Learning Probabilistic Automata with Variable Memory Length

Dana Ron Yoram Singer Naftali Tishby
Institute of Computer Science and
Center for Neural Computation
Hebrew University, Jerusalem 91904, Israel
Email: {danar,singer,tishby}@cs.huji.ac.il

Abstract

We propose and analyze a distribution learning algorithm for variable memory length Markov processes. These processes can be described by a subclass of probabilistic finite automata which we name *Probabilistic Finite Suffix Automata*. The learning algorithm is motivated by real applications in man-machine interaction such as handwriting and speech recognition. Conventionally used fixed memory Markov and hidden Markov models have either severe practical or theoretical drawbacks. Though general hardness results are known for learning distributions generated by sources with similar structure, we prove that our algorithm can indeed efficiently learn distributions generated by our more restricted sources. In Particular, we show that the KL-divergence between the distribution generated by the target source and the distribution generated by our hypothesis can be made small with high confidence in polynomial time and sample complexity. We demonstrate the applicability of our algorithm by learning the structure of natural English text and using our hypothesis for the correction of corrupted text.

1 Introduction

Statistical modeling of complex sequences is a fundamental goal of machine learning due to its wide variety of natural applications. The most noticeable examples of such applications are statistical models in human communication such as natural language and speech [7, 11], and statistical models of biological sequences such as DNA and proteins [9]. These kinds of complex sequences generally do not have any simple underlying statistical source, but they typically exhibit an exponentially decaying autocorrelation function. This observation suggests modeling such sequences by Markov and hidden Markov models. These statistical models capture a rich family of sequence distributions and moreover, they give efficient procedures both for generating sequences and for computing their probabilities.

The most powerful of these models is the Hidden Markov Model (HMM) [13], for which there exists a maximum likelihood estimation procedure which is widely used in many applications [12]. From the computational learning theory point of view however, the HMM has severe drawbacks. Abe and Warmuth [1] study the problem of *training* HMMs. The HMM training problem is the problem of approximating an arbitrary, unknown source distribution by distributions generated by HMMs. They prove that HMMs are *not* trainable in time polynomial in the alphabet size, unless $RP = NP$. Even if we allow the algorithm to run in time exponential in the alphabet size, then there are no known algorithms which run in time polynomial in the number of states of the target HMM. An additional drawback of the HMM is that its main benefit is in its Markov (finite memory) structure, but the full power of the HMM is seldom utilized at all. In this case the 'hidden' part of the HMM is not needed and the additional parameters just complicate the model unnecessarily.

Natural simpler alternatives, which are often used as well, are order L Markov chains or n-gram models. The size of full order L models is exponential in L and hence, if we want to capture correlations of substantial length in the sequences, then these model are clearly not practical.

In this paper we propose a simple adaptive model and describe its learning algorithm which is both theoretically and practically more effective. In many natural sequences, the lengths of the correlations depend on the context and are *not fixed*. The model we suggest is hence a variant of order L Markov chains, in which the order, or equivalently, the memory, is variable. We describe this model using a subclass of Probabilistic Finite Automata (PFA), which we name *Probabilistic Finite Suffix Automata* (PFSA). Each state in a PFSA is labeled by a string over an alphabet Σ. The transition function between the states is defined based on these labels, so that a walk on the underlying graph of the automaton, related to a given sequence of length at least L, always ends in a state labeled by a suffix of the sequence. The lengths of the strings labeling the states are bounded by some upper bound L, but different states may be labeled by strings of different length, and are viewed as having varying *memory length*. As mentioned above, the probability distributions these automata generate can be equivalently generated by Markov chains of order L, but the description using a PFSA may be much more succinct. Moreover, the estimation of the full order L Markov model requires data length and time exponential in L.

As our hypothesis class we choose a slightly different family of probabilistic machines named Probabilistic Suffix Trees (PST). Similar tree machines have been used before for universal data compression [14, 17, 18]. In the learning algorithm we 'grow' such a tree starting from a single root node, and adaptively add nodes (strings) for which there is strong evidence in the sample that they significantly affect the prediction properties of the tree.

Several measures of the quality of a hypothesis can be considered. Since we are mainly interested in models for statistical classification and pattern recognition, the most natural measure is the Kullback-Leibler (KL) divergence. Our results hold equally well for the variation (L_1) distance and other norms, which are upper bounded by the KL-divergence. Since the KL-divergence between Markov sources grows linearly with the length of the sequence, the appropriate measure is the KL-divergence per symbol. An ϵ-good hypothesis thus has at most ϵ KL-divergence per symbol to the target.

Our main result is the following. If both a bound L, on the memory length of the target PFSA, and a bound n, on the number of states the target has, are known, then for every given $\epsilon > 0$ and $0 < \delta < 1$, our learning algorithm outputs an ϵ-good hypothesis prediction suffix tree, with confidence $1 - \delta$, in time polynomial in L, n, $|\Sigma|$, $\frac{1}{\epsilon}$ and $\frac{1}{\delta}$. Furthermore, such a hypothesis can be obtained from a *single* sample sequence if the sequence length is also polynomial in a parameter related to the *mixing rate* (or the *correlation length*) of the target machine.

Höffgen [6] studies related families of distributions, but his algorithms depend exponentially and not polynomially on the order, or memory length, of the distributions. Freund *et. al.* [5] point out that their result for learning *typical* deterministic finite automata from random walks without membership queries, can be extended to learning *typical* PFAs. Unfortunately, there is strong evidence indicating that the problem of learning *general* PFAs is hard. Kearns *et. al.* [8] show that PFAs are not efficiently learnable under the assumption that there is no efficient algorithm for learning noisy parity functions in the PAC model.

Despite the intractability result mentioned above, our restricted model can be learned in a PAC-like sense efficiently. This has not been shown so far for any of the more popular sequence modeling algorithms. We demonstrate the true applicability of the algorithm by a small illustrative example in Appendix A.

2 Preliminaries

In this section we describe the family of distributions studied in this paper. We start with some basic notation that we use in the paper.

2.1 Basic Definitions and Notations

Let Σ be a finite alphabet. By Σ^* we denote the set of all possible strings over Σ. For any integer N, Σ^N denotes all strings of length N, and $\Sigma^{\leq N}$ denotes the set of all strings with length *at most* N. The empty string is denoted by e. For any string $s = s_1 \ldots s_l$, $s_i \in \Sigma$, we use the following notations:

- The longest prefix of s different from s is denoted by $\mathit{prefix}(s) \stackrel{\text{def}}{=} s_1 s_2 \ldots s_{l-1}$.
- The longest suffix of s different from s is denoted by $\mathit{suffix}(s) \stackrel{\text{def}}{=} s_2 \ldots s_{l-1} s_l$.
- The set of all suffixes of s is denoted by
$$\mathit{Suffix}^*(s) \stackrel{\text{def}}{=} \{s_i \ldots s_l \mid 1 \leq i \leq l\} \cup \{\mathsf{e}\} \ .$$
- Let s^1 and s^2 be two strings in Σ^*. If s^1 is a suffix of s^2 then we shall say that s^2 is a *suffix extension* of s^1.
- A set of strings S is called a *suffix free* set if
$$\forall s \in S,\ \mathit{Suffix}^*(s) \cap S = \{s\} \ .$$

2.2 Probabilistic Finite Automata and Prediction Suffix Trees

2.2.1 Probabilistic Finite Automata

A *Probabilistic Finite Automaton (PFA)* M is a 5-tuple $(Q, \Sigma, \tau, \gamma, \pi)$, where Q is a finite set of *states*, Σ is a finite *alphabet*, $\tau : Q \times \Sigma \to Q$ is the *transition function*, $\gamma : Q \times \Sigma \to [0, 1]$ is the *next symbol probability function*, and $\pi : Q \to [0, 1]$ is the *initial probability distribution* over the starting states. The functions γ and π must satisfy the following requirements: for every $q \in Q$, $\sum_{\sigma \in \Sigma} \gamma(q, \sigma) = 1$, and $\sum_{q \in Q} \pi(q) = 1$. We allow the transition function τ to be undefined only on states q and symbols σ, for which $\gamma(q, \sigma) = 0$. τ can be extended to be defined on $Q \times \Sigma^*$ as follows: $\tau(q, s_1 s_2 \ldots s_l) = \tau(\tau(q, s_1 \ldots s_{l-1}), s_l) = \tau(\tau(q, \mathit{prefix}(s)), s_l)$.

A PFA M generates strings of infinite length, but we shall always discuss probability distributions induced on prefixes of these strings which have some specified finite length. If P_M is the probability distribution M defines on infinitely long strings, then P_M^N, for any $N \geq 0$, will denote the probability induced on strings of length N. We shall sometimes drop the superscript N, assuming that it is understood from the context. The probability that M generates a string $r = r_1 r_2 \ldots r_N$ in Σ^N is

$$P_M^N(r) = \sum_{q^0 \in Q} \pi(q^0) \prod_{i=1}^{N} \gamma(q^{i-1}, r_i) \ , \qquad (1)$$

where $q^{i+1} = \tau(q^i, r_i)$.

2.2.2 Probabilistic Finite Suffix Automata

We are interested in learning a subclass of PFAs which we name *Probabilistic Finite Suffix Automata* (PFSA). These automata have the following property. Each state in a PFSA M is labeled by a string of finite length in Σ^*. The set of strings labeling the states is suffix free. We require that for every two states $q^1, q^2 \in Q$ and for every symbol $\sigma \in \Sigma$, if $\tau(q^1, \sigma) = q^2$ and q^1 is labeled by a string s^1, then q^2 is labeled by a string s^2 which is a suffix of $s^1 \cdot \sigma$. In order that τ be well defined on a given set of strings S, not only must the set be suffix free, but it must also have the following property. For every string s in S labeling some state q, and every symbol σ for which $\gamma(q, \sigma) > 0$, there exists a string which is a suffix of $s\sigma$. For our convenience, from this point on, if q is a state in Q then q will also denote the string labeling that state.

We also require that the underlying graph of M, defined by Q and $\tau(\cdot,\cdot)$, be *strongly connected*, and that for every state $q = q_1 \ldots q_l$ ($q_i \in \Sigma$) in Q, the initial probability of q, $\pi(q)$, be the stationary probability of q, $\Pi_M(q)$, satisfying

$$\Pi_M(q) = \sum_{q' \ s.t. \ \tau(q', q_l) = q} \Pi_M(q') \gamma(q', q_l) \quad . \tag{2}$$

For any given $L \geq 0$, the subclass of PFSAs in which each state is labeled by a string of length at most L is denoted by L-PFSA. An example 2-PFSA is depicted in Figure 1. A special case of these automata is the case in which Q includes *all* strings in Σ^L. These automata can be described as *Markov chains of order L*. The states of the Markov chain are the symbols of the alphabet Σ, and the next state transition probability depends on the last L states (symbols) traversed. Since every L-PFSA can be extended to a (possibly much larger) equivalent L-PFSA whose states are labeled by all strings in Σ^L, it can always be described as a Markov chain of order L. Alternatively, since the states of an L-PFSA might be labeled by only a small subset of $\Sigma^{\leq L}$, and many of the suffixes labeling the states may be much shorter than L, it can be viewed as a Markov chain with *variable order*, or *variable memory*.

Learning Markov chains of 'full' order L, i.e., L-PFSAs whose states are labeled by *all* Σ^L strings, is straightforward (though it takes time exponential in L). Since the 'identity' of the states (i.e., the strings labeling the states) is known, and since the transition function τ is uniquely defined, learning such automata reduces to approximating the next symbol probability function γ. For the more general case of L-PFSAs in which the states are labeled by strings of variable length, the task of an efficient learning algorithm is much more involved since it must reveal the identity of the states as well.

2.2.3 Prediction Suffix Trees

Though we are interested in learning PFSAs, we choose as our hypothesis class the class of *prediction suffix trees* (PST) defined in this section. We later show (Section 4) that for every PFSA there exists an equivalent PST of roughly the same size. We also show in the full paper that if a given PST has a special property then we can construct an equivalent PFSA which is not much larger. If the PST does not have that property then we can prove a slightly weaker claim. A similar version of these trees was introduced in [14] and has been used for other tasks such as compression and prediction (for examples see [17, 18]).

A PST T, over an alphabet Σ, is a tree of degree $|\Sigma|$. Each edge in the tree is labeled by a single symbol in Σ, such that from every internal node there is exactly one edge labeled by each symbol. The nodes of the tree are labeled by pairs (s, γ_s) where s is the string associated with the 'walk' starting from that node and ending in the root of the tree, and $\gamma_s : \Sigma \to [0,1]$ is the *next symbol probability function* related with s. We require that for every string s labeling a node in the tree, $\sum_{\sigma \in \Sigma} \gamma_s(\sigma) = 1$.

As in the case of PFAs, a PST T generates strings of infinite length, but we consider the probability distributions induced on finite length prefixes of these strings. The probability that T generates a string $r = r_1 r_2 \ldots r_N$ in Σ^N is

$$P_T^N(r) = \Pi_{i=1}^N \gamma_{s^{i-1}}(r_i) \quad , \tag{3}$$

where $s^0 = \mathsf{e}$, and for $1 \leq j \leq N-1$, s^j is the string labeling the deepest node reached by taking the walk corresponding to $r_i r_{i-1} \ldots r_1$ starting at the root of T. In view of this definition, the requirement that every internal node have *exactly* $|\Sigma|$ sons can be loosened, by allowing the omission of nodes labeled by substrings which are generated by the tree with probability 0. An example PST is depicted in Figure 1.

PSTs therefore generate probability distributions in a similar fashion to PFSAs. As in the case of PFSAs, symbols are generated sequentially and the probability of generating a symbol depends only on the previously generated substring of some bounded length. In both cases there is a simple procedure for determining this substring, as well as for determining the probability distribution on the next symbol conditioned on the substring. The main difference between these two models is that in the case of PFSAs the substrings considered in this procedure are directly related to states. This implies that for each substring (state) and symbol, the next state is well defined. In PSTs this property does not necessarily hold. PSTs, like PFSAs, can always be described as Markov chains of (fixed) finite order, but as in the case of PFSAs this description might be exponentially large.

We shall sometimes want to discuss only the structure of a PST and ignore its prediction property. In other words, we will be interested only in the string labels of the nodes and not in the values of $\gamma_s(\cdot)$. We refer to such trees as *suffix trees*. We now introduce two more notations. The set of leaves of a suffix tree T is denoted by $\mathcal{L}(T)$, and for a given string s labeling a node v in T, $T(s)$ denotes the subtree rooted at v.

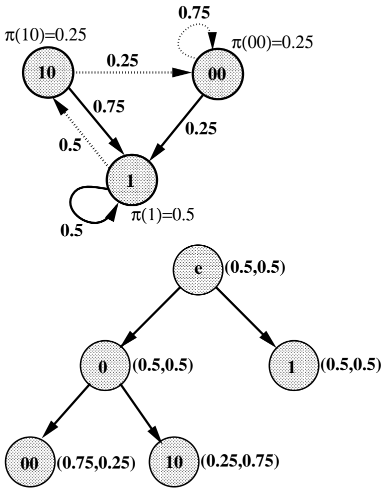

Figure 1: Top: A Probabilistic finite suffix automaton. The strings labeling the states are the suffixes corresponding to them. Bold edges denote transitions with the symbol '1', and dashed edges denote transitions with '0'. The transition probabilities are depicted on the edges. Bottom: A prediction suffix tree. The prediction probabilities of the symbols '0' and '1', respectively, are depicted beside the nodes, in parentheses.

3 The Learning Model

The learning model described in this paper is similar in spirit to that introduced in [8] in the sense that our approach is also motivated by the PAC model for learning boolean concepts from labeled examples. The two models differ slightly in their assumptions on the input to the learning algorithm and in the requirements from its output. We start by defining an ϵ-*good* hypothesis PST with respect to a given PFSA.

DEFINITION 3.1 *Let M be a PFSA and let T be a PST. Let P_M and P_T be the two probability distributions they generate respectively. We say that T is an* ϵ-**good** *hypothesis with respect to M, if for* **every** $N > 0$,

$$\frac{1}{N} D_{KL}[P_M^N || P_T^N] = \frac{1}{N} \sum_{r \in \Sigma^N} P_M^N(r) \log \frac{P_M^N(r)}{P_T^N(r)} \leq \epsilon \quad .$$

In this definition we chose the *Kullback-Leibler* (KL) divergence as a distance measure between distributions.[1] We would like to point out that the KL-divergence between distributions, generated by finite order markov chains, is proportional to the length of the strings over which the divergence is computed, when this length is longer than the order of the model. Hence, to obtain a measure independent of that length it is necessary to divide the KL-divergence by the length of the strings, N.

A learning algorithm for PFSAs is given the maximum length L of the strings labeling the states of the target PFSA M, and an upper bound n on the number of states in M. The second assumption can be easily removed by *searching* for an upper bound. This search is performed by testing the hypotheses the algorithm outputs when it runs with growing values of n. The algorithm is also given a confidence (security) parameter $0 < \delta < 1$ and an approximation parameter $0 < \epsilon < 1$. We analyze the following two learning scenarios. In the first scenario the algorithm has access to a source of sample strings of minimal length $L+1$, independently generated by M. In the second scenario it is given only a *single* (long) sample string generated by M. In both cases we require that it output a hypothesis PST $\hat{T}$, which with probability at least $1 - \delta$ is an ϵ-good hypothesis with respect to M.

The only drawback to having a PST as our hypothesis instead of a PFSA, is that the prediction procedure using a tree is somewhat less efficient (by at most a factor of L). Since no transition function is defined, in order to predict/generate each symbol, we must walk from the root until a leaf is reached. As mentioned earlier, we show in the full paper that every PST can be transformed into an 'almost equivalent' PFSA which may differ from it only in its prediction of the first L symbols. Hence, $\hat{T}$ can be transformed into a corresponding hypothesis PFSA $\hat{M}$, and we can use $\hat{T}$ to predict/generate the first L symbols and use $\hat{M}$ for the rest.

In order to measure the efficiency of the algorithm, we separate the case in which the algorithm is given a sample consisting of independently generated sample strings, from the case in which it is given a single sample string. In the first case we say that the learning algorithm is *efficient* if it runs in time polynomial in L, n, $|\Sigma|$, $\frac{1}{\epsilon}$ and $\frac{1}{\delta}$. In order to define efficiency in the latter case we need to take into account an additional property of the model – its mixing or convergence rate. To do this we should discuss another parameter of PFSAs.

For a given PFSA, M, let R_M denote the $n \times n$ stochastic transition matrix defined by $\tau(\cdot,\cdot)$ and $\gamma(\cdot,\cdot)$ when ignoring the transition labels. That is, if s^i and s^j are states in M and the last symbol in s^j is σ, then $R_M(s^i, s^j)$ is $\gamma(s^i, \sigma)$ if $\tau(s^i, \sigma) = s^j$, and 0 otherwise. Hence R_M is the transition matrix of a Markov chain. Since the underlying graph of M is strongly connected we know that R_M is irreducible. For simplicity we also assume that R_M is aperiodic and hence ergodic. [2] The

[1] Similar definitions can be considered for other distance measures such as the variation and the quadratic distances. Note that the KL-divergence bounds the variation distance as follows [3]: $D_{KL}[P_1||P_2] \geq \frac{1}{2}||P_1 - P_2||_1^2$. Since the L_1 norm bounds the L_2 norm, the last bound holds for the quadratic distance as well.

[2] R_M is *ergodic* if there exists a distribution $\pi' = \{\pi'_{s^i}\} > 0$ over the states, such that for every pair of states s^i and s^j, $\lim_{t \to \infty} R_M^t(s^i, s^j) = \pi'_j$. R_M is *irreducible* if for every s^i and s^j, there is an integer value t for which $R_M^t(s^i, s^j) > 0$. R_M is *aperiodic* if $\gcd\{t | R_M^t(s^i, s^j) > 0\} = 1$ for all pairs s^i and s^j. If R_M is both irreducible and aperiodic, then it is ergodic.

behavior of periodic chains can be analyzed similarly based on the aperiodic case.

Let $\tilde{R}_M$ denote the *time reversal* of R_M. That is,

$$\tilde{R}_M(s^i, s^j) = \frac{\Pi_M(s^j) R_M(s^j, s^i)}{\Pi_M(s^i)} \quad ,$$

where Π_M is the stationary probability vector of R_M as defined in Equation (2). Define the *multiplicative reversiblization* U_M of M by $U_M = R_M \tilde{R}_M$. Denote the second largest eigenvalue of U_M by $\lambda_2(U_M)$.

If the learning algorithm receives a single sample string, we allow the length of the string (and hence the running time of the algorithm) to be polynomial not only in L, n, $|\Sigma|$, $\frac{1}{\epsilon}$, and $\frac{1}{\delta}$, but also in $1/(1-\lambda_2(U_M))$. The rationale behind this is roughly the following. In order to succeed in learning a given PFSA, we must observe each state whose stationary probability is non-negligible enough times so that we can 'recognize' it, and so that we can compute (approximately) the next symbol probability function. When given several independently generated sample strings, we can easily bound the size of the sample needed by a polynomial in L, n, $|\Sigma|$, $\frac{1}{\epsilon}$, and $\frac{1}{\delta}$, using Chernoff bounds. When given one sample string, the given string must be long enough so as to ensure convergence of the probability of visiting a state to the stationary probability. This convergence rate can be bounded using expansion properties of a weighted graph related to U_M [10] or more generally,

using algebraic properties of U_M, namely, its second largest eigenvalue [4].

4 Emulation of PFSAs by PSTs

In this section we show that for every PFSA there exists an equivalent PST which is not much larger. This allows us to consider the PST equivalent to our target PFSA, whenever it is convenient. In the full paper we show that the opposite direction is true if the given PST has a special property and that a slightly weaker claim holds if it does not have this property.

Theorem 1 *For every L-PFSA $M = (Q, \Sigma, \tau, \gamma, \pi)$, there exists an equivalent PST T_M, with maximum depth L and at most $L \cdot |Q|$ nodes.*

Proof: (Sketch) We describe below the construction needed to prove the claim. The complete proof is provided in Appendix B.

Let T_M be the tree whose leaves correspond to the strings in Q. For each leaf s, and for every symbol σ, let $\gamma_s(\sigma) = \gamma(s, \sigma)$. This ensures that for every given string s which is a suffix extension of a leaf in T_M, and for every symbol σ, $P_M(\sigma|s) = P_{T_M}(\sigma|s)$. It remains to define the next symbol probability functions for the internal nodes of T_M. These functions must be defined so that T_M generates all strings related to its nodes with the same probability as M.

For each node s in the tree, let the *weight* of s, denoted by w_s, be $w_s \stackrel{\text{def}}{=} \sum_{s' \in Q,\, s \in \mathit{Suffix}^*(s')} \pi(s')$. In other words, the weight of a leaf in T_M is the stationary probability of the corresponding state in M; and the weight of an internal node labeled by a string s, equals the sum of the stationary probabilities over all states of which s is a suffix (which also equals the sum of the weights of the leaves in the subtree rooted at the node). Using the weights of the nodes we assign values to the γ_s's of the internal nodes s in the tree in the following manner. For every symbol σ let $\gamma_s(\sigma) = \sum_{s' \in Q,\, s \in \mathit{Suffix}^*(s')} \frac{w_{s'}}{w_s} \gamma(s', \sigma)$. The probability $\gamma_s(\sigma)$, of generating a symbol σ following a string s, *shorter* than any state in M, is thus a weighted average of $\gamma(s', \sigma)$ taken over all states s' which correspond to suffix extensions of s. The weight related with each state in this average, corresponds to its stationary probability. As an example, the probability distribution over the first symbol generated by T_M, is $\sum_{s \in Q} \pi(s)\gamma(s, \cdot)$. This probability distribution is equivalent, by definition, to the probability distribution over the first symbol generated by M.

Finally, if for some internal node in T_M, its next symbol probability function is equivalent to the next symbol probability functions of *all* of its descendants, then we remove all its descendents from the tree. □

An example of the construction described in the proof of Theorem 1 is illustrated in Figure 1. The PST on the bottom part was constructed based on the PFSA on the top, and is equivalent to it. Note that the next symbol probabilities related with the leaves and the internal nodes of the tree are as defined in the proof of the theorem.

5 The Learning Algorithm

We start with an overview of the algorithm. Let $M = (Q, \Sigma, \tau, \gamma, \pi)$ be the target L-PFSA we would like to learn, and let $|Q| \leq n$. According to Theorem 1, there exists a PST T, of size bounded by $L \cdot |Q|$, which is equivalent to M. We use the sample statistics to define the *empirical probability function*, $\tilde{P}(\cdot)$, and using $\tilde{P}$, we construct a suffix tree, $\bar{T}$, which with high probability is a subtree of T. We define our hypothesis automaton, $\hat{T}$, based on $\bar{T}$ and $\tilde{P}$,

The construction of $\bar{T}$ is done as follows. We start with a tree consisting of a single node (labeled by the empty string e) and add nodes which we have reason to believe should be in the tree. A node v labeled by a string s is added as a leaf to $\bar{T}$ if the following holds. The empirical probability of s, $\tilde{P}(s)$, is non-negligible, and for some symbol σ, the empirical probability of observing σ following s, namely $\tilde{P}(\sigma|s)$, differs substantially from the empirical probability of observing σ following *suffix*(s), namely $\tilde{P}(\sigma|\mathit{suffix}(s))$. Note that *suffix*$(s)$ is the string labeling the parent node of v. Our decision rule for adding v, is thus dependent on the ratio between

$\tilde{P}(\sigma|s)$ and $\tilde{P}(\sigma|\mathit{suffix}(s))$. We add a given node only when this ratio is substantially *greater* than 1. This suffices for our analysis (due to properties of the KL-divergence), and we need not add a node if the ratio is smaller than 1.

Thus, we would like to grow the tree level by level, adding the sons of a given leaf in the tree, only if they exhibit such a behavior in the sample, and stop growing the tree when the above is not true for any leaf. The problem is that the requirement that a node differ from its parent node is a necessary condition for belonging to the tree, but is not sufficient. The leaves of a PST must differ from their parents (or they are redundant) but internal nodes might not have this property. The PST depicted in Figure 1 illustrates this phenomena. In this example, $\gamma_0(\cdot) \equiv \gamma_{\mathsf{e}}(\cdot)$, but both $\gamma_{00}(\cdot)$ and $\gamma_{10}(\cdot)$ differ from $\gamma_0(\cdot)$. Therefore, we must continue testing further potential descendants of the leaves in the tree up to depth L.

As mentioned before, we do not test strings which belong to branches whose empirical count in the sample is small. This way we avoid exponential grow-up in the number of strings tested. The set of strings tested at each step, denoted by $\bar{S}$, can be viewed as a kind of potential 'frontier' of the growing tree $\bar{T}$, which is of bounded width. After the construction of $\bar{T}$ is completed, we define $\hat{T}$ by adding nodes so that all internal nodes have full degree, and defining the next symbol probability function for each node based on $\tilde{P}$. These probability functions are defined so that for every string s in the tree and for every symbol σ, $\gamma_s(\sigma)$ is bounded from below by γ_{min} which is a parameter that we set in the analysis of the algorithm. This is done by using a conventional 'smoothing' technique. Such a lower bound is needed in order to bound the KL-divergence between the target distribution and the distribution our hypothesis generates.

Let P denote the probability distribution generated by M. We now formally define the *empirical probability function* $\tilde{P}$, based on a given sample generated by M. For a given string s, $\tilde{P}(s)$ is roughly the relative number of times s appears in the sample, and for any symbol σ, $\tilde{P}(\sigma|s)$ is roughly the relative number of times σ appears after s. We give a more precise definition below.

If the sample consists of one sample string α of length m, then for any string s of length at most L, define $\chi_j(s)$ to be 1 if $\alpha_{j-|s|+1}\ldots\alpha_j = s$ and 0 otherwise. Let

$$\tilde{P}(s) = \frac{1}{m-L-1}\sum_{j=L}^{m-1}\chi_j(s) \quad ,$$

and for any symbol σ, let

$$\tilde{P}(\sigma|s) = \frac{\sum_{j=L}^{m-1}\chi_{j+1}(s\sigma)}{\sum_{j=L}^{m-1}\chi_j(s)} \quad .$$

If the sample consists of m' sample strings $\alpha^1,\ldots,\alpha^{m'}$, each of length $l \geq L+1$, then for any string s of length at most L, define $\chi_j^i(s)$ to be 1 if $\alpha^i_{j-|s|+1}\ldots\alpha^i_j = s$, and 0 otherwise. Let

$$\tilde{P}(s) = \frac{1}{m'(l-L-1)}\sum_{i=1}^{m'}\sum_{j=L}^{l-1}\chi_j^i(s) \quad ,$$

and for any symbol σ, let

$$\tilde{P}(\sigma|s) = \frac{\sum_{i=1}^{m'}\sum_{j=L}^{l-1}\chi_{j+1}(s\sigma)}{\sum_{i=1}^{m'}\sum_{j=L}^{l-1}\chi_j(s)} \quad .$$

For simplicity we assumed that all the sample strings have the same length. The case in which the sample strings are of different lengths can be treated similarly.

In the course of the algorithm and in its analysis we refer to several parameters ($\epsilon_0, \epsilon_1, \epsilon_2, \epsilon_3$, and γ_{min}). They are all simple functions of ϵ, δ, n, L and $|\Sigma|$. Their values are set in the analysis of the algorithm.

Algorithm Learn-PFSA

First Phase

1. Initialize $\bar{T}$ and $\bar{S}$: let $\bar{T}$ consist of a single root node (corresponding to e), and let $\bar{S} \leftarrow \{\sigma \mid \sigma \in \Sigma \text{ and } \tilde{P}(\sigma) \geq (1-\epsilon_1)\epsilon_0\}$.
2. While $\bar{S} \neq \emptyset$, do the following:

   Pick any $s \in \bar{S}$ and do:
   (a) Remove s from $\bar{S}$;
   (b) If there exists a symbol $\sigma \in \Sigma$ such that
   $$\tilde{P}(\sigma|s) \geq (1+\epsilon_2)\gamma_{min} \quad ,$$
   and
   $$\tilde{P}(\sigma|s)/\tilde{P}(\sigma|\mathit{suffix}(s)) > 1 + 3\epsilon_2$$
   then add to $\bar{T}$ the node corresponding to s and all the nodes on the path from the deepest node in $\bar{T}$ that is a suffix of s, to s;
   (c) If $|s| < L$ then for every $\sigma' \in \Sigma$, if
   $$\tilde{P}(\sigma' \cdot s) \geq (1-\epsilon_1)\epsilon_0 \quad ,$$
   then add $\sigma' \cdot s$ to $\bar{S}$.

Second Phase

1. Initialize $\hat{T}$ to be $\bar{T}$.
2. Extend $\hat{T}$ by adding all missing sons of internal nodes.
3. For each s labeling a node in $\hat{T}$, let
   $$\hat{\gamma}_s(\sigma) = \tilde{P}(\sigma|s')(1-|\Sigma|\gamma_{min}) + \gamma_{min} \quad ,$$
   where s' is the longest suffix of s in $\bar{T}$.

6 Analysis of the Learning Algorithm

In this section we state and prove our main theorem regarding the correctness and efficiency of the learning algorithm *Learn-PFSA*, described in Section 5.

Theorem 2 *For every target PFSA M, and for every given security parameter $0 < \delta < 1$, and approximation parameter $\epsilon > 0$, Algorithm* Learn-PFSA *outputs a hypothesis PST, $\hat{T}$, such that with probability at least $1-\delta$:*

1. *$\hat{T}$ is an ϵ-good hypothesis with respect to M.*
2. *The number of nodes in $\hat{T}$ is at most $|\Sigma| \cdot L$ times the number of states in M.*

If the algorithm has access to a source of independently generated sample strings, then its running time is polynomial in $L, n, |\Sigma|, \frac{1}{\epsilon}$ and $\frac{1}{\delta}$. If the algorithm has access to only one sample string, then its running time is polynomial in the same parameters and in $1/(1-\lambda_2(U_A))$.

In order to prove the theorem above we first show that with probability $1-\delta$, a large enough sample generated according to M is *typical* to M, where typical is defined subsequently. We then assume that our algorithm in fact receives a typical sample and prove Theorem 2 based on this assumption. Roughly speaking, a sample is typical if for every substring generated with non-negligible probability by M, the empirical counts of this substring and of the next symbol given this substring, are not far from the corresponding probabilities defined by M.

Definition 6.1 *A sample generated according to M is* **typical** *if for every string $s \in \Sigma^{\leq L}$ the following two properties hold:*

1. *If $s \in Q$ then $|\tilde{P}(s) - \pi(s)| \leq \epsilon_1\epsilon_0$;*
2. *If $\tilde{P}(s) \geq (1-\epsilon_1)\epsilon_0$ then for every $\sigma \in \Sigma$,*
$$|\tilde{P}(\sigma|s) - P(\sigma|s)| \leq \epsilon_2\gamma_{min} \quad .$$

Lemma 6.1

1. *There exists a polynomial M in L, n, $|\Sigma|$, $\frac{1}{\delta}$, ϵ_0, ϵ_1, ϵ_2 and γ_{min}, such that the probability that a sample of $m \geq M(L, n, |\Sigma|, \frac{1}{\delta}, \epsilon_0, \epsilon_1, \epsilon_2)$ strings each of length at least $L+1$ generated according to M is typical is at least $1-\delta$.*
2. *There exists a polynomial M' in L, n, $|\Sigma|$, $\frac{1}{\delta}$, ϵ_0, ϵ_1, ϵ_2, γ_{min}, and $\frac{1}{1-\lambda_2(U_A)}$, such that the probability that a single sample string of length $m \geq M'(L, n, |\Sigma|, \frac{1}{\delta}, \epsilon_0, \epsilon_1, \epsilon_2, \frac{1}{1-\lambda_2(U_A)})$ generated according to M is typical is at least $1-\delta$.*

The proof of Lemma 6.1 is provided in Appendix B.

Let T be the PST equivalent to the target PFSA M, as defined in Theorem 1. In the next lemma we prove two claims. In the first claim we show that the prediction properties of our hypothesis PST $\hat{T}$, and of T, are similar. We use this in the proof of the first claim in Theorem 2, when showing that the KL-divergence per symbol between $\hat{T}$ and M is small. In the second claim we give a bound on the size of $\hat{T}$ in terms of T, which implies a similar relation between $\hat{T}$ and M (second claim in Theorem 2).

Lemma 6.2 *If Learn-PFSA is given a typical sample then*

1. *For every string s in T, if $P(s) \geq \epsilon_0$ then*
$$\frac{\gamma_s(\sigma)}{\hat{\gamma}_{s'}(\sigma)} \leq 1 + \epsilon_3 \quad ,$$
where s' is the longest suffix of s corresponding to a node in $\hat{T}$.
2. *$|\hat{T}| \leq (|\Sigma| - 1) \cdot |T|$.*

Proof: (Sketch, the complete proofs of both claims are provided in Appendix B.)

In order to prove the first claim, we argue that if the sample is typical, then there cannot exist such strings s and s' which falsify the claim. We prove this by assuming that there exists such a pair, and reaching contradiction. If we choose our parameters (ϵ_2 and γ_{min}) correctly as functions of ϵ_3, then for such a pair, s and s', the ratio between $\gamma_s(\sigma)$ and $\gamma_{s'}(\sigma)$ must be bounded from below by $1 + \epsilon_3/2$. If $s = s'$, then we have already reached a contradiction. If $s \neq s'$, then we can show that the algorithm must add some longer suffix of s to $\bar{T}$, contradicting the assumption that s' is the longest suffix of $\hat{T}$.

In order to bound the size of $\hat{T}$, we show that $\bar{T}$ is a subtree of T. This suffices to prove the second claim, since when transforming $\bar{T}$ into $\hat{T}$, we add at most all $|\Sigma| - 1$ 'brothers' of every node in $\bar{T}$. We prove that $\bar{T}$ is a subtree of T, by arguing that in its construction, we did not add any string which does not correspond to a node in T. This follows from the decision rule according to which we add nodes to $\bar{T}$. □

Proof of Theorem 2: According to Lemma 6.1, with probability at least $1-\delta$ our algorithm receives a typical sample. Thus according to the second claim in Lemma 6.2, $|\hat{T}| \leq (|\Sigma| - 1) \cdot |T|$ and since $|T| \leq L \cdot |Q|$, then $|\hat{T}| \leq |\Sigma| \cdot L \cdot |Q|$ and the second claim in the theorem is valid.

Let $r = r_1 r_2 \ldots r_N$, where $r_i \in \Sigma$, and for any prefix $r^{(i)}$ of r, where $r^{(i)} = r_1 \ldots r_i$, let $s[r^{(i)}]$ and $\hat{s}[r^{(i)}]$ denote the strings corresponding to the deepest nodes reached upon taking the walk $r_i \ldots r_1$ on T and $\hat{T}$ respectively. In particular, $s[r^{(0)}] = \hat{s}[r^{(0)}] = \mathsf{e}$. Let $\hat{P}$ denote the probability distribution generated by $\hat{T}$. Then

$$\frac{1}{N}\sum_{r\in\Sigma^N} P(r)\log\frac{P(r)}{\hat{P}(r)} = \tag{4}$$

$$= \frac{1}{N}\sum_{r\in\Sigma^N} P(r)\cdot\log\frac{\prod_{i=1}^{N}\gamma_{s[r^{(i-1)}]}(r_i)}{\prod_{i=1}^{N}\hat{\gamma}_{\hat{s}[r^{(i-1)}]}(r_i)} \tag{5}$$

$$= \frac{1}{N}\sum_{r\in\Sigma^N} P(r)\cdot\sum_{i=1}^{N}\log\frac{\gamma_{s[r^{(i-1)}]}(r_i)}{\hat{\gamma}_{\hat{s}[r^{(i-1)}]}(r_i)} \tag{6}$$

$$\begin{aligned} = \frac{1}{N}\sum_{i=1}^{N}[& \sum_{\substack{r\in\Sigma^N \text{ s.t.}\\ P(s[r^{(i-1)}])<\epsilon_0}} P(r)\cdot\log\frac{\gamma_{s[r^{(i-1)}]}(r_i)}{\hat{\gamma}_{\hat{s}[r^{(i-1)}]}(r_i)} \qquad (7)\\ & + \sum_{\substack{r\in\Sigma^N \text{ s.t.}\\ P(s[r^{(i-1)}])\geq\epsilon_0}} P(r)\cdot\log\frac{\gamma_{s[r^{(i-1)}]}(r_i)}{\hat{\gamma}_{\hat{s}[r^{(i-1)}]}(r_i)} \ ] \ . \end{aligned}$$

Using Lemma 6.2, we can bound the last expression above by

$$\frac{1}{N}\cdot N[n\epsilon_0\log\frac{1}{\gamma_{min}} + \log(1+\epsilon_3)] \ . \tag{8}$$

If $\epsilon_0 \leq \epsilon/(2n\log(1/\gamma_{min}))$ and $\epsilon_3 \leq \epsilon/2$, then the KL-divergence per symbol between $\hat{P}$ and P is bounded by ϵ as required.

For each substring in the sample which has length at most L, we consider it at most a constant number of times. Each symbol in the sample is the last symbol of at most L such substrings. Thus, our algorithm can easily be implemented so that its running time is bounded by an order of L times the size of the sample. According to Lemma 6.1, and the bounds computed for all the relevant parameters, this bound is as required in the theorem statement. ■

Acknowledgments

We would like to thank Ronitt Rubinfeld and Yoav Freund for their helpful comments. This research has been supported in part by a grant from the Israeli Ministry of Science and Arts and by the Bruno Goldberg endowment fund. Dana Ron would like to thank the support of the Eshkol fellowship. Yoram Singer would like to thank the Clore foundation for its support.

References

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

[2] Lewis Carroll. *Alice's adventures in wonderland.* The Millennium Fulcrum edition 2.9.

[3] T. Cover and J. Thomas. *Elements of Information Theory.* Wiley, 1991.

[4] J.A. Fill. Eigenvalue bounds on convergence to stationary for nonreversible Markov chains, with an application to exclusion process. *Annals of Applied Probability*, 1:62–87, 1991.

[5] Y. Freund, M. Kearns, D. Ron, R. Rubinfeld, R.E. Schapire, and L. Sellie. Efficient learning of typical finite automata from random walks. In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, pages 315–324, 1993.

[6] K.-U. Höffgen. Learning and robust learning of product distributions. In *Proceedings of the Sixth Annual Workshop on Computational Learning Theory*, pages 97–106, 1993.

[7] F. Jelinek. Self-organized language modeling for speech recognition. Technical report, IBM T.J. Watson Research Center, 1985.

[8] M. Kearns, Y.Mansour, D. Ron, R. Rubinfeld, R.E. Schapire, and L. Sellie. On the learnability of discrete distributions. In *The 25th Annual ACM Symposium on Theory of Computing (to appear)*, 1994.

[9] A. Krogh, S.I. Mian, and D. Haussler. A hidden markov model that finds genes in E. coli DNA. Technical Report UCSC-CRL-93-16, University of California at Santa-Cruz, 1993.

[10] M. Mihail. Conductance and convergence of Markov chains - A combinatorial treatment of expanders. In *Proceedings 30th Annual Conference on Foundations of Computer Science*, 1989.

[11] A. Nadas. Estimation of probabilities in the language model of the IBM speech recognition system. *IEEE Trans. on ASSP*, 32(4):859–861, 1984.

[12] L.R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 1989.

[13] L.R. Rabiner and B. H. Juang. An introduction to hidden markov models. *IEEE ASSP Magazine*, 3(1):4–16, January 1986.

[14] J. Rissanen. A universal data compression system. *IEEE Transactions on Information Theory*, 29(5):656–664, 1983.

[15] D. Ron, Y. Singer, and N. Tishby. The power of amnesia. In *Advances in Neural Information Processing Systems*, volume 6. Morgan Kaufmann, 1993.

[16] H. Schütze and Y. Singer. Part-of-Speech tagging using a variable memory Markov model. In *Proceedings of ACL 32'nd*, 1994.

[17] M.J. Weinberger, A. Lempel, and J. Ziv. A sequential algorithm for the universal coding of finite-memory sources. *IEEE Transactions on Information Theory*, 38:1002–1014, May 1982.

[18] M.J. Weinberger, J. Rissanen, and M. Feder. A universal finite memory source. Submitted for publication.

A Applications

A slightly modified version of our learning algorithm was applied and tested on various problems such as: cleaning corrupted text, predicting DNA bases [15], and part-of-speech disambiguation resolving [16]. We are still exploring the applicative possibilities of the algorithm. Here we demonstrate how to clear corrupted text. We applied the algorithm to natural english text [2], in which the alphabet consists of the english letters and the blank character.

We took out the first chapter and it served as a test set. The output of the algorithm is an automaton having less than 500 states. The automaton constitutes of states that are labeled by strings of length 1, such as `'z'`, and on the other hand, states that are labeled by strings of length 5, such as `'in th'` and `'there'`. This indicates that the algorithm really captures the notion of variable memory length prediction which resulted in a compact yet accurate model. Building a full Markov model in this case is not practical since it requires $|\Sigma|^L = 27^5 = 14348907$ states. The test text was corrupted by a uniform noise which altered each character (including blanks) with probability 0.2. The most probable state sequence was found via dynamic programming. The 'cleaned' sequence is the most probable outcome given the knowledge of the error rate. An example of such a correction procedure is shown in Figure 2.

Original Text:
alice opened the door and found that it led into a small passage not much larger than a rat hole she knelt down and looked along the passage into the loveliest garden you ever saw how she longed to get out of that dark hall and wander about among those beds of bright flowers and those cool fountains but she could not even get her head through the doorway and even if my head would go through thought poor alice it would be of very little use without my shoulders

Corrupted Text:
alice opsneg fhe daor and fpund erat id led into umsnkll passabe not mxch lcrger rhjn fvrac holeeshesknelt down and looked alotg tve passagh into thc ltvbliest gardemxthuriverhsfw how snn longey towget out of that ark hall and wgnderaaboux amoig ghosewbeds of bridht faowers nnd xhhsefcoolrfeuntains but shh cozld not fjen gktnherqaevx whrougx kte dootwayzatd evzo if my heaf wouwd uo throqgh tzought poor alice it wjcwd bq of vlry litkle ust withaut my shoulberu

Corrected Text:
alice opened the door and found that it led into his all passage not much larger then forat hole she knelt down and looked along the passigh into the stabliest garded thuriver she how and longey to get out of that dark hall and wonder about along those beds of bright frowers and those cool feentains but she could not feen got her neve through the doo way and ever if my head would to through thought poor alice it would be of very little use without my shoulders

Figure 2: Cleaning corrupted text.

B Proofs of Technical Lemmas and Theorems

Proof of Theorem 1:

let T_M be the tree whose leaves correspond to the strings in Q (the states of M). For each leaf s, and for every symbol σ, let $\gamma_s(\sigma) = \gamma(s, \sigma)$. This ensures that for every string which is a suffix extension of some leaf in T_M, both M and T_M generate the next symbol with the same probability. The remainder of this proof is hence dedicated to defining the next symbol probability functions for the internal nodes of T_M. These functions must be defined so that T_M generates all strings related to nodes in T_M, with the same probability as M.

For each node s in the tree, let the *weight* of s, denoted by w_s, be defined as follows

$$w_s \stackrel{\text{def}}{=} \sum_{s' \in Q \text{ s.t. } s \in \mathit{Suffix}^*(s')} \pi(s') \ .$$

In other words, the weight of a leaf in T_M is the stationary probability of the corresponding state in M; and the weight of an internal node labeled by a string s, equals the sum of the stationary probabilities over all states of which s is a suffix. Note that the weight of any internal node is the sum of the weights of all the leaves in its subtree, and in particular $w_{\mathsf{e}} = 1$. Using the weights of the nodes we assign values to the γ_s's of the internal nodes s in the tree in the following manner. For every symbol σ let

$$\gamma_s(\sigma) = \sum_{s' \in Q \text{ s.t. } s \in \mathit{Suffix}^*(s')} \frac{w_{s'}}{w_s} \gamma(s', \sigma) \ .$$

According to the definition of the weights of the nodes, it is clear that for every node s, $\gamma_s(\cdot)$ is in fact a probability function on the next output symbol as required in the definition of prediction suffix trees.

What is the probability that M generates a string s which is a node in T_M (a suffix of a state in Q)? By definition of the transition function of M, for every $s^0 \in Q$, if $s' = \tau(s^0, s)$, then s' must be a suffix extension of s. Thus $P_M(s)$ is the sum over all such s' of the probability of reaching s', when s^0 is chosen according to the initial distribution $\pi(\cdot)$ on the starting states. But if the initial distribution is stationary then at any point the probability of being at state s' is just $\pi(s')$, and

$$P_M(s) = \sum_{s' \in Q \text{ s.t. } s \in \mathit{Suffix}^*(s')} \pi(s') = w_s \ .$$

We next prove that $P_{T_M}(s)$ equals w_s as well. We do this by showing that for every $s = s_1 \ldots s_l$ in the tree, where $|s| \geq 1$, $w_s = w_{prefix(s)}\gamma_{prefix(s)}(s_l)$. Since $w_e = 1$, it follows from a simple inductive argument that $P_{T_M}(s) = w_s$.

By our definition of PFSAs, $\pi(\cdot)$ is such that for every $s \in Q$, $s = s_1 \ldots s_l$,

$$\pi(s) = \sum_{s' \text{ s.t. } \tau(s', s_l) = s} \pi(s')\gamma(s', s_l) \quad . \tag{9}$$

Hence, if s is a leaf in T_M then

$$\begin{aligned} w_s = \pi(s) &\overset{(a)}{=} \sum_{s' \in \mathcal{L}(T_M) \text{ s.t. } s \in Suffix^*(s's_l)} w_{s'}\gamma_{s'}(s_l) \\ &\overset{(b)}{=} \sum_{s' \in \mathcal{L}(T_M(prefix(s))} w_{s'}\gamma_{s'}(s_l) \\ &\overset{(c)}{=} w_{prefix(s)}\gamma_{prefix(s)}(s_l) \ , \end{aligned}$$

where (a) follows by substituting $w_{s'}$ for $\pi(s')$ and $\gamma_{s'}(s_l)$ for $\gamma(s', s_l)$ in Equation (9), and by the definition of $\tau(\cdot,\cdot)$; (b) follows from our definition of the structure of prediction suffix trees; and (c) follows from our definition of the weights of internal nodes. Hence, if s is a leaf, $w_s = w_{prefix(s)}\gamma_{prefix(s)}(s_l)$ as required.

If s is an internal node then using the result above we get that

$$\begin{aligned} w_s &= \sum_{s' \in \mathcal{L}(T_M(s))} w_{s'} \\ &= \sum_{s' \in \mathcal{L}(T_M(s))} w_{prefix(s')}\gamma_{prefix(s')}(s_l) \\ &= w_{prefix(s)}\gamma_{prefix(s)}(s_l) \ . \end{aligned}$$

It is left to show that the resulting tree is not bigger than L times the number of states in M. The number of leaves in T_M equals the number of states in M, i.e. $|\mathcal{L}(T)| = |Q|$. If every internal node in T_M is of full degree (i.e. the probability T_M generates any string labeling a leaf in the tree is strictly greater than 0) then the number of internal nodes is bounded by $|Q|$ and the total number of nodes is at most $2|Q|$. In particular, the above is true when for every state s in M, and every symbol σ, $\gamma(s, \sigma) > 0$. If this is not the case, then the total number of nodes is bounded by $L \cdot |Q|$. ■

Proof of Lemma 6.1:

Several sample strings: We start with obtaining a lower bound for m so that the first property of a typical sample holds. Since the sample strings are generated independently, we may view $\tilde{P}(s)$, for a given state s, as the average value of m independent random variables. Each of these variables is in the range $[0, 1]$ and its expected value is $\pi(s)$. Using a variant of Hoeffding's Inequality we get that if $m \geq \frac{2}{\epsilon_1^2\epsilon_0^2}\ln\frac{4n}{\delta}$, then with probability at least $1 - \frac{\delta}{2n}$, $|\tilde{P}(s) - \pi(s)| \leq \epsilon_1\epsilon_0$. The probability that this inequality holds for every state is hence at least $1 - \frac{\delta}{2}$.

We would like to point out that since our only assumptions on the sample strings are that they are generated independently, and that their length is at least $L+1$, we use only the independence between the different strings when bounding our error. We do not assume anything about the random variables related to $\tilde{P}(s)$ when restricted to any one sample string, other than that their expected value is $\pi(s)$. If the strings are known to be longer, then a more careful analysis might be applied as described subsequently in the case of a single sample string.

We now show that for an appropriate m the second property holds with probability at least $1 - \frac{\delta}{2}$ as well. Let s be a string in $\Sigma^{\leq L}$. In the following lines, when we refer to *appearances* of s in the sample we mean in the sense defined by $\tilde{P}$. That is, we ignore all appearances such that the last symbol of s is before the Lth symbol in a sample string, or such that the last symbol of s is the last symbol in a sample string. For the ith appearance of s in the sample and for every symbol σ, let $X_i(\sigma|s)$ be a random variable which is 1 if σ appears after the ith appearance of s and 0 otherwise. If s is either a state or a suffix extension of a state, then for every σ, the random variables $\{X_i(\sigma|s)\}$ are independent random variables in the range $[0, 1]$, with expected value $P(\sigma|s)$. Let m_s be the total number of times s appears in the sample, and let $m_{min} = \frac{8}{\epsilon_2^2\gamma_{min}^2}\ln\frac{4|\Sigma|n}{\epsilon_0\delta}$. If $m_s \geq m_{min}$, then with probability at least $1 - \frac{\delta\epsilon_0}{2n}$, for every symbol σ, $|\tilde{P}(\sigma|s) - P(\sigma|s)| \leq \frac{1}{2}\epsilon_2\gamma_{min}$. If s is a suffix of several states $s^1, \ldots, s^k$, then for every symbol σ,

$$P(\sigma|s) \ = \ \sum_{i=1}^{k} \frac{\pi(s^i)}{P(s)} P(\sigma|s^i) \ , \tag{10}$$

(where $P(s) = \sum_{i=1}^{k} \pi(s^i)$) and

$$\tilde{P}(\sigma|s) \ = \ \sum_{i=1}^{k} \frac{\tilde{P}(s^i)}{\tilde{P}(s)} \tilde{P}(\sigma|s^i) \ . \tag{11}$$

Let ϵ_1 be bounded from above by $(\epsilon_2\gamma_{min})/(8n\epsilon_0)$. If for every state s^i, $|\tilde{P}(s^i) - \pi(s^i)| \leq \epsilon_1\epsilon_0$, and for each s^i satisfying $\pi(s^i) \geq 2\epsilon_1\epsilon_0$, $|\tilde{P}(\sigma|s^i) - P(\sigma|s^i)| \leq \frac{1}{2}\epsilon_2\gamma_{min}$ for every σ, then $|\tilde{P}(\sigma|s) - P(\sigma|s)| \leq \epsilon_2\gamma_{min}$, as required.

If the sample has the first property required of a typical sample (i.e., $\forall s \in Q$, $|\tilde{P}(s) - P(s)| \leq \epsilon_1\epsilon_0$), and for every state s such that $\tilde{P}(s) \geq \epsilon_1\epsilon_0$, $m_s \geq m_{min}$, then with probability at least $1 - \frac{\delta}{4}$ the second property of a typical sample holds for all strings which are either states or suffixes of states. If for every string s which is a suffix extension a state such that $\tilde{P}(s) \geq (1 - \epsilon_1)\epsilon_0$, $m_s \geq m_{min}$, then for all such strings the second property holds with probability at least $1 - \frac{\delta}{4}$ as well. Putting together all the bounds above, if $m \geq \frac{2}{\epsilon_1^2\epsilon_0^2}\ln\frac{4n}{\delta} + m_{min}/(\epsilon_1\epsilon_0)$, then with probability at least $1 - \delta$ the sample is typical.

A single sample string: In this case the analysis is somewhat more involved. We view our sample string generated according to M as a walk on the markov chain described by R_M (defined in Subsection 3). We may assume that the starting state is visible as well. We shall need the following theorem from [4] which gives bounds on the convergence rate to the stationary distribution of general ergodic Markov chains. This theorem is partially based on a work by Mihail [10], who gives bounds on the convergence in terms of combinatorial properties of the chain.

Markov Chain Convergence Theorem [4] *For any state s_0 in the Markov chain R_M, let $R^t_M(s_0,\cdot)$ denote the probability distribution over the states in R_M, after taking a walk of length t staring from state s_0. Then*

$$\left(\sum_{s\in Q}|R^t_M(s_0,s)-\pi(s)|\right)^2 \le \frac{(\lambda_2(U_M))^t}{\pi(s_0)} .$$

We first note that for each state s such that $\pi(s) < (\delta\epsilon_1\epsilon_0)/(2n)$, then with probability at least $1-\frac{\delta}{2n}$,

$$|\tilde{P}(s)-\pi(s)| \le \epsilon_1\epsilon_0 .$$

This can easily be verified by applying Markov's Inequality. It thus remains to obtain a lower bound on m', so that the same is true for each s such that $\pi(s) \ge (\delta\epsilon_1\epsilon_0)/(2n)$. We do this by bounding the variance of the random variable related with $\tilde{P}(s)$, and applying Chebishev's Inequality.

Let

$$t_0 = 5/(1-\lambda_2(U_M))\ln\frac{n}{\delta\epsilon_0\epsilon_1} . \tag{12}$$

It is not hard to verify that for every s satisfying $\pi(s) \ge (\delta\epsilon_1\epsilon_0)/(2n)$, $|R^{t_0}_M(s,s)-\pi(s)| \le \frac{\delta}{4n}\epsilon_1^2\epsilon_0^2$. Intuitively, this means that for every two integers, $t > t_0$, and $i \le t-t_0$, the event that s is the $(i+t_0)$th state passed on a walk of length t, is 'almost independent' of the event that s is the ith state passed on the same walk.

For a given state s, satisfying $\pi(s) \ge (\delta\epsilon_1\epsilon_0)/(2n)$, let X_i be a 0/1 random variable which is 1 *iff* s is the ith state on a walk of length t, and $Y = \sum_{i=1}^t X_i$. By our definition of $\tilde{P}$, in the case of a single sample string, $\tilde{P}(s) = Y/t$, where $t = m-L-1$. Clearly $E(Y/t) = \pi(s)$, and for every i, $Var(X_i) = \pi(s)-\pi^2(s)$. We next bound $Var(Y/t)$.

$$\begin{aligned}
Var\left(\frac{Y}{t}\right) &= \frac{1}{t^2}Var\left(\sum_{i=1}^t X_i\right)\\
&= \frac{1}{t^2}[\sum_{i,j}E(X_iX_j) - \sum_{i,j}E(X_i)E(X_j)]\\
&= \frac{1}{t^2}[\sum_{i,j \text{ s.t. } |i-j|<t_0}(E(X_iX_j)-E(X_i)E(X_j))\\
&\quad + \sum_{i,j \text{ s.t. } |i-j|\ge t_0}(E(X_iX_j)-E(X_i)E(X_j))]\\
&\le \frac{2t_0}{t}(\pi(s)-\pi^2(s)) + \frac{\delta}{4n}\epsilon_1^2\epsilon_0^2\pi(s) . \qquad (13)
\end{aligned}$$

If we pick t to be greater than $(4nt_0)/(\delta\epsilon_1^2\epsilon_0^2)$, then $Var(Y/t) < \frac{\delta}{2n}\epsilon_1^2\epsilon_0^2$, and using Chebishev's Inequality $Pr[|Y/t-\pi(s)| > \epsilon_1\epsilon_0] < \frac{\delta}{2n}$. The probability the above holds for any s is at most $\frac{\delta}{2}$. The analysis of the second property required of a typical sample is identical to that described in the case of a sample consisting of many strings. ■

Proof of Lemma 6.2:

1st Claim: Assume contrary to the claim that there exists a string labeling a node s in T such that $P(s) \ge \epsilon_0$ and for some $\sigma \in \Sigma$

$$\frac{\gamma_s(\sigma)}{\hat{\gamma}_{s'}(\sigma)} > 1+\epsilon_3, \tag{14}$$

where s' is the longest suffix of s in $\hat{T}$. For simplicity of the presentation, let us assume that there is a node labeled by s' in $\bar{T}$. If this is not the case (*suffix*(s') is an internal node in $\bar{T}$, whose son s' is missing), the analysis is very similar. If $s \equiv s'$ then we easily show below that our counter assumption is false. If s' is a proper suffix of s then we prove the following. If the counter assumption is true, then we added to $\bar{T}$ a (not necessarily proper) suffix of s which is longer than s'. This contradicts the fact that s' is the longest suffix of s in $\hat{T}$.

We first achieve a lower bound on the ratio between the two 'true' next symbol probabilities, $\gamma_s(\sigma)$ and $\gamma_{s'}(\sigma)$. According to our definition of $\hat{\gamma}_{s'}(\cdot)$,

$$\hat{\gamma}_{s'}(\sigma) \ge (1-|\Sigma|\gamma_{min})\tilde{P}(\sigma|s') .$$

We analyze separately the case in which $\gamma_{s'}(\sigma) \ge \gamma_{min}$, and the case in which $\gamma_{s'}(\sigma) < \gamma_{min}$. If $\gamma_{s'}(\sigma) \ge \gamma_{min}$, and we choose γ_{min} to be at most $\epsilon_2/|\Sigma|$, then

$$\begin{aligned}
\frac{\gamma_s(\sigma)}{\gamma_{s'}(\sigma)} &\ge \frac{\gamma_s(\sigma)}{\tilde{P}(\sigma|s')}\cdot(1-\epsilon_2) & (15)\\
&\ge \frac{\gamma_s(\sigma)}{\hat{\gamma}_{s'}(\sigma)}\cdot(1-\epsilon_2)(1-|\Sigma|\gamma_{min}) & (16)\\
&> (1+\epsilon_3)(1-\epsilon_2)^2 , & (17)
\end{aligned}$$

where Inequality (15) follows from our assumption that the sample is typical, Inequality (16) follows from our definition of $\hat{\gamma}_{s'}(\sigma)$, and Inequality (17) follows from our choice of γ_{min}. If $\epsilon_2 \le \epsilon_3/6$, and $\epsilon_3 < 1/2$ then we get that

$$\frac{\gamma_s(\sigma)}{\gamma_{s'}(\sigma)} > 1+\frac{\epsilon_3}{2} . \tag{18}$$

If $\gamma_{s'}(\sigma) < \gamma_{min}$, then since $\hat{\gamma}_{s'}(\sigma)$ is defined to be at least γ_{min} then

$$\frac{\gamma_s(\sigma)}{\gamma_{s'}(\sigma)} > 1+\epsilon_3 > 1+\frac{\epsilon_3}{2} \tag{19}$$

as well. If $s \equiv s'$ then the counter assumption (14) is evidently false, and we must only address the case in which $s \neq s'$, i.e., s' is a proper suffix of s.

Let $s = s_1 s_2 \ldots s_l$, and let s' be $s_i \ldots s_l$, for some $2 \leq i \leq l$. We now show that if the counter assumption (14) is true, then there exists an index $1 \leq j < i$ such that $s_j \ldots s_l$ was added to $\bar{T}$. Let $2 \leq r \leq i$ be the first index for which

$$\gamma_{s_r \ldots s_l}(\sigma) < (1 + 7\epsilon_2)\gamma_{min} \quad .$$

If there is no such index then let $r = i$. The reason we need to deal with the prior case is made more clear subsequently. In either case, if $\epsilon_2 \leq \epsilon_3/21$, and $\epsilon_3 < 1/2$, then

$$\frac{\gamma_s(\sigma)}{\gamma_{s_r \ldots s_l}(\sigma)} \; > \; 1 + \frac{\epsilon_3}{2}. \tag{20}$$

In other words

$$\frac{\gamma_s(\sigma)}{\gamma_{s_2 \ldots s_l}(\sigma)} \cdot \frac{\gamma_{s_1 \ldots s_l}(\sigma)}{\gamma_{s_3 \ldots s_l}(\sigma)} \cdot \ldots \cdot \frac{\gamma_{s_{r-1} \ldots s_l}(\sigma)}{\gamma_{s_r \ldots s_l}(\sigma)} \; > \; 1 + \frac{\epsilon_3}{2}.$$

This last inequality implies that there must exist an index $1 \leq j \leq i-1$, for which

$$\frac{\gamma_{s_j \ldots s_l}(\sigma)}{\gamma_{s_{j+1} \ldots s_l}(\sigma)} \; > \; 1 + \frac{\epsilon_3}{4L} \quad . \tag{21}$$

We next show that Inequality (21) implies that $s_j \ldots s_l$ was added to $\bar{T}$. We do this by showing that $s_j \ldots s_l$ was added to $\bar{S}$, that we compared $\tilde{P}(\sigma | s_j \ldots s_l)$ to $\tilde{P}(\sigma | s_{j+1} \ldots s_l)$, and that the ratio between these two values is at least $(1 + 3\epsilon_2)$. Since $P(s) \geq \epsilon_0$ then necessarily

$$\tilde{P}(s_j \ldots s_l) \geq (1 - \epsilon_1)\epsilon_0 \tag{22}$$

and $s_j \ldots s_l$ must have been added to $\bar{S}$. Based on our choice of the index r, and since $j < r$,

$$\gamma_{s_j \ldots s_l}(\sigma) \geq (1 + 7\epsilon_2)\gamma_{min}. \tag{23}$$

Since we assume that the sample is typical,

$$\tilde{P}(\sigma | s_j \ldots s_l) \geq (1 + 6\epsilon_2)\gamma_{min} > (1 + \epsilon_2)\gamma_{min} \quad , \tag{24}$$

which means that we must have compared $\tilde{P}(\sigma | s_j \ldots s_l)$ to $\tilde{P}(\sigma | s_{j+1} \ldots s_l)$.

We now separate the case in which $\gamma_{s_{j+1} \ldots s_l}(\sigma) < \gamma_{min}$, from the case in which $\gamma_{s_{j+1} \ldots s_l}(\sigma) \geq \gamma_{min}$. If $\gamma_{s_{j+1} \ldots s_l}(\sigma) < \gamma_{min}$ then

$$\tilde{P}(\sigma | s_{j+1} \ldots s_l) \leq (1 + \epsilon_2)\gamma_{min} \quad . \tag{25}$$

Therefore,

$$\frac{\tilde{P}(\sigma | s_j \ldots s_l)}{\tilde{P}(\sigma | s_{j+1} \ldots s_l)} \geq \frac{(1 + 6\epsilon_2)\gamma_{min}}{(1 + \epsilon_2)\gamma_{min}} \geq (1 + 3\epsilon_2) \quad , \tag{26}$$

and $s_j \ldots s_l$ would have been added to $\bar{T}$. On the other hand, if $\gamma_{s_{j+1} \ldots s_l}(\sigma) \geq \gamma_{min}$, and $\epsilon_2 \leq \frac{\epsilon_3}{24L}$, then the same would hold since

$$\begin{aligned}
\frac{\tilde{P}(\sigma | s_j \ldots s_l)}{\tilde{P}(\sigma | s_{j+1} \ldots s_l)} \; &\geq \; \frac{(1 - \epsilon_2)\gamma_{s_j \ldots s_l}(\sigma)}{(1 + \epsilon_2)\gamma_{s_{j+1} \ldots s_l}(\sigma)} \\
&> \; \frac{(1 - \epsilon_2)(1 + \frac{\epsilon_3}{4L})}{(1 + \epsilon_2)} \\
&\geq \; \frac{(1 - \epsilon_2)(1 + 6\epsilon_2)}{(1 - \epsilon_2)} \\
&> \; 1 + 3\epsilon_2 \quad .
\end{aligned} \tag{27}$$

This contradicts our initial assumption that s' is the longest suffix of s added to $\bar{T}$ in the first phase.

<u>*2nd Claim:*</u> We prove below that $\bar{T}$ is a subtree of T. The claim then follows directly, since when transforming $\bar{T}$ into $\hat{T}$, we add at most all $|\Sigma| - 1$ 'brothers' of every node in $\bar{T}$. Therefore it suffices to show that in the first phase we did not add to $\bar{T}$ any node which is not in T. Assume to the contrary that we add to $\bar{T}$ a node s which is not in T. According to the algorithm, the reason we add s to $\bar{T}$, is that there exists a symbol σ such that

$$\tilde{P}(\sigma | s) \; \geq \; (1 + \epsilon_2)\gamma_{min},$$

and

$$\tilde{P}(\sigma | s)/\tilde{P}(\sigma | \mathit{suffix}(s)) \; > \; 1 + 3\epsilon_2,$$

while both $\tilde{P}(s)$ and $\tilde{P}(\mathit{suffix}(s))$ are greater than $(1 - \epsilon_1)\epsilon_0$. If the sample string is typical then

$$P(\sigma | s) \geq \gamma_{min} \; ,$$

$$\tilde{P}(\sigma | s) \; \leq \; P(\sigma | s) + \epsilon_2\gamma_{min} \; \leq \; (1 + \epsilon_2)P(\sigma | s) \quad ,$$

and

$$\tilde{P}(\sigma | \mathit{suffix}(s)) \; \geq \; P(\sigma | \mathit{suffix}(s)) - \epsilon_2\gamma_{min} \quad .$$

If $P(\sigma | \mathit{suffix}(s)) \geq \gamma_{min}$ then

$$\tilde{P}(\sigma | \mathit{suffix}(s)) \; \geq \; (1 - \epsilon_2)P(\sigma | \mathit{suffix}(s)) \quad ,$$

and thus

$$\frac{P(\sigma | s)}{P(\sigma | \mathit{suffix}(s))} \; \geq \; \frac{(1 - \epsilon_2)}{(1 + \epsilon_2)}(1 + 3\epsilon_2) \quad ,$$

which is greater than 1 for $\epsilon_2 < 1/3$. If

$$P(\sigma | \mathit{suffix}(s)) < \gamma_{min}$$

then

$$P(\sigma | s)/P(\sigma | \mathit{suffix}(s)) > 1$$

as well, since $P(\sigma | s) \geq \gamma_{min}$. In both cases this ratio cannot be greater than 1 if s is not in the tree, contradicting our assumption. ∎