

A System for Data Extraction from Forms of Known Class

F. Cesarini

M. Gori
G. Soda

S. Marinai

Dipartimento di Sistemi e Informatica - Università di Firenze
Via S.Marta,3 - 50139 Firenze - Italia
tel: +39 55 4796362. fax: +39 55 4796363
e-mail: simone@mcculloch.ing.unifi.it

Abstract

In this paper, we describe a flexible and efficient system for processing forms of a known class. The model is based on attributed relational graphs and the system performs form registration and location of information fields using algorithms based on the hypothesize-and-verify paradigm. A special emphasis has been placed at the low level, where an autoassociator-based connectionist model has exhibited successful results in finding the instruction fields in very noisy forms.

1 Introduction

The task of a form reading system is that of scanning a form, convert it to an electronic format, locate information fields (fields containing the data to be extracted from the form) and convert them to a textual representation.

In order to solve this complex problem, in the literature, several approaches have been proposed to modeling forms. [10] proposed a model based on detecting lines as basic items. The information fields are located giving their position with respect to the lines. A more flexible way to identify an information field is that of describing its position with respect to an instruction field, that describes the content of the information field and helps to understand the form. An example is the system proposed in [5] that deals with fields that are into rectangles. The link between the rectangle containing an instruction field and the rectangle of the corresponding information field is given by the spatial relationship between the two rectangles. Another system [8] simultaneously analyzes the layout and recognizes keywords of text in documents in which spatial information is not sufficient to label the fields. For forms, this system locates information fields recognizing the text of the corresponding instruction field.

Obviously, the sequence of operations needed to analyze a form depends on whether or not the class of the form is known in advance. If the form class is *known*, that is if its layout is exactly defined, for instance by means of a form definition language, then the data extraction can naturally be based on a top-down process. In these systems, the form registration turns out to be the major problem, as we can subse-

quently extract data in the information fields directly in the location specified by the model. In the case of *unknown forms* a bottom-up approach similar to that used traditionally in image processing can be used. An intermediate case is that of forms of *known class*, where, the layout is not fixed but can only be inferred on the basis of the recognition of the instruction fields.

The model we propose lies in this intermediate case where the information fields can not be located neither by relying only on their absolute position on the image, nor by relying only on their relationship with the form lines [5]. Moreover, there are cases in which some fields can be located only after having recognized the corresponding instruction fields. Typical examples of such forms are those delivered by service companies for accounting (see Fig. 1), as well as for collecting data on the use of the service.

A bottom-up approach can be used for processing these documents but, however, this usually gives rises to computationally expensive algorithms, and does not seem the best way to deal with documents in which the structure is known. We suggest using a mixed method in which a top-down approach is switched to bottom-up on the basis of the information on the form class.

The model we propose is based on Attributed Relational Graphs [4] that gives an accurate and flexible description of the form class. Form registration and location of information fields are performed by using algorithms based on the hypothesize-and-verify paradigm [6].

To some extent, the approach we propose is related to the one described in [8], as the information fields can be identified only after having recognized the corresponding instruction field. However, in order to minimize the error during the very critical process of extracting instruction fields, instead of adopting an OCR-based reading system, we suggest using an autoassociator-based connectionist model, that is capable of dealing with different character fonts and high noise.

The paper is organized as follows. In section 2, we describe the class of forms considered by the proposed model by attribute relational graphs, while the algorithms for both form registration and location of

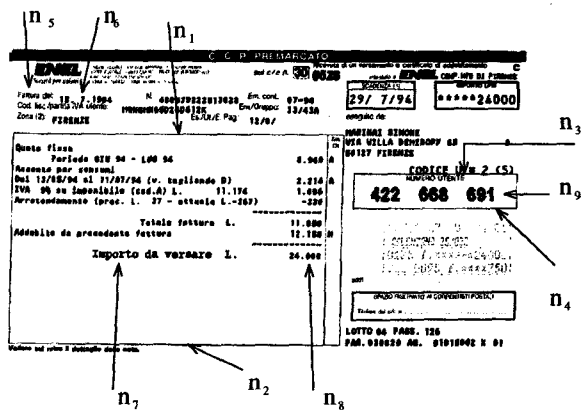


Figure 1: Example of a bill.

information fields are described in section 3. The low level algorithms used for detecting lines and words are described in section 4 and, finally, some conclusions are drawn in section 5.

2 Layout description

An example of the forms considered in this paper is shown in Fig. 1. As pointed out in the introduction, the basic assumption is that the form class is known in advance, though the document usually has no fixed layout.

There are at least three different fields for these forms (the examples refer to Fig. 1).

1) Fields which are in a fixed position with respect to the layout lines. E.g. field n_9 (422668691) is located by its position with respect to line n_3 .

2) The information field is located with respect to the instruction field that is printed in fixed positions. E.g. field n_6 (15.7.1994) is ever located into a region at the right of the instruction field n_5 (*Fattura*).

3) The instruction and the corresponding information fields are not always in the same position in the form, but their mutual position is defined in advance. The information field can be located only after recognizing the corresponding instruction field. E.g. field n_7 (*Importo*) allows to locate n_8 (24.000).

In order to deal with forms having this structure we use a model that is based on ARG (*Attributed Relational Graph* [4]), whose nodes describe objects or parts of objects, while its arcs describe, using numerical attributes, the mutual relationships between the elements represented by the nodes. In our application the objects represented by the nodes are lines, instruction fields and information fields. Each arc represents the mutual position of the elements corresponding to the linked nodes. Some of the nodes and arcs, properly marked by a flag (R), are used to register the position of the incoming form.

Some of the node attributes describe the position where the item corresponding to the node can be found. We have two alternatives to describe this position:

A) Give the coordinates (x_A, y_A, x_B, y_B) of the extreme points of an upright rectangle in which the item described in the node can be found. Varying the size

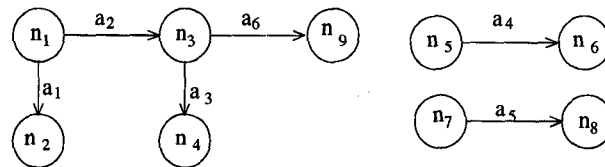


Figure 2: Graph describing the items of interest pointed in Figure 1.

of this rectangle, we can treat elements that are in fixed position into the page (with rectangle's size only slightly wider than the item) and also elements that can be located into an extended region of the form.

B) Define the size of the window where to look for the item, while the mutual position is described by an arc. The location of the window is determined after the form registration, considering its displacement with respect to the first item. For example consider the position of field n_8 with respect to field n_7 in Fig. 5.

Let us now describe the representation of forms by means of an attribute relational graph, a more detailed exposition of the graph can be found in [2].

- The nodes of the graph can be used to describe lines, instruction fields and information fields.

Line nodes (e.g. n_1, n_2, n_3 and n_4 , Fig. 1) contain information on line orientation (only horizontal and vertical lines are considered), on its thickness and length, and on its position, as previously described.

Instruction field nodes (e.g. n_5, n_7 of Fig. 1) contain a pointer to information used for recovering the instruction field¹ and on its position.

Information field nodes (e.g. n_6, n_8 and n_9 of Fig. 1) indicate whether the field is alphanumeric, alphabetic or numeric, the minimum and maximum number of characters that can be found in the field, and the position where the field can be found.

- The graph arcs define the mutual position of two nodes. The attributes describe the length and the orientation of the vector connecting the barycenters of the rectangles surrounding the elements corresponding to the two nodes.

3 Form registration and location of information fields

From a general point of view, the form processing can be viewed as a generalized *graph matching* in which for each node we look for a corresponding portion of form. We could use a bottom-up scheme that begins extracting all the items of the incoming form,

¹In practice, in our system, this index refers to the weights of an autoassociator-based connectionist model that is used for extracting the associated instruction field.

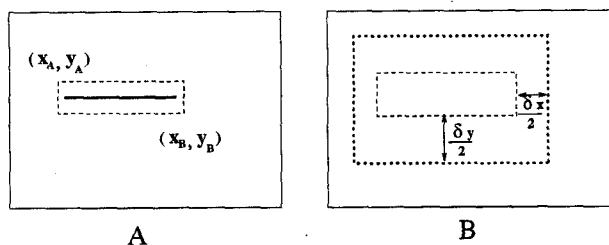


Figure 3: A) Region defining the position of a line in the reference model. B) The extended region for searching the line in the incoming form.

constructs an associated graph specifying mutual relationships between nodes and arcs and, finally, matches the created graph with the model of the class. This approach is likely to be very expensive from a computational point of view. In order to face this problem, in our approach the search of nodes is typically constrained in small windows. The system is based on some procedures that locate lines, instruction fields, and information fields belonging to upright rectangles. Each procedure gets the coordinates of the rectangle and the characteristics of the object to be found as input and provides the coordinates of the eventually found element. The procedure used to locate the instruction fields is described in detail in section 4.

The process for finding correspondences between nodes in the model and items into the image is divided in two consecutive steps.

Form registration

In this step, we register the form by looking for nodes and arcs of the graph with the R attribute. In so doing, we can determine an optimal roto-translational transformation that allows us to overlap the base layout of the form model with the incoming form.

Location of information fields

In this step, we locate the information fields by using the transformation found by form registration.

3.1 Form registration

A typical feature of the documents acquired by a scanner is the difference in rotation, *skew*, and position between the reference model and the incoming form. Many methods have been proposed in the literature for the skew evaluation [9]. A common feature of all these methods is that they require processing all the image in order to evaluate the incoming form position. In many applications, however, the form registration can be obtained simply by locating some specific layout items or the mutual position of other items. The problem with such approach may be the low flexibility arising when dealing with forms of the kind considered in this paper.

Our research is based on the assumption of dealing with forms of a given class that, however, has a flexible structure. The registration of such forms is obtained by defining some registration landmarks, that can be both lines and keywords (considered in the model as instruction fields) implemented by nodes with the R attribute. The mutual position between landmarks is

implemented by arcs with the R attribute. Moreover, in the reference model of the form we define an error margin $(\delta x, \delta y)$ that specifies the maximum allowable displacement for an image of an incoming form. This value is taken into account to deal with the skew and the translation of the form.

The process that registers the forms acts by using a variation of the hypothesize-and-verify algorithm [6].

The basic idea of the algorithm is the following. The hypothesize phase begins by looking for references in the incoming form that can be used for computing a roto-translational transformation aimed at performing an alignment with respect to the reference model of the form. Lines and also vectors associated with arcs of attributed relational graphs can conveniently be exploited as a reference for this matching process. In both cases we evaluate a roto-translation transformation that we hypothesize to be the actual transformation of the incoming form.

The verify phase searches items corresponding to all the registration nodes. In case of failure the hypothesis is wrong and the process continues by testing another hypothesis.

Hypothesis generation

The attribute relational graph defining the form has a special item, referred to as *first matching item* that is used for firing the hypothesize step². The search in the incoming form is carried out by using proper low level algorithms in the region defined by the *first matching item*. For each registration node, the model defines in fact a region where to look for geometric entities described in the node. This region is an upright rectangle centered on the item. Combining the rectangle coordinates and the error margins $(\delta x, \delta y)$ we obtain the region where to look for the element corresponding to the *first matching item* (Fig. 3).

If the *first matching item* is a line, then we simply match this item with a line found in the incoming form in order to evaluate the hypothesized registering transformation. This transformation can be viewed as an hypothesis on the actual transformation required for aligning the model and the incoming form.

In the case in which the *first matching item* is an arc then the alignment is carried out similarly, the only difference being that the reference line used for the registration is now derived from the vector associated with the arc. In both the cases we can easily find the coefficients of the hypothesized transformation.

Hypothesis verification

After finding the transformation described in the previous subsection, we use it to find the other nodes and arcs of the graph. This search is aimed at verifying whether or not the hypothesized transformation is

²This item can be a line node or a graph arc (convenient particularly in the case of forms with no lines). In this latter case the alignment process could take place considering a vector connecting the barycentres of two instruction fields.

correct. Of course, there are two possibilities depending on the result of the verify step.

1) If the hypothesized transformation generates a match for all the remaining items with the registration flag R , then the verify step is successful and the hypothesized transformation can be used for locating the information fields.

2) If, on the contrary, the matching fails for at least one item with the registration flag R , then the verify step fails. In this case, the search for the *first matching item* continues in the specified area of the incoming form until a new candidate item is found from which the hypothesize step is fired again.

If no successful result is found then we conclude that the incoming form has an excessive displacement or that it does not belong to the expected class.

3.2 Location of information fields

As shown in the previous subsection, the task of form registration is that of finding a proper transformation performing the alignment of the incoming form with the reference model. This transformation turns out to be very useful for locating the instruction fields or reference lines that, in turn, make the access to the information fields possible. The relationship between instruction and information fields has been specifically addressed in section 2 by the introduction of the attribute relational graph.

In the following, we give some details concerning the specific operations required to locate an information field (see Fig. 5).

1) Using the transformation found during the form registration, each instruction field turns out to be constrained into a rectangle. Similarly, lines that are used for referring to information fields are detected by a rectangle. The graph associated with the form makes it possible to establish whether the information field is obtained from reference lines or instruction fields.

2) Low level procedures are called in order to give a precise detection of reference lines or instruction fields.

3) The precise location of reference lines or instruction fields together with the graph makes it possible to determine the region the information field belongs to.

4) The image associated with the information field is used for the actual recognition process.

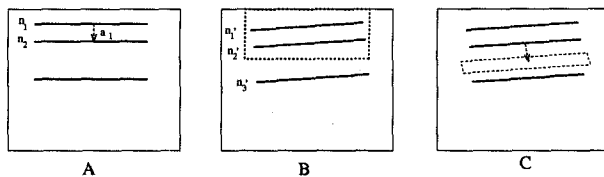


Figure 4: The registration algorithm. A) Form model with two line nodes (n_1 , n_2). B) A skewed incoming form, with the region where to look for the line corresponding to n_1 . C) A wrong hypothesis: the system hypothesizes that n'_2 corresponds with n_1 .

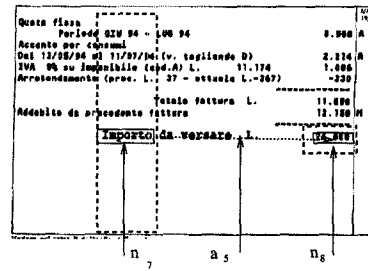


Figure 5: Location of information fields. Dashed rectangles describe regions where to search for the fields.

Notice that, in practice, the location of instruction fields depends significantly on the dimensions of the rectangle where to look for. Examples can be found where such location is very straightforward, as the form model offers precise information on the position of the field. On the contrary, in other cases, locating the instruction field is significantly more expensive, because of the "large" area, specified by the graph, where a low level recognition algorithm must look for. However, looking in large areas gives the model a higher flexibility and, in many cases, the computational burden may be affordable.

4 A connectionist-based approach for low level form processing

In this section, we give a brief description of the low level form processing that is currently implemented in a hybrid way by using both conventional and connectionist approaches. At this level we are interested in locating lines, instruction and information fields. Moreover, in addition to field location, we also ask this level to recognize the content of the instruction fields. This turns out to be very useful in the case of highly noisy forms and unknown character fonts. Unlike information fields, for which no reasonable bound can usually be given on the dimension of the dictionary of words, most of the times, the instruction fields are composed of words of a small dictionary. This hypothesis can conveniently be exploited in order to optimize the recognition of the instruction fields, where eventual errors turn out to be very critical.

Although at this level recognition can be entirely based on connectionist models, in our prototype, we locate lines by using RLSA (Run Length Smoothing Algorithm [9]). The particular connectionist approach suggested in the following for the recognition of instruction fields could be used also for other low level tasks but, so far, we have no experimental arguments for evaluating its effectiveness.

The word recognition has been extensively discussed in the literature with different approaches (see e.g.: [7]). Basically, different classes of algorithms arise depending on whether or not the word is segmented in single characters. The segmentation in characters turns out to be the only significant way of recognizing words of a very large dictionary, whereas very effective

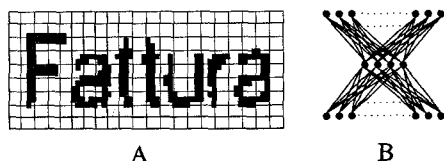


Figure 6: Input grid used to recognize a word and the corresponding autoassociator-based connectionist architecture adopted for the recognition process.

algorithms can be used for the recognition of words of small dictionaries when using the whole word as input to the classifier.

In the following we briefly describe the method proposed, a more detailed description can be found in [3]. The image is preprocessed in order to provide a simple scaling of the pixel representation aimed at matching the number of inputs assumed in the neural classifier. That number, as well as the other dimensions of the classifier, is established on the basis of a joint evaluation of convergence and generalization to new examples simply by trial and error. The neural classifier has an autoassociator-based architecture, where the input and the output layer have the same number of units (see Fig 6B). An hidden layer provides a compressed and nonlinear representation of the input information. The learning process consists of forcing the network to reproduce the input patterns to the outputs. Basically, for each word, an autoassociator is created where several instances of the same word are used for training. In particular, these instances may be associated with different fonts and, mainly, with different degree of noise. The learning set was created by using both actual and artificial images, where the noise was properly simulated [3]. Unlike more typical neural classifiers based on multilayered architecture, the use of the autoassociator mode, where the input is forced to be equal to the output, has the significant advantage of providing a reliable criterion for word rejecting [1], that is very important for the whole form reading process.

The low level recognition is an hybrid process that is based on the following scheme.

1) Locate the words in the image.

This is carried out by a run-length filtering followed by a connected components evaluation.

2) Autoassociator-based word recognition .

The incoming image obtained at the previous step is used for feeding the autoassociator models of the words that can be expected in the instruction field (Fig. 6A). For each network we can determine the degree of matching with the pattern obtained from the previous step by computing simply the distance from inputs and outputs. The lower the distance the higher is the degree of matching³.

A more detailed description of this approach to

³A theoretical foundation of this scheme can be found in [1]

word recognition is given in [3].

5 Conclusions

In this paper, we have described a flexible and efficient system for processing forms of a known class. The model, based on attributed relational graphs, performs form registration using algorithms based on the hypothesize-and-verify paradigm. The low level interaction has been based on a hybrid approach. In particular, autoassociator-based connectionist models have been used for recognizing the instruction fields with very successful results.

The proposed system requires still a massive experimental investigation of the interaction of the different modules, as well as an actual experimentation with many different kinds of forms, in order to assess the claimed flexibility.

REFERENCES

- [1] M. Bianchini, P. Frasconi and M. Gori, "Learning in Multilayered Networks Used as Autoassociators", To appear in *IEEE Transaction on Neural Networks*.
- [2] F. Cesarini, M. Gori, S. Marinai, G. Soda, "A System for Data Extraction from Forms of Known Class", *Technical Report DSI, 95-01*.
- [3] F. Cesarini, M. Gori, S. Marinai, G. Soda, "A Hybrid System for Locating Low Level Graphic Items", To appear in *Proc. of IAPR Workshop on Graphic Recognition, 1995*.
- [4] M.A. Eshera and K.S.Fu, "An Image Understanding System using Attributed Symbolic Representation and Inexact Graph-matching" *IEEE Transaction on PAMI*, Vol. 8, pp. 604-617, 1986.
- [5] H. Fujisawa, Y. Nakano, K. Kurino, "Segmentation Methods for Character Recognition: From Segmentation to Document Structure Analysis", *Proc. of the IEEE*, Vol. 80, pp. 1079-1092, 1992.
- [6] W.E.L. Grimson, *Object Recognition by Computer, the Role of Geometric Constraints*, MIT Press, 1990.
- [7] T.K. Ho, J.J. Hull, S.N. Srihari, "A computational Model for Recognition of Multifont Word Images", *Machine Vision and Applications*, No 6, pp. 157-168, 1993.
- [8] S.W. Lam, L. Javanbakht, S.N. Srihari, "Anatomy of a Form Reader" *Proc. of ICDAR 93*, pp. 506-509, 1993.
- [9] Y.Y. Tang, C.De Yan, C.Y. Suen, "Document Processing for Automatic Knowledge Acquisition" *IEEE Transaction on Knowledge and Data Engineering*, Vol. 6, pp. 3-20, 1993.
- [10] C.D. Yan, Y.Y. Tang, C.Y. Suen, "Form Understanding System Based on Form Description Language" *Proc. of ICDAR 91*, pp. 283-293, 1991.