

Syntactic Pattern Recognition by Error Correcting Analysis on Tree Automata.

Damián López and Ignacio Ríaga

Departamento de Sistemas Informáticos y Computación.

Universidad Politécnica de Valencia.

Camino de Vera s/n, 46071. Valencia (Spain).

phone: (+34) - 96 - 3877352

fax: (+34) - 96 - 3877359

e-mail: dlopez@dsic.upv.es

Abstract. Although the multidimensional primitives are more powerful than string primitives and there also exist some works concerning distance measure between multidimensional objects, there are no many applications of this kind of languages to syntactic pattern recognition tasks. In this work, multidimensional primitives are used for object modelling in a handwritten digit recognition task under a syntactic approach. Two well-known language inference algorithms are considered to build the models, using as error model an algorithm obtaining the editing distance between a tree automaton and a tree; the editing distance algorithm gives the measure needed to complete the classification. The experiments carried out show the good performance of the approach.

Keywords: Syntactic pattern recognition, editing distance tree automata, error correcting parsing.

1 Introduction

A wide range of problems are related with pattern recognition. This general problem has mainly two approximations: the geometric [3], and the structural or syntactic one [4][5].

When the choice is the syntactic approach, the first step to consider is the modelling of the object domain. This phase is usually followed by an inference process to build grammatical models able to keep the structure of the different classes. After this, a parse over the models gives a classification by ownership.

Although the modelling phase has usually been performed by using strings of a given alphabet, mainly to take advantage of the huge quantity of tools that exist for string automata, there also exist more powerful primitives to model the object

domain, for instance trees and graph primitives. And since hierarchical features or information concerning connections between related areas of the pattern are inherent to these primitives, these features could be easily modelled by them [4][5].

Much work has been done on graphs and graph grammars [17], and although there even exist graph language inference algorithms [9], the temporal complexity involved causes make these methods not easy to use in applied tasks. Besides, due to the high representation power of these primitives, even very restricted families of graphs or graph languages might be useful in pattern recognition [10].

The description power of the tree primitives and the existence of time efficient tools to handle tree representations make this approach interesting. In fact there exist several tree language inference algorithms. Some of them use complete presentation in the inference process [7], others characterize a specified family [6], or adapt previous string language algorithms. Still others, even though they have not been developed to this purpose, could be seen as a tree language inference algorithm since the set of structured samples of a string language form a tree language [18].

Other works deal with the tree edition problem [20], establish a distance model between trees[14][15][22], or study the distance computation complexity [23][24].

When a real application is wanted, the availability of a big enough set of samples that allows the variability of the classes to be retained is a serious problem. So when an object is going to be classified, it usually does not fit any model, and therefore it is necessary to obtain the nearest model to the object structure. There are several methods where string languages [2][19] or even tree languages [12][11] are involved, but the latter have not been frequently used.

This work uses a tree based pattern recognition approach in a handwritten digit recognition task where the classification is carried out by using a recently proposed method which obtains a distance measure between a tree and a tree automaton. First of all, this work introduces the notation needed. Then the approach to be used is explained: the feature extraction procedure, the tree language algorithms used and the error correcting algorithm which will provide the error model, and the series of experiments carried out. Finally, a summary of the best results, the conclusions and the proposed future lines of work are exposed.

2 Theoretical Concepts and Notation

Let V be an alphabet and \mathbb{N} the set of natural numbers, a *ranked alphabet* is defined as the association of V with a finite relation $r \subseteq (V \times \mathbb{N})$. V_n denotes the subset $\{\sigma \in V \mid (\sigma, n) \in r\}$.

Let V^T be the set of finite trees whose nodes are labelled with symbols in V , where a tree is defined inductively as follows:

$$\begin{aligned}
V_0 &\subseteq V^T \\
\sigma(t_1, \dots, t_n) &\in V^T : & \forall t_1, \dots, t_n \in V^T, \\
& & \sigma \in V_n
\end{aligned}$$

Let the root of a tree t , denoted by $root(t)$, be:

$$\begin{aligned}
root(a) &= a : & \forall a \in V_0. \\
root(\sigma(t_1, \dots, t_n)) &= \sigma : & \forall t_1, \dots, t_n \in V^T, \\
& & \sigma \in V_n
\end{aligned}$$

Let the size of a tree t ($|t|$) be:

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

A *deterministic tree automaton* is defined as the four-tuple $A = (Q, V, \delta, F)$ where Q is a finite set of states; V is a ranked alphabet; $F \subseteq Q$ is a set of final states and $\delta = (\delta_0, \dots, \delta_m)$ is a finite set of functions defined as:

$$\begin{aligned}
\delta_n : (V_n \times (Q \cup V_0)^n) &\rightarrow Q & n = 1, \dots, m \\
\delta_0(a) &= a & \forall a \in V_0
\end{aligned}$$

δ can be extended to operate on trees as follows:

$$\begin{aligned}
\delta : V^T &\rightarrow Q \cup V_0 \\
\delta(\sigma(t_1, \dots, t_n)) &= \delta_n(\sigma, \delta(t_1), \dots, \delta(t_n)) & \text{if } n > 0 \\
\delta(a) &= a & \forall a \in V_0
\end{aligned}$$

A tree $t \subseteq V^T$ is accepted by A if $\delta(t) \in F$. The set of trees accepted by A is defined as $L(A) = \{t \in V^T \mid \delta(t) \in F\}$

3 Syntactic Pattern Recognition

Although there exist different approaches in the literature to perform a syntactic classification [4][5], the traditional way to undertake the object modelling, is the use of strings from a given alphabet.

Once the grammatical models are obtained, several methods exist to work out a distance measure between the samples to be classified and the models [2][4][21][19], allowing the classification of noisy samples.

There also exist several works which establish an edit distance between trees [14][15][20][22][23], or between a tree and a tree automaton [11][12], but however there are few applications of these works to real tasks [16].

In this work, we consider two tree language inference algorithms [6][18] to obtain tree automata as class models in a handwritten digit recognition task. Both algorithms deal with a kind of trees without labels in their internal nodes, named skeletons, so, we will transform trees into skeletons making from now on no distinction between them.

Once the automata are obtained, the samples are classified considering the distance obtained by application of the error correcting analysis algorithm proposed in [11]. The experiments carried out show the good performance of the algorithm.

3.1 Feature Extraction

All the binary images of handwritten digit samples used in this work,¹ both in the training and test phases, were previously thinned using the Arcelli and Sanniti algorithm [1]. These simplified images were the starting point of the tree representation procedure, which may be summarized as follows:

1. The upper leftmost pixel of the thinned image was assigned to the root node of the tree, and “@” is assigned as label.
2. Each node of the tree has as many branches as neighbours has the pixel assigned to the node.
3. Each branch stretches until one of the following conditions is fulfilled:
 - The branch reaches the length established in the parameter *window*.
 - There are no more neighbours to the pixel (a final pixel is found).
 - A pixel with more than one neighbour is found, (an intersection pixel).
 Once the end of the branch is found, its final pixel is assigned to a new node. The node label comes from the scheme explained in figure 1.
4. For each node with neighbours, go to step 2.

The trees which are obtained by this procedure have labels in their internal nodes. To transform these trees into skeletons without loss of information the operator “*Sk*” was applied:

$$\begin{aligned}
 Sk(a) = a & : & \forall a \in V_0. \\
 Sk(\alpha(t_1, \dots, t_n)) = \sigma(\alpha, Sk(t_1), \dots, Sk(t_n)) & : & \forall t_1, \dots, t_n \in V^T, \\
 & & \alpha \in V_n, \\
 & & \sigma \notin V
 \end{aligned}$$

One original sample, the image after the application of the thinning algorithm and the tree representation, are shown in figure 2.

¹ from the data set “NIST SPECIAL DATABASE 3, NIST Binary Images of Handwritten Segmented Characters”.

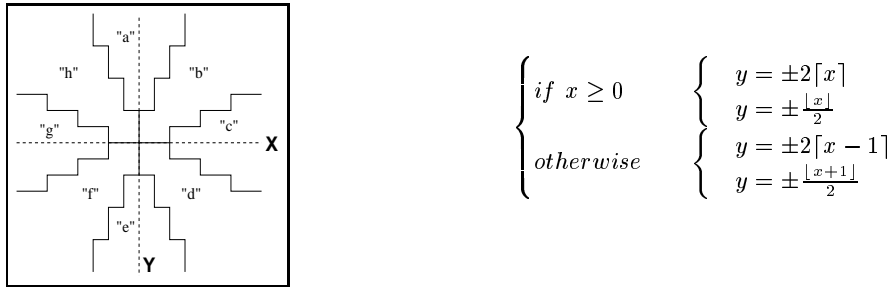


Fig. 1. These equations divide the 2-D space into eight regions, as shown on the left; let the northern one be the region with label “a”, and clockwise, let the rest be labelled consecutively. When the label of a segment has to be obtained, the starting pixel shall be placed at the origin of the axes; the labels is assigned to the final pixel depending on its relative situation to the starting one.

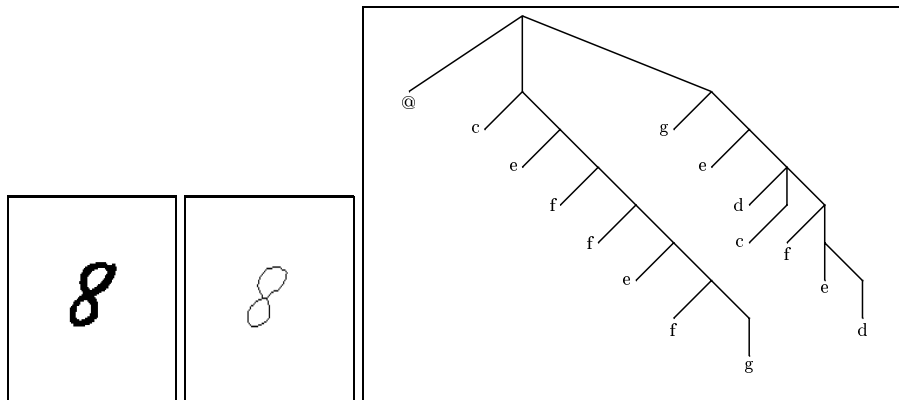


Fig. 2. A digit sample, the thinned sample obtained, and the tree representation when the window is equal to 8 are shown.

3.2 Error Correcting Analysis

Due to the fact that almost all the samples to be classified in real tasks are distorted or have some kind of noise, the development of error models is crucial in syntactic pattern recognition applications.

The algorithm we test in this work [11] explores each tree in postorder, calculating the cost of reducing each subtree to each one of the states of the automaton. Briefly, the method works out the distance of every subtree to every state of the automaton, to do that the algorithm compares the successors of a node with the different ways the automaton can produce each state.

Although the insertion, deletion and substitution costs vary depending on the node being analyzed, a dynamic programming scheme allows the algorithm to obtain the distance with polynomial time complexity with respect to the size

of the tree and the automaton. The authors prove that the distance obtained is the minimum one according to the edit operations proposed.

3.3 Experimentation and Results

In order to test the behaviour of the edit distance algorithm, two tree language inference algorithms were used [6][18], both of them working with positive data.

The first of them [6], obtains a *k-Testable in strict sense tree automaton* (k-TSS tree automaton), which is basically defined by a set of substructures of size k , allowed to appear in the members of the tree language. In case that the target language were not a k-testable language, the algorithm obtains the smallest k-TSS tree language that contains the training set. Varying the values of k , a hierarchy of languages could be obtained. In that way, the higher the value of k , the bigger class; this is the reason why several values of the parameter k have been tested in the experimentation.

The second algorithm [18], introduces a context-free normal form named *reversible context free grammars*, and gives an algorithm to learn context free grammars in this normal form with positive samples and structural information of them, that is, the derivation skeleton of the samples. Since every context free grammar has an equivalent reversible one, the algorithm learns the context free class with positive structural information. Because this structural information (sets of trees) could be seen as members of a tree language, the adaptation to this kind of multidimensional languages is straightforward.

window	in-order string representation
16	(@(d(f(e(g))(g)))(f(e(f))))
8	(@(c(e(f(f(e(f(g)))))))(g(e(d(c)(f(e(d)))))))
4	(@(c(c(e(f(f(g(f)))))))(g(f(e(d(c(e(e(g(g)))))))(g(f(e(e(d))))))))

Fig. 3. Differences between trees obtained with different window sizes. All the trees represent the same digit (the one in figure 2). Notice that the bigger the window, the more compact the representation.

The classification was carried out with and without considering the editing measure. Due to the fact that the parameter window modifies the tree representation (as shown in figure 3), it was also considered in the experiments.

Moreover, a *classification scheme by voting* was implemented. Into this scheme, being D_k and D_r the lower distance among the k-testable and reversible models respectively, and C_k and C_r the set of models with distance D_k and D_r , the classification was completed following the algorithm 1.1.

Several sizes of the training set were tested (100, 200, 300 and 400 samples), testing the models with the same 3000-sample set. When the k-testable algorithm was used, values of k between 2 and 6 were considered. A brief comparative of the results obtained is showed in figure 4. An extended version of these results can be found in [13]

Algorithm 1.1 Classification by voting scheme.

Input: C_k, C_r, D_k, D_r
 t , sample to be classified

Output: C , best class to classify t

Method: **if** $|C_k| == |C_r| == 1$ **and** $D_k == D_r$
 $C = C_k$
fi
if $|C_k| == |C_r| == 1$ **and** $D_i < D_j : i, j \in \{k, r\}$
 $C = C_i$
fi

if $|C_k| \geq 1$ **or** $|C_r| \geq 1$
 if $|C_k \cap C_r| == 1$
 $C = C_k \cap C_r$
 else
 if $D_i < D_j$ **and** $|C_i| == 1 : i, j \in \{k, r\}$
 $C = C_i$
 else t could not be classified
 fi
fi
fi

EndMethod:

In every case, the votation strategy gave the best results, being able to classify correctly up to a 30% more samples than the models obtained directly by inference.

As expected, the experimentation showed that 100 samples were not enough to model all the variance in the classes. Using 400 samples, an early over-generalization was observed together with very similar distances between classes when the error model is taken into account, increasing in that way the ambiguity rate.

The best results without error model were obtained with the 3-TSS, 6 as window size and a training set of 300 samples (23.53%). All the training set sizes proved, gave reversible models with poor generalization, obtaining therefore bad results. Using an error model, the votation scheme, training set of 300 samples and 6 as window size, the 83.83% was reached.

4 Conclusions and Future Work

In this work a syntactic pattern recognition task is carried out. Multidimensional (tree) primitives were used to model the objects in a handwritten digit recognition task. These primitives give more representation power than string primitives. Two tree language inference algorithms [6][18] were used to build the models that, together with a mixed strategy, were considered to carry out the

$ M^T $	algorithm	window size	%
200	3-TSS	4	15.17 / 43.50
200	4-TSS & rev	4	9.57 / 80.10
300	3-TSS	6	23.53 / 49.43
300	4-TSS & rev	6	10.23 / 83.83
400	3-TSS	4	21.30 / 36.77
400	4-TSS & rev	4	17.74 / 79.70

Fig. 4. Summary of results. From left to right: size of the training set, algorithm used (k-testable or votation), size of the window and classification rate (with and without error model).

classification. A editing distance algorithm between trees and tree automata was used to obtain an error model.

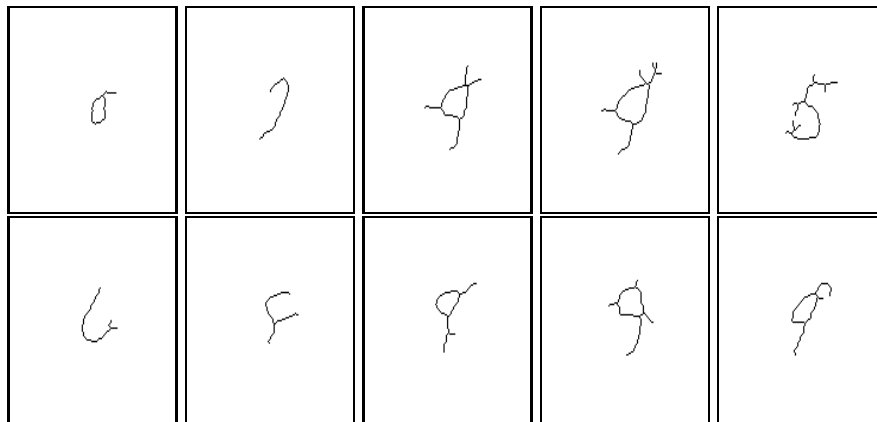


Fig. 5. Examples of noisy digits. From left to right and downwards: zero, two, four, four, five, six, eight, eight, nine, and nine.

The results obtained prove the validity of the approach. Nonetheless several of the thinned samples showed noise (figure 5) which opens the possibility of improving the results introducing modifications in the tree extraction algorithm to get rid of this noise and giving the possibility of representing new features (for instance, loops).

Furthermore, considering the whole scheme of this work (tree representation, inference algorithms and error model), the editing distance algorithm gives a somewhat inaccurate measure, which causes ambiguity. Another alternative tree representation (perhaps qtrees [8]), a previous weight learning operation step, or the use of probability as an addition to the scheme, might help to improve the performance.

5 Acknowledgements

The authors would like to thank José M. Sempere for his comments on the revision of this paper and to Segundo González for his help.

References

1. C. ARCELLI, G. SANNITI DI BAJA A width-independent fast thinning algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 7. No. 4. pp. 463-474. 1985.
2. H. BUNKE String matching for structural pattern recognition. In *Syntactic and Structural Pattern Recognition: Theory and Applications*. Eds. H. Bunke, A. Sanfeliu. World Scientific. 1990.
3. R. O. DUDA, P. E. HART *Syntactic pattern recognition and applications*. John Wiley and sons. 1973.
4. K. S. FU *Syntactic pattern recognition and applications*. Prentice-Hall, 1982.
5. R. GONZÁLEZ AND M. THOMASON *Syntactic Pattern Recognition. An Introduction*. Addison-Wesley Publishing Company. 1978.
6. P. GARCÍA Learning k-testable tree sets from positive data *Technical Report DSIC II/46/1993*. Departamento de Sistemas Informáticos y Computación. Universidad Politécnica de Valencia. 1993.
7. P. GARCÍA, J. ONCINA Inference of recognizable tree sets. *Technical Report DSIC II/47/1993*. Departamento de Sistemas Informáticos y Computación. Universidad Politécnica de Valencia. 1993.
8. G. M. HUNTER, K. STEIGLITZ Operations on images using quad trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 1. No. 2. pp. 145-153. 1979.
9. E. JELTSCH, H-J. KREOWSKI Grammatical inference based on hyperedge replacement. In *LNCS 532 Graph grammars and their application to computer science*. pp. 461-474. Springer-Verlag 1991.
10. D. LÓPEZ, J. M. SEMPERE Handwritten digit recognition through inferring graph grammars. A first approach. In *LNCS 1451 Advances on Pattern Recognition*. Joint of the SSPR and SPR International Workshops. pp. 483-491. Springer-Verlag 1998.
11. D. LÓPEZ, J. M. SEMPERE, P. GARCÍA Error correcting analysis for tree languages. *To be published on IJPRAI*.
12. S. Y. LU, K. S. FU Error-correcting tree automata for syntactic pattern recognition. *IEEE Transactions on Computers*. C-27. No. 11. pp. 1040-1053. 1978.
13. I. PIÑAGA Inferencia de lenguajes de árboles. *Proyecto final de carrera*. Facultad de Informática. Universidad Politécnica de Valencia. 1999.
14. B. J. OOMMEN, K. ZHANG, W. LEE Numerical Similarity and Dissimilarity Measures Between Two Trees. *IEEE Transactions on Computers*. To be published.
15. B. J. OOMMEN, R. K. S. LOKE On the Recognition of Noisy Subsequence Trees. In *LNCS 1451 Advances on Pattern Recognition*. Joint of the SSPR and SPR International Workshops. pp. 169-180. Springer-Verlag 1998.
16. J. R. RICO Off-line cursive handwritten word recognition based on tree extraction and an optimized classification distance. In *VIII National Symposium on Pattern Recognition and Image Analysis. M.I. Torres and A. Sanfeliu (Eds.)* Vol. 2, pp. 15-16. 1999.

17. G. ROZENBERG (ED.) *Handbook of Graph Grammars and Computing by Graph Transformation, Volume 1: Foundations* World Scientific, 1997.
18. Y. SAKAKIBARA Efficient Learning of Context-Free Grammars from Positive Structural Examples. *Information and Computation* 97, 23-60. 1992
19. G. A. STEPHEN String search. *Technical Report TR-92-gas-01*. School of Engineering Science. University College of North Wales. Gwynedd. United Kingdom. October 1992.
20. K. C. TAI The tree to tree correction problem *Journal of the ACM* Vol. 26, No 3, pp. 422-433. 1979.
21. E. TANAKA, K. S. FU Error-correcting parsers for formal languages *IEEE Transactions on Computers* C-27 No. 7. pp. 605-616. 1978.
22. K. ZHANG, D. SHASHA Simple Fast Algorithms for the Editing Distance between Trees and Related Problems. *SIAM J. Computation* Vol. 18, No 6. 1245-1262. 1989
23. K. ZHANG, R. STATMAN, D. SHASHA On the Editing Distance between unordered labeled Trees. *Information Processing Letters* 42, 133-139. 1992
24. K. ZHANG, T. JIANG Some MAX SNP-hard results concerning unordered labeled trees. *Information Processing Letters* 49, 249-254. 1994