Handwritten Digit Recognition Through Inferring Graph Grammars. A First Approach

Damián López and José M. Sempere

Departamento de Sistemas Informáticos y Computación.
Universidad Politécnica de Valencia.
Camino de Vera s/n, 46071. Valencia (Spain).
email:{dlopez, jsempere}@dsic.upv.es.

Abstract. Graph grammars have been widely used in areas such as program semantics, concurrence and parallelism, or graphic generation. This work is concerned with pattern recognition tasks and how to use a class of graph grammars (web grammars) to modelize the objects in both the learning and recognition phases. First, we give some definitions and basic notation referring to web grammars. We then propose an adaptation of a previous method for learning web grammars and we report some preliminary results on a handwritten digit recognition task together with future work to be carried out.

Keywords: Pattern Recognition, Graph Grammars, Grammatical Inference.

1 Introduction

The traditional way of modelizing objects in syntactic pattern recognition applications has been done, principally, by using string languages. This way of modelizing the objects of the problem does not provide a correct treatment of the *knots* of the objects (for instance, the intersection point of digit eight).

In the 60's, several works appeared that introduced a new approach based on the graph theory, so the concepts of graph grammar and graph language were introduced as a generalization of strings and tree languages. These contributions provide a more powerful tool for object modelling than strings which can only partially modelize.

Since then, a whole theory about graph grammars has been developed and the properties of different families of graph grammars have been established [1] [2]. This theory has also been applied to several fields.

In this work, this approach is applied to a pattern recognition task. To this purpose, a family of graph grammars (web grammars) was used and an inference method to obtain web grammars from examples is proposed. To test the method, a handwritten digit recognition task was carried out.

The scheme of the paper is the following one. First, some basic notation and definitions concerning web grammars are presented; next, the inference algorithm is proposed. Then, the task and the guidelines for solving it with this approach are explained and the results of the experimentation carried out are reported. These results are promising for eventually constructing a satisfying final version of the method. Finally, some conclusions and suggestions for future which will increase the performance of the method are presented.

2 Theoretical concepts and notation

We introduce some basic definitions related to graph grammars theory. We present these concepts in a way similar to other works such as [3] [2] and [1]

Definition 1. A *web* is defined as an undirected graph which is labeled in its nodes[1]. Although the literature distinguishes between the concepts graph and web, this paper does not.

Definition 2. Given a graph W, A is a subgraph from W if all the nodes in A are also in W, if the nodes are equally labeled and if the edges in A join the same nodes as in W. Given a graph α, we will denote the set of labels taken by the nodes of α by N_α.

Definition 3. A graph grammar is defined as a tuple (N, Σ, P, S) where:

- N is an auxiliary alphabet.
- Σ is a terminal alphabet ($N \cap \Sigma = \oslash$).
- S is the axiom of the grammar or initial graph ($N_S = \{S\} \subseteq N$).
- P is a finite set of productions each of which is a tuple (α, β, F) where:
  - α and β are graphs such that $N_\alpha, N_\beta \subseteq \{N \cup \Sigma\} \wedge N_\alpha \cap N \neq \oslash$
  - F is a non-empty set of connectivity functions

  $$f : N_\beta \times N_\alpha \rightarrow \{2^{N \cup \Sigma} \cup \{normal\}\}$$

  Its meaning, as explained in [3], establishes that given $f(A, B) = \{C, D\}$, the application of the rule where f belongs, will connect the node A ($A \in \beta$) with the neighbors of node B ($B \in \alpha$ which are labeled with C or D); node B and all its edges are then deleted. If the function f takes the *normal* value, it is possible to substitute B for A in any context.

Definition 4. Let W be a graph and p a be production (α, β, F) of the grammar G such that α is a subgraph of W. Let U be the graph resulting from eliminating the subgraph α of W, and V the result of connecting β to U using the functions in F.

Then, it is said that W and V are in direct derivation relationship inside the grammar G, denoted by $W \Rightarrow V$, or $W \underset{G}{\Rightarrow} V$ in case of ambiguity.

Definition 5. Let a derivation relationship denoted by $\overset{*}{\underset{G}{\Rightarrow}}$ or $\overset{*}{\Rightarrow}$ be the reflexive and transitive closure of the relation $\Rightarrow$ previously defined.

So, given a graph set $\{W_1, W_2, W_3, ...W_m\}$ such that:

$$W_1 \underset{G}{\Rightarrow} W_2 \underset{G}{\Rightarrow} W_3 \underset{G}{\Rightarrow} ... \underset{G}{\Rightarrow} W_m$$

[1] Montanari [6] distinguishes between directed and undirected webs.

We say that W_1 derives (is in derivation relationship with) W_m and it is denoted by:

$$W_1 \overset{*}{\underset{G}{\Rightarrow}} W_m$$

Definition 6. Let the language generated by the graph grammar G, L_G, be the set of graphs whose nodes are all labeled by symbols in Σ that can be derived by applying the rules in G starting from the axiom S. That is:

$$L_G = \{\alpha \mid S \overset{*}{\underset{G}{\Rightarrow}} \alpha \wedge N_\alpha \subseteq \Sigma\}$$

Several papers classify the grammars taking into account many features. Therefore, looking at the set of connectivity functions of the productions, we can talk about Normal Web Grammars [6]. We can talk about context sensitive, context free or regular (linear) web grammars if we take into account the form of the graphs in these productions [1]. Therefore, in this work, we are dealing with context free, normal, web grammars[2].

3 Inference Method

The proposed inference method is a modification of a well known and tested one used on the regular string languages inference. This method [7] builds a *prefix tree acceptor* (PTA) using the positive sample and attempts to merge the states of the automaton in lexicographic order. The method determines whether the resulting automaton is consistent with the sample (positive and negative) thus allowing the join or in case that the automaton accepts a negative sample undoing the join. The method goes on until no more merges between states of the automaton are possible.

In order to apply the previous method, we represent the grammar as a *states transition diagram* (STD) where the states are associated to the non-terminal symbols of the grammar and the transitions are determined by the productions that substitute the non-terminal symbol of the origin state by a graph [5]. A state is a final state when no rule can be applied from it, while the initial state is the axiom (Figure 1).

The method uses this grammar representation to construct a graph grammars acceptor based on the idea of the PTA. In this way, using the positive sample, an STD which accepts the positive sample is constructed. In addition, some other samples which are not in this set are also accepted.

Once the initial STD is built, the method attempts to merge states when the join produces no ambiguity, taking into account that the final STD must not accept any sample from the negative sample set. When no more merges are possible, the STD is considered a consistent correct conjecture [5].

[2] As has been advanced, although the literature distinguishes between graph grammars and web grammars, and because we just work with web grammars, we shall not do it

α	β	f
$\overset{S}{\bullet}$	$\overset{@}{\bullet}\text{———}\overset{A}{\bullet}$	$f(@, S) = \{normal\}$
$\overset{A}{\bullet}$	$\overset{c}{\bullet}\text{———}\overset{B}{\bullet}$	$f(c, A) = \{normal\}$
$\overset{B}{\bullet}$	$\overset{f}{\bullet}\text{———}\overset{C}{\bullet}$	$f(f, B) = \{normal\}$
$\overset{B}{\bullet}$	$\overset{B}{\bullet}\text{———}\overset{E}{\bullet}$	$f(B, B) = \{normal\}$
$\overset{B}{\bullet}$	$\overset{B}{\bullet}\text{———}\overset{G}{\bullet}$	$f(B, B) = \{normal\}$
$\overset{C}{\bullet}$	$\overset{a}{\bullet}\text{———}\overset{D}{\bullet}$	$f(a, C) = \{normal\}$
$\overset{D}{\bullet}$	$\overset{b}{\bullet}\text{———}\overset{c}{\bullet}$	$f(b, D) = \{normal\}$
$\overset{E}{\bullet}$	$\overset{g}{\bullet}\text{———}\overset{F}{\bullet}$	$f(g, E) = \{normal\}$
$\overset{F}{\bullet}$	$\overset{f}{\bullet}\text{———}\overset{e}{\bullet}$	$f(f, F) = \{normal\}$
$\overset{G}{\bullet}$	$\overset{e}{\bullet}\text{———}\overset{H}{\bullet}$	$f(e, G) = \{normal\}$
$\overset{H}{\bullet}$	$\overset{f}{\bullet}\text{———}\overset{g}{\bullet}$	$f(f, H) = \{normal\}$

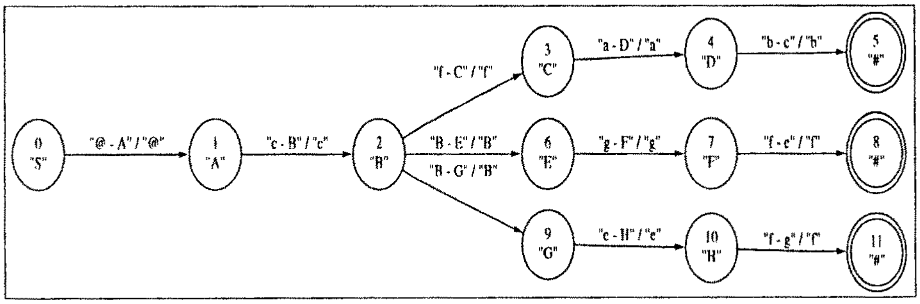

Fig. 1. Example of a representation of a graph grammar with a states transition diagram (STD). Every edge is labeled by β and the label of the node to be substituted. The final states are always labeled by '#'

4 Pattern recognition task

In order to test the behavior of the inference method proposed above a handwritten digit recognition task was considered. The modelization of each digit with a graph that could represent all the underlying features of the sample was the first step to be performed.

Once all the samples had been reduced to a graph, the inference method was used to obtain a structural model for each class of the pattern recognition problem. With these models, a test phase was implemented to test the capabilities of the proposed algorithm.

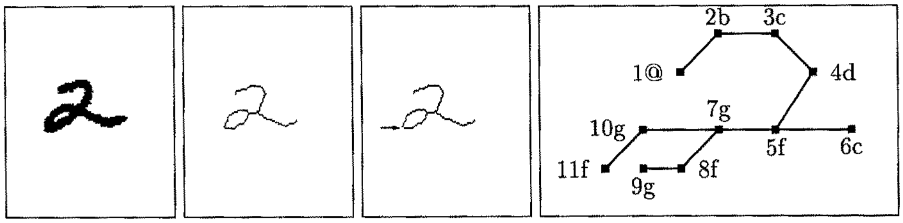

Fig. 2. The whole process is shown in this figure: the first box shows the initial digit, the second one shows the thinned digit, the arrow in the third one indicates the cut point and, finally the underlying graph is presented in the fourth box.

4.1 Preprocess

The digits[3] which were used in both the training and test phases, were represented as bitmaps where '0' dots stood for background and '1' dots for points belonging to the digit. Each sample was preprocessed to obtain the graph that modelized it. This process followed the scheme below[5]:

- **Thinning of the digit:** several algorithms were studied. The Arcelli-Sanniti one was used due to its performance with the samples, [11] (second box in figure 2).
- **Deletion of the cycles of the digit:** The sample recognition can be reduced to the recognition of repeated and interconnected simple patterns. Moreover, this step reduces the complexity of the problem, allowing us to work with a more restricted family of graph grammars (third box in Figure 2).
- **Construction of the graph** using a parameter *window*[4] to obtain a graph which varies in complexity. i.e. the higher the value of the window parameter, the simpler the graph obtained. Every edge of the graph represents a

[3] All samples were obtained from the data base "NIST SPECIAL DATABASE 3, NIST Binary Images of Handwritten Segmented Characters".

[4] A window is the number of dots to be taken into account before considering a node creation and a new edge representing a change of direction.

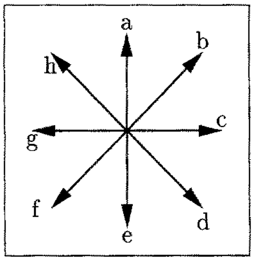

Fig. 3. Set of primitives used to represent the directions in the nodes.

segment of the digit which maintains the same direction according to the set of primitives (Figure 3). Every node has the label corresponding to the direction that has been used in the segment of the digit. i.e. label "3b" means that the segment which arrives to node 3 uses the direction "b" according with the set of primitives.

4.2 Experimentation and results

Once the samples had been reduced to graphs, the inference algorithm was used to obtain a grammatical model for each digit.

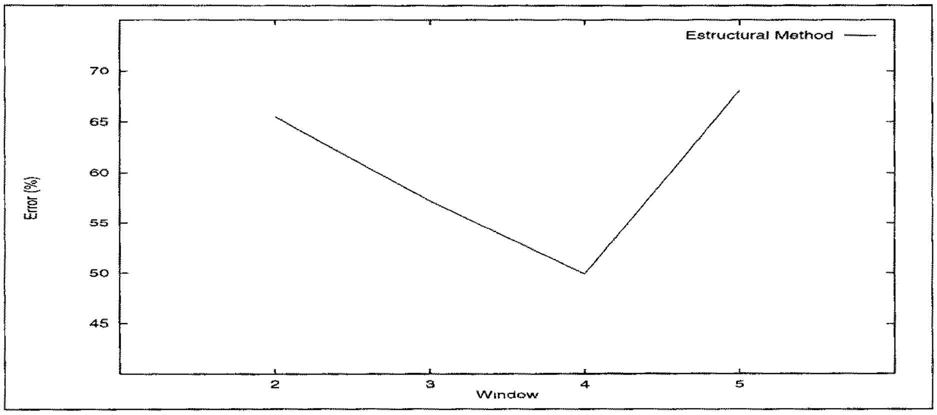

Fig. 4. Average of the error rates obtained for each of the values given to the window parameter. The X axis shows the size of the window and the Y axis indicates the average error rate of the five iterations carried out for each window. This rate was calculated as the number of correctly classified samples minus the total number of misclassified samples divided by the total number of samples.

The number of elements in the initial set was 1000 (100 in each class), a *leaving-k-out* strategy was used, and five iterations were carried out, constructing a grammatical model for each of them with a positive set containing 80 digits and a negative set consisting of 360 samples (40 digits for each of the 9 remaining classes). A test set was formed in each iteration with 200 digits (20 for each of the 10 classes). For each iteration four different values of the parameter window (2, 3, 4 and 5) were taken into account.

A summary chart of the results obtained is shown in Figure 4.

window 4	0	1	2	3	4	5	6	7	8	9
0	72	-	-	-	1	-	1	-	1	-
1	-	95	-	-	-	-	-	87	-	-
2	-	-	20	-	-	-	-	-	-	1
3	1	-	-	71	-	1	-	-	-	-
4	-	-	-	-	62	-	-	-	-	-
5	1	-	-	2	-	78	4	-	-	-
6	1	-	-	-	-	2	59	-	-	4
7	-	-	3	-	-	-	-	87	-	1
8	-	-	-	-	-	-	-	-	8	2
9	-	-	-	-	-	-	4	-	1	67

Fig. 5. Summary of the results obtained after the five iterations using a window of size 4. Each column (c) shows a trained model and each row (r) one digit class. Each intersection (c, r) indicates the number of digits of a class belonging to a model. For instance, the model of the digit *three* accepts two *fives*.

After an analysis of the error rate, the main conclusion was that the ambiguity in the classification produced almost all of the errors (as shown in the table in Figure 5). Therefore, a way to correct this ambiguity was tested. Once the model of the digit was built, the frequency of use of every production used to process the whole positive sample was performed. This frequency became a probability when normalized and was assigned to each production of the grammar, so that an ownership probability to each class was assigned to the samples of the test set.

The results in Figure 6 show that the error rate was considerably reduced.

5 Conclusions and future work

On the one hand, the new approach to pattern recognition problems proposed in this paper gives a more powerful representation. On the other hand this method may well have to be improved by adding an error correcting procedure however the results are already promising.

In order to extend these results, we are planning to carry out some future work:

- The development of an error-correcting method for this family of grammars. References about the measures of distance between non-unidimensional models are proposed by Oommen [8][9].

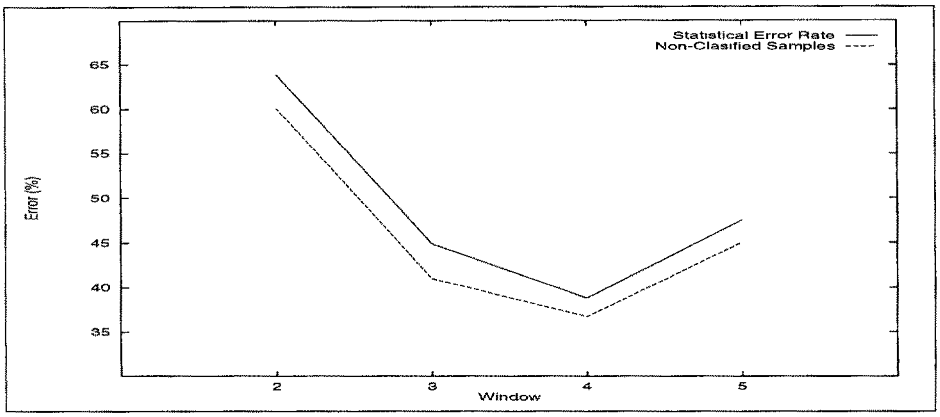

Fig. 6. Summary of the error rate obtained after the five iterations using probabilities to eliminate the ambiguity. The chart follows the same scheme as Figure 4 and the error rate is calculated in the same way. The refusal rate is calculated as the number of refused samples divided by the number of samples in the class.

- The application of more general families of grammars, by means of a new parsing procedure.

Once we improve our results, we think that this method could be a good way to solve other related problems such as the following:

- The application to other sample sets, for instance: alphabetic characters, other alphabets (Hebrew, Cirilic, Arabic), etc.
- The application to the recognition of continuous writing. A general scheme for this might be as follows:
  - Learning phase of the models of each character.
  - Segmentation of a continuous writing sample with a segmentation procedure.
  - Application of the method to the segmented sample, by combining the basic learned models.

References

1. N. Abe, M. Mizumoto, J. Toyoda and H. Tanaka Web Grammars and Several Graphs. *Journal of Computer and System Sciences* 7, 37-65. 1973.
2. G.Rozenberg (ed.) *Handbook of Graph Grammars and Computing by Graph Transformation, Volume 1: Foundations* World Scientific, 1997.
3. R. González and M. Thomason *Syntactic Pattern Recognition. An Introduction.* Addison-Wesley Publishing Company. 1978.
4. J. Hopcroft, J. Ullman *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley Publishing Company. 1979.
5. D. López Inferencia de Gramáticas de Grafos para una tarea de Reconocimiento de Dígitos Manuscritos. Technical Report, DSIC-II/34/97 Departamento de Sistemas Informáticos y Computación, *Polytechnic University of Valencia.* September 1997.

6. UGO G. MONTANARI Separable Graphs, Planar Graphs and Web Grammars. *Information and Control* 16, pp 243-267. 1970.
7. J. ONCINA, P. GARCÍA Inferring Regular Languajes in Polinomial Update Time. *Pattern Recognition and Image Analysis. Selected Papers from the IVth Spanish Symposium.* Series in Machine Perception and Artificial Inteligence. Vol. 1. World Scientific. 1992.
8. B. J. OOMMEN, K. ZHANG, W. LEE Numerical Similarity and Dissimilarity Measures Between Two Trees. *IEEE Transactions on Computers.* To be published.
9. B. J. OOMMEN, R. K. S. LOKE On the Recognition of Noisy Subsecuence Trees. SSPR'98.
10. JOHN L. PFALTZ Web Grammars and Picture Description. *Computer Graphics and Image Processing* 1, 193-220. 1972.
11. S. YEBRA Implementación y estudio de diversos algoritmos de Thinning. *Proyecto Final de Carrera E.U.I, Polytechnic University of Valencia.* 1995.