



ELSEVIER

Pattern Recognition Letters 23 (2002) 1–12

Pattern Recognition
Letters

www.elsevier.com/locate/patrec

Error-correcting tree language inference [☆]

Damián López ^{*}, Salvador España

Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, Camino de Vera s/n, 46071 Valencia, Spain

Received 17 November 1999; received in revised form 4 April 2001

Abstract

A new tree language inference algorithm is proposed in this work. This algorithm extends a string language inference algorithm which is based on error correction (ECGI). The algorithm proposed here uses the substructures which have already been taken into account in a tree automaton, modifying the automaton in order to force it to accept the new structures presented in the identification process. The proposed algorithm allows the use of more powerful representation primitives in pattern recognition tasks than the string primitives. It also takes advantage of the thoroughly tested ECGI features used in speech and planar shape recognition tasks. © 2002 Elsevier Science B.V. All rights reserved.

Keywords: Error-correcting analysis; Tree automata; Grammatical inference; Syntactic pattern recognition

1. Introduction

The grammatical inference problem can be considered as the process of obtaining the set of rules of an unknown grammar G from a set of samples. The samples may or may not belong to the language $L(G)$.

Among other important factors, the identification on the limit paradigm (Gold, 1967) shows that the presentation (positive or complete) of the information is key to obtaining a method which is capable of learning a single given class.

Due to the scarce availability of negative samples (samples which do not belong to the target language) in real problems, and the impossibility of inferring in the limit even the regular language

class with only a positive representation (Angluin, 1980), several approximations were developed to make up for the lack of negative samples.

One of the techniques which has been tested previously was the use of *structured samples*, that is, skeletons which represent the derivations of strings in a grammar (Sakakibara, 1990, 1992). From that point on, other algorithms were developed to infer tree languages (García, 1993; García and Oncina, 1993), thus allowing the objects to be modelled with more powerful primitives in real tasks.

This work makes use of an error-correcting algorithm introduced in (López et al., 2000) which gives a distance between a tree and a tree automaton to propose a tree language inference algorithm as an extension of an already existing string language inference algorithm, the error-correcting grammar inference algorithm (ECGI) (Rulot, 1992).

The ECGI algorithm, which was developed to be applied on speech recognition tasks, has proven to be efficient in other fields where the differential

[☆] Work supported by the Spanish CICYT under contract TIC2000-1153.

^{*} Corresponding author.

E-mail address: dlopez@dsic.upv.es (D. López).

features of the representation are based on the recognition of substructures, the length of these substructures and their relative position in the sample representation. This algorithm has been widely applied with good performance results in speech recognition and in recognition of two-dimensional shapes (Prieto et al., 1994; Vidal et al., 1993).

In this work, for each new sample t , the tree language inference algorithm proposed establishes the set of edit operations needed to force the automaton to accept t . In this way, the path with minimum cost will guide the creation of new states and transitions in the automaton.

There are two reasons why this new approach can be considered as a proper one for syntactic pattern recognition (Fu, 1982; González and Thomason, 1978). On the one hand, the representation power of tree primitives is considerably higher than string primitives. On the other hand, this algorithm takes advantage of the ECGI algorithm features that considers the substructures which are already in the language as well as, adding the new ones when they are presented in the inference process.

This work begins with the introduction of the concepts and the notation which is going to be used, followed by the description of the algorithm and the study of its complexity. It is then shown how the increased representation power of the domain objects does not imply an exponential increase in the complexity of the recognition process. A discussion explains the uniqueness of the inference result, and then the results obtained in the experimentation are discussed. Finally, the conclusions and the future lines of work are presented.

2. Notation and definitions

Given an alphabet V and the set of natural numbers \mathbb{N} , let a *ranked alphabet* be defined as the association of V with a finite relation r in $(V \times \mathbb{N})$. Let V_n denote the subset $\{\sigma \in V \mid (\sigma, n) \in r\}$. Now let V^T be the set of finite trees whose nodes are labelled with symbols in V , and let a tree be defined inductively as follows:

$$V_0 \subseteq V^T, \\ \sigma(t_1, \dots, t_n) \in V^T : \quad \forall t_1, \dots, t_n \in V^T, \sigma \in V_n.$$

Given a tree t , let the set of subtrees of t (denoted by $\text{Sub}(t)$) be defined as follows:

$$\text{Sub}(a) = \emptyset \quad \forall a \in V_0, \\ \text{Sub}(\sigma(t_1, \dots, t_n)) = \{t_1, \dots, t_n\} \cup \bigcup_{i=1, \dots, n} \text{Sub}(t_i) \\ \forall t_1, \dots, t_n \in V^T, \sigma \in V_n.$$

Let the size of a tree t , denoted by $|t|$, be defined inductively as follows:

$$|a| = 1 : \quad \forall a \in V_0, \\ |\sigma(t_1, \dots, t_n)| = 1 + \sum_{i=1, \dots, n} |t_i| : \\ \forall t_1, \dots, t_n \in V^T, \sigma \in V_n.$$

Given a tree t , let $\text{leaves}(t)$ be the set defined as follows:

$$\text{leaves}(a) = \{a\} : \quad \forall a \in V_0, \\ \text{leaves}(\sigma(t_1, \dots, t_n)) = \bigcup_{i=1, \dots, n} \text{leaves}(t_i) \\ \forall t_1, \dots, t_n \in V^T, \sigma \in V_n.$$

Let a *deterministic tree automaton* be defined as a tuple $A = (Q, V, \delta, F)$, where Q is a finite set of states; V is a ranked alphabet with $Q \cap V = \emptyset$; $F \subseteq Q$ is a set of final states and $\delta = (\delta_0, \dots, \delta_m)$ is a finite set of transitions defined as follows:

$$\delta_n : (V_n \times (Q \cup V_0)^n) \rightarrow Q, \quad n = 1, \dots, m, \\ \delta_0(a) = a \quad \forall a \in V_0.$$

δ could be extended to operate on trees as follows:

$$\delta : V^T \rightarrow Q \cup V_0, \\ \delta(\sigma(t_1, \dots, t_n)) = \delta_n(\sigma, \delta(t_1), \dots, \delta(t_n)) \quad \text{if } n > 0, \\ \delta(a) = a.$$

A given tree, $t \in V^T$, is accepted by A if $\delta(t) \in F$. The sequence of transitions applied to accept a tree is named *path*. The set of trees accepted by A is defined by $L(A) = \{t \in V^T \mid \delta(t) \in F\}$.

In this work, σ will be used to denote all the symbols with arities which are greater than or

equal to 1. Given Σ , the alphabet of symbols with arity 0, we will deal with trees in $V = \{\sigma\} \cup \Sigma$. These trees are called *skeletons*, and we will make no distinction between them and the trees in the rest of the paper.

An algorithm that obtains the minimum distance from a tree to a tree automaton is proposed in (López et al., 2000), using the extended edit operations over trees to consider the insertion, deletion and substitution of a subtree and a state of the automaton. In that work it was also proven that this distance was the minimum distance with respect to the edit operations considered.

Given $A = (Q, V, \delta, F)$, a tree automaton, and $t = \sigma(\tau_1, \tau_2, \dots, \tau_p)$, a tree, let *minimum cost path* (denoted by Δ_t , or by Δ where no confusion is possible) be a sequence of transitions that should be applied to obtain the minimum distance from t to A under the scheme proposed in (López et al., 2000). The set of edit operations with minimum cost may not be unique as shown in Fig. 1; nevertheless, this will not affect the inference algorithm as will be shown.

Given one subtree $\tau_i = \sigma(\tau_{i1}, \tau_{i2}, \dots, \tau_{ik})$ from t , let $d_{\tau_i}^t \in \Delta$ be the transition $\delta(\sigma, u_{i1}, u_{i2}, \dots, u_{ik}) = q_i$ which is applied to reduce τ_i , where

$$u_{ij} = \begin{cases} \tau_{ij} & \text{if } \tau_{ij} \in V_0, \\ q | d_{\tau_{ij}}^t \equiv \delta(\sigma, z_1, z_2, \dots, z_h) & \\ = q : q \in Q & \text{otherwise.} \end{cases} \quad (1)$$

When no confusion arises, it will be denoted by d_{τ_i} instead of $d_{\tau_i}^t$.

3. Inference algorithm

The algorithm proposed in this work is an extension of a string language inference algorithm (ECGI), which was proposed in (Rulot, 1992). It maintains the same ECGI heuristic features, and so the obtained result depends on the order in which the samples are presented to the method. Although the ECGI algorithm is a non-characterizable inference method, it has shown good

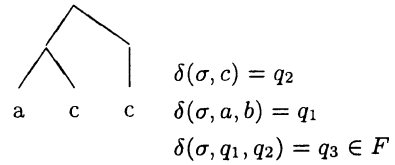


Fig. 1. The automaton accepts only the tree $\sigma(\sigma(a, b), c)$. When the tree is analyzed, there exist two ways of correcting the subtree $\sigma(a, c)$ in order to be accepted by the automaton: by using the first production of the automaton and adding a new leaf, or by using the second production of the automaton and modifying the leaf with label c . Both actions incur the same cost.

performance when applied to syntactic pattern recognition tasks (Prieto et al., 1994; Vidal et al., 1993).

Both algorithms are incremental: given an automaton, it accepts all the processed samples together with the samples that the algorithm is able to infer from them. When a new sample is processed, the algorithm applies an error-correcting analysis against the available automaton; this analysis establishes which productions have to be modified or added to the automaton in order to force it to accept the sample.

Let the sum of every edit operation cost applied in Δ be called *accepting cost* (according to the nomenclature proposed in López et al., 2000). Once this cost is established, the transitions in Δ could be divided into two disjoint subsets following the notation used in (Rulot, 1992): Δ^P which includes all the transitions that do not add error cost, and Δ^N which includes the transitions that actually do add error cost.

Once the error correcting analysis is carried out, the algorithm uses the transitions in Δ^P and adds the ones in Δ^N in such a way that the addition creates no loops in the resulting automaton. From this absence of loops, it follows that the inferred languages will never be infinite. This consequence is discussed in (Rulot, 1992), where it is concluded that this feature is not important in real pattern recognition tasks.

Algorithms 1.1 and 1.2 describe the inference process using a set of trees \mathcal{C} belonging to the target language. The cost of the inference process is proven to be polynomial in the following theorem.

Algorithm 1.1 (*Error-correcting tree language inference algorithm*).

Input: A set of samples
 $\mathcal{C} = t_1, t_2, \dots, t_n$.

Output: A tree automaton
 $A = (Q, V, \delta, F)$ which accepts at least the sample set.

Method: */* Initial Automaton */*
 $Q = \text{Sub}(t_1) \cup \{t_1\}$
 $V = \text{leaves}(t_1) \cup \{\sigma\}$
 $F = \{t_1\}$
 $\delta(a) = a \quad \forall a \in \text{leaves}(t_i)$
if $u_1, u_2, \dots, u_p \in Q$
 $\wedge \sigma(u_1, u_2, \dots, u_p) \in Q$
 $\delta \supseteq \delta(\sigma, u_1, u_2, \dots, u_p)$
 $= \sigma(u_1, u_2, \dots, u_p)$
fi
/ Processing of the set */*
 $\forall t_i \in \mathcal{C} : i : 2, \dots, n$
 $A = \text{Expand}(t_i, A)$
f
EndMethod.

Algorithm 1.2 (*Expand(t_i, A) method*).

Input: A tree $t = \sigma(\tau_1, \tau_2, \dots, \tau_n)$.
A tree automaton
 $A = (Q, V, \delta, \{q_f\})$.

Output: A tree automaton that accepts at least $\{t\} \cup L(A)$.

Variables: $\text{Red}[1, \dots, |t|]$ array of
 $Q \cup \text{leaves}(t)$
 $A' = (Q', V', \delta', F')$

Method: $Q' = Q$
 $V' = V \cup \text{leaves}(t)$
 $\delta' = \delta$
 $F' = F$
Obtain the minimum cost path
 A_t
 $\forall z \in \text{Sub}(t) :$
 $d_z = \delta(\sigma, u_{z1}, u_{z2}, \dots, u_{zp}) = q_z$
if $z \in \text{leaves}(t)$
 $\text{Red}[z] = z$

else
 $\text{Red}[z] = q_z$
if
f
 $\forall z = \sigma(z_1, z_2, \dots, z_p) \in \text{Sub}(t)$ */**
in postorder **/*
if $d_z \in A_t^N$
 $Q' = Q' \cup \{q_N\}$ */* add*
a new state **/*
 $\delta' \supseteq \delta(\sigma, \text{Red}[z_1],$
 $\text{Red}[z_2], \dots,$
 $\text{Red}[z_p]) = q_N$
 $\text{Red}[z] = q_N$
else
if $\exists d_{z_j} \in A_t^N$
 $\delta' \supseteq \delta(\sigma, \text{Red}[z_1],$
 $\text{Red}[z_2], \dots,$
 $\text{Red}[z_p]) = \text{Red}[z]$
fi
fi
f
/ Accept transition */*
 $\delta' \supseteq \delta(\sigma, \text{Red}[\tau_1], \text{Red}[\tau_2], \dots,$
 $\text{Red}[\tau_n]) = q_f$
Return A'

EndMethod.

Theorem 1. *The temporal complexity of the inference Algorithm 1.2 is polynomial with respect to the size of the sample and the initial automaton.*

Proof. The distance algorithm used works out the accepting cost from a tree t to the automaton A with complexity $\mathcal{O}(q(\max\{m, p\}))$, where $p = |A|$, $m = |t|$ and q is a polynomial function (López et al., 2000). Once the accepting cost between the tree and the tree automaton is obtained, the algorithm uses the minimum cost path to modify the automaton in order to add the non-existing transitions.

Since the number of transitions needed to process the tree is bounded by the quantity of internal nodes of the tree, the number of transitions to be modified is also bounded by this factor. What is more, the cost of adding/modifying a transition is linear with the size of the tree, which makes the complexity $\mathcal{O}(m^2 + q(\max\{m, p\}))$. Although the degree of q is not established in (López et al.,

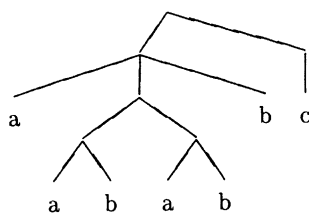
2000), the algorithm described uses a generic string error-correcting algorithm. This algorithm has a cost of at least second-degree polynomial (Stephen, 1992), which allows the asymptotic complexity of the algorithm proposed here to be set as $\mathcal{O}(q(\max\{m, p\}))$. \square

Given n trees in the whole inference process, with M being the maximum size of the set of trees, the complexity of the learning process is bounded by $\mathcal{O}(n \cdot q(\max\{M, p\}))$.

Example 1. Consider the automaton in Fig. 2 as an initial automaton, with three tree examples presented to the algorithm. The result of each

$$\begin{aligned} \delta(\sigma, a, q_1, b) &= q_1 \\ \delta(\sigma, a, b) &= q_1 \\ \delta(\sigma, a, q_2) &= q_2 \\ \delta(\sigma, c) &= q_2 \\ \delta(\sigma, q_1, q_2) &= q_3 \in F \end{aligned}$$

Fig. 2. Initial automaton.



D_A	1	2	3	4	5	6
q_1	0	0	5	5	2	9
q_2	2	2	6	7	0	10
q_3	7	7	2	9	6	5

Fig. 3. The first tree example and distance analysis against the initial automaton. The first two nodes are analyzed without error. The third node produces an edit operation with a cost of five. This error remains at a cost of five.

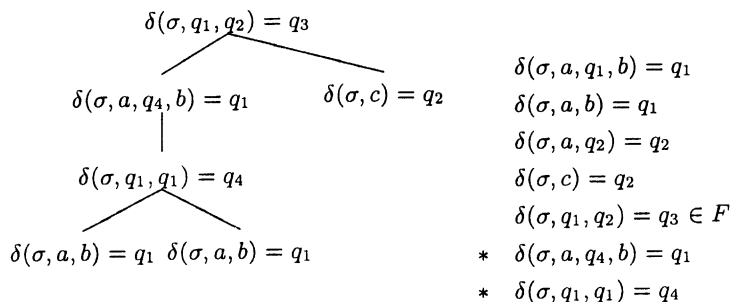


Fig. 4. Result of the first iteration of the inference algorithm and the automaton obtained from the first tree example (Fig. 3) and the initial automaton (Fig. 2). The productions added are marked with an asterisk.

error-correcting analysis is shown in a matrix where the rows represent the states of the automaton, and the columns represent the postorder enumeration of the internal nodes of the tree. The bold figures in each matrix show the minimum cost path of the analysis. The first tree example and its distance matrix to the initial automaton are shown in Fig. 3.

The scheme of the inference process and the resulting automaton are shown in Fig. 4. Note how a new state q_4 is created to consider the subtree which is rooted at the third node, and how another production is added to consider this new state and link it to the rest of the automaton (both marked with an asterisk in the figure).

The second tree example and its distance analysis are shown in Fig. 5. The result of the inference step is shown in Fig. 6. Note that the error analysis is due to the fifth node and that a new state q_5 is added in the way q_4 was. Furthermore, a production is added to consider the state q_4 as a child of the root node. Also note how the absence of this production, which produces no error in the analysis, is considered in the inference process.

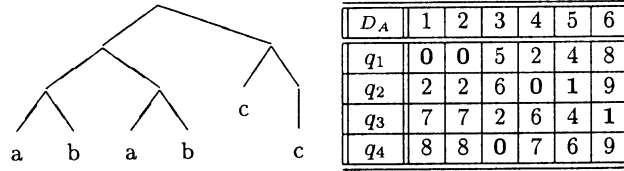


Fig. 5. The second tree and its error analysis matrix. Note how the state which was added in the previous iteration belongs to the minimum cost path.

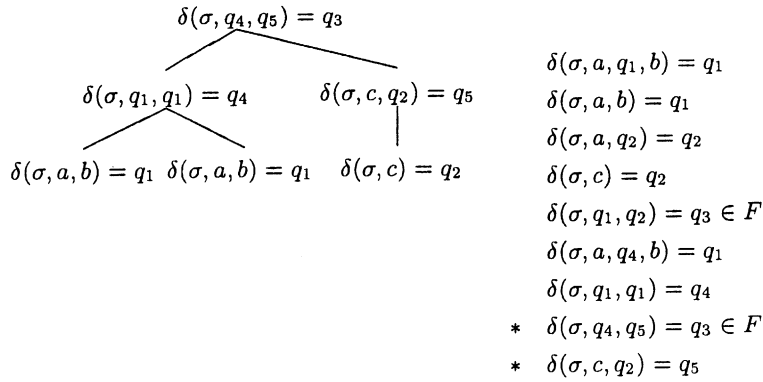


Fig. 6. Scheme of the inference process and the resulting automaton when the second tree example is considered.

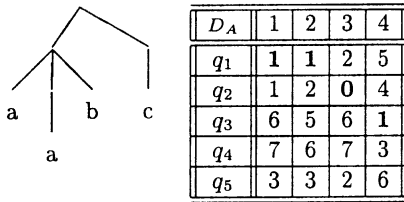


Fig. 7. The third tree example and its error-correcting analysis. Note that there are two ways of analyzing the first node to obtain the same cost. This does not affect the obtention of a minimum cost path.

The last tree example and its distance analysis matrix are shown in Fig. 7. The final automaton of the inference process is shown in Fig. 8. Note how the tree $\sigma(\sigma(a, \sigma(\sigma(a, \sigma(a, b)), b), \sigma(a, b)), b), \sigma(c)$ is now accepted by the automaton.

4. Edit distance to a tree automaton

Under a syntactic approach to pattern recognition tasks, the availability of efficient tools to

carry out a correct classification is as important as a good object representation or the obtention of good models which are able to consider as much variability as possible, among all the different classes. Before considering a change in object representation primitives, it is necessary to study whether this change produces an exponential time increase with respect to the initial representation.

When the tree approach is used, the existence of polynomial algorithms to obtain a tree representation (e.g., Hunter and Steiglitz, 1979) or to infer tree languages (García, 1993; García and Oncina, 1993) together with an error-correcting tree language analysis algorithm (López et al., 2000), offers an advantage over the classic string object representation. This is due to the increase in the representation power, together with a polynomial cost in the whole process.

The fact that there could be more than one set of edit operations (as set in Fig. 1) does not affect the result of the inference algorithm, because once an editing error is detected, a new production and

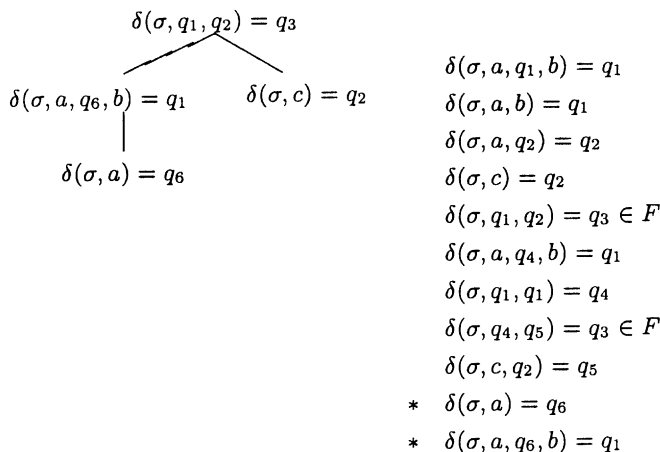


Fig. 8. Processing scheme to consider the last example and final automaton obtained.

a new state $q \in Q$ are added, whether or not there exist one or more possible editing operations having the same cost. The addition of q forces us to create another production to link the new one with the rest of the productions of the automaton. A new state is created unless the editing-operation takes place in a child of the root node, in which case the final state is used (see, for instance, the tree and its distance analysis in Fig. 5 and the result of the inference step in Fig. 6).

Fig. 9 could illustrate another conflicting situation, but it is clear that no confusion arises. No-

vice that there exist two different ways to edit the first node, using the first production or using the second production of the automaton. However, this fact does not affect the algorithm that creates the new state q_4 and the transitions which are needed to link this state to the rest of the initial automaton in order to accept the tree example (these transitions are marked with an asterisk).

In general, the only way for Δ not to be unique is that there be two productions of the automaton $\delta_n = (\sigma, u_1, u_2, \dots, u_n) = p$ and $\delta_m = (\sigma, v_1, v_2, \dots, v_n) = q$, where $u_i = v_i \forall i : 1, \dots, n$.

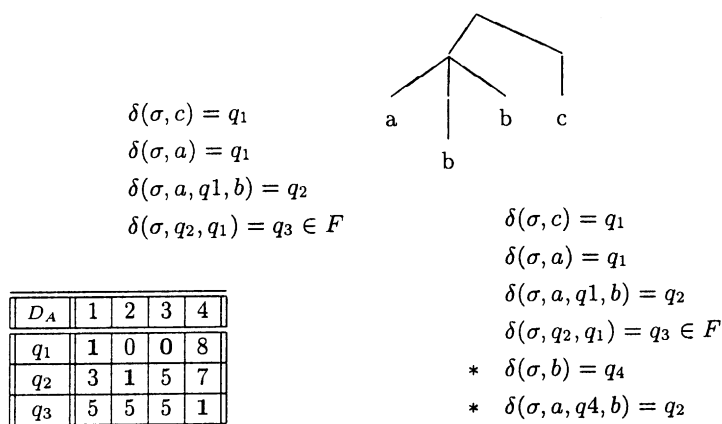


Fig. 9. Let the automaton on the left of the figure be the initial one. Also let the tree on the right be an example of a tree. The matrix on the left gives the distance analysis of the tree where the rows represent the states of the automaton, the columns represent the internal nodes of the tree in postorder, and the bold figures show Δ_i . Let the automaton on the bottom right be the result of the inference algorithm.

The initial automaton considered in the inference process does not fulfill the condition (as can be seen in Algorithm 1.1), and when the inference process considers a production $\delta_j = (\sigma, u_1, u_2, \dots, u_k) = r$ to be added, it implies that no other $\delta_k = (\sigma, v_1, v_2, \dots, v_k) = s$, where $u_i = v_i \forall i: 1, \dots, k$ belongs to the automaton. In that case, δ_k would belong to Δ and, therefore, there would be no need to add δ_j . Thus, given a sequence of samples, the result of the algorithm is always the same.

5. Experimental results

To test the performance of the tree language inference algorithm proposed in this work, a handwritten digit recognition task was considered. Several successful approaches to this task have been used. (e.g., Vidal et al., 1993). Since the results here reported were obtained without any kind of stochastic information, we consider them encouraging even if they do not reach the best classification rates reported. Furthermore, the rates obtained are comparable with other tree-based approaches reported (López and Piñaga, 2000; Rico, 1999).

The task carried out in the experimentation was the classification of images of isolated handwritten digits from '0' to '9'. For each one of the ten classes (digits from 0 to 9), a set of 500 such images was extracted from the data set 'NIST SPECIAL DATABASE 3, NIST Binary Images of Handwritten Segmented Characters' (Garris, 1992).

In order to model the features of the objects to classify, two different approaches were tested. First, a variation of the quad trees (q -trees) (Hunter and Steiglitz, 1979) was used. In a standard q -tree representation, the root of the tree is associated with the whole image to model, and any other internal node with one of the four quadrants of its parent image. When a node represents a one-colour square, the node becomes a leaf labelled with the color of the region. The variation of the q -tree representation used in this work allows us to set the maximum q -trees' depth. Thus, when the selected depth is reached, a leaf is added labelled with 'white' ('black') if the image is at least 75%

white (black); otherwise, the leaf is labelled with a 'grey' label. Note that this representation needs no thinning of the image to be represented.

Second, an 8-neighbour segment labelling scheme was used (López and Piñaga, 2000). This representation needs the samples to be thinned and the algorithm used was the one described in (Carrasco and Forcada, 1995). Once the image was thinned, the upper leftmost point of each image became the initial point of its representation. From this point, segments were extracted taking into account the value of the parameter 'window', and each segment was labelled using an eight neighbour scheme. This representation gives more compact trees but it is quite sensitive to the noise the thinning algorithm may cause. A digit example and some tree representations is shown in Fig. 10; note the effect of the parameters in the representation.

To test the behaviour of the algorithm, experiments were carried out using training sets of 200, 300 and 400 samples out of the 500 available samples per class, using the remaining as test set (300×10 , 200×10 and 100×10 samples per test set, respectively). In order to take profit from the data, each experiment entailed five iterations. Each iteration took into account a different set of samples for training and testing the automata inferred.

The string inference algorithm is sensitive to the order in the presentation of the samples (because when the larger samples are presented first, the automata have more available structures with which to work out the editing distance), as shown in (Rulot, 1992). Therefore, to test this behaviour in the algorithm, given a training set, three different orderings were considered: ascending, descending and random order (without applying any ordering).

The use of a table of distances between the symbols which label the leaves was also tested. In this way, the substitution of 'closer' leaves was 'cheaper' than the substitution of 'farther' leaves. The substitution table considered that takes into account the 8-neighbour-based representation is shown in Table 1. When the q -tree-based representation was used, the cost to substitute the symbols 'a' and 'b' was one; otherwise the cost was 0.5. When these tables were not used in the ex-

Table 2
Classification rates using the 8-neighbour-based representation

Samples ordering	Use of distances	Training size	Window			
			2	4	6	8
Descending	Yes	200	78.46	82.29	83.36	81.82
		300	82.94	82.92	85.36	84.22
		400	83.08	84.66	86.54	85.10
	No	200	82.79	84.14	85.28	83.31
		300	84.16	86.70	86.78	85.28
		400	82.12	86.84	84.24	86.44
Ascending	Yes	200	81.86	80.32	82.58	82.11
		300	73.02	76.98	81.76	83.87
		400	73.00	78.70	82.94	84.48
	No	200	74.78	79.57	81.43	83.38
		300	61.32	78.10	80.80	83.75
		400	72.36	78.46	82.44	84.18
No order	Yes	200	79.19	81.10	82.87	82.14
		300	79.22	83.84	84.20	83.69
		400	80.54	83.28	82.52	84.82
	No	200	79.58	81.57	83.59	83.21
		300	79.14	83.51	84.00	84.45
		400	79.64	83.66	85.72	85.22

The table shows the behaviour of the algorithm depending on the order in the presentation of the samples, the use of a table of distances both in the inference and analysis procedures, the size of the training set and the value of the parameter window which controls the size of the segments in the representation. The best results are marked in bold print.

Table 3
The classification rates when the q -tree-based representation is used

Samples ordering	Use of distances	Training size	Depth		
			3	4	5
Descending	Yes	200	76.36	81.80	78.98
		300	79.43	83.81	80.99
		400	81.50	84.86	84.28
	No	200	77.48	81.59	78.53
		300	80.39	84.01	81.25
		400	81.92	85.42	83.90
Ascending	Yes	200	76.59	80.53	77.28
		300	79.23	82.38	79.89
		400	80.98	83.02	80.50
	No	200	77.74	80.20	77.28
		300	80.28	82.83	79.71
		400	82.02	83.36	80.86
No order	Yes	200	75.97	80.62	77.59
		300	79.07	82.96	80.65
		400	80.25	83.88	82.56
	No	200	77.03	80.47	77.79
		300	79.66	82.99	80.85
		400	81.52	84.36	82.64

Note that bigger training sets are needed with this representation in order to obtain similar classification rates, and, even in that case, the results do not reach those obtained with the other representation used. Nevertheless, no matter which representation is used, the guidelines of the algorithm's behaviour are the same.

discussed in (Rulot, 1992). If the larger samples are presented first, the automata have more available structures with which to work out the editing distance, thus obtaining more compact automata and better results in the classification.

The experimentation obtained better results when the 8-neighbour representation was used. This is due to the fact that the main features of the object are modelled in a brief way. Thus, the modification or substructures are relatively more important in the whole edition process.

The use of a table of distances in the inference and classification steps gives worse results. It seems that the use of the table penalizes the insertion and deletion of structures, giving worse results. This penalization is greater when the 8-neighbour representation is used, because the representation is more compact and the table size is also bigger.

It should also be noted that less training data is needed to obtain the same classification rate when lower values of the 8-neighbour representation window parameter (larger representation of the images) are used.

6. Conclusions and future work

The development of an error-correcting analysis algorithm for tree automata (López et al., 2000), allows us to benefit from the ECGI string language inference algorithm strategy proposed in (Rulot, 1992). In this work, an extension of the ECGI inference algorithm is proposed in order to deal with tree-like representations.

This work also shows how the increase in the representation power allowed by the use of multi-dimensional primitives does not imply an exponential increase in the temporal complexity of the entire recognition process. It also shows how the inference algorithm does not create productions in the automaton which could lead to multiple minimum cost paths in the distance analysis of future samples. Therefore, given a sequence of samples, the result of the inference process is unique.

The proposed inference method has been tested in a handwritten digit recognition task and the results obtained show the validity of the approach. The use of other tree representations of the objects

(images or other objects), as well as the use of stochastic information should be studied. This could be directions for future work. It might also be interesting to study the influence of other combinations of the costs of the editing operations in real tasks.

Acknowledgements

The author would like to thank José M. Sempere for his helpful comments during the revision of this paper, as well as thank the anonymous referees for their suggestions which greatly helped to improve the paper.

References

- Angluin, D., 1980. Inductive inference of formal languages from positive data. *Inf. Control* 45, 117–135.
- Carrasco, R.C., Forcada, M.L., 1995. A note on the Nagendrapasad–Wang–Gupta thinning algorithm. *Pattern Recognition Lett.* 16, 539–541.
- Fu, K.S., 1982. *Syntactic Pattern Recognition and Applications*. Prentice-Hall, Englewood Cliffs, NJ.
- García, P., 1993. Learning k -testable tree sets from positive data. Technical Report DSIC II/46/1993, Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia.
- García, P., Oncina, J., 1993. Inference of recognizable tree sets. Technical Report DSIC II/47/1993, Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia.
- Garris, M.D., 1992. Design and collection of a handwriting sample image database. *Social Sci. Comput. J.* 10, 196–214.
- Gold, E.M., 1967. Language identification in the limit. *Inf. Control* 10, 447–474.
- González, R., Thomason, M., 1978. *Syntactic Pattern Recognition: An Introduction*. Addison-Wesley, Reading, MA.
- Hunter, G.M., Steiglitz, K., 1979. Operations on images using quad trees. *IEEE Trans. Pattern Anal. Machine Intell.* 1 (2), 145–153.
- López, D., Piñaga, I., 2000. Syntactic pattern recognition by error correcting analysis on tree automata. In: Ferri, F., Iñesta, J., Amin, A., Pudil, P. (Eds.), *Joint IAPR Internat. Workshops SSPR 2000 and SPR 2000, Lecture Notes in Computer Science*, Vol. 1876, pp. 133–142.
- López, D., Sempere, J.M., García, P., 2000. Error correcting analysis for tree languages. *IJPRAI* 14 (3), 357–368.
- Prieto, N., Sanchis, E., Palmero, L., 1994. Continuous speech understanding based on automatic learning of acoustic and semantic models. In: *Proc. 1994 Internat. Conf. on Spoken Language Process., ICSLP 94*.

- Rico, J.R., 1999. Off-line cursive handwritten word recognition based on tree extraction and an optimized classification distance. In: Torres, M.I., Sanfeliu, A. (Eds.), Proc. VIII SNRFAI.
- Rulot, H.M., 1992. ECGI. Un algoritmo de inferencia gramatical mediante corrección de errores. Ph.D. Dissertation, Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia.
- Sakakibara, Y., 1990. Learning context-free grammars from structural data in polynomial time. *Theoret. Comput. Sci.* 76, 223–242.
- Sakakibara, Y., 1992. Efficient learning of context-free grammars from positive structural examples. *Inf. Comput.* 97, 23–60.
- Stephen, G.A., 1992. String search. Technical Report TR-92-gas-01, School of Engineering Science, University College of North Wales, Gwynedd, UK, October 1992.
- Vidal, E., Rulot, H., Valiente, J.M., Andreu, G., 1993. Application of the error correcting grammatical inference algorithm (ECGI) to planar shape. In: *Grammatical Inference: Theory, Applications and Alternatives*. IEE. Digest No: 1993/092.