

Locally Threshold Testable Languages in Strict Sense: Application to the Inference Problem

José Ruiz, Salvador España, Pedro García

Depto. de Sistemas Informáticos y Computación.
Universidad Politécnica de Valencia. Valencia (Spain).

jruiz@dsic.upv.es
sespana@iti.upv.es
pgarcia@dsic.upv.es

Abstract. The aim of this paper is to define a new family of regular languages, the *Locally Threshold Testable Languages in Strict Sense (LTTSS)*. This family includes the well known family of locally testable languages in strict sense (*LTSS*) and is included in the family of locally threshold testable languages (*LTT*). Membership of a word to a *LTTSS* language can be decided by means of local scanning, using a sliding window of a fixed length k , although in this case, we have to care that the number of occurrences of certain segments of length $\leq k$ in the words is not greater than a level of restriction, less than a threshold r . As *LTTSS* languages may be of interest in disciplines such as Pattern Recognition and specially in Speech Recognition, an inference algorithm that identifies the family of (k, r) -*TTSS* languages from positive data in the limit is proposed. Finally we also report some results aiming to reflect the evolution of the behavior of this algorithm for different values of k and r when it is used in a handwritten digits recognition task.

1 Introduction

One of the families of languages that has received more attention in the literature is the family of *Locally Testable Languages (LT)*. Membership of a word to a *LT* language is determined by the set of factors of a fixed length k and by the prefixes and suffixes of length less than k of the word. The number of occurrences of the factors in the word or the order in which they appear are not relevant. Local languages, known either by their ability to generate the family of regular languages by means of morphisms [12] or by Chomsky-Schützenberger's theorem for Context Free Languages, are included in the family of *LT*. In fact they constitute a particular instance of the so called *Locally Testable Languages in Strict Sense (LTSS)*.

The words of a *LTSS* language L , for a given value of $k \geq 1$, are defined by means of three finite sets: A set A of prefixes of length $< k$, a set B of suffixes of length $< k$ and a set C of segments of length k not allowed to appear in its strings. Membership of a word to a *LTSS* language, for a given value of k (k -*TSS*) can be decided exploring the word through a sliding window of length

k and testing if its prefix belongs to A , its suffix belongs to B and also that it does not contain any segment of length k in C .

An extension of the family of LT languages is the family of locally threshold testable languages (LTT). LTT languages are defined in a similar way as LT languages are, with the difference that in this case, frequency of factors of length $\leq k$ are count up to a threshold $r \geq 1$ (LT are a particular instance of LTT , the case $r = 1$). If a word x belongs to a LTT language L for given values of k and r , any word y will also belong to L iff it meets the three following conditions:

1. Begins and ends with the same segments of length $k - 1$ that x .
2. The frequency of each segment of length $\leq k$ in y is the same as in x , if this number is less than r .
3. If the frequency of a factor of length $\leq k$ in x is $\geq r$, then the frequency of that factor in y is also $\geq r$.

It seems natural, in this context, to define the family of *Locally Threshold Testable Languages in Strict Sense* ($LTTSS$), which, on one hand, is a restriction the family of LTT and, on the other, is an extension of $LTSS$ languages. We define $LTTSS$ languages by means of the sets of prefixes and suffixes of length $< k$ and by a set of *restricted segments* of length $\leq k$. Each segment in the set of restricted segments is associated with a level of restriction, which is less than a fixed threshold r . The segments for which this level is zero are forbidden. The language defined this way contains the words that begin and end with elements of the indicated sets and such that none of the restricted segments appears in them a number of times beyond its level of restriction. For each value of k , if we set r to 1 we obtain the family of k -Testable Languages in Strict Sense (k - TSS). The family of $LTTSS$ languages has been algebraically characterized by means of a property of the idempotents of their syntactic semigroup (see [6] for further details).

Besides its theoretical interest, another reason to study $LTTSS$ languages is the possibility of being used in Pattern Recognition (PR) [5], [9]. Stochastic $LTSS$ languages, also known as N -grams, are frequently used in PR , particularly in Speech Recognition, both in Acoustic-Phonetics Decoding as in Language Modeling [14]. We show in this work that, like k - TSS languages [7], (k,r) - $TTSS$ languages are learnable from positive data, and we present some results obtained by using learned (k,r) - $TTSS$ languages for handwritten digits classification task. The main goal of the experiments is to show the evolution of the classification rates when k and r vary.

2 Preliminaries and Notation

2.1 Formal Languages

We assume that the reader is familiar with the rudiments of formal language theory. Any concept not mentioned here can be found in [10].

Let Σ be a finite alphabet and Σ^* the free monoid generated by Σ with concatenation as the binary operation and λ as neutral element. A language L over Σ is a subset of Σ^* , its elements will be referred as *words*. The set of all words of length k will be represented as Σ^k , also $\Sigma_1^k = \bigcup_{i=1}^k \Sigma^i$. Given $x \in \Sigma^*$, if $x = uvw$ with $u, v, w \in \Sigma^*$, then u (resp. w) is called *prefix* (resp. *suffix*) of x , while v (also u and w) is said to be a *factor* or *segment* of x . $\text{Pr}(L)$ (resp. $\text{Suf}(L)$) denotes the set of prefixes (resp. suffixes) of L . The *length* of a word x is represented by $|x|$. The frequency of appearance of a word w as a segment of x (its number of occurrences in x) is written as $|x|_w$. Given $u \in \Sigma^*$ and $L \subseteq \Sigma^*$, $u^{-1}L = \{v \in \Sigma^* : uv \in L\}$. Also, if $R \subseteq \Sigma^*$ is a finite set, $|R|$ denotes its cardinal.

A Deterministic Finite-state Automaton (*DFA*) is a 5-tuple $A = (Q, \Sigma, \delta, q_0, F)$ where Q is a finite set of states, Σ is a finite alphabet, $q_0 \in Q$ is the initial state, $F \subseteq Q$ is the set of final states and δ is a partial function, mapping $Q \times \Sigma$ to Q that can be extended to words defining $\delta(q, \lambda) = q$ and $\delta(q, xa) = \delta(\delta(q, x), a)$, $\forall q \in Q, \forall x \in \Sigma^*, \forall a \in \Sigma$. A word x is accepted by A if $\delta(q_0, x) \in F$. The set of words accepted by an automaton A is denoted by $L(A)$.

Given $R \subseteq \Sigma^*$, $PTA(R)$ denotes the prefix tree acceptor for R , that is $PTA(R) = (Q, \Sigma, \delta, q_0, F)$, being $Q = \text{Pr}(R)$, $q_0 = \lambda$, $F = R$ and $\forall u, ua \in \text{Pr}(R)$, $\delta(u, a) = ua$. Given $A_0 = PTA(R)$, if \sim is an equivalence relation defined in Q , A_0/\sim denotes the quotient automaton (obtained by merging equivalent states).

2.2 Language inference from positive presentation

A *positive sample* of a language L is any finite subset of L . A *positive presentation* of L is any enumeration of L . Let \mathcal{L} a family of languages over Σ , a *class of representations* \mathcal{H} for the languages in \mathcal{L} must accomplish that for any $L \in \mathcal{L}$ there exists at least $h \in \mathcal{H}$ such that $L(h) = L$. For example, if \mathcal{L} is a family of regular languages, \mathcal{H} can be taken as the set of *DFA* over Σ . An *inference algorithm* for \mathcal{L} from positive samples and representations in \mathcal{H} is an algorithm A that, with input of a finite set $R \subseteq \Sigma^*$, outputs a hypothesis $h \in \mathcal{H}$. If $L \in \mathcal{L}$ and $\langle x_1, x_2, \dots \rangle$ is a positive presentation of L , we denote as t_k the finite sequence $\langle x_1, x_2, \dots, x_k \rangle$. A identifies the family \mathcal{L} with respect to \mathcal{H} from positive samples in the limit if for any $L \in \mathcal{L}$ and any positive presentation $\langle x_1, x_2, \dots \rangle$ of L there exists a natural number n_0 such that for $n \geq n_0$, $h_n = h_{n_0}$ and $L(h_n) = L$, where $h_n = A(t_n)$. A family \mathcal{L} is identifiable with respect to \mathcal{H} in the limit if there exists an algorithm A that identifies \mathcal{L} [8], [2].

Let R a positive sample of L and A a *DFA* such that $L(A) = L$. We say that R is *structurally complete* for A iff every transition of A is used in the acceptance by A of some of the words of R .

A well known technique in regular language inference is the one known as *state clustering* [4], [2]. It is basically carried out, given a finite set $R \subseteq \Sigma^*$, obtaining $A_0 = PTA(R)$, defining an equivalence relation \sim in $\text{Pr}(R)$, and outputting A_0/\sim .

A class \mathcal{L} is learnable in the limit from positive data with conjectures updated in polynomial time if there exists an algorithm \mathcal{A} for \mathcal{L} such that $\mathcal{A}(t_i) = M_i$ and there exists a polynomial P such that $\forall L \in \mathcal{L}$, the time used by \mathcal{A} between receiving the sample x_i and outputting the hypothesis M_i is $P(n, m_1 + \dots + m_i)$, where $m_j = |x_j|$ and n is the number of states of the minimal automaton accepting L .

3 Locally Threshold Testable Languages in Strict Sense

Given positive integers k and r . The equivalence relation $\approx_{k,r}$ over Σ^* is defined as follows:

- If $|x| < k$, then $x \approx_{k,r} y$ iff $x = y$.
- Otherwise $x \approx_{k,r} y$ iff:
 1. $\text{Pr}(x) \cap \Sigma^{k-1} = \text{Pr}(y) \cap \Sigma^{k-1}$.
 2. $\text{Suf}(x) \cap \Sigma^{k-1} = \text{Suf}(y) \cap \Sigma^{k-1}$.
 3. $\forall w \in \Sigma_1^k (|x|_w = |y|_w < r \vee (|x|_w > r \wedge |y|_w > r))$.

The relation $\approx_{k,r}$ is a congruence of finite index. A language is (k, r) -TT iff it is saturated by $\approx_{k,r}$. A language L is *Locally Threshold Testable (LTT)* if there exist integers $k, r \geq 1$ such that L is (k, r) -TT. If $r = 1$ this family coincides with the well known family of Locally Testable Languages (LT).

A subclass of the LT languages is the class of Locally Testable Languages in Strict Sense (LTSS). A language L over Σ is *LTSS* iff there exists an integer $k \geq 1$ and sets $A, B \subseteq \Sigma^{k-1}$ and $C \subseteq \Sigma^k$ such that

$$\Sigma^{k-1} \Sigma^* \cap L = A \Sigma^* \cap \Sigma^* B - \Sigma^* C \Sigma^*$$

We are going to define a new family of languages, called *Locally Threshold Testable Languages in Strict Sense (LTTSS)* that are related to LTT languages in the same way as LTSS are related to LT. The case $r = 1$ constitutes the family of LTSS languages.

Given an alphabet Σ and integers $k, r \geq 1$ we define the 4-tuple $(I_k, F_k, T_{k,r}, g)$ where

- $I_k, F_k \subseteq \Sigma^{k-1}$, which are respectively the sets of prefixes and suffixes of length $k - 1$.
- $T_{k,r} \subseteq \Sigma_1^k$, the set of restricted factors of length k .
- $g : T_{k,r} \rightarrow \{0, 1, \dots, r - 1\}$, a function that defines the level of restriction of each of the restricted factors.

We say that L is a *k-Testable Language in Strict Sense with Threshold r* (in the sequel this family will be referred as (k,r) -TTSS) iff

$$\Sigma^{k-1} \Sigma^* \cap L = I_k \Sigma^* \cap \Sigma^* F_k - \left(\bigcup_{w \in T_k} \{x \in \Sigma^* : |x|_w > g(w)\} \right)$$

L is obviously regular. Language L , except for a finite number of words of length less than k , is the set of all words that:

- begin with a prefix of length $k - 1$ in I_k ,
- end with a suffix of length $k - 1$ in F_k and,
- if they contain segments of length k belonging to $T_{k,r}$, the number of occurrences of these segments is less than or equal to the level of restriction given by the function g .

Example 1. The language $a^+ba^+b \cup a^+b$ is $(2,3)$ -*TTSS*, as it fulfils the definition with $I_2 = \{a\}$, $F_2 = \{b\}$, $T_{k,r} = (b, ab, ba, bb)$ with $g(b) = 2$, $g(ab) = 2$, $g(ba) = 1$, $g(bb) = 0$.

A language L is threshold locally testable in strict sense (*LTTSS*) if there exist two positive integers k and r such that L is (k, r) -*TTSS*.

$\forall k, r \geq 1$, the family of (k, r) -*TTSS* languages is included in the family of (k, r) -*TT*. From the definition of (k, r) -*TTSS* languages, it follows that any language belonging to that family is saturated by the relation $\approx_{k,r}$, so $LTTSS \subset LTT$. It is easily seen that this inclusion is strict from the fact that (k, r) -*TTSS* languages are not closed under boolean operations.

4 Smallest (k, r) -*TTSS* Language That Contains a Positive Sample S

Given $x \in \Sigma^*$ we define the initial and final segments of x of length $\leq k$ as follows:

$$i_k(x) = \begin{cases} x & \text{if } |x| < k \\ u \in \Sigma^k : x = uv, v \in \Sigma^* & \text{if } |x| \geq k \end{cases}$$

$$f_k(x) = \begin{cases} x & \text{if } |x| < k \\ v \in \Sigma^k : x = uv, u \in \Sigma^* & \text{if } |x| \geq k \end{cases}$$

Let S be a positive sample of L , that is, $S \subset L$ finite. $\forall w \in \Sigma^k$ we define $n_{S,k}(w) = \max_{x \in S} |x|_w$, and let

- $I_k(S) = \{i_{k-1}(x) : x \in S\}$.
- $F_k(S) = \{f_{k-1}(x) : x \in S\}$.
- $T_{k,r}(S) = \{w \in \Sigma_1^k : n_{S,k}(w) < r\}$ and the function
- $g : T_{k,r}(S) \rightarrow \{0, 1, \dots, r-1\}$ such that $g(w) = n_{S,k}(w)$.

It is evident that the language (k, r) -*TTSS* defined by $(I_k(S), F_k(S), T_{k,r}(S), g)$ is the smallest (k, r) -*TTSS* language that contains S . It will be denoted as $L_{k,r}(S)$.

4.1 Properties of $L_{k,r}(S)$.

The following properties follow directly from the definition of $L_{k,r}(S)$:

1. $S \subseteq S' \Rightarrow L_{k,r}(S) \subseteq L_{k,r}(S')$.
2. $k \leq k' \vee r \leq r' \Rightarrow L_{k',r'}(S) \subseteq L_{k,r}(S)$.
3. $r > \max_{x \in S} |x| - k + 1 \Rightarrow L_{k,r}(S) = \text{Pr}(S) \cap \Sigma^* F_k(S)$.
4. $k > \max_{x \in S} |x| - r + 1 \Rightarrow L_{k,r}(S) = S$.

4.2 Automaton accepting the smallest (k,r) -TTSS language that contains a positive sample S

The following automaton recognizes $L_{k,r}(S)$, the smallest (k,r) -TSS language that contains a positive sample S .

Given the sample S we obtain the 4-tuple $(I_k(S), F_k(S), T_{k,r}(S), g)$. In the sequel $\mathbf{0}$ will represent the $|T_{k,r}(S)|$ -dimensional null vector. Vectors \mathbf{v} have also dimension $|T_{k,r}(S)|$.

Let $A = (Q, \Sigma, \delta, q_0, F)$ with

- $Q = \{[u, \mathbf{0}] : u \in \text{Pr}(I_k(S))\} \cup \{[v, \mathbf{v}] : v \in \Sigma^{k-1}, \mathbf{v}_w \leq g(w), \forall w \in T_{k,r}(S)\}$.
- $q_0 = [\lambda, \mathbf{0}]$.
- $F = \{[v, \mathbf{v}] \in Q : v \in F_k(S)\}$.
- The transition function δ defined as follows:
 1. $\forall u, ua \in \text{Pr}(I_k(S)), |u| < k - 1, \delta([u, \mathbf{0}], a) = [ua, \mathbf{0}]$.
 2. $\forall aub \in T_{k,r}(S)$, where $a, b \in \Sigma$
 - $\delta([au, \mathbf{v}], b) = \emptyset$ if $\mathbf{v}_{aub} = g(aub)$
 - $\delta([au, \mathbf{v}], b) = [ub, \mathbf{v}']$ if $\mathbf{v}_{aub} < g(aub)$ where $\mathbf{v}'_{aub} = \mathbf{v}_{aub} + 1$ and $\mathbf{v}'_w = \mathbf{v}_w \forall w \in T_{k,r}(S) - \{aub\}$.
 3. $\forall aub \in \Sigma^k - T_{k,r}(S), \delta([au, \mathbf{v}], b) = [ub, \mathbf{v}]$.

Lemma 1. *The transition function δ defined above can be extended to the words of S , that is, $\forall x \in \text{Pr}(L_{k,r}(S)), \delta([\lambda, \mathbf{0}], x) = [f_{k-1}(x), \mathbf{v}]$, with $\mathbf{v}_w = |x|_w, \forall w \in T_{k,r}(S)$.*

Proof. We proceed by induction in $|x|$.

For $x = \lambda$ it evidently holds.

Let us suppose that the proposition is true $\forall x \in \text{Pr}(L_{k,r}(S))$ with $|x| = m$ and let $x = ya$ with $|y| = m$,

1. If $m < k - 1$,
 $\delta([\lambda, \mathbf{0}], ya) = \delta(\delta([\lambda, \mathbf{0}], y), a) = \delta([y, \mathbf{0}], a) = [ya, \mathbf{0}] = [f_{k-1}(ya), \mathbf{0}]$
2. If $m \geq k - 1$, $\delta([\lambda, \mathbf{0}], ya) = \delta(\delta([\lambda, \mathbf{0}], y), a) = \delta([f_{k-1}(y), \mathbf{v}], a)$, with $\mathbf{v}_w = |y|_w, \forall w \in T_{k,r}(S)$
 - (a) If $f_{k-1}(ya) \in \Sigma^k - T_{k,r}(S)$, it follows that $\delta([f_{k-1}(y), \mathbf{v}], a) = [f_{k-2}(y)a, \mathbf{v}] = [f_{k-1}(ya), \mathbf{v}]$ and the proposition holds, as these words don't modify the associated vector of any of the segments of length k , that is, $\forall w \in T_{k,r}(S), |ya|_w = |y|_w$.
 - (b) If $f_{k-1}(ya) \in T_{k,r}(S)$ it follows that $|y|_{f_{k-1}(y)a} < g(f_{k-1}(y)a)$, otherwise $ya \notin \text{Pr}(L_{k,r}(S))$. Then $\delta([f_{k-1}(y), \mathbf{v}], a) = [f_{k-1}(ya), \mathbf{v}']$, with

$$\begin{cases} \mathbf{v}'_{f_{k-1}(y)a} = |y|_{f_{k-1}(y)a} + 1 = |ya|_{f_{k-1}(y)a} \\ \mathbf{v}'_w = |ya|_w = |y|_w \quad \forall w \in T_{k,r}(S) : w \neq f_{k-1}(y)a \end{cases}$$

□

Proposition 1. $L_{k,r}(S) = L(A)$.

Proof. – If $x \in L_{k,r}(S)$, then $x \in \text{Pr}(L_{k,r}(S)) \cap \Sigma^* F_k(S)$, that is, $f_{k-1}(x) \in F_k(S)$ and by lemma 1 $\delta([\lambda, \mathbf{0}], x) = [f_{k-1}(x), \mathbf{v}] \in F$, therefore $x \in L(A)$.

– Let $x \in L(A)$. It can be shown by induction in $|x|$ that $\forall x \in \text{Pr}(L(A))$, $\delta([\lambda, \mathbf{0}], x) = [f_{k-1}(x), \mathbf{v}(x)]$. It follows that $[f_{k-1}(x), \mathbf{v}(x)] \in F$, and by the definition of A , this implies that $f_{k-1}(x) \in F_k(S)$. By the definition of Q , $\mathbf{v}_w(x) \leq \mathbf{v}_w(S)$. Moreover $\delta([\lambda, \mathbf{0}], i_{k-1}(x)) = [i_{k-1}(x), \mathbf{0}]$, therefore $i_{k-1}(x) \in I_k(S)$ and then $x \in L_{k,r}(S)$. □

5 Inference of (k, r) -TTSS Languages from Positive Data

The class of regular languages is not identifiable from positive data in the limit [8]. Even though, interesting subclasses of the family of regular languages that can be inferred this way [3], [7], [13]. Conditions for language identifiability can be found in [2], [11].

Given $k, r \geq 1$, the family of (k, r) -TTSS languages is identifiable from positive data in the limit. That comes from the fact that, given an alphabet Σ , the family of (k, r) -TTSS languages is finite.

Let L an unknown (k, r) -TTSS language and S a positive sample for L . An algorithm that, on input of k, r and S outputs a finite automaton accepting the smallest (k, r) -TTSS language that contains S , converge in el limit to an automaton accepting L . In this sense, section 4 provides an algorithm that identifies in the limit the family of (k, r) -TTSS languages. The main disadvantage of that algorithm is its high computational cost.

In this section we propose a new algorithm, less expensive in time, and though it needs more data to converge, it identifies the family of (k, r) -TTSS languages also. It is a states-merging algorithm and is shown in Figure 1. We will refer it as (k, r) -TTSSI algorithm.

<p>INPUT: $S \subseteq \Sigma^*, k \geq 1, r \geq 1$ OUTPUT: AFD $A_{k,r}$ compatible with S ($S \subseteq L(A_{k,r})$) METHOD: Obtain $T_{k,r}(S)$ $A_0 = (Q, \Sigma, \delta, q_0, F)$ the PTA(S) $A'_0 = (Q, \Sigma, \delta, q_0, F')$ with $F' = \{u \in \text{Pr}(S) : \exists x \in S, f_{k-1}(u) = f_{k-1}(x)\}$ Compute \sim, the equivalence in $Q = \text{Pr}(S)$, defined: $\forall u, v \in \text{Pr}(S), u \sim v \iff$ $f_{k-1}(u) = f_{k-1}(v) \wedge \forall w \in \Sigma_1^k (w \in T_k(S) \Rightarrow u _w = v _w)$ $A_{k,r} := A'_0 / \sim$</p> <p>End.</p>
--

Fig. 1. Algorithm (k, r) -TTSSI for inference of the family of (k, r) -TTSS languages.

It is easy to see that $L(A_{k,r}) \subseteq L_{k,r}(S)$. Moreover, the automaton $A_{k,r}$ is a subautomaton of the one defined in section 4. It is evident that if S is structurally complete with respect to the automaton of section 4, we have that $L(A_{k,r}) = L_{k,r}(S)$. So (k,r) -TTSSI algorithm identifies the family of (k, r) -TTSS languages in the limit.

As far as its application to Pattern Recognition tasks is concerned, the variation of parameters k and r permits a double control of the degree of generalization obtained from the sample, which may be interesting for that purpose. It is easily seen that for any fixed value of k and r ,

1. $L(A_{k+1,r}) \subseteq L(A_{k,r})$.
2. $L(A_{k,r+1}) \subseteq L(A_{k,r})$.

On the other hand, the algorithm can be implemented so that it works in an incremental way, that is, if a new input is considered, the new hypothesis can be obtained only from the current hypothesis and from this new input.

5.1 Example of run

Let $k = 2$, $r = 2$ and $S = \{aababa, abaaba\}$. Then, $I_2(S) = F_2(S) = \{a\}$, $T_{2,2}(S) = \{aa, bb\}$, with $g(aa) = 1$ and $g(bb) = 0$.

Figure 2 shows the automaton A'_0 for the sample S . Observe that A'_0 has the same set of states that $PTA(S)$, the only difference between them is the set of final states.

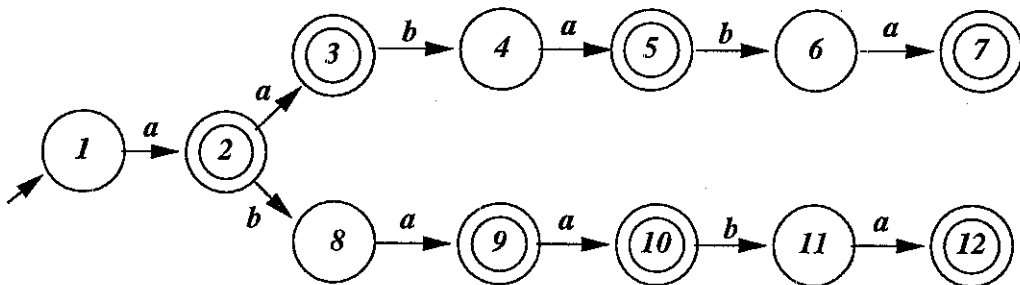


Fig. 2. Automaton A'_0 obtained from sample $S = \{aababa, abaaba\}$

Table 1 summarizes the values of $[f_{k-1}(Pr(S)), v]$ for the states of A'_0 and the values $k = 2, r = 2$ in order to obtain A'_0 / \sim . The resulting automaton is shown in Figure 3(A). Figure 3(B) shows the automaton obtained from the same input data for the values $k = 2$ and $r = 1$.

5.2 Time complexity.

Let us suppose that (k, r) -TTSSI algorithm has obtained automaton A_{i-1} from the sample $w_1 \dots w_{i-1}$ and receives $w_i = w_{i_1} w_{i_2} \dots w_{i_{n_i}}$ as input. For each symbol

Table 1. Values $[f_{k-1}(\text{Pr}(S)), v]$ of the states of A'_0

State	1	2	3	4	5	6	7	8	9	10	11	12
$f_{k-1}(\text{Pr}(S))$	λ	a	a	b	a	b	a	b	a	a	b	a
$v _{aa}$	0	0	1	1	1	1	0	0	1	1	1	1
$v _{bb}$	0	0	0	0	0	0	0	0	0	0	0	0

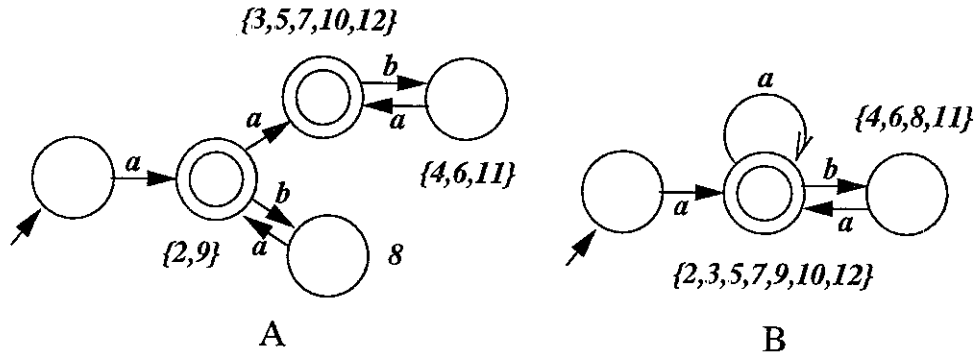


Fig. 3. Quotient automaton A'_0/\sim obtained after merging states in A'_0 : (A) for $k = 2$, $r = 2$. (B) for $k = 2$, $r = 1$. Numbers in the states correspond to the states merged in A'_0 .

w_{i_j} of w_i the algorithm has to decide if the state that corresponds to the prefix $w_{i_1}w_{i_2}\dots w_{i_j}$ has already been created and, if this is not the case, it has to construct it. Afterwards, it has to update the transitions table. With convenient data structures this operation can be attained in $k \log |\Sigma| + |\Sigma|^k \log r$ steps for each symbol in the worst case, so the total cost of obtaining automaton A_i from A_{i-1} is $(k \log |\Sigma| + |\Sigma|^k \log r)|w_i|$ steps.

6 Experiments with Handwritten Digits

We present the results of an experiment aiming to show the behavior of the algorithm in a classification task for different values of k and r . In the experiment, a database with 2400 images of handwritten digits has been used. It was composed of 240 repetitions of each digit written by 12 different writers. The images were processed and an eight-symbol chaincode was achieved following the contour of the image from the left end superior point. As the original chaincodes were long and with great amount of information, the resolution was decreased using separation grids of 6 and 10 pixels. In the sequel they will be referred as $G6$ and $G10$. Examples of this representation for several digits are shown in Figure 4.

As the amount of samples in the database is small, in order to obtain significant results, twelve different runs of each experiment for different values of k and r were carried out using the *leaving k-out* (with $k = 2$) scheme. So, in each run, the database was split into two sets: one (with 200 images of each digit) for learning and the other (with 40 images of each digit) for testing. Therefore,

0:	102122222233345656666666770
	01102212322233446465666666677
1:	3222222224666766663376760
	2222222224666676642476766
2:	1223333320000244445777767644433470770
	10233233320007024444346776777664443770

Fig. 4. Two randomly chosen digits from some of the 10 classes.

the well classification rates reported below are the average result of the twelve runs of each experiment. Each string α was labelled as belonging to the class represented by the automata A where the Levensthein distance $D_L(\alpha, L(A))$ is minimum. In case of ties it is classified in the class that achieves smaller number of substitutions. If the tie still remains, it is classified as "wrong".

Calculus of distances were done using an extension of the Viterbi algorithm that finds a minimum cost path through a general directed cyclic graph [1].

The confusion matrix for the 12 runs of the experiment with a separation grid of 6 pixels and values $k = 3$ and $r = 3$ are described in Table 2. The column labeled as "W" shows the number of ties for each digit. One should observe the special behavior of the digit "8", which has been classified as "0" 115 times, while digit "0" has been classified as "8" only 10 times. This situation is a constant in most of the experiments (either digit "0" is classified as "8" or digit "8" is classified as "0", but not both in the same experiment).

We have also observed that the number of ties of each experiment decreases as k or r increase. In case of a tie between two or more classes, one of them is the correct class in more than 98% of the cases.

Table 2. Confusion matrix of the experiment with grid 6, $k = 3$ and $r = 3$

	0	1	2	3	4	5	6	7	8	9	W	%
0	459	0	0	0	8	0	0	0	10	2	1	95.6
1	0	465	3	0	2	0	0	3	0	0	7	96.9
2	2	0	474	0	0	0	0	0	4	0	0	98.8
3	0	0	6	456	2	1	0	3	2	8	2	95.0
4	0	17	0	0	448	0	0	2	0	6	7	93.3
5	1	0	0	3	0	475	0	0	0	0	1	99.0
6	1	0	0	0	0	0	477	0	2	0	0	99.4
7	0	26	4	0	8	0	0	421	0	15	6	87.7
8	115	2	4	1	3	0	5	8	316	9	17	65.8
9	2	5	0	15	10	0	0	0	0	446	2	92.9

The well classification rates for values of k , $2 \leq k \leq 5$ and r , $1 \leq r \leq 5$ are summarized in Table 3.

Table 3. Correct classification rates of the experiment for different values of k and r .

G10		k				G6		k			
		2	3	4	5			2	3	4	5
r	1	0.67	31.00	77.12	87.90	r	1	0.31	20.27	63.15	80.83
	2	23.06	84.17	88.31	89.06		2	15.52	61.17	94.56	94.56
	3	71.92	87.50	89.23	89.94		3	26.44	92.44	95.44	95.04
	4	82.92	88.87	89.27	90.00		4	73.00	93.62	96.24	95.73
	5	86.18	88.52	89.27	90.44		5	84.06	95.11	96.41	95.81

We can observe that:

- As k , r (or both) increase, the correct classification rates become greater too, with only one exception (the case $k = 3$ and $r = 5$ in grid 10).
- The classification rates increase faster when using grid 10 (shorter chains containing less information) than with grid 6 for small values of k and r , but the later achieves better classification rates than the former for greater values of k and r .

Some partial experiments have been done using stochastic automata. The methodology for those experiments was essentially the same except that in each run, the set of 200 images used before to learn the automata was split into two sets: 120 images to learn the stochastic automata and 80 images to estimate the probability of the error rules. In this case each string was classified as belonging to the class that maximizes the a posteriori probability.

The classification rates achieved by stochastic automata are much better for the values $k = 2, 3$, but decrease with respect to non stochastic when $k = 4, 5$ and $r > 2$ (this can be explained from the fact that each stochastic automata was learned from 120 images, while non stochastic were learned from 200 images).

7 Conclusions

A new family of regular languages, the *Locally Threshold Testable Languages in Strict Sense* has been introduced. This family includes the family of locally testable languages in strict sense (for each value of k , they constitute the case $r = 1$) and is included in the family of locally threshold testable languages. An algorithm for inference of (k, r) -TTSS languages from positive data in the limit and some experiments that show the evolution of the learning process as k and r vary have also been presented.

References

1. Amengual, J.C. and Vidal E. *Two different approaches for Cost-efficient Viterbi Parsing with Error Correction*. LNCS (1121): Advances in Structural and Syntactic Pattern Recognition. Percier, Wang and Rosenfeld (Eds.) Springer Verlag pp 30-39, 1996.
2. Angluin, D. *Inductive inference of formal languages from positive data*. Information and Control, 45. pp. 117-135, 1980
3. Angluin, D. *Inference of reversible languages*. Journal of the ACM 29 (3). pp. 7741-765, 1982.
4. Biermann A.W. and Feldman, J.A. *On the synthesis of finite state machines from samples of their behavior*. IEEE Trans. on Computers, C-21:592-597, 1972.
5. Fu, K S. *Syntactic Pattern Recognition and Applications*. Prentice Hall, 1982.
6. García, P. and Ruiz, J. *Locally Threshold Testable Languages in Strict Sense*. Research Report DSIC-II/36/96. Univ Politécnic de Valencia. 1996.
7. García, P. and Vidal, E. *Inference of k -testable Languages in the Strict Sense and application to Syntactic Pattern Recognition*. IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. PAMI-12 pp.920-925, 1990.
8. Gold, E.M. *Language identification in the limit*. Information and Control, 10. pp. 447-474, 1967.
9. González R.C. and Thomason, M.G. *Syntactic Pattern Recognition: An Introduction*. Addison Wesley, 1978.
10. Hopcroft, J. Ullman, J. *Introduction to Automata theory, Languages and Computation*. Addison-Wesley. 1979.
11. Kapur, S. and Bilardi, G. *Language learning without overgeneralization*. Theoretical Computer Science 141, pp. 151-162, 1995.
12. Medvedev, Y. T. *On the Class of Events Representable in a Finite Automaton*. Sequential Machines-Selected Papers, ed E. F. Moor, Addison-Wesley, pp 227-315, 1964.
13. Ruiz, J. and García, P. *Learning k -Piecewise Testable Languages from Positive Data*. Proc. of the 3rd Intern. Conference on Grammatical Inference. LNCS 1147. Springer Verlag, pp. 203-210. 1996.
14. Vidal, E., Casacuberta, F and García, P. *Grammatical Inference and Automatic Speech Recognition*. In Speech Recognition and Coding. Springer Verlag, pp 175-191. 1995.