Syntactic Modeling and Recognition of Document Images*

Ignacio Perea and Damián López

Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
Camino de Vera s/n, 46022 Valencia, Spain
{nperea,dlopez}@dsic.upv.es

Abstract. In this work, we propose a new scheme for the recognition of document images under a syntactic approach. We present a new method to model the layout of the document using a tree-like representation of the form. The syntactic representation of the documents are used to infer a tree automaton for each one of the classes involved in the task. An error-correcting analysis of tree languages allows us to carry out the classification. The experimentation carried out showed the good behaviour of the approach: error rate of 1.18%.

1 Introduction

Document processing [1],[20] can be decomposed into two phases, document analysis and document understanding. Where the former attempts to obtain a hierarchical representation of the document that embeds its geometric structure (i.e. [9]), the latter builds the logical structure of the document from the information obtained in the first analysis [4].

Another line of work related to paper documents is *Document Image Recognition*. The techniques applied to this task can be divided into two categories, those based on matching local features (i.e. [2][16]) and those that combine both local features and layout information (i.e. [17][19][21]).

In this work, we face the problem of form classification under a syntactic approach. A new procedure is presented to represent the layout of the document. Briefly, this method measures the importance of each of the components of the image and then represents the relative position of each component using tree-like components [7]. Once the documents are reduced to a tree-like structure, this work uses a tree language inference algorithm [10] which builds a tree automaton for each class of the problem using a minimum edition distance criterion.

In real applications, the documents to be classified usually do not fit exactly in any class. Therefore, it is necessary to obtain the class that is nearest to the document structure. In our work, to obtain that measure, we use an error

* Work partially supported by the Spanish CICYT under contracts TIC2000-1153 and TIC2003-09319-C03-02.

correcting analysis algorithm on tree automata [14]. This scheme and has shown good performance in handwritten digit recognition [11].

This work is structured as follows. First of all, the required notation is presented. Then, the approach to be used is explained: the tree-like representation procedure, the main features of the error-correcting analysis algorithm and the tree language inference method. The experimentation carried out shows the good performance of the approach (1.18% of error rate). Finally, the conclusions and the proposed future lines of work are presented.

2 Notation and Definitions

A *ranked alphabet* is defined as an association of a finite alphabet V with a finite relation $r \subseteq (V \times \mathbb{N})$. Let V_n denote the subset $\{\sigma \in V : (\sigma, n) \in r\}$.

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

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

Note that the set of symbols in V_n are the valid labels of the nodes with arity n ; therefore V_0 is the set of valid leaf labels.

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:

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

A tree $t \subseteq V^T$ is accepted by A if $\delta(t) \in F$. This set of trees accepted by A is defined as $L(A) = \{t \in V^T \mid \delta(t) \in F\}$, this set is also referred to as the *tree language* accepted by A .

3 Document Image Classification

3.1 Tree-Like Representation

In this work, we used the database of structured forms *NIST Special Database 6 (SD6)*[3]. The database contains 12 different tax forms together with different

schedules and form faces. Hence, 20 different form faces (classes) are represented in the database. The images have been automatically derived and synthesized and therefore contain no “real” tax data. Nevertheless, all the images appear to be real hand-printed forms prepared by individuals. There are 900 simulated tax submissions represented in the database averaging 6.22 form faces per submission. All the images are stored in a bitlevel black and white raster format.

Algorithm 3.1 Algorithm to obtain a tree-like model of the document images.

1. Blur the image and reduce it by one eighth. /* result: 256 grey-level image */
 2. Detect the connected components (four connected pixels) of the image.
 3. For each connected component of the image:
 - Reduce it to its smallest rectangle.
 - Reduce each rectangle to its central point and assign a value that is proportional to the area of the rectangle to this point.
 4. Set M to the maximum but two /* to smooth the value */ and m to the minimum values in the image.
 5. Let $I = (M + m)/2$;
 6. Consider the following correspondence between intervals and symbols:
 - Interval $[0, \lfloor I/2 \rfloor]$; symbol ‘A’
 - Interval $[\lfloor I/2 \rfloor, \lfloor I \rfloor]$; symbol ‘B’
 - Interval $[\lfloor I \rfloor, \lfloor (3I)/2 \rfloor]$; symbol ‘C’
 - Interval $[\lfloor (3I)/2 \rfloor, \lfloor (23I)/10 \rfloor]$; symbol ‘D’
 - Interval $[\lfloor (23I)/10 \rfloor, \infty]$; symbol ‘E’
 7. Create the root node of the tree.
 8. Set *node* to the root node of the tree
 9. Divide the image in four quadrants
 10. Create a child node to *node* for each quadrant (from left to right and downwards).
 11. For each node (and its corresponding quadrant):
 - If the quadrant contains more than one point, then label the (internal) node with symbol σ and go to step 9.
 - If the quadrant has no points, then label the node as a leaf using the symbol *empty*.
 - Otherwise, assign a symbol to the node (see step 6).
 12. Return the tree obtained.
-

Quad trees (q-tree) [7] has been broadly used to obtain tree-like representations. These representations have shown to be useful in pattern recognition tasks [10][11]. In order to capture only the more relevant features of the images, the algorithm we propose takes into account a reduced image.

The first step in modelling the layout of the documents was to perform a light blur of the images (we considered 5×5 pixels) and a reduction of the image by one eighth. Then, the resulting 256 grey-level images were processed to detect the connected-components. In this process, only four connected pixels were considered. Each connected-component was then substituted by the minimum rectangle that contains the component.

The image of the document was then simplified to an image where all the rectangles were reduced to a pixel in the middle of the rectangle with a weight that is proportional to its area.

In a standard q-tree representation, the root of the tree is associated to the whole image to model and any other internal node with one of the four quadrants of its parent image. The process is repeated recursively until a node represents a one-colour square, then, the node becomes a leaf labeled with the color of the region. The variation of the q-tree representation used in this work ends the recursion when the quadrant does not contain any point. The region is then labeled with the *empty* leaf. When the region contains only one point, the leaf is marked with a symbol that is proportional to the weight of the point (five different labels were considered). A scheme of the entire process is shown in Algorithm 3.1. An example of the run is shown in Figure 1. Note that no preprocessing of the image (slant correction, rescaled or similar) was performed.

3.2 Error-Correcting Analysis

The treatment of noisy or distorted samples has always been a problem in pattern recognition tasks, these distortions usually do appear as a side effect to the acquisition, preprocessing or primitive extraction phases.

This work makes use of an error-correcting analysis algorithm that was introduced in [14] and that gives a distance between a tree and a tree automaton.

To calculate the distance between a tree t to a tree automaton A , the algorithm explores the tree, and calculates the cost of reducing each subtree of the tree to each state of the automaton. To do this, the algorithm compares the successors of a node with the different transitions of the automaton.

The method visits the tree nodes in postorder. Therefore, when a tree is going to be analyzed, all the distances between its subtrees and the states of the automaton have already been calculated. These distances are stored in a matrix that is indexed by the nodes of the tree and the states of the automaton, thereby avoiding calculations carried out previously (for details see [14]). The whole process is carried out in polynomial time with respect to the size of the tree and the number of states of the automaton. The authors prove that the distance obtained is the minimum one according to the edit operations proposed.

3.3 Tree Language Inference

In our work, we model each class with a tree language. In order to obtain these languages, it is possible to use several algorithms (for instance [5][6][8][13][15]). In this work, we used a tree language inference algorithm that, in each step of the inference process, considers the automaton of the previous step and a new sample of the training set. Then, the inference algorithm performs an error-correcting analysis on the tree automaton. The modification of the automaton comes determined by the editing operations needed to force the automaton to accept the sample.

Fig. 1. Example of the run for Algorithm 3.1. From left to right and downwards: The original image (here reduced eight times), the blurred and reduced image, the smallest rectangles that contain a connected component, the final division in quadrants, and the resulting tree. A more compact notation for the tree is: $\sigma(\sigma(\text{empty}, \sigma(\text{empty}, \text{empty}, \sigma(A, C, \text{empty}, \text{empty})), \text{empty}), \text{empty}, B), \text{empty}, \text{empty}, D)$.

This algorithm has proven to be efficient when the differential features of the representation are based on the presence of substructures, the length of these substructures and their relative position in the sample representation.

There are two reasons why this new approach can be seriously considered for syntactic pattern recognition. On the one hand, the representation power of tree primitives is very high. On the other hand, this algorithm takes advantage of the features (substructures) that are already considered in the automaton, adding the new ones when they are presented in the inference process. As noted before, this approach has been used previously in pattern recognition tasks with good results.

4 Experimentation

Once the tree representation of the database was obtained, a tree automaton was inferred to represent each class. In order to infer these automata, we used the inference algorithm described above.

Fig. 2. Example images of each class in the dataset. Note the slant of the first image.

Once the automata were inferred, we used the error-correcting parser on tree automata to obtain the minimum distance of the test sample for each automaton. The document images were classified into the class with a minimum value criterion. This approach has previously been used in other pattern recognition tasks with good performance (e.g. [12][11]).

Our approach needs some samples from each class in order to train the models. Therefore, the four classes with greatest number of available images were selected from the database. These classes were: Form 1040 page 1, Form 1040 page 2, Form 1040 Schedules A and B. Example images are shown in Figure 2.

Two different experiments were performed. The models of the first experiment were inferred using 200 samples per class. In the second experiment, we used 300 samples per class. The results of the experimentation are shown in Table 1. Note that due to the number of samples available, the number of samples in the test set is not homogeneous.

Table 1. Results of the experimentation. The table on the left shows the confusion matrix when 200 training samples were used. The table on the right shows the confusion matrix when 300 samples were used to infer the automata.

class	0	1	2	3
0	399	1	0	0
1	1	399	0	0
2	1	17	268	4
3	1	0	2	377

class	0	1	2	3
0	398	2	0	0
1	0	399	1	0
2	1	7	178	4
3	0	0	0	280

The results show the excellent performance of our approach, an error rate of 1.84% was obtained when 200 samples were considered to infer the automata, and an error rate of 1.18% was obtained when 300 samples were used.

5 Conclusions

In this work, we tackle the classification of hand-filled forms using a syntactic approach. We propose a new procedure for modelling the layout of the forms which obtains a tree-like representation. This representation allows us to use a tree language inference algorithm to obtain a model for each class of the classification task. An error-correcting analysis is carried out to classify the test samples, taking into account the minimum editing-distance.

We used a subset of the *NIST Special Database 6* of structured forms and our syntactic approach to carry out the classification. The results of the experiments show the excellent behaviour of the approach: an error rate of 1.18%.

As a future line of work, the preprocessing of the image (for instance, applying methods to correct the slant of the image), should improve the results. Note that the approach does not consider the use of probabilities. Another line of work is the modification of the model to obtain a stochastic one which should also lead to better results.

Finally, the representation process could easily be modified for other pattern recognition tasks and should be explored further. Also, other methods for modelling the physical structure of forms [18] could be used in a syntactic approach to classification to produce good results.

References

1. R. Cattoni, T. Coianiz, S. Messelodi, and C.M. Modena. Geometric layout analysis techniques for document image understanding. Technical report, Instituto Trentino di Cultura, 1998.
2. F. Cesarini, M. Gori, S. Marinai, and G. Soda. Informys: a flexible invoice-line form-reader system. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(7):730–745, 1995.
3. D.L. Dimmick and M.D. Garriss. Nist special database 6. structured forms database 2. Technical report, National Institute of Standards and Technology. Advanced Systems Division. Image Recognition Group, 1992.
4. D. Dori, D. Doermann, C. Shin, R. Haralick, I. Phillips, M. Buchman, and D. Ross. *Handbook on Optical Character Recognition and Document Image Analysis. The representation of document structure: a generic object-process analysis*. World Scientific Pub. Co., 1996.
5. H. Fernau. Learning tree languages from text. *LNCS*, 2375:153–168, 2002. 15th Annual Conference on Computational Learning Theory (COLT 2002).
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. Available on: <http://www.dsic.upv.es/users/tlcc/tlcc.html>
7. G. M. Hunter and K. Steiglitz. Operations on images using quad trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(2):145–153, 1979.
8. T. Knuutila. *Advances in Structural and Syntactic Pattern Recognition: Proc. of the International Workshop*, chapter Inference of k -Testable Tree Languages, pages 109–120. World Scientific, 1992.
9. S.W. Lee and D.S. Ryu. Parameter-free geometric document layout analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1240–1256, 2001.
10. D. López and S. España. Error correcting tree language inference. *Pattern Recognition Letters*, 23(1–3):1–12, 2002.
11. D. López. *Inferencia de lenguajes de árboles*. PhD thesis, Departamento de Sistemas Informáticos y Computación. Universidad Politécnica de Valencia, 2003. in spanish.
12. D. López and I. Piñaga. Syntactic pattern recognition by error correcting analysis on tree automata. *LNCS*, 1876:133–142, 2000. Joint IAPR International Workshops SSPR and SPR (S+SSPR 2000).
13. D. López, J. Ruiz, and P. García. *Pattern Recognition and String Matching*, chapter Inference of k -piecewise testable tree languages. Kluwer, 2003.
14. D. López, J. M. Sempere, and P. García. Error correcting analysis for tree languages. *International Journal on Pattern Recognition and Artificial Intelligence*, 14(3):357–368, 2000.
15. E. Mäkinen. On inferring linear single-tree languages. *Information Processing Letters*, 73:1–3, 2000.

16. H. Peng, F. Long, and Z. Chi. Document image recognition based on template matching of component block projections. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(9):1188–1192, 2003.
17. H. Peng, F. Long, Z. Chi, and W-C. Siu. Document image template matching based on component block list. *Pattern Recognition Letters*, 22:1033–1042, 2001.
18. S. Ramdane, B. Taconet, and A. Zahour. Classification of forms with handwritten fields by planar hiddenmarkov models. *Pattern Recognition*, 36:1045–1060, 2003.
19. R. Safari, N. Narasimhamurthi, M. Shridhar, and M. Ahmadi. Document registration using projective geometry. *IEEE Transactions on Image Processing*, 6(9):1337–1341, 1997.
20. Y.Y. Tang, M. Cheriet, J. Liu, J.N. Said, and C. Y. Suen. *Handbook of Pattern Recognition and Computer Vision, Document analysis and recognition by computers*. World Scientific Pub. Co., 1999.
21. T. Watanabe, Q. Luo, and N. Sugie. Layout recognition of multi-kinds of table-form documents. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(4):432–445, 1995.