases we will present them in comparison with the probabilistic finite-state automata. The comparison will be algorithmic: techniques (when existing) allowing to transform one model into another, equivalent in the sense that the same distribution is represented, will be provided. From the simpler to the more complex objects we will study n-grams and stochastic k-testable languages (in section II-A), and HMMs (in section II-C). We will include in section II-B a probabilistic extension of an important result in the classical theory of formal languages, known as the *morphism theorem*.

A. N-grams and stochastic k-testable automata

N-grams have been the most widely used models in natural language processing, speech recognition, continuous handwritten text recognition, etc. As will be seen below, under certain assumptions, n-grams are equivalent to a class of DPFA known as *stochastic k-testable automata*. Despite the popularity and success of these models, we shall prove that they cannot model all distributions that can be modeled by DPFA.

N-gram models:

N-grams are traditionally presented as an approximation to a distribution of strings of *fixed length*. For a string x of length m, the chain rule is used to (exactly) decompose the probability of x as [21]:

$$\Pr(x) \;=\; \Pr(x_1) \cdot \prod_{l=2}^{m} \Pr(x_l \mid x_1, \ldots, x_{l-1}). \tag{1}$$

The n-gram approximation makes the assumption that the probability of a symbol depends only on the $n-1$ previous symbols; that is:[1]

$$\Pr(x) \;\approx\; \prod_{l=1}^{m} \Pr(x_l \mid x_{l-n+1}, \ldots, x_{l-1}) \;.$$

As the exact equation (1), this approximation also defines a probability distribution over Σ^m. Nevertheless, for practical reasons it is often interesting to *extend* it to define a probability distribution over $\Sigma^\star$. To this end, the set of events, Σ, which are predicted by the $n-1$ previous symbols, is extended by considering an additional end-of-string event (denoted by $\#$), with probability $\Pr(\#|x_{m-n+2}, \ldots, x_m)$. As a result, the probability of any string $x \in \Sigma^\star$ is approximated as:

$$\Pr(x) \;\approx\; \Big(\prod_{l=1}^{|x|} \Pr(x_l \mid x_{l-n+1}, \ldots, x_{l-1})\Big) \cdot \Pr(\# \mid x_{|x|-n+2}, \ldots, x_{|x|}) \;. \tag{2}$$

By making use of our convention that a string such as $x_i \ldots x_j$ denotes λ if $i > j$, this approximation accounts for the empty string. In fact, if $x = \lambda$, the right-hand side of equation (2) is $1 \cdot \Pr(\#|\lambda)$, which may take values greater then 0. The resulting approximation will be referred to as *"extended n-gram model"*. The parameters of this model are estimates

[1]For the sake of simplifying the notation, if $i \leq 1$ the expression $\Pr(x_j|x_i, \ldots, x_{j-1})$ is assumed to denote $\Pr(x_j|x_1, \ldots, x_{j-1})$. If $j = 1$, it is just $\Pr(x_1|\lambda)$, interpreted as $\Pr(x_1)$.

of $\Pr(a|z)$, $a \in \Sigma \cup \{\#\}$, $z \in \Sigma^{<n}$, which will be referred to as $P_n(a|z)$. The model assigns a probability $\Pr_n(x)$ for any string $x \in \Sigma^\star$ as:

$$\Pr_n(x) \;=\; \Big(\prod_{l=1}^{|x|} P_n(x_l \mid x_{l-n+1}, \ldots, x_{l-1})\Big) \cdot P_n(\# \mid x_{|x|-n+2}, \ldots, x_{|x|})\Big) \;. \tag{3}$$

Note that (unlike the classical n-gram model for fixed-length strings) $\Pr_n(x)$ can be *deficient*. This may happen for certain "degenerate" values of $P_n(a|z)$, $a \in \Sigma \cup \{\#\}, z \in \Sigma^{<n}$, which may lead to infinite-length strings with non-null probability. Disregarding these degenerate cases and provided that

$$\sum_{a \in \Sigma} P_n(a|z) \;+\; P_n(\#|z) \;=\; 1 \quad \forall z \in \Sigma^{<n},$$

this model is *consistent*; i.e., it defines a probability distribution, $\mathcal{D}_n$, over $\Sigma^\star$.

It follows from the above definition that, if $\mathcal{D}_n$ is described by an extended n-gram, for any $n' > n$ there is an extended n'-gram which describes a distribution $\mathcal{D}_{n'}$ such that $\mathcal{D}_n = \mathcal{D}_{n'}$. In other words, there is a natural hierarchy of *classes* of n-grams, where the classes with more expressive power are those with larger n. The simplest interesting class in this hierarchy is the class for $n = 2$, or *bigrams*. This class is interesting for its generative power, in the sense discussed later (Section II-B).

On the other hand, perhaps the most interesting feature of n-grams is that they are easily learnable from training data. All the parameters of a n-gram model can be maximum-likelihood estimated by just counting the relative frequency of the relevant events in the training strings [21]. If S is a training sample, $P_n(a \mid z)$ is estimated as $f(za) \,/\, f(z)$, $a \in \Sigma \cup \{\#\}, z \in \Sigma^{<n}$, where $f(y)$ is the number of times the substring y appears[2] in the strings of S. Interestingly, the degenerate cases mentioned above can never happen for n-grams trained in this way and the resulting trained models are always consistent.

The n-grams estimated in this way from a fixed S exhibit an interesting hierarchy for decreasing values of n. Let $\mathcal{D}_\mathcal{S}$ be the empirical distribution associated with S and let $L_n = \prod_{x \in S} \Pr_{\mathcal{D}_n}(x)$ be the likelihood with which an extended n-gram generates S. Then for $m = max_{x \in S}|x|$, $\mathcal{D}_S = \mathcal{D}_m$ and for all $m'' < m' < m$, $L_{m''} \leq L_{m'}$. In other words, starting with $n = m$, the sample S is increasingly generalized for decreasing values of n.

[2]For substrings shorter than n, $f(y)$ is the number of times that y appears as a *prefix* of some string in S.

Stochastic k-testable automata :

N-grams are closely related to a family of regular models called *k-testable stochastic automata* (k-TSA) [22].[3] In fact, we shall see that for every *extended* k-gram model there is a k-TSA which generates the same distribution.

In the traditional literature, a k-testable language is characterized by two sets of strings, corresponding to permitted prefixes and suffixes of length less than k, and a set of permitted substrings of length k [22]–[24]. A straightforward probabilistic extension adequately assigns probabilities to these substrings, thereby establishing a direct relation with n-grams. For the sake of brevity, we will only present the details for 2-testable distributions, also called *stochastic local languages*.

Definition 1: A *stochastic local language* (or 2-testable stochastic language) is defined by a four-tuple $Z = \langle \Sigma, P_I, P_F, P_T \rangle$, where Σ is the alphabet, and $P_I, P_F : \Sigma \to [0,1]$, and $P_T : \Sigma \times \Sigma \to [0,1]$ are, respectively, *initial*, *final*, and *symbol transition* probability functions. $P_I(a)$ is the probability that $a \in \Sigma$ is a starting symbol of the strings in the language and, $\forall a \in \Sigma$, $P_T(a', a)$ is the probability that a follows a', while $P_F(a')$ is the probability that no other symbol follows a' (*i.e.* a' is the last symbol) in the strings of the language.

As in the case of n-grams, this model can be easily extended to allow the generation of *empty strings*. To this end, P_F can be redefined as $P_F : \Sigma \cup \{\lambda\} \to [0,1]$, interpreting $P_F(\lambda)$ as the probability of the empty string, according to the following normalization conditions:

$$\begin{aligned} \sum_{a \in \Sigma} P_I(a) \ + \ P_F(\lambda) \ &= \ 1, \\ \sum_{a \in \Sigma} P_T(a', a) \ + \ P_F(a') \ &= \ 1 \quad \forall a' \in \Sigma \ . \end{aligned}$$

Disregarding possible "degenerate" cases (similar to those of extended n-grams discussed

[3]In the traditional literature, a *k-testable automaton* (K-TA) is (more properly) referred to as a *k-testable automaton in the strict sense* (K-TSA) [23], [24]. In these references, the name *k-testable automaton* is reserved for more powerful models which are defined as boolean compositions of K-TSA. A stochastic extension of K-TA would lead to models which, in some cases, can be seen as *mixtures* of stochastic k-TSA.

above), the model Z is consistent; i.e., it defines a probability distribution $\mathcal{D}_{\mathcal{Z}}$ on $\Sigma^\star$ as:

$$\Pr_Z(x) \;=\; \begin{cases} P_F(\lambda) & \textit{if } x = \lambda\,, \\ P_I(x_1) \cdot \prod_{i=2}^{|x|} P_T(x_{i-1}, x_i) \;\cdot\; P_F(x_{|x|}) & \textit{if } x \in \Sigma^+. \end{cases} \tag{4}$$

Comparing equation (3) and (4), the equivalence of *local language* and extended *bigram* distributions can be easily established by letting:

$$\begin{aligned} P_I(a) &= P_2(a),\ \forall a \in \Sigma\,, \\ P_F(a) &= P_2(\# \mid a),\ \forall a \in \Sigma \cup \{\lambda\}\,, \\ P_T(a', a) &= P_2(a \mid a'),\ \forall a, a' \in \Sigma\,. \end{aligned}$$

Therefore, the following proposition holds:

Proposition 1: For any extended *bigram* distribution $\mathcal{D}_2$ there exists a *local language* model Z such that $\mathcal{D}_{\mathcal{Z}} = \mathcal{D}_2$, and vice versa.

A stochastic 2-testable model $Z = \langle \Sigma, P_I, P_F, P_T \rangle$ can be straightforwardly represented by a 2-testable stochastic automaton (2-TSA). This automaton is a DPFA $\mathcal{A} = \langle Q, \Gamma, \delta, q_0, F, P \rangle$ built as follows:

$$\begin{aligned} &\Gamma = \Sigma,\ Q = \Sigma \cup \{\lambda\},\ q_0 = \lambda, \\ &\delta = \{(\lambda, a, a) \mid a \in \Sigma,\ P_I(a) > 0\} \bigcup \{(a'', a, a) \mid a, a'' \in \Sigma,\ P_T(a'', a) > 0)\}, \\ &\forall a, a'' \in \Sigma,\ P(a'', a, a) = P_T(a'', a),\ P(\lambda, a, a) = P_I(a),\ F(a) = P_F(a),\ F(\lambda) = P_F(\lambda)\ . \end{aligned} \tag{5}$$

An example of this construction is shown in figure 3 (middle), page 11, corresponding to example 2 below. Definition 1, proposition 1 and the above construction (5) can be easily extended to show the equivalence of extended k-grams and k-TSA for any finite k.

As in the case of n-grams, k-TSA can be easily learned from training data [22]. Given the equivalence with extended n-grams, k-TSA exhibit the same properties for varying values of k. In particular, in this case, the m-TSA obtained from a training sample S for $m = max_{x \in S}|x|$ is an *acyclic* DPFA which is identical to the *probabilistic prefix tree automaton* representation of S.

N-grams and k-TSA are less powerful than DPFA:

We now show that extended n-grams or stochastic k-testable automata do not have as much modeling capabilities as DPFA have.

Proposition 2: There are regular deterministic distributions that cannot be modeled by a k-TSA or extended k-gram, for any finite k.

This is a direct consequence of the fact that every regular language is the support of at least one stochastic regular language, and there are regular languages which are not k-testable. The following example illustrates this lack of modeling power of extended n-grams or k-TSA.

Example 1: Let $\Sigma = \{a, b, c, d\}$ and let $\mathcal{D}$ be a probability distribution over $\Sigma^\star$ defined as:

$$\Pr_{\mathcal{D}}(x) = \begin{cases} 1/2^{i+1} & \text{if } x = ab^ic \vee x = db^ie,\ \ i > 0\,, \\ 0 & \text{otherwise}\,. \end{cases}$$

This distribution can be exactly generated by the DPFA of figure 1, but it can not be properly approached by any k-TSA for any given k. The best k-TSA approximation of $\mathcal{D}$, $\mathcal{D}_k$, is:

$$\Pr_{\mathcal{D}_k}(x) = 1/2^{i+1}, \quad \Pr_{\mathcal{D}_k}(x') = 0 \qquad \forall i \leq k-2\ ,$$

$$\Pr_{\mathcal{D}_k}(x) = \Pr_{\mathcal{D}_k}(x') = 1/2^{i+2} \qquad \forall i > k-2\ ,$$

for any string x of the form ab^ic or db^ie, and x' of the form db^ic or ab^ie.

In other words, using probabilistic k-testable automata or extended k-grams, only the probabilities of the strings up to length k can be approached while, in this example, the error ratio[4] for longer strings will be at least $1/2$ (or larger if k-TSA probabilities are estimated from a finite set of training data). As a result, for all finite values of k the *logarithmic distance* $d_{\log}(\mathcal{D},\mathcal{D}_k)$ is infinite.

This can be seen as a probabilistic manifestation of the well known over/under-generalization behavior of conventional k-testable automata [25].

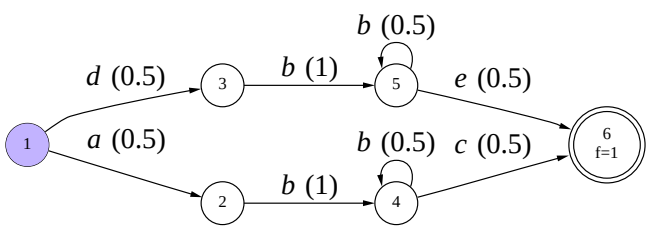

Fig. 1. A DPFA which generates a regular deterministic distribution that cannot be modeled by any k-TSA or n-gram.

[4]The error-ratio for a string x is the quotient between the true and the approximated probabilities for x.

B. Stochastic morphism theorem

In classical formal language theory the *morphism theorem* [26] is a useful tool to overcome the intrinsic limitations of k-testable models and to effectively achieve the full modelling capabilities of regular languages in general. Thanks to this theorem, the simple class of 2-testable languages becomes a *"base set"* from which all the regular languages can be generated.

However, no similar tool existed so far for the corresponding stochastic distributions. This section extends the standard construction used in the proof of the morphism theorem so that a similar proposition can be proved for stochastic regular languages.

Theorem 3 (Stochastic morphism theorem): Let Σ be a finite alphabet and $\mathcal{D}$ a stochastic regular language on $\Sigma^\star$. There exists then a finite alphabet Σ', a letter-to-letter morphism $h : \Sigma'^\star \to \Sigma^\star$, and a stochastic local language over Σ', $\mathcal{D}_2$, such that $\mathcal{D} = h(\mathcal{D}_2)$; *i.e.*,

$$\forall x \in \Sigma^\star \quad \Pr_{\mathcal{D}}(x) \; = \Pr_{\mathcal{D}_2}(h^{-1}(x)) \; = \sum_{y \in h^{-1}(x)} \Pr_{\mathcal{D}_2}(y)\,, \tag{6}$$

where $h^{-1}(x) = \{y \in \Sigma'^\star \mid x = h(y)\}$.

The proof of this proposition is in the Appendix A.

The following example illustrates the construction used in this proof and how to obtain exact 2-TSA-based models for given, possibly non-deterministic stochastic regular languages.

Example 2: Consider the following distribution $\mathcal{D}$ over $\Sigma = \{a, b\}$:

$$\Pr(x) = \begin{cases} \Pr(i) & \textit{if} \;\; x = ab^i, \quad i \geq 0\,, \\ 0 & \textit{otherwise}, \end{cases}$$

with $\Pr(i) = p_1 \cdot (1 - p_2) \cdot {p_2}^i + (1 - p_1) \cdot (1 - p_3) \cdot {p_3}^i$ and $p_1 = 0.5$, $p_2 = 0.7$ and $p_3 = 0.9$.

This distribution (which is similar to that used in part I [1] to prove that the mean of two deterministic distributions may not be deterministic) is exactly generated by the PFA shown in figure 3 (left). From a purely structural point of view, the strings from the language underlying this distribution constitute a very simple local language that can be exactly generated by a trivial 2-testable automaton. However, from a probabilistic point of view, $\mathcal{D}$ is not regular deterministic, nor by any means *local*. In fact, it can not be approached with arbitrary precision by any k-TSA, for any finite value of k. The best approximations for $k = 2, 3, 4, 5$ produce error-ratios greater than 2 for strings longer than 35, 40, 45 and 52, respectively,

as it is shown in figure 2. In fact, the *logarithmic distance* between the true and k-TSA-approximated distributions is *infinite* for any finite k. Nevertheless, the construction given by the stochastic morphism theorem yields a stochastic finite-state automaton that exactly generates $\mathcal{D}$.

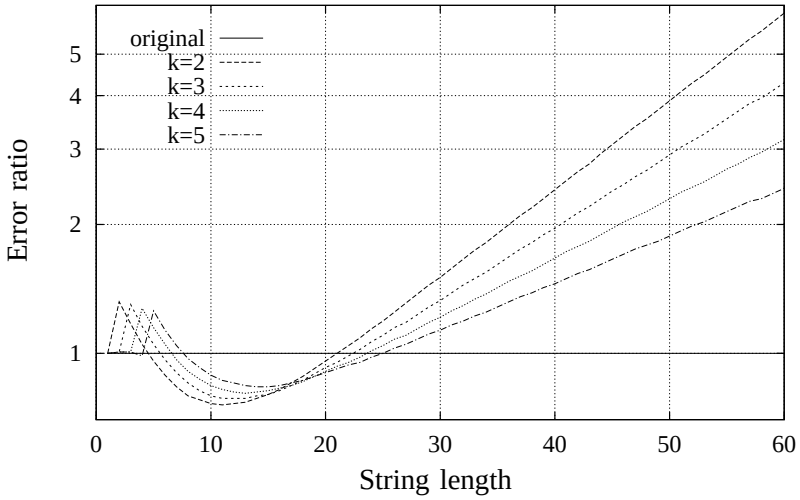

Fig. 2. Error-ratio of the probabilities provided by different k-testable automata that best approach the stochastic language of example 2, with respect to the true probability of strings in this language.

Using the construction of the proof of the stochastic morphism theorem, a 2-TSA, $Z = \langle \Sigma', P_I, P_F, P_T \rangle$, is built from the DPFA $\mathcal{A}_0 = \langle Q, \Sigma, \delta, q_0, F, P \rangle$ shown in figure 3 (left) as follows:

$$\begin{aligned}
&\Sigma' = \{a_2, a_3, b_2, b_3\}\,, \\
&P_I(a_2) = P_I(a_3) = P(1, a, 2) = P(1, a, 3) = 0.5\,, \\
&P_F(a_2) = P_F(b_2) = F(2) = 0.3, \quad P_F(a_3) = P_F(b_3) = F(3) = 0.1\,, \\
&P_T(a_2, b_2) = P_T(b_2, b_2) = P(2, b, 2) = 0.7, \quad P_T(a_3, b_3) = P_T(b_3, b_3) = P(3, b, 3) = 0.9\,,
\end{aligned} \tag{7}$$

all the other values of P_I, P_F and P_T are zero.

The corresponding 2-TSA is shown in figure 3 (middle). Applying the morphism h (*i.e.* dropping sub-indexes) to this automaton yields the PFA $\mathcal{A}$ shown in figure 3 (right). For any string x of the form ab^i, we have:

$$\Pr_{\mathcal{A}}(x) = 0.5 \cdot 0.3 \cdot 0.7^i \;+\; 0.5 \cdot 0.1 \cdot 0.9^i \quad \forall i \geq 0.$$

which is exactly the original distribution, $\mathcal{D}$. □

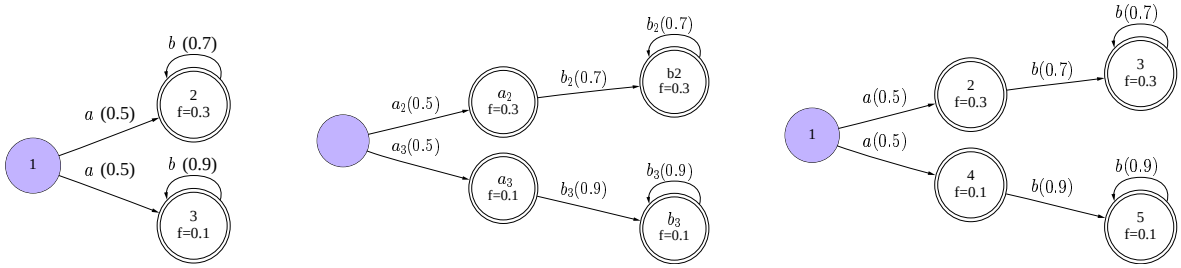

Fig. 3. Left: finite-state stochastic automaton which generates the stochastic language of example 2; Center and right: automata obtained through the construction used in the proof of the stochastic morphism theorem.

C. Hidden Markov models

Nowadays, hidden Markov models (HMMs) are basic components of the most successful natural language processing tasks, including speech [21], [27], [28] and handwritten text recognition [29], [30], speech translation [31], [32] and shallow parsing [33], to name but a few. HMMs have also proved useful in many other pattern recognition and computer vision tasks, including shape recognition, face and gesture recognition, tracking, image database retrieval and medical image analysis [34], [35] and other less conventional applications such as financial returns modeling [36].

There exist many variants of Markov models, including differences as to whether the symbols are emitted at the states or at the transitions. See for example [21], [27], [28], [37].

Definition 2: A HMM is a 6-tuple $\mathfrak{M} = \langle \mathrm{Q}, \Sigma, \mathrm{I}, \mathrm{q_f}, \mathrm{T}, \mathrm{E} \rangle$, where

- Q is a finite set of states,
- Σ is a finite alphabet of symbols,
- $\mathrm{T} : (\mathrm{Q} - \{\mathrm{q_f}\}) \times \mathrm{Q} \to \mathbb{R}^+$ is a state to state transition probability function,
- $\mathrm{I} : \mathrm{Q} - \{\mathrm{q_f}\} \to \mathbb{R}^+$ is an initial state probability function,
- $\mathrm{E} : (\mathrm{Q} - \{\mathrm{q_f}\}) \times \Sigma \to \mathbb{R}^+$ is a state-based symbol emission probability function,
- $\mathrm{q_f} \in \mathrm{Q}$ is a special (final) state,

subject to the following normalization conditions:

$$\begin{aligned} \sum_{q \in \mathrm{Q} - \{\mathrm{q_f}\}} \mathrm{I}(q) &= 1\,, \\ \sum_{q' \in \mathrm{Q}} \mathrm{T}(q, q') &= 1, \quad \forall q \in \mathrm{Q} - \{\mathrm{q_f}\}\,, \\ \sum_{a \in \Sigma} \mathrm{E}(q, a) &= 1, \quad \forall q \in \mathrm{Q} - \{\mathrm{q_f}\}\,. \end{aligned}$$

We will say that the model $\mathfrak{M}$ generates (or *emits*) a sequence $x = x_1 \ldots x_k$ with probability $\Pr_{\mathfrak{M}}(x)$. This is defined in two steps. First let θ be a valid path of length k; i.e., a sequence $(s_1, s_2, \ldots, s_k)$ of states, with $s_k = \mathrm{q_f}$. The probability of θ is:

$$\Pr_{\mathfrak{M}}(\theta) \;=\; \mathrm{I}(s_1) \cdot \prod_{2 \leq j \leq k} \mathrm{T}(s_{j-1}, s_j)\,.$$

and the probability of generating x through θ is:

$$\Pr_{\mathfrak{M}}(x \mid \theta) \;=\; \prod_{1 \leq j < k} \mathrm{E}(s_j, x_j)\,.$$

Then if $\Theta_{\mathfrak{M}}(x)$ is the set of all valid paths for x, the probability that $\mathfrak{M}$ generates x is:

$$\Pr_{\mathfrak{M}}(x) \;=\; \sum_{\theta \in \Theta_{\mathfrak{M}}(x)} \Pr_{\mathfrak{M}}(x \mid \theta) \cdot \Pr_{\mathfrak{M}}(\theta)\,.$$

It should be noticed that the above model cannot emit the empty string. Moreover, as in the case of PFA, some HMMs can be deficient. Discarding these degenerate cases, it can easily be seen that $\sum_{x \in \Sigma^+} \Pr_{\mathfrak{M}}(x) = 1$. Correspondingly, a HMM $\mathfrak{M}$ defines a probability distribution $\mathcal{D}_{\mathfrak{M}}$ on Σ^+.

In some definitions, states are allowed to remain silent or the final state $\mathrm{q_f}$ is not included in the definition of a HMM. As in the case of n-grams, this latter type of model defines a probability distribution on Σ^n for each n, rather than on Σ^+ [37].

Some relations between HMMs and PFA are established by the following propositions.

Proposition 4: Given a PFA $\mathcal{A}$ with m transitions and $\Pr_{\mathcal{A}}(\lambda) = 0$, there exists a HMM $\mathfrak{M}$ with at most m states, such that $\mathcal{D}_{\mathfrak{M}} = \mathcal{D}_{\mathcal{A}}$.

Proposition 5: Given a HMM $\mathfrak{M}$ with n states there exists a PFA $\mathcal{A}$ with at most n states such that $\mathcal{D}_{\mathcal{A}} = \mathcal{D}_{\mathfrak{M}}$.

In order to have a self-contained article, the proofs of propositions 4 and 5 are given in the appendix (sections B and C). They nonetheless also appear in [8] using a slightly different method regarding proposition 4.

III. Learning probabilistic automata

Over the years researchers have attempted to learn, infer, identify or approximate PFA from a given set of data. This task, often called language modeling [38] is seen as essential when considering pattern recognition [27], machine learning [39], computational linguistics

[40] or biology [14]. The general goal is to construct a PFA (or some alternative device) given data assumed to have been generated from this device, and perhaps the partial knowledge of the underlying structure of the PFA. A recent review on probabilistic automata learning appears in [8]. Here only a quick, in most cases complementary review, along with a set of relevant references, will be presented. We will distinguish here between the estimation of the probabilities given an automaton structure and the identification of the structure and probabilities altogether.

A. *Estimating* PFA *probabilities*

The simplest setting of this problem arises when the underlying structure corresponds to a n-gram or a k-TSA. In this case, the estimation of the parameters is as simple as the identification of the structure [21], [22].

We assume more generally that the *structural components*, Σ, Q, and δ, of a PFA $\mathcal{A}$ are given. Let S be a finite sample of training strings drawn from a regular distribution $\mathcal{D}$. The problem is to estimate the probability parameters I, P, F of $\mathcal{A}$ in such a way that $\mathcal{D}_{\mathcal{A}}$ approaches $\mathcal{D}$.

Maximum likelihood (ML) is one of the most widely adopted criteria for this estimation:

$$(\hat{I}, \hat{P}, \hat{F}) = \operatorname*{argmax}_{I,P,F} \prod_{x \in S} \Pr_{\mathcal{A}}(x). \tag{8}$$

Maximizing the likelihood is equivalent to minimizing the empirical cross entropy $\hat{\mathcal{X}}(S, \mathcal{D}_{\mathcal{A}})$ (see section VI of [1]). It can be seen that, if $\mathcal{D}$ is generated by some PFA $\mathcal{A}'$ with the same *structural components* of $\mathcal{A}$, optimizing this criterion guarantees that $\mathcal{D}_{\mathcal{A}}$ approaches $\mathcal{D}$ as the size of S goes to infinity [41].

The optimization problem (8) is quite simple if the given automaton is deterministic [42]. Let $\langle Q, \Sigma, \delta, q_0, F, P \rangle$ be the given DPFA whose parameters F and P are to be estimated. For all $q \in Q$, a ML estimation of the probability of the transition $P(q, a, q')$ is obtained by just counting the number of times this transition is used in the *deterministic* derivations of the strings in S and normalizing this count by the frequency of use of the state q. Similarly, the final state probability $F(q)$ is obtained as the relative frequency of state q being final through the parsing of S. Probabilistic parameters of *non-ambiguous* PFA or λ-PFA can also be easily ML-estimated in the same way.

However, for general (non-deterministic, ambiguous) PFA or λ-PFA, multiple derivations are possible for each string in S and things become more complicated. If the values of I, P and F of $\mathcal{A}$ are constrained to be in $\mathbb{Q}^+$, the decisional version of this problem is clearly in **NP** and the conjecture is that this problem is at least **NP**-Hard. In practice, only locally optimal solutions to the optimization (8) are possible.

As discussed in [5], the most widely used algorithmic solution to (8) is the well known expectation-maximization (EM) *Baum-Welch algorithm* [2], [3], [6]. It iteratively updates the probabilistic parameters (I, F and P) in such a way that the likelihood of the sample is guaranteed not to decrease after each iteration. The parameter updating is based on the forward *and* backward dynamic programming recurrences to compute the probability of a string discussed in section-III of [1]. Therefore the method is often referred to as *backward-forward* re-estimation. The time and space complexities of each *Baum-Welch* iteration are $\mathcal{O}(M \cdot N)$ and $\mathcal{O}(K \cdot L + M)$, respectively, where $M = |\delta|$ (number of transitions), $K = |Q|$ (number of states) $N = ||S||$ (number of symbols in the sample), and $L = \max_{x \in S} |x|$ (length of the longest training string) [5].

Using the *optimal path* (Viterbi) approximation rather than the true (*forward)* probability (see [1], sections III-B and III-A, respectively) in the function to be optimized (8), a simpler algorithm is obtained, called the *Viterbi re-estimation algorithm*. This is discussed in [5], while re-estimation algorithms for other criteria different from ML can be found in [7], [43]–[45].

Baum-Welch and Viterbi re-estimation techniques adequately cope with the multiple-derivations problem of ambiguous PFA. Nevertheless, they can also be applied to the simpler case of *non-ambiguous* PFA and, in particular, the deterministic PFA discussed above. In these cases, the following properties hold:

Proposition 6: For non-ambiguous PFA (and for DPFA in particular),

1) the *Baum-Welch* and the *Viterbi* re-estimation algorithms produce the same solution;
2) the *Viterbi* re-estimation algorithm stops after only one iteration;
3) the solution is unique (global maximum of equation (8)).

B. Learning the structure

We will first present informally the most classic learning paradigms and discuss their advantages and drawbacks. We will then present the different results of learning.

Learning paradigms: In the first learning paradigm, proposed by Gold [46], [47], there is an infinite source of examples that are generated following the distribution induced by a hidden target. The learning algorithm is expected to return after each new example some hypothesis, and we will say that the class is identifiable in the limit with probability one if whatever the target the algorithm identifies the target (*i.e.* there is a point from which the hypothesis is equivalent to the target) with probability one.

The main drawbacks of this paradigm are:

- it does not entail complexity constraints;
- we usually don't know if the amount of data needed by the algorithm is reached;
- an algorithm can be proven to identify in the limit and might return arbitrary bad answers if the required amount of data is not provided.

Despite these drawbacks, the identification in the limit paradigm can be seen as a necessary condition for learning a given class of model. If this condition is not met, that means that some target is not learnable.

A second learning paradigm was proposed by Valiant and extended later [48]–[52]. This paradigm, called probably approximatively correct (PAC) learning, requires that the learner returns a *good* approximation of the target with *high* probability. The words *good* and *high* are formalized in a probabilistic framework and are function of the amount of data provided.

These frameworks have been adapted to the cases where the target concept is a probabilistic model [19], [53]–[58].

Finally, another framework comes from traditional methods for HMM estimation. In this framework, the structure of the model is somehow parameterized and learning is seen as a problem of parameter estimation. In the most general statement of this problem for PFA, only the alphabet (of size n) and the number of states (m) are given and the problem is to estimate the probabilities of all the $n \cdot m^2$ possible transitions. As discussed in section III-A, the Baum-Welch (or the Viterbi) algorithm can be used for a locally optimal estimation of these parameters. However, given the very large amount of parameters, this general method

has seldom proved useful in practice. Related approaches where the amount of parameters to estimate is explicitly constrained are discussed in [8].

What can be learned?: This section addresses previous works related to the learning of probabilistic finite-state automata. The first results came from Horning [53] who showed that any recursively enumerable class of languages can be identified in the limit with probability one. The problem of the proof —among others of the same spirit [54], [55]— is that it does not provide us with a reasonable algorithm to perform the learning task.

A more constructive proof, relying on a reasonable algorithm, was proposed in [57]: Identification in the limit of DPFA is shown. This proof is improved in [59] with results concerning the identification of rational random variables.

Work has also been done in the Probably Approximately Correct (PAC) learning paradigm. The results are rather different depending on the object we want to infer and/or what we know about it. Actually, Abe and Warmuth [17] showed that non-deterministic acyclic automata that defined a probability distribution over Σ^n, with n and Σ known, could be approximated in polynomial time. Moreover, they showed that learnability is not polynomial in the size of the vocabulary. Kearns *et al.* [18] showed that an algorithm that aims at learning a probabilistic function cannot reach its goal[5] if the probability distribution can be generated by a DPFA over $\{0,1\}^n$. Thus knowing the class of the object we want to infer helps a lot the inference since the object dealt with in [17] are more complex than the ones addressed in [18]. Following this idea, Ron and *al.* [19] proposed a practical algorithm that converges in a PAC like framework that infers a restricted class of acyclic automata. More recently Clark and Thollard [58] showed that the result holds with cyclic automata as soon as a bound on the expected length of the generated strings is known.

Some algorithms: If we restrict ourselves to the class of n-gram or k-TSA distributions, as previously mentioned, learning both the structure and the probabilities of n-grams or k-TSA is simple and already very well known [21], [22]

For more general PFAs, another strategy can be followed: first the probabilistic prefix tree automaton (PPTA), which models the given training data with maximum-likelihood, is

[5]Actually, the authors showed that this problem was as hard as learning parity functions in a noisy setting for the non-probabilistic PAC framework. This problem is generally believed to be untractable.

constructed. This PPTA is then generalized using state-merging operations. This is usually called the *state-merging* strategy.

Following this strategy, Carrasco and Oncina [60] proposed the ALERGIA algorithm for DPFA learning. Stolcke and Omohundro [20] proposed another learning algorithm that infer DPFA based on Bayesian learning. Ron and *al.* [19] reduced the class of the language to be learned and provided another state-merging algorithm and Thollard and *al.* [61] proposed the MDI algorithm under the same framework. MDI has been shown to outperform ALERGIA on a natural language modeling task [61] and on *shallow parsing* [62]. A recent variant of ALERGIA was proposed in [63] and evaluated on a natural language modeling task. A modification of this algorithm was also used in [64] to discover the underlying model in structured text collections.

Other learning approaches: While not a learning algorithm in itself, a (heuristic) general learning scheme which is worth mentioning can be derived from the *stochastic morphism theorem* shown in Section II-B. In fact, the use of the conventional morphism theorem [26] was already proposed in [65] to develop a general methodology for learning general regular languages, called "*morphic generator grammatical inference*" (MGGI). The basic idea of MGGI was to rename the symbols of the given alphabet in such a manner that the syntactic restrictions which are desirable in the target language can be described by simple *local languages*. MGGI constitutes an interesting engineering tool which has proved very useful in practical applications [25], [65].

We briefly discuss here how the stochastic morphism theorem can be used to obtain a stochastic extension of this methodology, which will be called *stochastic* MGGI (SMGGI).

Let S be a finite sample of training sentences over Σ and let Σ' be the alphabet required to implement an adequate *renaming function* $g\colon S \to \Sigma'^{\star}$. Let $h\colon \Sigma'^{\star} \to \Sigma^{\star}$ be a letter-to-letter morphism; typically one such that $h(g(S)) = S$. Then, a 2-TSA model can be obtained and the corresponding transition and final-state probabilities max-likelihood estimated from $g(S)$ using conventional bigram learning or the 2-TSI algorithm [22].

Let $\mathcal{D}_2(g(S))$ be the stochastic local language generated by this model. The final outcome of SMGGI is then defined as the regular distribution $\mathcal{D} = h(\mathcal{D}_2(g(S))$; that is:

$$\forall x \in \Sigma^\star,\ \ \Pr_{\mathcal{D}}(x) = \sum_{y \in h^{-1}(x)} \Pr_{\mathcal{D}_2(g(S))}(y)\ , \tag{9}$$

where $h^{-1}(x) = \{y \in \Sigma'^\star :\ y = h(x)\}$.

From a practical point of view, the morphism h is just applied to the terminal symbols of the 2-TSA generating $\mathcal{D}_2(g(S))$. While this automaton (defined over Σ') has deterministic structure and therefore is unambiguous, after applying h the resulting automaton is often ambiguous, thus precluding a simple max-likelihood estimation of the corresponding transition and final state probabilities. Nevertheless, equation (9) allows us to directly use the the 2-TSA probabilities with the guarantee that they constitute a proper estimation for the possibly ambiguous resulting automaton.

C. Smoothing issues

The goal of smoothing is estimating the probability of events that have never been seen in the training data available. From the theoretical point of view, smoothing must be taken into account since estimates must behave well on the whole set $\Sigma^\star$. From the practical point of view, we saw that the probability of a sequence is computed using products of probabilities associated with the symbols. Smoothing is necessary to distinguish a very probable sequence with a unique unknown symbol (*e.g.* in natural language modeling this can be a sentence with an unknown proper noun) from a sequence composed of impossible concatenations of symbols.

Even though some work has been done in order to theoretically justify some smoothing techniques – *e.g.* the Good-Turing estimator [39], [66] – smoothing has mainly been considered from the practical point of view. The main line of research is considering the n-gram model as the base model and a back-off strategy as the smoothing technique [10], [38], [67]–[69]. In the back-off strategy another model is used (usually a more general one) in order to estimate the probability of a sequence; for example, if there is no trigram to estimate a conditional probability, a bigram can be used to do it. In order to guaranty an overall consistent model, several variants have been considered. After the backing-off, the trigram can again be used to estimate the probabilities.

Smoothing PFA is a harder problem. Even if we can think about backing-off to simpler and more general models, it is not easy to use the PFA to continue the parsing after the backing-off. A first strategy consists in backing-off to a unigram and finishing the parsing in the unigram [70] itself. A more clever strategy is proposed by Llorens et al. [71], which use a (recursively smoothed) n-gram as a back-off model. The *history* of each PFA state is computed in order to associate it with the adequate n-gram state(s). Parsing can then go back and forth through the full hierarchy of PFA *and* m-gram states, $0 < m \leq n$, as needed for the analysis of any string in $\Sigma^\star$. This strategy performs better in term of predicting power, but is obviously more expensive in terms of computing time. An error correcting approach can also be used, which consists in looking for the string generated by the PFA that with maximum likelihood may have been "distorted" (by an error model) into the observed string [11], [72].

Smoothing can be considered either as a distribution estimation technique or as a post-processing technique used to improve the result of a given estimator. Some other pre/post processing techniques have been proposed in order to improve a machine learning algorithm.

In the spirit of pre-processing the data, [73] cluster the data using a statistical clustering algorithm [74]. The inference algorithm will then provide a class-model. This technique allows to work on tasks with large vocabularies (*e.g.* 65,000 words). Moreover, it seems to improve the power of prediction of the model. Another way of dealing with the data is by typing it. For example, in natural language processing, we can type a word using some syntactic information such as the part of speech it belongs to. The idea is to take external information into account during the inference. A general framework for taking into account typed data for the inference of PFA was studied in [75].

Another technique that pre-processes the data is *bagging* [76]. It was successfully adapted to the inference of PFA applied on a noun phrase chunking task [62].

IV. PROBABILISTIC EXTENSIONS

A number of natural extensions of the PFA and DPFA have been proposed. We mention in the sequel some of the most important ones. These include *probabilistic finite-state transducers*, and *stochastic finite-state tree automata*. These models are related with the more general *stochastic context-free grammars*, for which a short account is also given.

A. Probabilistic finite-state transducers

Stochastic finite-state transducers (SFSTs) are similar to PFA, but in this case two different alphabets are involved: source (Σ) and target (Δ) alphabets. Each transition in a SFST has attached a source symbol and a (possible empty) string of target symbols.

Different types of SFSTs have been applied with success in some areas of machine translation and pattern recognition [77]–[83]. On the other hand, in [40], [84], [85], *weighted finite-state transducers* are introduced. Another (context-free) generalization, *head transducer models*, was proposed in [86], [87].

A SFST $\mathcal{T}$ is defined as an extension of PFA: $\mathcal{T}$=$\langle Q, \Sigma, \Delta, \delta, I, F, P\rangle$, where: Q is a finite set of *states*; Σ and Δ are the source and target *alphabets*, respectively, $\delta \subseteq Q \times \Sigma \times \Delta^* \times Q$ is a *set of transitions*; $I : Q \to \mathbb{R}^+$ and $F : Q \to \mathbb{R}^+$ are the *initial-* and *final-state probabilities*, respectively; and $P : \delta \to \mathbb{R}^+$ are the *transition probabilities*, subject to the following normalization constraints:

$$\sum_{q\in Q} I(q) = 1, \quad \text{and} \quad \forall q \in Q,\ F(q) \ + \sum_{a\in\Sigma,\ q'\in Q, y\in\Delta^*} P(q, a, y, q') \ = \ 1.$$

A particular case of SFST is the *deterministic* SFST, where $(q, a, u, r) \in \delta$ and $(q, a, v, s) \in \delta$ implies $u = v$ and $r = s$. A slightly different type of deterministic SFST is the *subsequential transducer (SST)* which can produce an additional target substring when the end of the input string has been detected.

Much in the same way a PFA generates an inconditional distribution on $\Sigma^\star$, if a SFST has no useless states it generates a joint distribution $\Pr_{\mathcal{T}}$ on $\Sigma^\star \times \Delta^\star$.

Given a pair $(t, x) \in \Delta^* \times \Sigma^*$, the computation of $\Pr_{\mathcal{T}}(t, x)$ is quite similar to the computation of $\Pr_{\mathcal{A}}(x)$ for a PFA $\mathcal{A}$ [81]. Other related problems arise in the context of SFST [7], [88]. One of the most interesting is the *stochastic translation problem*: Given a SFST $\mathcal{T}$ and $x \in \Sigma^\star$, compute[6]:

$$\operatorname*{argmax}_{t\in\Delta^\star} \Pr_{\mathcal{T}}(t, x) \ . \tag{10}$$

This problem has been proved to be **NP**-Hard [88], but an approximate solution can be computed in polynomial time by using an algorithm similar to the Viterbi algorithm for PFA [7], [43].

[6]SFSTs can be used in statistical machine translation, where the problem is to find a target-language sentence that maximizes the conditional probability $\Pr(t \mid x)$. This is equivalent to equation 10; i.e., $\max_t \Pr(t \mid x) = \max_t \Pr(t, x)$.

For certain particular cases of SFSTs, the (exact) stochastic translation problem is computationally tractable. If the SFST $\mathcal{T}$ is *non-ambiguous in the translation sense* ($\forall x \in \Sigma^\star$ there are not two target sentences $t, t' \in \Delta^\star$, $t \neq t'$, such that $\Pr_{\mathcal{T}}(t, x) > 0$ and $\Pr_{\mathcal{T}}(t', x) > 0$), the translation problem is polynomial. Moreover, if $\mathcal{T}$ is simply *non-ambiguous* ($\forall x \in \Sigma^\star$ and $\forall t \in \Delta^\star$ there are not two different sequences of states that deal with (x, t) with probability greater then zero), the translation problem is also polynomial. In these two cases, the computation can be carried out using an adequate version of the Viterbi algorithm. Finally, if $\mathcal{T}$ is *subsequential*, or just *deterministic* with respect to the input symbol, the stochastic translation problem is also polynomial, though in this case the computational cost is $\mathcal{O}(|x|)$, independent of the size of $\mathcal{T}$.

The components of a SFST (states, transitions and the probabilities associated to the transitions) can be learned from training pairs in a single process or in a two-step process. In the latter case, first the structural component is learned and next the probabilistic components are estimated from training samples. The GIATI (*Grammatical Inference and Alignments for Translator Inference*)[7] is a technique of the first type [81], [89], while OSTIA (*Onward Subsequential Transducer Inference Algorithm*) and OMEGA (OSTIA *Modified for Employing Guarantees and Alignments*) are techniques for learning the structural component of a SFST [79], [80]. Only a few other techniques exist to infer finite-state transducers [77], [90]–[92]

To estimate the probabilistic component in the two-step approaches, *maximum likelihood* or other criteria can be used [7], [45], [93]. One of the main problems associated with the learning process is the modeling of events not seen in the training set. As previously discussed for PFA, this problem can be tackled by using smoothing techniques; either in the estimation of the probabilistic components of the SFSTs [94] or within of the process of learning both components [81].

B. Stochastic context-free grammars

Stochastic context-free grammars are the natural extension of probabilistic finite-state transducers. These models are defined as a tuple $\langle Q, \Sigma, S, R, P \rangle$, where Q is a set of non-terminal symbols, Σ is an finite alphabet, $S \in Q$ is the initial symbol, R is a set of rules

[7]In earlier papers this technique was called MGGI (*Morphic Generator Transducer Inference*).

$A \to \omega$ with $\omega \in (Q \cup \Sigma)^\star$ and $P : R \to \mathbb{R}^+$ is the set of probabilities attached to the rules such that $\sum_{\omega \in (Q \cup \Sigma)^\star} P(A \to \omega) = 1$ for all $A \in Q$.

In general, parsing strings with these models is in $\mathcal{O}(n^3)$ (although quadratic algorithms can be designed for special types of stochastic context-free grammars) [4], [5]. Approximations to stochastic context-free grammars using probabilistic finite-state automata have been proposed in [95], [96]. Algorithms for the estimation of the probabilities attached to the rules are basically the *inside-outside algorithm* [4], [97] and a *Viterbi-like algorithm* [98]. The relation between the probability of the optimal path of states and the probability of generating a string has been studied in [99]. The structure of stochastic context-free grammars (the non-terminal symbols and the rules) can currently be learned from examples [100]–[102] in very limited settings only (e.g., when grammars are *even linear*). An alternative line is to learn the context-free grammar from the examples and by ignoring the distribution: Typically, Sakakibara's *reversible* grammars [103] have been used for this purpose; then, the inside-outside algorithm is used to estimate the probabilities.

There are also extensions of stochastic context-free grammars for translation: *stochastic syntax-directed translation schemata* [104] and *head transducer models* were proposed in [86], [87].

C. Stochastic finite-state tree automata

Stochastic models that assign a probability to a tree can be useful, for instance, in natural language modeling to select the best parse tree for a sentence and resolve structural ambiguity. For this purposes, finite-state automata that operate on trees can be defined [15]. In contrast to the case of strings, where the automaton computes a state for every prefix, a frontier-to-root tree automaton processes the tree bottom-up and state is computed for every subtree. The result depends on both the node label and the states obtained after the node subtrees. Therefore, a collection of transition functions, one for each possible number of subtrees, is needed. This probabilistic extension defines a probability distribution over the set T_Σ of labeled trees.

A *probabilistic finite-state tree automaton* (PTA) is defined as $M = (Q, \Sigma, \Delta, P, \rho)$, where

- Q is a finite set of states;

- Σ is the alphabet;
- $\Delta = \{\delta_0, \delta_1, \ldots, \delta_M\}$ is a collection of transition sets $\delta_m \subset Q \times \Sigma \times Q^m$;
- P is a collection of functions $P = \{p_0, p_1, p_2, \ldots, p_M\}$ of the type $p_m : \delta_m \to [0, 1]$;
- ρ are the root probabilities $\rho : Q \to [0, 1]$.

The required normalizations are

$$\sum_{q \in Q} \rho(q) = 1 \ , \tag{11}$$

and, for all $q \in Q$,

$$\sum_{a \in \Sigma} \sum_{m=0}^{M} \sum_{\substack{i_1, \ldots, i_m \in Q: \\ (q, a, i_1, \ldots, i_m) \in \delta_m}} p_m(q, a, i_1, \ldots, i_m) \quad = \quad 1 \ . \tag{12}$$

The probability of a tree t in the stochastic language generated by A is defined as

$$p(t|A) = \sum_{q \in Q} \rho(q) \cdot \pi(q, t) \ , \tag{13}$$

where $\pi(q, t)$ is recursively defined as:

$$\pi(q, t) = \begin{cases} p_0(q, a) \ \ \text{if } t = a \in \Sigma \, , \\ \displaystyle\sum_{\substack{i_1, \ldots, i_m \in Q: \\ (q, a, i_1, \ldots, i_m) \in \delta_m}} p_m(\ q, a, \delta(t_1), \ldots, \delta(t_m)) \cdot \pi(i_1, t_1) \cdots \pi(i_m, t_m) \, , \\ \quad \text{if } t = a(t_1 \cdots t_m) \in T_\Sigma - \Sigma \, , \\ 0 \ \ \text{otherwise.} \end{cases} \tag{14}$$

As in the case of PFA it is possible to define deterministic PTA as those where the set $\{q \in Q : (q, a, i_1, ..., i_m) \in \delta_m\}$ has size at most 1 for all $a \in \Sigma$, for all $m \geq 0$ and for all $i_1, ..., i_m \in Q$. In such a case, a minimal automaton can be defined and it can be identified from samples [15].

In contrast, the consistency of probabilistic tree automata is not guaranteed by the normalizations (11) and (12) even in the absence of useless states. Consistency requires that the *spectral radius* of the production matrix Λ defined below is strictly smaller than 1 [42]:

$$\Lambda_{ij} \ = \ \sum_{a \in \Sigma} \sum_{m=1}^{M} \sum_{\substack{i_1, i_2, \ldots, i_m \in Q: \\ (i, a, i_1, \ldots, i_m) \in \delta_m}} p_m(i, a, i_1, i_2, ..., i_m) \cdot (1(j, i_1) + \cdots + 1(j, i_m)) \ , \tag{15}$$

where $1(i, j)$ is Kronecker's delta defined before.

V. Conclusion

We have in this paper proposed a survey of the properties concerning deterministic and non-deterministic probabilistic finite-state automata. A certain number of results have been proved and others can be fairly straightforwardly derived from them. On the other hand, we have left many questions not answered in this work. They correspond to problems that to our knowledge are open or, even in a more extensive way, to research lines that should be followed. Here are some of these:

1) We studied in the section concerning topology of part I [1] the questions of computing the distances between two distributions represented by PFA. In the case where the PFA are DPFA the computation of the L_2 distance and of the Kullback-Leibler divergence can take polynomial time, but what about the L_1, L_∞ and logarithmic distances?
2) In the same trend it is reasonably clear that if at least one of the distributions is represented by a PFA, the problem of computing or even approximating the L_1 (or L_∞) is **NP**-hard. What happens for the other distances? The approximation problem can be defined as follows: Given an integer m decide if $d(\mathcal{D}, \mathcal{D}') < \frac{1}{m}$.
3) In [105] the question of computing the weight of a language inside another (or following a regular distribution) is raised. Technically, it requires computing $\sum_{w \in L_{\mathcal{A}}} \Pr \mathcal{B}(w)$ where $\mathcal{A}$ is a DFA and $\mathcal{B}$ is a DPFA. Techniques for special cases are proposed in [105] but the general question is not solved. The problem is clearly polynomially solvable; the problem is that of finding a fast algorithm.
4) The equivalence of HMM has been studied in [106], where it is claimed that it can be tested in polynomial time. When considering the results from our section II-C it should be possible to adapt the proof in order to obtain an equivalent result for PFA.
5) We have provided a number of results on distances in the section concerning distances of part I [1]. Yet a comparison of these distances, and how they relate to learning processes would be of clear interest. From the theoretical point of view, in probabilistic PAC learning framework, the error function used is usually the Kullback-Leibler divergence [17]–[19], [56], [58]. As we mentioned many other measures exist and it should be interesting to study learnability results while changing the similarity measure.
6) Smoothing is a crucial issue for language modeling (see section III-C). Good smooth-

ing techniques for PFA and DPFA would surely improve the modeling capacities of these models, and it can be conjectured that they might perform better than standard techniques.

7) Testing the closeness of two distributions from samples is also an issue that matters: Whether to be able to use larger data sets for learning or to be able to decide merging in learning algorithms, one wishes to be able to have a simple test to decide if two samples come from the same (or sufficiently similar) distribution or not.
8) Following [88], we recall that the problem of deciding whether the probability of the most probable string is more than a given fraction is **NP**-hard. It is not known if the problem belongs to **NP**.

Obviously there are many topics related with PFA that require further research efforts and here only few are mentioned. To mention but one of these topics, probabilistic (finite or context-free) transducers are increasingly becoming important devices, where only a few techniques are known to infer finite-state transducers from training pairs or to smooth probabilistic finite-state transducers when the training pairs are scarce.

Solving some of the above problems, and in a more general way, better understanding how PFA and DPFA work, would necessarily increase their importance and relevance in a number of fields, and specifically those that are related to pattern recognition.

ACKNOWLEDGMENTS

The authors wish to thank the anonymous reviewers for their careful reading and in-depth criticisms and suggestions.

APPENDIX

A. Proof of the Theorem 3

Theorem 3 (Stochastic morphism theorem)
Let Σ be a finite alphabet and $\mathcal{D}$ a stochastic regular language on $\Sigma^\star$. There exist then a finite alphabet Σ', a letter-to-letter morphism $h : \Sigma'^\star \to \Sigma^\star$, and a stochastic local language over Σ', $\mathcal{D}_2$, such that $\mathcal{D} = h(\mathcal{D}_2)$; i.e.,

$$\forall x \in \Sigma^\star, \quad \Pr_{\mathcal{D}}(x) \;=\; \Pr_{\mathcal{D}_2}(h^{-1}(x)) \;=\; \sum_{y \in h^{-1}(x)} \Pr_{\mathcal{D}_2}(y) \;, \tag{16}$$

where $h^{-1}(x) = \{y \in \Sigma'^\star \mid x = h(y)\}$.

Proof: By proposition 11 of [1], $\mathcal{D}$ can be generated by a PFA with a *single initial state*. Let $\mathcal{A} =< Q, \Sigma, \delta, q_0, F, P >$ be such a PFA. Let $\Sigma' = \{a_q | (q', a, q) \in \delta\}$ and define a letter-to-letter morphism $h : \Sigma' \to \Sigma$ by $h(a_q) = a$. Next, define a stochastic local language, $\mathcal{D}_2$, over Σ' by $Z = (\Sigma', P_I, P_F, P_T)$, where

$$P_I(a_q) = P(q_0, a, q), \quad P_F(a_q) = F(q), \quad P_T(a'_{q'}, a_q) = P(q', a, q) \;. \tag{17}$$

Now, let $x = x_1 \ldots x_n$ be a non-empty string over Σ, with $\Pr_{\mathcal{D}}(x) > 0$. Then, at least one valid path exists for x in $\mathcal{A}$. Let θ be one of these paths, with $s_0 = q_0$:

$$\theta = (s_0, x_1, s_1) \ldots (s_{n-1}, x_n, s_n) \;.$$

Associated with θ, define a string y over Σ' as:

$$y \;=\; y_1 \ldots y_n \;=\; x_{1s_1} \ldots x_{ns_n} \;.$$

Let Y be the set of strings in $\Sigma'^\star$ associated with all the valid paths for x in $\mathcal{A}$. Note that for each $y \in Y$ there is a unique path for x and vice-versa. Note also that $x = h(y)$. Therefore $Y = h^{-1}(x)$.

If $x = \lambda$, it has a unique degenerate path consisting only in q_0; that is $Y = \{\lambda\}$ and $\Pr_{\mathcal{D}_2}(\lambda) = F(q_0) = \Pr_{\mathcal{D}}(\lambda)$. Otherwise, from equations (4) and (17), the probability of every $y \in Y$ is:

$$\begin{aligned} \Pr_{\mathcal{D}_2}(y) \;&=\; P_I(x_{1s_1}) \cdot \prod_{i=2}^{n} P_T(x_{i-1s_{i-1}}, x_{is_i}) \cdot P_F(x_{ns_n}) \\ &=\; P(s_0, x_1, s_1) \cdot \prod_{i=2}^{n} P(s_{i-1}, x_i, s_i) \cdot F(s_n) \;; \end{aligned}$$

which, according to equation (1) in section II-F of [1] (and noting that in our PFA $I(q_0) = 1$), is the probability of the path for x in $\mathcal{A}$ y is associated with. Finally, following equation (2) in section II-F of [1] (that gives the probability of generating a string),

$$\sum_{y \in Y} \Pr_{\mathcal{D}_2}(y) = \Pr_{\mathcal{A}}(x) \quad \forall x : \Pr \mathcal{D}(x) > 0 \ .$$

On the other hand, if $\Pr_{\mathcal{D}}(x) = 0$, then $Y = \emptyset$, leading to $\sum_{y \in Y} \Pr_{\mathcal{D}_2}(y) = 0$. Therefore, since $Y = h^{-1}(x)$, we have $h(\mathcal{D}_2) = \mathcal{D}$. ■

This proof is a probabilistic generalization of the proof for the classical morphism theorem [26]. Given the non-equivalence of PFA and DPFA, the present construction required the use of non-deterministic and possibly ambiguous finite-state automata.

B. Proof of the Proposition 4

Proposition 4 *Given a* PFA $\mathcal{A}$ *with* m *transitions and* $\Pr_{\mathcal{A}}(\lambda) = 0$*, there exists a* HMM $\mathfrak{M}$ *with at most* m *states, such that* $\mathcal{D}_{\mathfrak{M}} = \mathcal{D}_{\mathcal{A}}$*.*

Proof: Let $\mathcal{A} = \langle Q, \Sigma, \delta, I, F, P \rangle$ be a PFA. We create an equivalent HMM $\mathfrak{M} = \langle \mathrm{Q}, \Sigma, \mathrm{I}, \ \mathrm{q_f}, \mathrm{T}, \mathrm{E} \rangle$ as follows

- $\mathrm{Q} = Q \times Q$;
- $\mathrm{I}(q, q') = I(q) \cdot \sum_{a \in \Sigma} P(q, a, q')$ for all $(q, q') \in \mathrm{Q}$;
- $\mathrm{T}((q, q'), (q', q'')) = \sum_{a \in \Sigma} P(q', a, q'')$ and $\mathrm{T}((q, q'), \mathrm{q_f}) = F(q')$;
- $\mathrm{E}((q, q'), a) = \frac{P(q,a,q')}{\sum_{b \in \Sigma} P(q,b,q')}$ if $P(q, a, q') \neq 0$

For each $x = x_1 \ldots x_{|x|} \in \Sigma^\star$ with $\Pr_{\mathcal{A}}(x) \neq 0$, there is at least a sequence of states $(s_0, \ldots, s_{|x|})$ that generates x with probability:

$$I(s_0) \cdot P(s_0, x_1, s_1) \cdot \cdots \cdot P(s_{|x|-1}, x_{|x|}, s_{|x|-1}, s_{|x|}) \cdot F(s_{|x|}) \ .$$

And in $\mathfrak{M}$,

$$\mathrm{I}(s_0, s_1) \cdot \mathrm{E}((s_0, s_1), x_1) \cdot \mathrm{T}((s_0, s_1), (s_1, s_2)) \ldots \mathrm{E}((s_{|x|-1}, s_{|x|}), x_{|x|}) \cdot \mathrm{T}((s_{|x|-1}, s_{|x|}), \mathrm{q_f}) \ .$$

For each path in $\mathcal{A}$ there is one and only one path in HMM, the theorem holds. ■

C. Proof of the Proposition 5

Proposition 5 *Given a* HMM $\mathfrak{M}$ *with* n *states there exists a* PFA $\mathcal{A}$ *with at most* n *states such that* $\mathcal{D}_{\mathcal{A}} = \mathcal{D}_{\mathfrak{M}}$.

Proof: Let $\mathfrak{M} = \langle \mathrm{Q}, \Sigma, \mathrm{I}, \mathrm{q_f}, \mathrm{T}, \mathrm{E} \rangle$ be a HMM. We create an equivalent PFA $\mathcal{A}' = \langle Q, \Sigma, I, \delta, F, P \rangle$ as follows:

$Q = \mathrm{Q}$;

$I(q) = \mathrm{I}(q)$, for all $q \in Q \diagdown \{\mathrm{q_f}\}$, and $I(\mathrm{q_f}) = 0$;

$\delta = \{(q, a, q') : \mathrm{T}(q, q') \neq 0 \text{ and } \mathrm{E}(q, a) \neq 0\}$;

$F(q) = 0$ for all $q \in Q \diagdown \{\mathrm{q_f}\}$, and $F(\mathrm{q_f}) = 1$;

$P(q, a, q') = \mathrm{E}(q, a) \cdot \mathrm{T}(q, q')$.

For each $x = x_1 \ldots x_{|x|} \in \Sigma^\star$ with $\Pr_{\mathfrak{M}}(x) \neq 0$, there is at least a sequence of states $(s_1, \ldots, s_{|x|}, \mathrm{q_f})$ that generates with x with probability:

$$\mathrm{I}(s_1) \cdot \mathrm{E}(s_1, x_1) \cdot \mathrm{T}(s_1, s_2) \cdots \mathrm{T}(s_{|x|-1}, s_{|x|}) \cdot \mathrm{E}(s_{|x|}, x_{|x|}) \cdot \mathrm{T}(s_{|x|}, \mathrm{q_f}) \ .$$

And in $\mathcal{A}'$,

$$I(s_1) \cdot P(s_1, x_1, s_2) \cdots P(s_{|x|}, x_{|x|}, \mathrm{q_f}) \ .$$

For each path in $\mathfrak{M}$ there is one and only one path in $\mathcal{A}'$. Moreover, by construction, $I(s_1) = \mathrm{I}(s_1)$ and $P(q, a, q') = \mathrm{E}(q, a) \cdot \mathrm{T}(q, q')$; Therefore $\mathcal{D}_{\mathcal{A}'} = \mathcal{D}_{\mathfrak{M}}$. Finally, by proposition 11 of [1], we can build a PFA $\mathcal{A}$, with at most $|Q| = n$ states, such that $\mathcal{D}_{\mathcal{A}'} = \mathcal{D}_{\mathcal{A}}$.

■

REFERENCES

[1] E. Vidal, F. Thollard, C. de la Higuera, F. Casacuberta, and R. C. Carrasco, "Probabilistic finite state automata – part I," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. Special Issue-Syntactic and Structural Pattern Recognition, 2004.

[2] L. E. Baum, "An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes," *Inequalities*, vol. 3, pp. 1–8, 1972.

[3] C. F. J. Wu, "On the convergence properties of the EM algorithm," *Annals of Statistics*, vol. 11, no. 1, pp. 95–103, 1983.

[4] F. Casacuberta, "Statistical estimation of stochastic context-free grammars," *Pattern Recognition Letters*, vol. 16, pp. 565–573, 1995.

[5] ——, "Growth transformations for probabilistic functions of stochastic grammars," *International Journal on Pattern Recognition and Artificial Intelligence*, vol. 10, no. 3, pp. 183–201, 1996.

[6] G. J. McLachlan and T. Krishnan, *The EM Algorithm and Extensions*. Wiley, 1997.

[7] D. Picó and F. Casacuberta, "Some statistical-estimation methods for stochastic finite-state transducers," *Machine Learning Journal*, vol. 44, no. 1, pp. 121–141, 2001.

[8] P. Dupont, F. Denis, and Y. Esposito, "Links between probabilistic automata and hidden markov models: probability distributions, learning models and induction algorithms," *Pattern Recognition*, 2004, to appear.

[9] I. H. Witten and T. C. Bell, "The zero frequency problem: Estimating the probabilities of novel events in adaptive test compression," *IEEE Trans.*, vol. IT-37, no. 4, pp. 1085–1094, 1991.

[10] H. Ney, S. Martin, and F. Wessel, *Corpus-Based Statiscal Methods in Speech and Language Processing*. S. Young and G. Bloothooft, Kluwer Academic Publishers, 1997, ch. Statistical Language Modeling Using Leaving-One-Out, pp. 174–207.

[11] P. Dupont and J.-C. Amengual, "Smoothing probabilistic automata: an error-correcting approach," ser. Lecture Notes in Computer Science, A. de Oliveira, Ed., vol. 1891. Berlin, Heidelberg: Springer-Verlag, 2000, pp. 51–6.

[12] Y. Sakakibara, M. Brown, R. Hughley, I. Mian, K. Sjolander, R. Underwood, and D. Haussler, "Stochastic context-free grammars for tRNA modeling," *Nuclear Acids Res.*, vol. 22, pp. 5112–5120, 1994.

[13] T. Kammeyer and R. K. Belew, "Stochastic context-free grammar induction with a genetic algorithm using local search," in *Foundations of Genetic Algorithms IV*, R. K. Belew and M. Vose, Eds. University of San Diego, CA, USA: Morgan Kaufmann, 1996.

[14] N. Abe and H. Mamitsuka, "Predicting protein secondary structure using stochastic tree grammars," *Machine Learning*, vol. 29, pp. 275–301, 1997.

[15] R. C. Carrasco, J. Oncina, and J. Calera-Rubio, "Stochastic inference of regular tree languages," *Machine Learning Journal*, vol. 44, no. 1, pp. 185–197, 2001.

[16] M. Kearns and L. Valiant, "Cryptographic limitations on learning boolean formulae and finite automata," in *21st ACM Symposium on Theory of Computing*, 1989, pp. 433–444.

[17] N. Abe and M. Warmuth, "On the computational complexity of approximating distributions by probabilistic automata," *Machine Learning*, vol. 9, pp. 205–260, 1992.

[18] M. Kearns, Y. Mansour, D. Ron, R. Rubinfeld, R. E. Schapire, and L. Sellie, "On the learnability of discrete distributions," in *Proc. of the 25th Annual ACM Symposium on Theory of Computing*, 1994, pp. 273–282.

[19] D. Ron, Y. Singer, and N. Tishby, "On the learnability and usage of acyclic probabilistic finite automata," in *Proceedings of COLT 1995*, 1995, pp. 31–40.

[20] A. Stolcke and S. Omohundro, "Inducing probabilistic grammars by bayesian model merging," ser. Lecture Notes in Computer Science, R. C. Carrasco and J. Oncina, Eds., no. 862. Berlin, Heidelberg: Springer Verlag, 1994, pp. 106–118.

[21] F. Jelinek, *Statistical Methods for Speech Recognition*. Cambridge, Massachusetts: The MIT Press, 1998.

[22] P. García and E. Vidal, "Inference of k-Testable languages in the strict sense and application to syntactic pattern recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-12, no. 9, pp. 920–925, Sept. 1990.

[23] Y. Zalcstein, "Locally testable languages," *Journal of Computer and System Sciences*, vol. 6, pp. 151–167, 1972.

[24] R. McNaughton, "Algebraic decision procedures for local testability," *Mathematical System Theory*, vol. 8, no. 1, pp. 60–67, 1974.

[25] E. Vidal and D. Llorens, "Using knowledge to improve N-Gram Language Modelling through the MGGI methodology," in *Proceedings of ICGI '96*, ser. Lecture Notes in Computer Science, L. Miclet and C. de la Higuera, Eds. Berlin, Heidelberg: Springer Verlag, 1996, no. 1147.

[26] S. Eilenberg, *Automata, Languages and Machines. Vol. A*. New York Academic, 1974.

[27] L. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recoginition," *Proceedings of the IEEE*, vol. 77, pp. 257–286, 1989.

[28] J. Picone, "Continuous speech recognition using hidden Markov models," *IEEE ASSP magazine*, vol. 7, no. 3, pp. 26– 41, 1990.

[29] I. Bazzi, R. Schwartz, and J. Makhoul, "An omnifont open-vocabulary OCR system for english and arabic," *IEEE Trans. on PAMI*, vol. 6, no. 21, pp. 495–504, 1999.

[30] A. Toselli, A. Juan, D. Keysers, J. González, I. Salvador, H. Ney, E. Vidal, and F. Casacuberta, "Integrated handwriting recognition and interpretation using finite state models," *Int. J. Patt. Recognition and Artifial Intelligence*, 2004.

[31] F. Casacuberta, "Finite-state transducers for speech-input translation," in *Proceedings of the Workshop on Automatic Speech Recognition and Understanding*, ITC-IRST. IEEE, dec 2001.

[32] F. Casacuberta, E. Vidal, and J. M. Vilar, "Architectures for speech-to-speech translation using finite-state models," in *Proceedings of the Workshop on Speech-to-Speech Translation: Algorithms and Systems*. Philadelphia: ACL, July 2002, pp. 39–44.

[33] A. Molina and F. Pla, "Shallow parsing using specialized HMMs," *Journal on Machine Learning Research*, vol. 2, pp. 559–594, March 2002.

[34] H. Bunke and T. Caelli, Eds., *Hidden Markov Models applications in Computer Vision*, ser. Series in Machine Perception and Artificial Intelligence. World Scientific, 2001, vol. 45.

[35] R. Llobet, A. H. Toselli, J. C. Perez-Cortes, and A. Juan, "Computer-aided prostate cancer detection in ultrasonographic images," in *Proceedings of the 1st Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA)*, vol. 1, Puerto de Andratx (Mallorca, Spain), 2003, pp. 411–419.

[36] Y. Bengio, V.-P. Lauzon, and R. Ducharme, "Experiments on the application of IOHMMs to model financial returns series," *IEEE Transaction on Neural Networks*, vol. 12, no. 1, pp. 113–123, 2001.

[37] F. Casacuberta, "Some relations among stochastic finite state networks used in automatic speech recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 7, pp. 691–695, 1990.

[38] J. Goodman, "A bit of progress in language modeling," Microsoft Research, Tech. Rep., 2001.

[39] D. McAllester and R. E. Schapire, "On the convergence rate of Good-Turing estimators," in *Proc. 13th Annu. Conference on Comput. Learning Theory*. Morgan Kaufmann, San Francisco, 2000, pp. 1–6.

[40] M. Mohri, F. Pereira, and M. Riley, "The design principles of a weighted finite-state transducer library," *Theoretical Computer Science*, vol. 231, pp. 17–32, 2000.

[41] R. Chaudhuri and S. Rao, "Approximating grammar probabilities: Solution to a conjecture," *Journal of the Association for Computing Machinery*, vol. 33, no. 4, pp. 702–705, 1986.

[42] C. S. Wetherell, "Probabilistic languages : A review and some open questions," *Computing Surveys*, vol. 12, no. 4, 1980.

[43] F. Casacuberta, "Probabilistic estimation of stochastic regular syntax-directed translation schemes," in *VI Spanish Symposium on Pattern Recognition and Image Analysis*, R. Moreno, Ed. AERFAI, 1995, pp. 201–297.

[44] ——, "Maximum mutual information and conditional maximum likelihood estimation of stochastic regular syntax-directed translation schemes," in *Grammatical Inference: Learning Syntax from Sentences*, ser. Lecture Notes in Computer Science, L. Miclet and C. de la Higuera, Eds., vol. 1147. Springer-Verlag, 1996, pp. 282–291.

[45] D. Picó and F. Casacuberta, "A statistical-estimation method for stochastic finite-state transducers based on entropy measures," in *Advances in Pattern Recognition*, ser. LNCS. Springer-Verlag, 2000, vol. 1876, pp. 417–426.

[46] E. M. Gold, "Language identification in the limit," *Information and Control*, vol. 10, no. 5, pp. 447–474, 1967.

[47] ——, "Complexity of automaton identification from given data," *Information and Control*, vol. 37, pp. 302–320, 1978.

[48] L. G. Valiant, "A theory of the learnable," *Communications of the Association for Computing Machinery*, vol. 27, no. 11, pp. 1134–1142, 1984.

[49] L. Pitt and M. Warmuth, "The minimum consistent DFA problem cannot be approximated within any polynomial," *Journal of the Association for Computing Machinery*, vol. 40, no. 1, pp. 95–142, 1993.

[50] F. Denis, C. d'Halluin, and R. Gilleron, "PAC learning with simple examples," in *13th Symposium on Theoretical Aspects of Computer Science, STACS'96*, ser. LNCS, 1996, pp. 231–242.

[51] F. Denis and R. Gilleron, "PAC learning under helpful distributions," in *Algorithmic Learning Theory, ALT'97*, 1997.

[52] R. Parekh and V. Honavar, "Learning DFA from simple examples," in *Workshop on Automata Induction, Grammatical Inference, and Language Acquisition, ICML-97*, 1997.

[53] J. J. Horning, "A procedure for grammatical inference," *Information Processing*, vol. 71, pp. 519–523, 1972.

[54] D. Angluin, "Identifying languages from stochastic examples," Yale University, Tech. Rep. YALEU/DCS/RR-614, March 1988.

[55] S. Kapur and G. Bilardi, "Language learning from stochastic input," in *Proceedings of the fifth conference on Computational Learning Theory*, Pittsburgh, July 1992, pp. 303–310.

[56] N. Abe and M. Warmuth, "On the computational complexity of approximating distributions by probabilistic automata," in *Proceedings of the Third Workshop on Computational Learning Theory*. Morgan Kaufmann, 1998, pp. 52–66.

[57] R. Carrasco and J. Oncina, "Learning deterministic regular grammars from stochastic samples in polynomial time," *RAIRO (Theoretical Informatics and Applications)*, vol. 33, no. 1, pp. 1–20, 1999.

[58] A. Clark and F. Thollard, "Pac-learnability of probabilistic deterministic finite state automata," *Journal of Machine Learning Research*, vol. 5, pp. 473–497, May 2004.

[59] C. de la Higuera and F. Thollard, "Identification in the limit with probability one of stochastic deterministic finite automata," ser. Lecture Notes in Computer Science, A. de Oliveira, Ed., vol. 1891. Berlin, Heidelberg: Springer-Verlag, 2000, pp. 15–24.

[60] R. Carrasco and J. Oncina, "Learning stochastic regular grammars by means of a state merging method," ser. Lecture Notes in Computer Science, R. C. Carrasco and J. Oncina, Eds., no. 862. Berlin, Heidelberg: Springer Verlag, 1994, pp. 139–150.

[61] F. Thollard, P. Dupont, and C. de la Higuera, "Probabilistic dfa inference using Kullback-Leibler divergence and minimality," in *Proc. 17th International Conf. on Machine Learning*. Morgan Kaufmann, San Francisco, CA, 2000, pp. 975–982.

[62] F. Thollard and A. Clark, "Shallow parsing using probabilistic grammatical inference," in *Int. Coll. on Grammatical Inference*, M. v. Z. P. Adriaans, H. Fernau, Ed., vol. 2484, ICGI. Amsterdam: Springer, September 2002, pp. 269–282.

[63] C. Kermorvant and P. Dupont, "Stochastic grammatical inference with multinomial tests," ser. Lecture Notes in Computer Science, P. Adriaans, H. Fernau, and M. van Zaannen, Eds., vol. 2484. Berlin, Heidelberg: Springer-Verlag, 2002, pp. 149–160.

[64] M. Young-Lai and F. W. Tompa, "Stochastic grammatical inference of text database structure," *Machine Learning*, vol. 40, no. 2, pp. 111–137, 2000.

[65] P. García, E. Vidal, and F. Casacuberta, "Local languages, the succesor method, and a step towards a general methodology for the inference of regular grammars," *IEEE Trans. on Pat. Analysis and Machine Intelligence*, vol. 9, no. 6, pp. 841–845, 1987.

[66] A. Orlitsky, N. P. Santhanam, and J. Zhang, "Always Good Turing: Asymptotically optimal probability estimation," in *44th Annual IEEE Symposium on Foundations of Computer Science (FOCS'03)*, October 11 - 14 2003, p. 179.

[67] S. Katz, "Estimation of probabilities from sparse data for the language model component of a speech recognizer," *IEEE Transactions on Acoustic, Speech and Signal Processing*, vol. 35, no. 3, pp. 400–401, 1987.

[68] R. Kneser and H. Ney, "Improved backing-off for m–gram language modeling," in *IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, vol. I, Detroit MI, 1995, pp. 181–184.

[69] S. F. Chen and J. Goodman, "An empirical study of smoothing techniques for language modeling," in *Proceedings of the Thirty-Fourth Annual Meeting of the Association for Computational Linguistics*, A. Joshi and M. Palmer, Eds., Association for Computational Linguistics. San Francisco: Morgan Kaufmann Publishers, 1996, pp. 310–318.

[70] F. Thollard, "Improving probabilistic grammatical inference core algorithms with post-processing techniques," in *Proc. 18th International Conf. on Machine Learning*. Morgan Kaufmann, San Francisco, CA, 2001, pp. 561–568.

[71] D. Llorens, J. M. Vilar, and F. Casacuberta, "Finite state language models smoothed using n-grams," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 16, no. 3, pp. 275–289, 2002.

[72] J. Amengual, A. Sanchis, E. Vidal, and J. Benedí, "Language simplification through error-correcting and grammatical inference techniques," *Machine Learning*, vol. 44, no. 1, pp. 143–159, 2001.

[73] P. Dupont and L. Chase, "Using symbol clustering to improve probabilistic automaton inference," ser. Lecture Notes in Computer Science, V. Honavar and G. Slutski, Eds., no. 1433. Berlin, Heidelberg: Springer-Verlag, 1998, pp. 232–243.

[74] R. Kneser and H. Ney, "Improved clustering techniques for class-based language modelling," in *European Conference on Speech Communication and Technology*, Berlin, 1993, pp. 973–976.

[75] C. Kermorvant and C. de la Higuera, "Learning languages with help," ser. Lecture Notes in Computer Science, P. Adriaans, H. Fernau, and M. van Zaannen, Eds., vol. 2484. Berlin, Heidelberg: Springer-Verlag, 2002.

[76] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

[77] S. Bangalore and G. Riccardi, "Stochastic finite-state models for spoken language machine translation," in *Proceedings of the Workshop on Embeded Machine Translation Systems, NAACL*, Seattle, USA, May 2000, pp. 52–59.

[78] ——, "A finite-state approach to machine translation," in *Proceedings of the North American ACL2001*, Pittsburgh, USA, May 2001.

[79] J. Oncina, P. García, and E. Vidal, "Learning subsequential transducers for pattern recognition interpretation tasks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 5, pp. 448–458, 1993.

[80] J. M. Vilar, "Improve the learning of subsequential transducers by using alignments and dictionaries," in *Grammatical Inference: Algorithms and Applications*, ser. Lecture Notes in Artificial Intelligence. Springer-Verlag, 2000, vol. 1891, pp. 298–312.

[81] F. Casacuberta, "Inference of finite-state transducers by using regular grammars and morphisms," in *Grammatical Inference: Algorithms and Applications (proc. of ICGI-2000)*, ser. Lecture Notes in Artificial Intelligence. Springer-Verlag, 2000, vol. 1891, pp. 1–14.

[82] F. Casacuberta, H. Ney, F. J. Och, E. Vidal, J. M. Vilar, S. Barrachina, I. García-Varea, D. Llorens, C. Martínez, S. Molau, F. Nevado, M. Pastor, D. Picó, A. Sanchis, and C. Tillmann, "Some approaches to statistical and finite-state speech-to-speech translation," *Computer Speech and Language*, 2003.

[83] F. Casacuberta and E. Vidal, "Machine translation with inferred stochastic finite-state transducers," *Computational Linguistics*, vol. 30, no. 2, pp. 205–225, 2004.

[84] M. Mohri, "Finite-state transducers in language and speech processing," *Computational Linguistics*, vol. 23, no. 3, pp. 269–311, 1997.

[85] M. Mohri, F. Pereira, and M. Riley, "Weighted finite-state transducers in speech recognition," *Computer Speech and Language*, vol. 16, no. 1, pp. 69–88, 2002.

[86] H. Alshawi, S. Bangalore, and S. Douglas, "Head transducer model for speech translation and their automatic acquisition from bilingual data," *Machine Translation*, 2000.

[87] ——, "Learning dependency translation models as collections of finite state head transducers," *Computational Linguistics*, vol. 26, 2000.

[88] F. Casacuberta and C. de la Higuera, "Computational complexity of problems on probabilistic grammars and transducers," ser. Lecture Notes in Computer Science, A. de Oliveira, Ed., vol. 1891. Berlin, Heidelberg: Springer-Verlag, 2000, pp. 15–24.

[89] F. Casacuberta, E. Vidal, and D. Picó, "Inference of finite-state transducers from regular languages," *Pattern Recognition*, p. In press, 2004.

[90] E. Mäkinen, "Inferring finite transducers," University of Tampere, Tech. Rep. A-1999-3, 1999.

[91] E. Vidal, P. García, and E. Segarra, "Inductive learning of finite-state transducers for the interpretation of unidimensional objects," in *Structural Pattern Analysis*, R. Mohr, T. Pavlidis, and A. Sanfeliu, Eds. World Scientific pub, 1989, pp. 17–35.

[92] K. Knight and Y. Al-Onaizan, "Translation with finite-state devices," in *Proceedings of the 4th. ANSTA Conference*, ser. Lecture Notes in Artificial Intelligence. Springer-Verlag, 1998, vol. 1529, pp. 421–437.

[93] J. Eisner, "Parameter estimation for probabilistic finite-state transducers," in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL 2002)*, Philadelphia, USA, July 2002.

[94] D. Llorens, "Suavizado de autómatas y traductores finitos estocásticos," Ph.D. dissertation, Universitat Politècnica de València, 2000.

[95] M.-J. Nederhoff, "Practical experiments with regular approximation of context-free languages," *Computational Linguistics*, vol. 26, no. 1, 2000.

[96] M. Mohri and M.-J. Nederhof, *Robustness in Language and Speech Technology*. Kluwer Academic Publisher, 2000, ch. Regular Approximations of Context-Free Grammars through Transformations, pp. 252–261.

[97] K. Lari and S. Young, "The estimation of stochastic context-free grammars using the inside-outside algorithm," *Computer Speech and Language*, no. 4, pp. 35–56, 1990.

[98] J. Sánchez and J. Benedí, "Consistency of stocastic context–free grammars from probabilistic estimation based on growth transformation," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 19, no. 9, pp. 1052–1055, 1997.

[99] J. Sánchez, J. Benedí, and F. Casacuberta, "Comparison between the inside-outside algorithm and the Viterbi algorithm for sotchastic context-free grammars," in *Advances in Structural and Syntactical Pattern Recognition*, ser. Lecture Notes in Computer Science, P. Perner, P. Wang, and A. Rosenfeld, Eds. 8th Int. Workshop SSPR'96, Leipzig: Springer, 1996, vol. 1121, pp. 50–59.

[100] Y. Takada, "Grammatical inference for even linear languages based on control sets," *Information Processing Letters*, vol. 28, no. 4, pp. 193–199, 1988.

[101] T. Koshiba, E. Mäkinen, and Y. Takada, "Learning deterministic even linear languages from positive examples," *Theoretical Computer Science*, vol. 185, no. 1, pp. 63–79, 1997.

[102] ——, "Inferring pure context-free languages from positive data," *Acta Cybernetica*, vol. 14, no. 3, pp. 469–477, 2000.

[103] Y. Sakakibara, "Learning context-free grammars from structural data in polynomial time," *Theoretical Computer Science*, vol. 76, pp. 223–242, 1990.

[104] F. Maryanski and M. G. Thomason, "Properties of stochastic syntax-directed translation schemata," *International Journal of Computer and information Science*, vol. 8, no. 2, pp. 89–110, 1979.

[105] A. Fred, "Computation of substring probabilities in stochastic grammars," in *Grammatical Inference: Algorithms and Applications*, ser. Lecture Notes in Computer Science, A. de Oliveira, Ed. Berlin, Heidelberg: Springer-Verlag, 2000, vol. 1891, pp. 103–114.

[106] V. Balasubramanian, "Equivalence and reduction of Hidden Markov Models," Massachusetts Institute of Technology, Tech. Rep. AITR-1370, 1993.

[107] R. C. Carrasco and J. Oncina, Eds., *Grammatical Inference and Applications, ICGI-94*, ser. Lecture Notes in Computer Science, no. 862. Berlin, Heidelberg: Springer Verlag, 1994.

[108] A. de Oliveira, Ed., *Grammatical Inference: Algorithms and Applications, ICGI '00*, ser. Lecture Notes in Computer Science, vol. 1891. Berlin, Heidelberg: Springer-Verlag, 2000.

[109] P. Adriaans, H. Fernau, and M. van Zaannen, Eds., *Grammatical Inference: Algorithms and Applications, ICGI '00*, ser. Lecture Notes in Computer Science, vol. 2484. Berlin, Heidelberg: Springer-Verlag, 2002.