MULTISTRATEGY LEARNING FOR DOCUMENT RECOGNITION

FLORIANA ESPOSITO, DONATO MALERBA, and GIOVANNI SEMERARO
Dipartimento di Informatica, Università degli Studi di Bari, Bari, Italy

In this paper, a methodology for document classification and understanding is proposed. It is based on a multistrategy approach to learning from examples. By document classification, we mean the process of identification of the particular class to which a document belongs. Document understanding is defined as the process of detecting the logical structure of a document. The multistrategy approach for document classification and understanding has been implemented in a system called PLRS, which embeds two empirical learning systems: RES and INDUBI/H. Given a set of documents whose layout structure has already been detected and such that the membership class has been defined by the user, RES generates the knowledge base of an expert system devoted to the classification of a document. The language used to describe both the layout of the training documents and the learned rules is a first-order language. The learning methodology adopted for the problem of learning classification rules integrates both a parametric and a conceptual learning method. As to the problem of document understanding, INDUBI/H can be used to generate the recognition rules, provided that the user is able to supply examples of the logical structure. RES and INDUBI/H are implemented in C language. PLRS is a module of IBIsys, a software environment for office automation distributed by Olivetti.

Processing paper documents is currently one of the important tasks in office automation. It involves much more than just a simple acquisition of a paper document by means of a scanner. Generally speaking, a paper document is a collection of printed objects (characters, columns, paragraphs, titles, figures, and so on), each of which needs to be detected and then processed in the most appropriate way. Therefore, the main objective of a document processing system is the transformation of a digitized document into a collection of information to be stored, classified, retrieved, combined, and updated. In order to do all that, the structures of a document must first be defined.

According to the office document architecture/office document interchange formats (ODA/ODIF) standard (Horak, 1985), any document is characterized by

Address correspondence to Floriana Esposito, Dipartimento di Informatica, Università degli Studi di Bari, via Orabona 4, 70126 Bari, Italy.

The authors are grateful to the staff of Olivetti Systems and Networks, and Syntax Sistemi Software for their valuable collaboration. In particular, they wish to thank Enrico Annese for having made available the set of documents used in their experiments, and Carlo Cito and Giuseppe Cannillo for their technical support during the experiments.

Applied Artificial Intelligence, 8:33–84, 1994

0883-9514/94 $10.00 + .00

two different document structures, representing both its content and its internal organization: the *geometric* (or layout) structure and the *logical* structure.

The geometric structure is the result of repeatedly dividing the contents of a document into increasingly smaller parts, on the basis of the *presentation* (Tang et al., 1991). Generally, this structure associates the contents of the document with a *hierarchy* of *layout objects*, such as text lines, vertical/horizontal lines, graphic elements, photographic elements, columns, and pages (see Figure 1). The leaves of a *layout tree*, representing a hierarchical geometrical structure, are called *basic objects* (or *basic blocks*), and they typically correspond to rectangular areas that delimit portions of contents on the presentation medium. All the internal nodes of

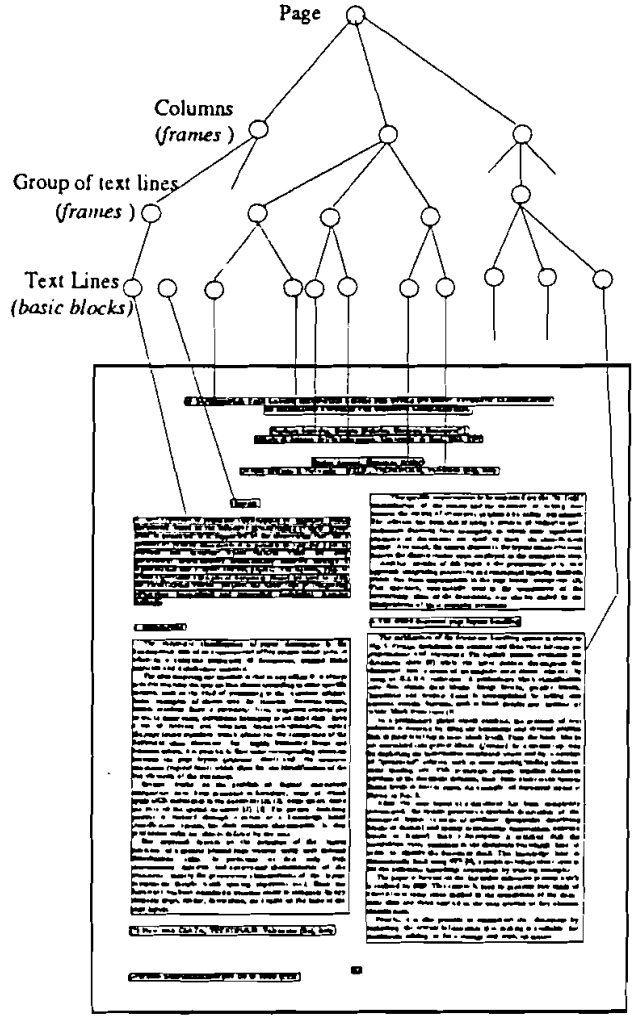


Figure 1. The hierarchical layout structure of a document is the result of repeatedly dividing the content of a document into increasingly smaller parts, on the basis of the presentation.

the layout tree are named *composite objects*, since they are obtained by grouping together basic blocks as well as composite objects of lower level. The root of the layout tree represents the whole document. In multipage documents, such as books, a composite object can represent a set of pages, where a *page* is a rectangular area that corresponds to a unit of the presentation medium. All other internal blocks are named *frames* and correspond to rectangular areas within a page. Henceforth, we will consider only single-page documents; thus, the root of the layout tree will always be a page.

The logical structure is the result of repeatedly dividing the content of a document into increasingly smaller parts, on the basis of the *human-perceptible* meaning of the content. Generally, this structure associates the contents of the document with a hierarchy of logical objects, such as title, abstract, paragraphs, sections, chapters, tables, figures, and footnotes (see Figure 2). Even for the logical structure, it is possible to distinguish *basic logical objects*, which appear at the bottom of the logical tree, from the *composite logical objects*, which are represented as internal nodes of the logical tree. Obviously, the types of logical objects in a document are strongly application dependent; thus, for a journal it is possible to define title, abstract, subtitle, paragraph, header, footnote, page number and caption (Tsujimoto & Asada, 1990), while for a letter it is sensible to look for *sender, receiver, date, logotype, reference number, body,* and *signature*.

Both layout and logical objects can be described by a set of *attributes*. For instance, layout objects can be characterized by

- the type of enclosed content (text, picture, etc.)
- their absolute position on the page according to an orthogonal coordinate system
- their shape (if not always rectangular)
- their dimensions
- numerical properties of their bitmaps

while logical objects can be described by

- the type (abstract, paragraph, etc.)
- some keywords contained in the text (date, figure, etc.)
- formatting properties (spacing, indentation, etc.)

Relationships among different objects are also possible. Of course, the hierarchy in the layout/logical structures defines some hierarchical relationships among objects of the same structure. However, other more interesting relationships exist among layout objects (*layout-layout* relationships) and among logical objects (*logical-logical* relationships). An example of a layout-layout relationship is the mutual position of two layout objects, while the cross reference of a caption to a figure or

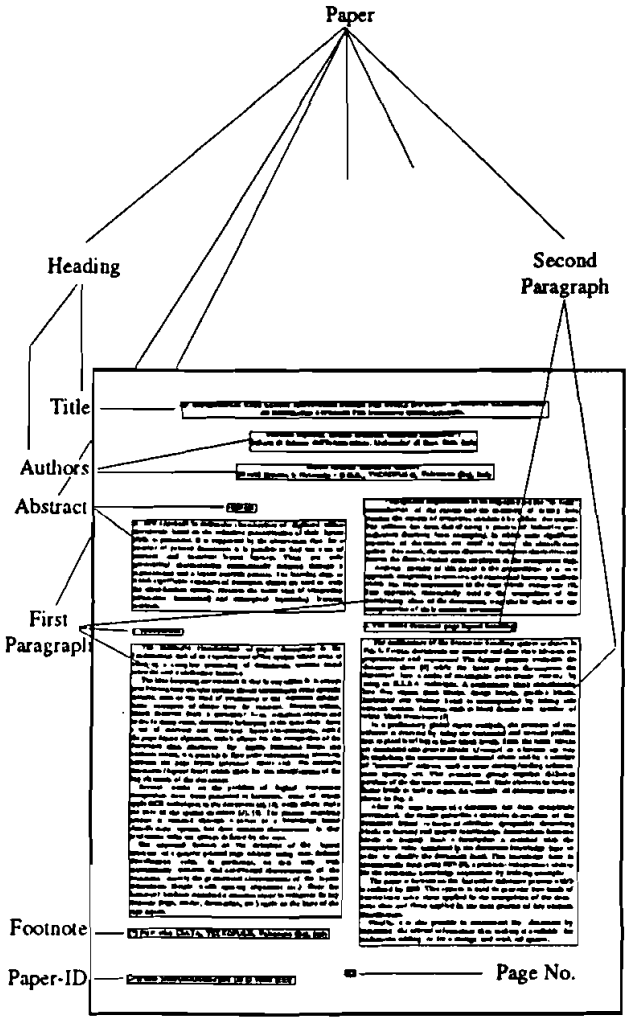


Figure 2. The hierarchical logical structure of a document is obtained by repeatedly dividing the content of a document into increasingly smaller parts, on the basis of the human-perceptible meaning.

the reading order of some parts of a document are two examples of logical-logical relationships.

Finally, *layout-logical* relationships between one or more elements of the layout hierarchy and one element of the logical hierarchy can be defined. These relationships are the most interesting for our task. Indeed, they allow us to identify some logical components of a document without necessarily reading its content by means of an optical character recognizer (OCR) but using only layout characteristics. For instance, in a standard English letter, the date is under the sender's address, which is in turn, in the top right-hand corner. Thus, this simple layout information can be

profitably exploited by a document management system to reconstruct the logical structure of a document, starting from its layout structure. Furthermore, the distinction between text and picture regions is also important in order to limit the application of the OCR, so that only information useful for storing and retrieval purposes is read.

In literature the process of breaking down a document image into several layout components (blocks), without any knowledge regarding the specific format, is called *document analysis* (Tsujimoto & Asada, 1990). The document analysis process produces the layout tree of a document. On the contrary, the term *document understanding* denotes the process of identifying the logical structure of a document.

Several approaches to the problem of document analysis have been presented in the literature. They can be classified as *top-down* and *bottom-up* (Srihari, 1987; Nagy et al., 1988). Top-down (or model-driven) approaches first divide the document into major regions, which are further divided into subregions, while bottom-up approaches first extract individual connected components and then group them together. The run length smoothing algorithm (RLSA) (Wong et al., 1982) and the projection profile cuts (Nagy et al., 1986) are two examples of top-down approaches, while the neighborhood line density method (Kubota et al., 1984) and the connected component analysis (Fletcher & Kasturi, 1988) follow the bottom-up approach. In some cases, document analysis is performed by a rule-based system, where layout and composition rules are defined for specific classes of documents (Nagy et al., 1986). In other cases, a rule-based system is used in order to discriminate text from nontext regions in a segmented document (Fisher et al., 1990).

As to document understanding systems, they can be characterized to a certain extent by the degree of specialized domain knowledge used. Indeed, some systems apply an OCR to the document in order to recognize the logical components by means of keywords (Eirund & Kreplin, 1988; Pagurek et al., 1990), while others make use only of the spatial structure (Dengel & Barth, 1988; Nagy et al., 1986). However, a characteristic shared by all such systems is that production/grammar rules are always defined by an expert of the application domain. This means that approaches proposed so far analyze succesfully only those documents having a previously defined structure and they fail when a new kind of form has to be analyzed and understood.

On the contrary, our approach to the problem aims at a complete automatization of the different processes, so that no prior knowledge on the specific format of a document should be provided by the user. Such a requirement leads to a great flexibility of the document processing system, which should be customized quickly and easily, but it forces us to distinguish clearly between the two problems of document analysis and document understanding.

As to document analysis, we use both a top-down approach for segmenting a page into basic blocks, namely, the RLSA (Wong et al., 1982), and a bottom-up approach for grouping layout blocks into higher level frames (Ciardiello et al., 1988).

Heuristic knowledge on the structure of a printed document is employed in the latter step; for instance, "long text blocks with the same spacing can be grouped together" or "near blocks of the same type can be grouped together."

As to document understanding, we use a rule-based approach in order to map the geometrical structure of a document into its logical structure. Our approach does not require that the user define the rules of a knowledge-based system but simply that he is able to provide the system with some *examples* of documents whose logical structures have been previously defined. Thus, the problem of understanding a document can be cast as a problem of *inductive generalization* (learning from examples), since the rules for the recognition of logical objects can be inferred. It is worthwhile to note that, in this way, we deal only with *automatically* detected and constructed characteristics of the document, namely, the geometrical characteristics of the layout objects (height, width, spacing, alignment, etc.).

Nevertheless, when several kinds of documents have to be automatically handled, document understanding becomes a difficult process due to the different layout-logical relationships met in each kind of document. For instance, letters from various companies will present different writing standards, so the identification of the sender or receiver would be hard if only layout information were used. Thus, an intermediate step becomes necessary: *document classification,* that is, the identification of the particular class the document belongs to. The idea is that in any office it is always possible to group documents into classes according to some specific criteria, such as the kind of processing or the common subject. Some examples of classes may be invoices, business letters, letters received from a particular firm, and magazine indexes. In many cases, documents belonging to the same class have a set of relevant and invariant layout characteristics, called *page layout signature,* which allow for recognizing the document class. Our approach is based on the idea that humans are generally able to classify office documents (invoices, letters, order forms, papers, indexes, etc.) from a perceptive point of view, by recognizing the structure of a form or by reading only the content of particular parts of the document. Once again, we expect that the user is able to provide some examples of documents whose class has already been established (such documents are the same used for the document understanding problem). Thus, it is possible to discover the layout similarities and to derive the discrimination rules employed in the recognition step by means of a process of inductive generalization.

To sum up, in order to automatically recognize a document, that is, to classify and understand it, three processing steps are necessary: first document analysis, then document classification, and finally, document understanding. All these steps raise questions that can be partially solved by using AI techniques. A novelty of our approach is the application of inductive learning algorithms to both document classification and document understanding in order to get a document processing system that can be easily customized. The methods have been implemented and tested in an application for automated recognition of paper documents using layout

information developed in the workpackage AP (application for automatic classification of documents) of the INTREPID[1] (innovative techniques for recognition and processing of documents) Project (1993). Indeed, the main activity in the workpackage concerns the automated acquisition of the knowledge base of an expert system devoted to the recognition of office documents, namely, the recognition of documents as a whole (document classification) and the recognition of logical objects within a given document (document understanding) based upon the layout structure alone.

The architecture of the implemented system, named PLRS (page layout recognition system), is shown in Figure 3. After having *scanned* a document, the system detects a possible skew in the document image and corrects it by means of a rotation. The bitmap of the deskewed document is then *segmented* by using the RLSA. In a preliminary global *layout analysis*, the presence of text columns is detected by using the horizontal and vertical profiles. Then, the basic blocks are assembled into greater blocks (frames) in a bottom-up way, by exploiting information on text columns and by using a number of perceptive criteria, such as same starting/ending columns and same spacing. This procedure groups together different portions of the document contents, from basic elements to various frame levels, as well as pages. When the page layout of a document has been completely determined, the system generates a numeric/symbolic description of the document layout. The descriptions of the training documents are exploited by a learning system to generate recognition rules both for document classification and document understanding. Such rules constitute the knowledge base of an expert system devoted to the classification and subsequent understanding of a new (test) document. Finally, the document can be reconstructed by exploiting textual information read by the OCR in those logical objects that are of interest for the application. Since textual, graphic, and layout information is properly managed, the document can be efficiently stored and retrieved.

In the next section we will present in detail the document analysis performed by PLRS, that is, the preprocessing phase that precedes the learning steps. In the third section, we will describe a hybrid inductive learning methodology for document classification that integrates both a parametric and a conceptual learning method. We will also present a solution to the problem of classifying documents when their layout descriptions are noisy. It is based on a distance measure between structural symbolic descriptions that allows us to realize a form of flexible matching. The third section is completed by some experimental results on large sets of real office documents. The problem of learning rules for document understanding, which is presented in the fourth section, is more complex than the problem of learning

[1]The work in the INTREPID project is done within the framework of the ESPRIT program and partly funded by the Commission of the European Community. The following partners form the consortium: AEG Electrocom (D), CTA (E), Nottingham Polytechnic (GB), Olivetti Systems & Networks (I), Pacer Systems Ltd. (GB), University of Bari (I), University of Koblenz (D), and University of Naples (I).

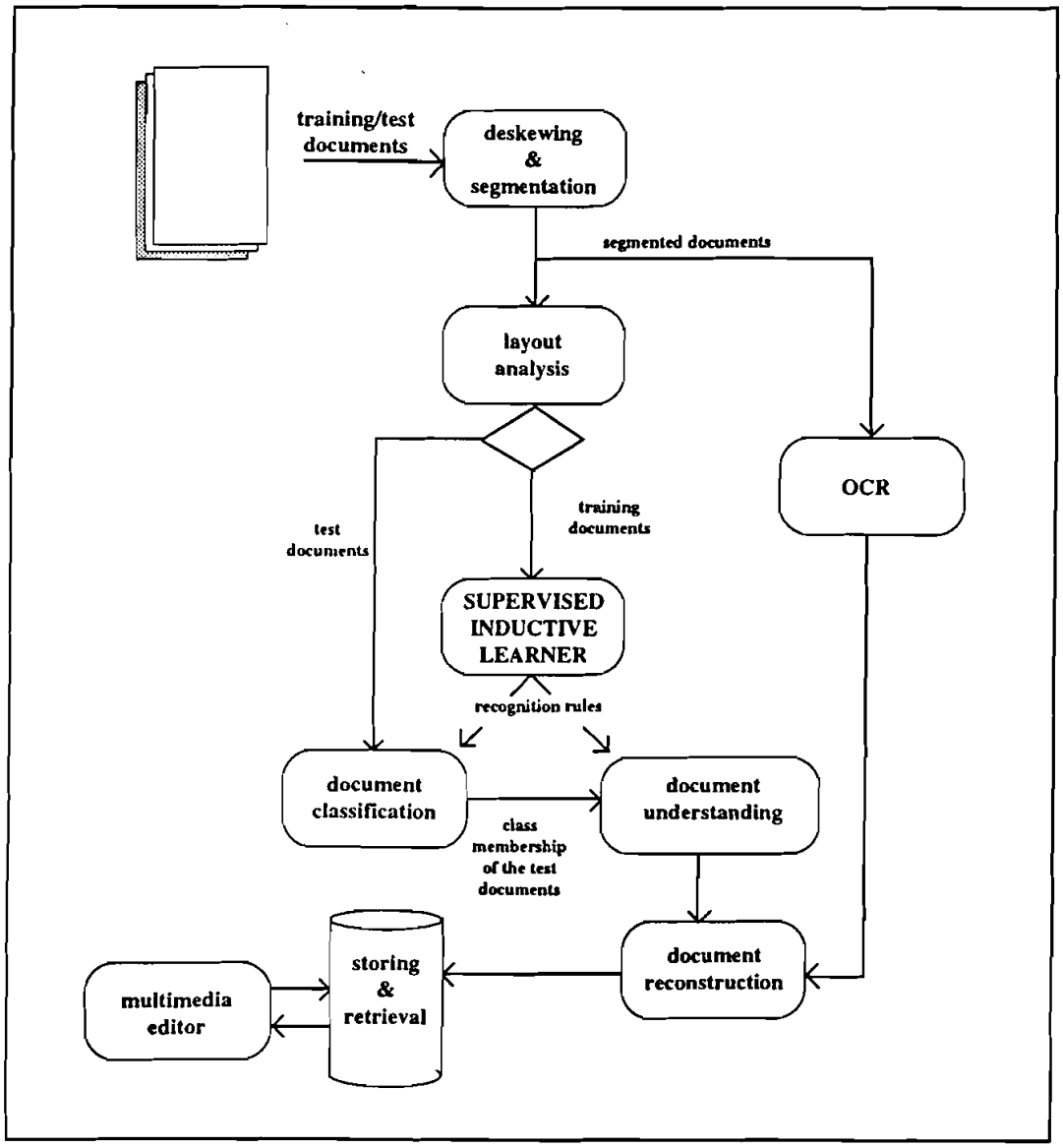


Figure 3. Architecture of PLRS. After having scanned a document, the system detects a possible skew in the document image and corrects it by means of a rotation. Then the document image is segmented into blocks, which are grouped together by the layout analysis. The supervised inductive learner produces rules useful for both document classification and document understanding. Finally, the document is reconstructed by exploiting information read by the OCR.

recognition rules for classifying documents. The main difficulty is that the concepts to be learned refer to a part of a document rather than to the whole document, and since parts of a document may be related to each other according to logical-logical relationships, this leads to the problem of learning *contextual rules*. In the fourth section, we hypothesize that contextual rules have a higher predictive accuracy than rules learned by assuming concept independence. Such a hypothesis is empirically evaluated by means of experiments on a set of letters belonging to the same class. Positive results confirm that learning contextual rules is a better approximation of reality.

DOCUMENT ANALYSIS

The first step of a document processing system is that of obtaining an electronic representation of the printed page by means of a scanner. In order to do that, we have to define two parameters: the resolution, expressed in dots per inch (dpi), and the amount of information for each dot. PLRS acquires black-and-white bitmaps of documents with a resolution of 300 dpi (Figure 4a). This means that each dot is represented by a single bit, so a bitmap of an A4 document takes 2,496 × 3,500 = 1,092,000 bytes when stored.

After having scanned a document, its bitmap is enhanced in order to reduce the noise level or in order to correct some flaws that may affect the subsequent processing steps. In particular, PLRS detects and corrects the skew of a document, since the RLSA algorithm used to segment the document is generally ineffective when applied to skewed documents. Deskewing is accomplished by evaluating a horizontal projection profile (Figure 4b), and then by selecting sample regions for computing the average density of black pixels per row (Figure 4c). The sample

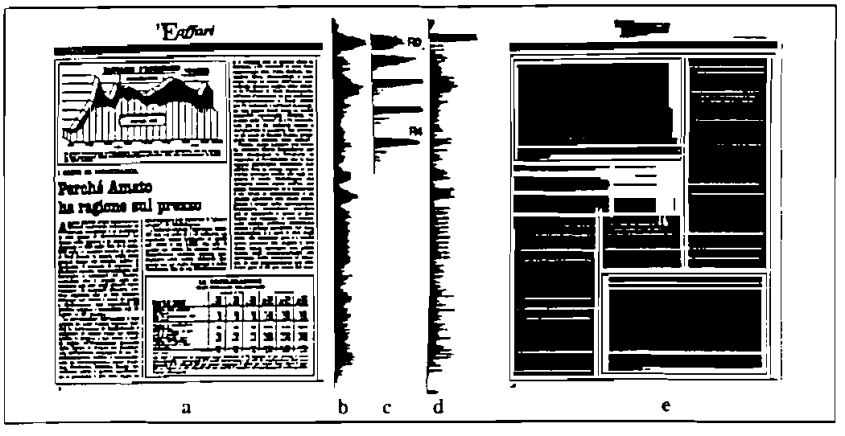


Figure 4. (a) Scanned document, (b) horizontal histogram, (c) sample region R0 and its successive rotations R1, R2, R3, R4, where the last three determine the points of the interpolation, (d) the horizontal histogram after the rotation, (e) result of the smoothing process.

region is rotated by an angle of 0.7°, and the relative histogram is reevaluated until the mean-square deviation of the histogram is maximized. The actual document skew is determined by evaluating the vertex of the parabola obtained by the interpolation of the last three points in the space rotation-angle/mean-square deviation. If required, the document is rotated (Ciardiello et al., 1988) (Figure 4d).

Even though the skew detection and correction are accomplished on the original bitmap, it is not necessary to deal with so much detailed information for the subsequent phase of segmentation. Therefore, the document image is reduced to a resolution of 75 dpi, which allows the RLSA to work more efficiently. The basic RLSA was developed by Wong, Casey and Whal and is described by Wong et al. (1982). It is applied to a binary sequence of black (represented by 1) and white (represented by 0) pixels, and transforms an input sequence **x** into an output sequence **y** according to the following rules:

(1) The 0s in **x** are changed into 1s in **y** if the number of adjacent 0s is less than or equal to a constant C.

(2) The 1s in **x** are unchanged in **y**.

In practice, the RLSA has the effect of linking together neighboring black areas that are separated by less than C pixels. Obviously, the choice of C is crucial in order to get areas including a common data type (text or graphic). The transformation process presented above is applied to both the horizontal and the vertical sequences, yielding two distinct bitmaps, which are then ANDed together. After an additional horizontal smoothing process (Figure 4e), we get the segmented document. Each block is described by a set of features such as

- total number of black pixels in the block after the AND operation
- total number of black pixels in the original bitmap delimited by the block
- number of horizontal white-black transitions in the original bitmap delimited by the block
- eccentricity
- mean length of the black runs.

Such features, which allow us to cluster blocks into classes, are used to classify each block into one of the following classes: *text, horizontal line, vertical line, graphic,* and *picture* (halftone image). The classifier is a linear parametric classifier, whose coefficients were defined by Wong et al. (1982) after having considered a variety of documents.

Information on the segments of the page layout can be effectively exploited by an OCR, since text and graphics are now separated, but it is too detailed for the tasks of document classification and understanding. In this case, it is necessary to locate columns, paragraphs, words, titles, and captions, which are then useful for the

reconstruction of the document (O'Gorman & Kasturi, 1992). Such a process is called *layout analysis*. PLRS detects the layout of a page by

(1) analyzing the projection profile cuts
(2) grouping blocks and frames according to general heuristic criteria

Basically, the first step is accomplished by creating two arrays, one for the horizontal profile and the other for the vertical profile. Each contains the number of black pixels of text blocks that fall in a point of the horizontal/vertical projection axis (in a reduced document there are 640 points on the horizontal axis and 875 points on the vertical axis). The graphical representation of an array, by means of histograms, is a waveform whose deep valleys correspond to the blank areas of the document along one of the projection axes. The deep valleys with a base larger than a fixed threshold can be cut, thus indicating two parallel edges of a possible frame. At the end of the process, we obtain information on which columns and paragraphs are present in the document.

The second step is, in turn, subdivided into three distinct phases, according to the kind of heuristic criteria adopted for grouping blocks:

(2.1) Adjacent text blocks with almost the same height and belonging to the same column are grouped together (in this way, we group fragments of a text line in a column).

(2.2) A group of text blocks in the same column, with the same spacing and length, forms a frame corresponding to a paragraph. Paragraphs are grouped together if they are in the same column.

(2.3) A block/frame B2 is grouped with a block/frame B1 if either B2 is just below B1, or B2 is aligned with B1, or B2 has almost the same width as B1, or B2 partially overlaps B1.

As to the type of a composite layout object, if the object is obtained by grouping together simpler objects of the same type, then it inherits the type of the composing objects; otherwise, it is defined as *mixture* type. Figure 5 illustrates the result of the segmentation process and layout detection of the cover page of an issue of *IEEE Transactions on Computers*.

The final output of the document analysis process is an ASCII file containing three tables: *BasicBlocks*, *Frames1*, and *Frames2*. Each entry in the tables describes a layout object by means of two couples of coordinates: those of the top left-hand corner and those of the bottom right-hand corner. For each logical block, the type code is also reported (1 for text, 2 for horizontal line, 3 for picture, 4 for vertical line, 5 for graphics, and 6 for mixture). There are two levels of frames, which are described in distinct tables. For each frame the list is provided of layout components that have been grouped together to obtain the frame, while for each basic block a

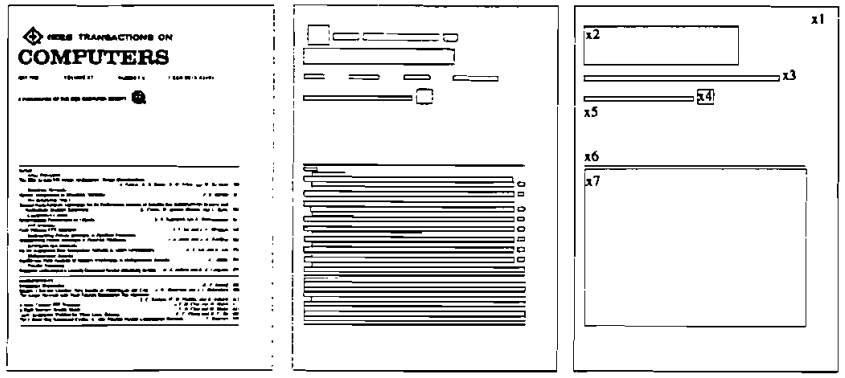


Figure 5. Sample document: (a) 75 dpi bitmap, (b) segmented document, (c) final layout.

set is reported of numerical features, which are useful to derive the statistical descriptors (as illustrated in the next section).

DOCUMENT CLASSIFICATION

As already pointed out in the introduction, document classification aims at identifying the membership class of a document, provided that the user is able to define a set of classes that are relevant for her/his specific application. We approach such a classification task by means of machine learning techniques in order to directly acquire classification rules from a set of training documents. The learning system, devoted to the subtask of document classification and embedded in the supervised inductive learner of PLRS, is called RES (Esposito, 1990). It implements a hybrid approach to learning from examples. More precisely, RES integrates a parametric method for linear classification [Fisher's (1936) discriminant functions] with a conceptual learning method [hypothesis generation and test by means of the STAR methodology (Michalski, 1983)]. The integration aims at obtaining a more powerful and efficient learning system, by combining the advantages of both methods. Indeed, the conceptual learning methods

- use a symbolic representation of knowledge, including qualitative dependencies between subparts
- work on the structure of an object and treat relations between subparts
- generate hypotheses that depend on the context and satisfy the background knowledge of the trainer
- produce easily understandable, human-oriented decision rules

On the contrary, the parametric methods

- efficiently manipulate quantitative knowledge
- work on the global characteristics of a class of objects
- are suitable when a deep background knowledge on the problem is not available
- tolerate irrelevant and noisy features

As a result, an integrated approach allows for

- coping with a heterogeneous set of features (qualitative and quantitative)
- reducing the effects of noise on quantitative features
- effectively describing the background knowledge
- improving the computational efficiency of the learning process by realizing a form of constructive induction (Michalski, 1980)
- producing useful posterior probabilities of class membership

Before presenting the integrated approach to inductive generalization implemented in RES, it is necessary to explain how the layout of a document is described, in order to clarify where the numeric/symbolic features come from.

Page Layout Description Language for Document Classification

According to Muggleton (1992), in the general inductive problem we are provided with

- L_O, the language of observations
- L_B, the language of background knowledge
- L_H, the language of hypotheses
- O, a set of examples or observations described by means of L_O
- B, some background knowledge described by means of L_B

and we want to find a hypothesis H described by means of L_H, such that

$$B,H \vdash O$$

where $\vdash$ is the symbol of logic derivation.

In the application of document classification, L_O is the language used to describe the training documents, namely, the VL_{21} logic language (Michalski, 1980). Such a language is a variable-valued extension of the first-order predicate logic (FOPL). The main difference with respect to FOPL lies in the definition of an atomic formula. Indeed, in VL_{21} the equivalent to the FOPL atom is the notion of selector or relational statement, which is written as

$$[L = R]$$

where L, named *referee*, is a function symbol with its arguments and R, named *reference*, is a set of values in the referee's domain.

The semantics of a selector is the following: it is true if L takes one of the values in R. Function symbols of referees are called *descriptors*. They are n-ary typed functions ($n \geq 1$), mapping onto one of three different kinds of domains: *nominal*, *linear*, and *tree structured*.

A nominal, or *categorical*, domain is a domain in which no relation is imposed on its values. For instance, the domain of the descriptor *type*, used to describe the type of a block of a page layout, is nominal, since there is no relation between the values *text*, *hor_line*, *ver_line*, etc.

A linear domain is one in which a total order is imposed on its values. For instance, the domain of the descriptor *width* is linear, since it is possible to say that

$$very\_very\_small < very\_small < small < medium\_small < \ldots < very\_very\_large$$

Finally, a tree-structured domain is a partially ordered set whose elements can be represented as nodes of a tree. There are no tree-structured domains in the description language we adopted to represent documents. However, as we will show later, we feel the need to define a fourth and more general kind of domain, called *dag-structured* domain, in which values can be represented as nodes of a directed acyclic graph (dag).

In VL_{21}, monadic functions are named *attributes*, while n-adic functions, with $n > 1$, are called *relations*. Moreover, a *predicate* is considered as a particular function whose nominal domain is {false, true}.

The set of descriptors used for the symbolic description of the page layout is reported in Table 1. The domains of *width* and *height* are derived by discretizing two numeric intervals according to a preliminary study of the statistical distribution of the integer values on such intervals. As to the position of blocks, the descriptor *contain_in_pos* takes nine values, since an A4 page has been split into nine areas (this is a sort of discretization of the numerical coordinates of the center of each block). Figure 6 illustrates the way in which a page has been partitioned. Special attention must be paid to the relations *on_top*, *to_right*, and *aligned*. In fact, DDOCUM, the module of PLRS that produces page layout descriptions for the problem of document classification, generates the selector

$$[aligned(X, Y) = both\_columns]$$

if two blocks, X and Y, are almost aligned by columns (there is a tolerance due to noise), while it generates the selector

$$[aligned(X, Y) = starting\_col]\ ([aligned(X, Y) = ending\_col])$$

Table 1. Descriptors used for the symbolic description of the page layout in document classification

Descriptors	Definition
width(block)	Linear domain: *very_very_small, very_small, small, medium_small, medium, medium_large, very_large, very_very large*
height(block)	Linear domain: *smallest, very_very_small, very_small, small, medium_small, medium, medium_large, large, very_large, very_very_large, largest*
type(block)	Nominal domain: *text, picture, graphic, hor_line, ver_line, mixture*
contain_in_pos(doc,block)	Nominal domain (denotes the relative position of *block* in *doc*): *north_west, north, north_east, west, center, east, south_west, south, south_east*
on_top(block1,block2)	True if *block1* is above *block2*
to_right(block1,block2)	True if *block2* is to the right of *block1*
aligned(block1,block2)	Nominal domain (denotes the mutual alignment between *block1* and *block2*): *starting_row, ending_row, middle_row, both_rows, starting_col, ending_col, middle_col, both_columns, no_alignment*

if two blocks, X and Y, are almost aligned by left (right) column *alone*. In this application the predicate *aligned* is not commutative. For instance,

$$[aligned(X, Y) = starting\_col]$$

means that X and Y are aligned by left column and X is above Y, while

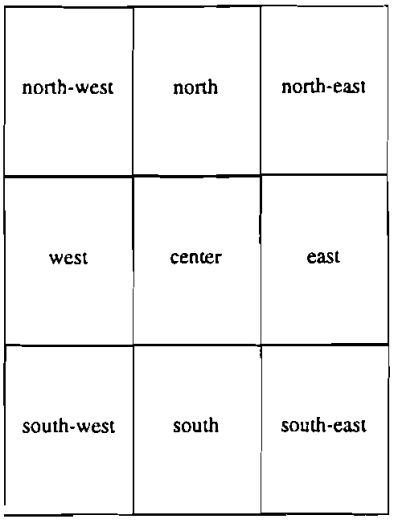


Figure 6. Partitioning of an A4 page into nine areas. Each area name denotes a possible value of the descriptor *contain_in_pos.*

$$[aligned(Y, X) = starting\_col]$$

means that X and Y are aligned by left column and Y is above X. However, this does not imply that every time X and Y are aligned by columns that X is *on_top* Y, since the selector $[on\_top(X, Y) = true]$ means that X is above Y and it is not too far from Y (less than 50 points on the vertical axis). Analogously, it should be noted that

$$[aligned(X, Y) = starting\_row]$$

means that X and Y are aligned by the upper row and X is to the right of Y. Once again, this does not imply that every time X and Y are aligned by row that X is *to_right* Y, since

$$[to\_right(X, Y) = true]$$

means that X is to the right of Y and their distance is less than 100 points on the horizontal axis.

Another property that does not hold for the descriptor *aligned* is the transitive property. This is due to the introduction of a tolerance in the alignment, that is, two blocks are aligned by left/right/middle/both column(s)/row(s) even if the coordinates of the blocks (or their centroids) are not exactly the same. Therefore,

$$[aligned(x1, x2) = starting\_col] \text{ and } [aligned(x2, x3) = starting\_col]$$

means that $x1$ and $x2$ have approximately the same abscissa, as do $x2$ and $x3$. However, this does not mean that $x1$ and $x3$ have approximately the same abscissa, since the algebraic sum of the differences along the horizontal axis can be greater than a fixed threshold.

Descriptors listed in Table 1, which are called *conceptual descriptors,* are not the only ones used to describe the page layout of a document. Indeed, another ninety-three numerical attributes, called *statistical descriptors,* are used by the parametric method. While conceptual descriptors are derived from the table Frame2 of the layout description, statistical descriptors are derived from the table Basic-Blocks produced by the segmentation process. Some examples of statistical descriptors are the mean length of text blocks (*mean_l1*) or the maximum area of image blocks (*max_a3*). All statistical descriptors refer to the characteristics of the whole document; therefore the only variable contained in the referee of a selector with a statistical descriptor is always the same and denotes the document. The same statistical descriptor, therefore, cannot occur more than once in the same example. Due to the fact that in the hybrid approach each feature should be processed by the proper method, only interval or ratio level measurements are specified as statistical descriptors, while nominal, ordinal, and partially ordered level measurements are treated as conceptual descriptors.

VL_{21} selectors can be combined by applying different operators, such as $\wedge$, $\vee$, :: (decision operator), and => (logic implication) in order to define

- *decision rules*, used for representing examples from a class (the action part of a rule specifies the class to which the observation belongs)
- *inference rules*, used for representing background knowledge, that is, the relationships among various descriptors
- *generalization rules* applied to concept descriptions with the aim of including more examples

Henceforth, we will adopt the convention of omitting the $\wedge$ operator between selectors, as well as the reference of a predicate when it is true. Figure 7 shows an example of a VL_{21} decision rule describing the page layout of the document depicted in Figure 5. For the sake of brevity, only some of the statistical descriptors have been reported, namely, those that will be selected by the stepwise feature selection algorithm used by the parametric method. It is worthwhile to observe that constants x_i introduced in the referees of selectors denote distinct parts of the document (the whole document is denoted by $x1$). Variables appearing in the generalization rules are considered existentially quantified and distinct. This means that variables with different names denote distinct objects (*object identity*) (Esposito et al., 1993f), and this must be taken into account while matching a rule against the description of a document.

Finally, an important difference between the language of observation L_O and the languages L_B and L_H is that the reference of each selector in an example is made up of exactly one value.

Conceptual Method

As we have already pointed out, page layout is made up of frames often strongly related to each other. Nevertheless, the structural nature of the domain makes the task of inductively developing efficient and effective classification rules for office documents more difficult, so that many well-known learning systems are unsuitable,

[contain_in_pos(x1,x2)=north_west] [contain_in_pos(x1,x3)=north] [contain_in_pos(x1,x4)=north]
[contain_in_pos(x1,x5)=north_west] [contain_in_pos(x1,x6)=center] [contain_in_pos(x1,x7)=centre]
[width(x2)=large] [width(x3)=very_large][width(x4)=medium_small] [width(x5)=large] [width(x6)=very_large]
[width(x7)=very_large] [height(x2)=medium] [height(x3)=very_very_small] [height(x4)=small]
[height(x5)=very_very_small] [height(x6)=smallest] [height(x7)=very_large)][type(x2)=mixture] [type(x3)=text]
[type(x4)=picture] [type(x5)=text] [type(x6)=hor_line] [type(x7)=mixture] [on_top(x2,x3)] [on_top(x3,x4)]
[on_top(x3,x5)] [on_top(x6,x7)] [to_right(x5,x4)] [aligned(x2,x3)=starting_col] [aligned(x3,x5)=starting_col]
[aligned(x5,x4)=middle_row] [aligned(x5,x6)=starting_col] [aligned(x6,x7)=both_columns][sd_11(x1)=223.874]
[min_e3(x1)=1.091] [max_e3(x1)=9.838] [dens_t(x1)=0.21] [symm_y(x1)=-0.049] [sd_p(x1)=893.429]
::> [class=icomp]

Figure 7. Description of the page layout of the document in Figure 5.

A learning system that is able to deal with structural knowledge is INDUBI (Semeraro, 1987), a system developed in C language at the Department of Informatics of the University of Bari. INDUBI is a problem-independent system for the automated acquisition of classification rules from examples, that draws its inspiration from the STAR methodology (Michalski, 1983). Such a methodology is definable as a general framework for empirical supervised learning and consists of a conceptual method for generating disjunctive covers, one for each class.

At a high level, INDUBI implements a separate-and-conquer search strategy (cf. Algorithm 1), while it adopts a beam search strategy at the low level, that is, for the construction of a single disjunct. More precisely, INDUBI starts with a positive example e^+ (defined as *seed*) and generates a set MQ of at least m distinct maximally general conjunctive generalizations, which are consistent and match against e^+. The set MQ is called *Star*, hence the name of the methodology. The best generalization is selected from MQ according to a lexicographic evaluation functional (LEF) (Larson, 1977), which takes into account the number of positive examples covered, the complexity expressed by the number of selectors, as well as the total cost of each generalization. Then, positive examples covered by the best generalization are removed from the set of positive examples, and a new disjunct is generated if the set of remaining positive examples is not empty.

INDUBI outputs symbolic discriminant descriptions of the concepts (classes), expressed in the form of classification rules, such as

$$Pattern :: > C$$

where *Pattern* is a concept description in disjunctive normal form (*Pattern* = $D_1 \vee D_2 \vee \ldots \vee D_n$, for some n) that satisfies both the *completeness* and the *consistency* conditions. Here, completeness means that every training example of a class C must

Algorithm 1: High-level strategy for learning a set of VL_{21} generalization rules

```
E+ := set of positive examples
E- := set of negative examples
LearnedRules := Ø
while E+ ≠ Ø do
      randomly select e+ from E+
      MQ := Beam_Search_for_consistent_hypotheses(e+, E+, E-, m)
      Best := FindBest(MQ)
      LearnedRules := LearnedRules ∪ {Best}
      E+ := E+ - Covers(Best, E+)
endwhile
output the set LearnedRules of VL21 generalization rules
```

match against the class description for C, while consistency means that no training example of a given class C matches against the description of a class different from C. In the domain of office document automatic classification, *Pattern* represents the *page layout signature* of the document, while C represents the class of a document. In Figure 8 an example of decision rule is reported, generated by INDUBI for the class *icomp*, to which the document described in Figure 7 belongs. As we would expect, the rule captures some regularities in the layout of the covers of the *IEEE Transactions on Computers*.

Parametric Method

INDUBI implements a conceptual learning method that produces human-oriented classification rules, but it is not very suitable for numeric descriptors. Furthermore, due to the consistency and completeness criteria described above, the method produces poor results when various sources of noise affect the document descriptions. These limits are overcome by traditional techniques studied in statistical pattern recognition.

In the statistical approach, examples are represented in terms of N numerical features (*feature vector representation*) and viewed as points in an N-dimensional space, named *feature space* (Duda & Hart, 1973). Then, the problem can be simply stated as follows:

- *Feature selection:* select M features, $M \leq N$, such that examples belonging to different classes will occupy different regions in the reduced feature space.
- *Discrimination:* establish decision boundaries in the feature space to separate examples belonging to different classes.

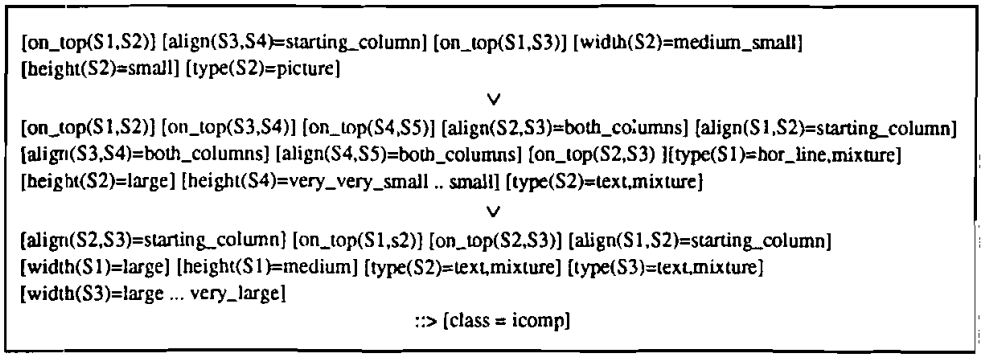


Figure 8. Generalization rule for the class *icomp* of documents. It contains three *or-atoms*, the first of which matches against the document description of Figure 7, according to the substitution $\sigma = \{S1 \leftarrow x3, S2 \leftarrow x4, S3 \leftarrow x5, S4 \leftarrow x6\}$.

There are two main reasons for performing feature selection. First, computing decision boundaries in an M-dimensional space is less expensive than dealing with the data in a higher dimensional space. Furthermore, a lower misclassification rate can sometimes be achieved by using a reduced feature space. The latter reason is explained by both theoretical and empirical considerations. In particular, it has been proved that the performance of a classifier, based on estimated densities, improves up to a point, then starts deteriorating as further features are added to the feature space (Hand, 1981). This phenomenon is also known as the *curse of dimensionality.* Generally, the number of training samples per class should be at least five to ten times the number of features used for computing the decision surfaces (Trunk, 1979).

There are two aspects concerning the problem of feature selection:

(1) A measure of class separability for observations described by a certain set of features is required.

(2) An algorithm for finding the best (or an optimal) subset of features is needed.

As to point (1), we have studied the properties of Wilks' lambda, which is defined as follows:

$$\Lambda = |W| \,/\, |W + B|$$

where W is the *within-class* scatter matrix and B is the *between-class* scatter matrix (Hand, 1981). Notice that as $|B|$ increases relative to $|W|$, the value of Λ decreases; thus Λ close to 0 indicates a good separation between classes and a high cohesion (homogeneity) within each class, while Λ close to 1 indicates a lack of discriminating power of the selected features. The Λ test of the null hypothesis

$$H_0: \mu_1 = \mu_2 = \ldots = \mu_k = \mu$$

with a significance level $\alpha = P(\Lambda < c_1 \,|\, H_0)$ can be cast as the problem of evaluating the probability $P(F > c_2 \,|\, H_0)$, where c_2 is the upper 100α percentage point of a particular F distribution (Cooley & Lohnes, 1971). This is a test of the statistical significance of the separation of the *class centroids* (class means) when a subset of variables has already been chosen.

As to point (2), we have implemented the *stepwise selection* (Hand, 1981). At each step, a feature is considered for selection if the statistical significance of the amount of class centroids separation added by that feature is sufficiently high, given the separation due to the previously selected features. This involves a test on the partial F ratio. Then the most discriminating feature that passes the test for inclusion is added to the set of selected features (the discriminating power is evaluated by using the Λ statistic). After the inclusion, the amount of separation between classes,

added by each of the selected features, given the other selected variables, is evaluated. Indeed, it is possible that some of the features entered earlier are no longer relevant for discrimination purposes. In such a case, at least one of such features can be removed from the set without affecting the discriminant power of the whole set. The stepwise selection stops when no feature passes both the inclusion and the removal tests. The description of the feature selection algorithm is provided in Algorithm 2.

In discrimination, once the variables have been selected, it is necessary to define the decision boundaries. They are described by mathematical functions, usually linear or quadratic, which are determined by the statistical distribution of the examples. Generally, such distributions are not known a priori, so an assumption of their form is made. In this case, the problem of establishing the decision boundaries is cast as a parameter estimation problem, in which the unknown parameters of the assumed distribution are estimated from the training examples. This is why the above functions are also called parametric classifiers. A popular choice for parametric methods is the normality of the class-conditional densities. In fact, the normal form is a good model for many naturally occurring phenomena (as a consequence of the central limit theorem) and it is uniquely defined by a mean vector μ and a variance-covariance matrix Σ. Under this assumption, the decision surfaces are generally

Algorithm 2: stepwise feature selection.

1. All the features are in SELECTABLE
2. The set SELECTED is empty
3. Select the first variable by a univariate-F test,
   add such a feature to SELECTED
   remove such a feature from SELECTABLE
4. While a new variable is added to SELECTED
   - 4.1. Repeat
     - 4.1.1. The set POSSIBLE_REMOVALS is empty
     - 4.1.2. For each feature in SELECTED
       - 4.1.2.1. Evaluate the partial F-ratio $F_{\text{to-remove}}$
       - 4.1.2.2. If $F_{\text{to-remove}}$ is too small then add the feature to the set of POSSIBLE_REMOVALS
     - 4.1.3. If POSSIBLE_REMOVALS is not empty then select the feature with the minimum $F_{\text{to-remove}}$ and perform the Λ test on it
     - 4.1.4. If the Λ test is accepted then remove such a feature from SELECTED and add it to SELECTABLE

     until no more removals are possible
   - 4.2. The set of POSSIBLE_INCLUSIONS is empty
   - 4.3. For each feature in SELECTABLE
     - 4.3.1. Evaluate the partial F-ratio $F_{\text{to-enter}}$
     - 4.3.2. If $F_{\text{to-enter}}$ is not too small then add the feature to the set of POSSIBLE_INCLUSIONS
   - 4.4. If POSSIBLE_INCLUSIONS is not empty then select the feature with the maximum $F_{\text{to-enter}}$ and perform the Λ test on it
   - 4.5. If the Λ test is accepted then remove such a feature to SELECTABLE and add such a feature to SELECTED
5. The optimal set of features is SELECTED

quadratic, and they become linear under the further hypothesis of an identical variance-covariance matrix for all the classes. The linear discriminant functions also present the advantage of robustness when data are nonnormal (Lachenbruch, 1975).

In PLRS, we have implemented a linear parametric classifier, namely, Fisher's (1936) linear discriminant functions, taking the following form:

$$q_i(\mathbf{x}) = \mathbf{v} \cdot \mathbf{x} - v_0$$

where

$\mathbf{x}$ is an M-dimensional feature vector representing an observation (in our application domain it is the collection of numerical attributes describing the layout of a single page document).

$\mathbf{v}$ is a vector of coefficients, which in turn, are functions of the estimated parameters μ and Σ.

v_0 is a constant term.

For instance, for the class *icomp*, to which the document described in Figure 7 belongs, the parametric classifier defined on the set of selected features is

$$\begin{aligned} q_{\text{icomp}}(sd\_l1, min\_e3, max\_e3, dens\_t, symm\_y, sd\_p) \\ = 1.134 \cdot sd\_l1 - 3.719 \cdot min\_e3 + 18.2 \cdot max\_e3 + 76.055 \cdot dens\_t \\ + 142.89 \cdot symm\_y + 0.003 \cdot sd\_p - 216.42 \end{aligned}$$

Finally, it is important to note that the higher $q_i(\mathbf{x})$ is, the higher the likelihood that the observation described by $\mathbf{x}$ belongs to class i. Therefore, an observation will be assigned to the class with the maximum value of $q_i(\mathbf{x})$, since i has the highest posterior probability $P(i \mid \mathbf{x})$.

Integrated Method

The conceptual and the statistical methods mentioned above present some similarities, as follows:

- Both approaches are *model-driven*, that is, the examples are considered all together in order to generate the decision rules.
- Zeroing a coefficient in a linear discriminant function is equivalent to the dropping-condition rule used in the STAR methodology.

However, they also exhibit some relevant differences:

- A symbolic hypothesis describes a concept, while a discriminant function delimits a boundary surface.
- The symbolic method works in a sequential way, generalizing from a single example each time, while discriminant analysis works on all observations together.
- Conceptual methods allow multiple-entity (structural) representations, while the statistical methods manage only single-entity representations (nonstructured objects).

A first step toward an integration of the parametric and the symbolic methods concerns the representation language VL_{21}. Indeed, each document is described by a set of statistical and conceptual descriptors, so we have a unique representation formalism, both for conceptual and for statistical discrimination.

Only the statistical descriptors are considered by the statistical analysis in order to compute the coefficients of the linear discriminant functions. These are also used for computing the posterior probabilities of the class membership of each document, $P(i \mid x)$, so that the maximum probability determines the class to which each recognized document is assigned. The classification result is encoded in the following VL_{21} selector:

$$[class\_disc\_analysis(variable)=value]$$

where

- *class_disc_analysis* is a descriptor specifying the class assigned to the document by the discriminant analysis: it represents the value of a mathematical function mapping the feature vector associated with the document to the class identifier,
- *variable* is the name of the document as a whole, and
- *value* is the class identifier.

Such a selector synthesizes the numerical information on the common geometrical characteristics of a class and is appended to the symbolic description of each document. The selector can play an important role in making the concept learning process efficient, since it realizes a form of constructive induction (Esposito et al., 1993a).

INDUBI, the symbolic rule generator of RES, extracts the selectors containing the conceptual descriptors and the selector *class_disc_analysis* from the page layout description of each training document. The results are VL_{21} discriminant rules generated according to the STAR methodology and stored in the document knowledge base, together with the document descriptions and the results of the discriminant analysis (see Figure 9).

The main purpose of the page classifier is to match the symbolic descriptions of new documents against the previously stored rules. First, a test phase is necessary.

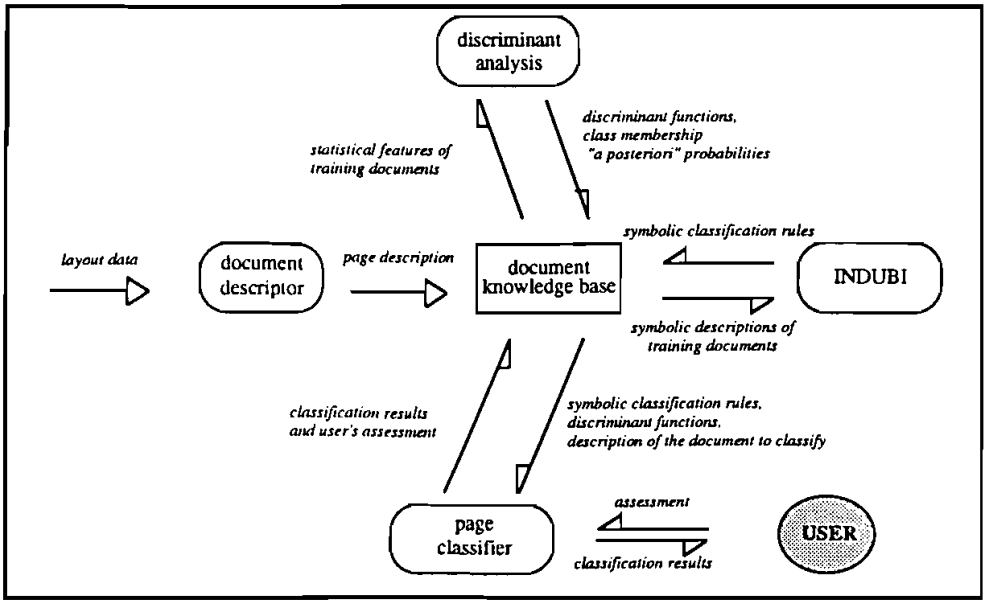


Figure 9. Schema of the integrated method implemented for the layout-based document classification. From the page layout description of training examples, a set of statistical features is selected. It is used by the discriminant analysis, which produces the discriminant functions, and the posterior probabilities of each class for all training examples. Posterior probabilities are used in order to define the value of the selector *class_disc_analysis*, which is appended to the symbolic descriptions of the training examples. The latter are used to train INDUBI, which produces symbolic classification rules. Finally, the page classifier exploits both the discriminant functions and the symbolic rules in order to classify new test documents.

Some statistical features of the document descriptions are used for computing Fisher's discriminant scores. The posterior probabilities are evaluated, and the class with the highest probability defines the value of the selector *class_disc_analysis*. The posterior probabilities are also exploited to define the order in which the VL_{21} classification rules have to be matched against the document descriptions. This speeds up the classification process, provided that the true error rate of the discriminant analysis is low enough. The result of classification may be a single class with probability equal to 1, that is, the system is sure of its recognition of the new document. However, it might occur that the document does not match against any classification rule. In such a case, the system shows a set of probabilities representing the degree of similarity between the document description and that of each class. If the system fails in the recognition process, the user can give a negative assessment together with the proper class and force the system to produce new classification rules.

For instance, the classification rule produced by the hybrid method for the class *icomp*, is as follows:

[*contain_in_pos*(*S*1,*S*2) = *north*] [*on_top*(*S*2,*S*3)]
[*class_disc_analysis*(*S*1) = *icomp*] [*type*(*S*2) = *text,hor_line,picture,mixture*]
[*width*(*S*2) = *very_small*. . *very_large*]
[*height*(*S*2) = *smallest*. . *large*] :: > [*class* = *icomp*]

By comparing this rule with that presented in Figure 8, it is evident that the recognition rule is now simpler. Consequently, the learning process takes less time to generate it. Time requirement is an important factor for our real world application. Indeed, another workpackage of the INTREPID project, namely, HD (HW/SW architecture definition and prototype implementation), is aimed at defining suitable architecture for an efficient implementation of a real-time recognition system. As we will show in the section devoted to experimental results, the classification accuracy of the above rule improved significantly as well.

Flexible Matching for Noisy Structural Descriptions

When noise is present or the training sample is insufficient to properly train the system, the predictive accuracy cannot be guaranteed. A classifier working in an all-or-none fashion is useless, and it is necessary to rely on a continuous measure of *similarity* between objects. In Esposito et al. (1992a, 1992b), we propose a complementary measure, the *distance,* based on a probabilistic interpretation of the matching predicate. It is used to classify examples that do not present exactly all the regularities appearing in the corresponding recognition rule.

The distance measure, Δ, suitable for structural deformations, is defined on the space of the VL_{21} well-formed formulas (*wff*s). Let $F1$ and $F2$ be two VL_{21} *wff*s, then

$$\Delta(F1, F2) = 1 - flex\_match(F1, F2)$$

where *flex_match*($F1$,$F2$) is a function taking on values in [0,1] that represent the probability that $F1$ perfectly matches against $F2$. Therefore,

$$flex\_match(F1, F2) = P[match(F1, F2)]$$

where *match* is the canonical matching predicate defined on the same space. The latter definition marks the transition from deterministic to probabilistic matching.

The main application of the distance measure is concept recognition in noisy environments. Therefore, from now on, $F1$ will denote the description of a concept, and $F2$ the observation to be classified.

The function *flex_match* is computed according to a top-down evaluation scheme:

(1) $F1$ and $F2$ are disjunctions of conjuncts:

$$flex\_match(F1,F2) = P[match(or\_atom_1,F2) \vee \ldots \vee match(or\_atom_N,F2)]$$

where or_atom_i is the generic conjunction of selectors in $F1$, while $match(or\_atom_i,F2)$, with $i \in [1,N]$, are independent events.

(2) $F1$ and $F2$ are conjunctions of selectors:

$$flex\_match(F1,F2) = \begin{cases} \max_{\sigma_j} \prod_i w_i \cdot flex\_match_j\left(sel_i, F2\right) & \text{if there exists a substitution } \sigma_j \\ & \text{such that } F2 \Rightarrow \sigma_j\,(F1') \\ 0 & \text{otherwise} \end{cases}$$

where

σ_j is one of the possible substitutions among variables,
sel_i is the generic selector in $F1$,
$flex\_match_j$ is the measure of fitness between sel_i and $F2$ when the variable substitutions are fixed by σ_j,
w_i denotes the weight of the function in sel_i, and
$F1'$ is the shortest conjunction of selectors in $F1$, such that all the distinct variables in $F1$ are also in F1′.

(3) $F1$ and $F2$ are selectors:

$$flex\_match_j\,(F1,F2) = \max_{i \in [1,s]} P[equal(g_i,e)]$$

where

g_i is one of the values of the reference of $F1$,
e is the only element in the reference of the observation $F2$, and
$equal(x,y)$ denotes the matching predicate defined on any two values x and y of the same domain.

The term $P[equal(g_i,e)]$ is defined as the probability that an observation e may be considered a distortion of g_i, that is,

$$P[equal(g_i,e)] = P[\delta(g_i,X) \geq \delta(g_i,e)]$$

where

X is a random variable assuming values in the domain of the function contained in $F1$ (or $F2$) and
δ is the distance defined on the domain itself.

The definition of $P[equal(g_i,e)]$ takes into account both the type of function and the probability distribution of its domain.

The function *flex_match*($F1$,$F2$) actually computes the probability that any observation of the concept described by $F1$ would be farther from the centroid $F1$ than the case $F2$ being considered. If *flex_match*($F1$,$F2$) is too small, it signals the possibility that $F2$ is not an instance of the class described by $F1$, even though it is the closest. The function *flex_match*($F1$,$F2$) can be considered from a theoretical point of view. In fact, the probability that the referee's value changes is a deformation probability, so finding the most similar concept description is equivalent to detecting the most probable deformation. In Esposito et al. (1991c), the definition of distance measure has also been extended in order to deal with incomplete descriptions.

Decision making based on a distance measure is more expensive than a true/false matching procedure; therefore, a multilayered framework is more suitable (Esposito et al., 1991a). Moreover, two distinct methods have been applied in order to reduce the computational complexity: a branch-and-bound algorithm and a heuristic approach (Esposito et al., 1991b). The former is an algorithm that performs quickly on average, while the latter produces acceptable answers in an acceptable amount of time. From an empirical comparison of the two methods, we draw the following conclusion: classification results do not change at all if the heuristic method is used and the class corresponding to the highest value of similarity is taken as the membership class. However, by considering the throughput time, we conclude that the branch-and-bound method needs much more time than the heuristic method, and this is a great limitation for a real-time document processing system.

Here we provide an example of application of the distance measure to the problem of document classification. In a first experimentation described by Esposito et al. (1990a), we considered a set of seventy-two single-page documents belonging to nine different classes. Four classes are business printed letters, sent or received by four different companies, while four other classes are magazine indexes. The ninth class is a reject class, representing the rest of the world. Fifty instances were selected as training examples, leaving the remaining twenty-two documents for the test phases. In Table 2 the recognition rules for each class and the corresponding flexible matching values (*FM*) for the document in Figure 10 are reported.

As an example of computation of the flexible matching, let us consider the recognition rule for the third class. When $F1$ is equal to the first *or-atom*, then $F1'$ becomes the following conjunct:

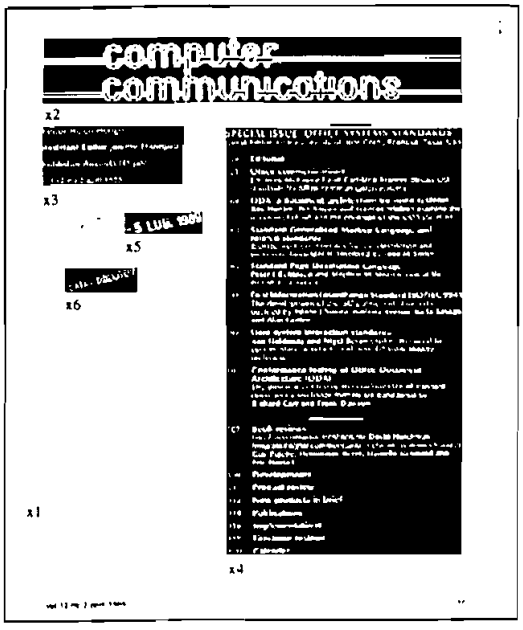
$$F1' = [to\_right(S1,S2)][to\_right(S1,S3)][to\_right(S4,S2)]$$

which does not strictly match against the description of the test document, that is $F2$. Consequently, we have

$$flex\_match(or\_atom_1,F2) = 0$$

Table 2. Flexible matching of the test document against each rule

Class	VL_{21} description	Value
1	[*to_right*(*S*1,*S*2)][*aligned*(*S*1,*S*2) = *starting_col*][*height*(*S*1) = *medium_small*] [*width*(*S*2) = *medium*]	0.0
2	[*to_right*(*S*1,*S*2)][*aligned*(*S*1,*S*2) = *ending_row*][*height*(*S*1) = *very_small*] [*width*(*S*1) = *medium_small*][*width*(*S*2) = *very_small,small*]	0.40
3	[*to_right*(*S*1,*S*2)][*to_right*(*S*1,*S*3)][*to_right*(*S*4,*S*2)][*to_right*(*S*3,*S*2)] [*height*(*S*2)= *very_very_large*] ∨ [*on_top*(*S*1,*S*2)][*to_right*(*S*2,*S*3)][*on_top*(*S*1,*S*3)][*aligned*(*S*1,*S*2) = *starting_col*] [*height*(*S*1) = *medium*][*height*(*S*2) = *medium_small*][*height*(*S*3) = *medium .. very_very_large*]	0.909
4	[*aligned*(*S*1,*S*2) = *ending_col*][*width*(*S*1) = *medium_small*][*height*(*S*1) = *medium_small*]	0.444
5	[*to_right*(*S*1,*S*2)][*height*(*S*1) = *very_large .. largest*]	0.454
6	[*to_right*(*S*1,*S*2)][*to_right*(*S*3,*S*1)][*on_top*(*S*1,*S*4)][*on_top*(*S*5,*S*1)] [*width*(*S*2) = *medium_small*][*height*(*S*2) = *very_very_small*]	0.0
7	[*contain_in_pos*(*S*1,*S*2) = *north_west*][*contain_in_pos*(*S*1,*S*3) = *center*] [*aligned*(*S*2,*S*3) = *starting_col*][*height*(*S*3) = *very_very_large,largest*]	0.90
8	[*aligned*(*S*1,*S*2) = *middle_row*][*to_right*(*S*3,*S*4)][*on_top*(*S*5,*S*1)][*on_top*(*S*1,*S*3)]	0.0



[width(x2)=large]
[width(x3)=medium]
[width(x4)=medium_large]
[width(x5)=medium_small]
[width(x6)=medium_small]
[height(x2)=medium_small]
[height(x3)=medium_small]
[height(x4)=very_very_large]
[height(x5)=very_small]
[height(x6)=small]
[contain_in_pos(x1,x2)=north]
[contain_in_pos(x1,x3)=north_west]
[contain_in_pos(x1,x4)=centre]
[contain_in_pos(x1,x5)=west]
[contain_in_pos(x1,x6)=west]
[on_top(x2,x3)]
[on_top(x2,x4)]
[on_top(x3,x5)]
[to_right(x3,x4)]
[to_right(x5,x4)]
[align(x2,x3)=first_column]
[align(x2,x4)=last_column]
[align(x3,x4)=first_row]
::> [class=three]

Figure 10. The page layout of a test document and its description in VL_{21}.

On the contrary, when $F1$ is equal to the second *or-atom*, then we find two substitutions, such that

$$F2 \Rightarrow \sigma_j(F1') \qquad (F1' = [on\_top(S1,S2)][to\_right(S2,S3)])$$

namely, $\sigma_1 = \{S1 \leftarrow x2, S2 \leftarrow x3, S3 \leftarrow x4\}$ and $\sigma_2 = \{S1 \leftarrow x3,\ S2 \leftarrow x5, S3 \leftarrow x4\}$. According to the substitution σ_1, only the fifth selector of the *or-atom* does not strictly match the description $F2$; therefore, we have

$$\prod_{i=3}^{7} flex\_match_1(sel_i,F2) = 1 \cdot 1 \cdot {}^{10}\!/_{11} \cdot 1 \cdot 1 = 0.909$$

since the linear domain of the descriptor *height* has eleven values and we are assuming that each value of the domain has the same probability [for further explanation, see Esposito et al. (1991a)]. It is easy to verify that the flexible matching computed according to the substitution σ_2 gives a lower value, so we can conclude that

$$flex\_match(or\_atom_2, F2) = 0.909$$

and then $flex\_match(F1, F2) = 0.909$. The highest FM value in Table 2 is that associated with the third class; therefore the document is correctly classified. The document is not rejected because the FM values are all higher than the fixed threshold (0.85). Results for all the test documents are reported by Esposito et al. (1991b). Here we give some hints on the system performance: about 6 s for generating the symbolic description of a document and about 3 s for matching and classifying a document.

These results refer to an implementation in C on an Olivetti PC M280. The total processing time for a single document is less than 1 minute, including the scanning process.

Experimental Results

As already pointed out, RES has been used to automatically produce the rules of a knowledge-based system for the recognition of optically scanned documents. In order to produce the classification rules, some significant examples of document classes, which may be of interest in a specific environment, were used as a training sample to discover the layout similarities within each class.

In a second experiment, we considered a set of 161 single-page documents, namely, printed letters and magazine indexes, belonging to 8 different classes (the last was a *reject* class, representing the rest of the world). Fifty-one instances were selected as training cases, leaving the remaining 110 documents for the testing phase.

Once again, all the sample documents were real letters, received or sent by companies, or copies of the indexes of international magazines, so that several forms of noise actually affected them.

Working on the layout structural characteristics, the conceptual method produced seven maximally specific classification rules, expressed as VL_{21} decision rules, satisfying both the completeness and consistency conditions. Most of these rules pointed out the invariant parts of the physical layouts, such as logos and titles of fixed sizes in fixed positions, and their relations with other parts of the documents, such as alignment with other blocks. However, due to the limited number of training cases, the predictive accuracy was low. In fact, only 76 of the 110 test cases were correctly classified, although the recognition rate increased by using the flexible matching (see Table 3).

A training set of 121 examples was then selected, independently of the 110 test samples. The training set included the old 51 instances. The predictive accuracy for the whole set increased up to 92%, but disjunctive and trivial rules were generated. Two disjunctive rules with three *or-atoms* were generated both for *IEEE Transactions on PAMI* indexes and for *IEEE Transactions on Computers* indexes (see Figure 8), because of the variability of page layout and the presence of noise. Table 4 illustrates the results of the testing process. The learning process in the training phase took about 4 hours and 30 min on a Sun3/280. Half of the training time was spent in generating these rules.

With a consistent number of training cases, it is possible to use and to evaluate the performance of the parametric classifier on the statistical descriptors concerning the layout. Initially, ninety-three features were chosen to describe each document, but only six were selected by the stepwise variable selection process, minimizing Wilks' lambda:

(1) maximum eccentricity of image blocks (*max_e3*).
(2) standard deviation of the number of black pixels (*sd_p*).

Table 3. Results of the application of the conceptual method by a strict matching and a flexible matching (training set 51 cases; testing set 110 cases)

Class	No. of cases	No. correct classifications	Strict matching, %	Flexible matching, %
Olivetti letter	27	20	74	96
Sifi letter	20	12	60	90
Sims letter	15	7	46	91
Sogea letter	10	8	80	98
PAMI index	21	17	81	85
Spectrum index	8	4	50	70
Computer index	9	8	88	90

Table 4. Confusion matrix for the conceptual method (training set 121 cases; testing set 110 cases)

	Olivetti	Sifi	Sims	Sogea	PAMI	Spectrum	Computer	Reject
Olivetti	96							4
Sifi		100						
Sims			100					
Sogea				100				
PAMI					90			10
Spectrum						75		25
Computer	11						67	22

Values are percentages.

(3) standard deviation of the length of text blocks (*sd_l*1).
(4) minimum eccentricity of image blocks (*min_e*3).
(5) symmetry along the vertical direction (*symm_y*).
(6) percentage of the textual part (*dens_t*).

It is fairly evident that image blocks, such as logos and magazine headings, are important in the discrimination process. Another selected feature is the standard deviation of the length of text blocks, which is relevant in order to discriminate justified letters from those not justified, and magazine indexes with a high variability in the length of some entries from others that are better organized. The discriminant rules generated by discriminant analysis are not sufficient to recognize the documents. In fact, some misclassifications occurred when the same training documents were reclassified. This means that the eight classes are not linearly separable. When using the features selected, the re-substitution error rate is about 8%, nearly the same as that obtained using the leaving-one-out criterion (Lachenbruch, 1975).

The throughput time for selecting the variables and computing the discriminant functions was about 4 min on a Sun3/280. Time needed for classifying a single document was less than 1 s. Table 5 shows the results of the testing phase when only the statistical method was applied.

In the integrated approach, the new selector [*class_disc_analysis*(*variable*) = *value*], resulting from the application of discriminant analysis, was appended to each training/testing example. Moreover, a lower cost was assigned to *class_disc_analysis* than to the other descriptors, due to the high discriminant power revealed by the statistical method.

With the application of INDUBI, the generated hypotheses often changed significantly, especially for those classes characterized by disjunctive rules. This was due to the fact that the new appended selectors alone were sufficient to characterize some classes. The only rules that remained unchanged were those

Table 5. Confusion matrix for the statistical method (training set 121 cases; testing set 110 cases)

	Olivetti	Sifi	Sims	Sogea	PAMI	Spectrum	Computer	Reject
Olivetti	96							4
Sifi		95	5					
Sims		7	93					
Sogea				100				
PAMI					95		5	
Spectrum						100		
Computer							100	

Values are percentages.

concerning two classes of letters, while all the others changed by including the new selector in the condition part. The results of the testing phase are reported in Table 6. The overall predictive accuracy of the combined approach is 98%. It is interesting to notice the complementarity of the symbolic and the statistical methods, which justifies the robustness of the integrated approach. Moreover, the application of the distance measure is necessary when the training sample size is small, while it becomes less important when the training sample size increases. This is due to the presence of the new descriptor, which summarizes the class regularities and is robust against the noise.

Finally, the time needed to train the system was nearly halved, since it took only 2 hours and 30 min, including the statistical analysis. This is a good result, bearing in mind that the learning system handles about nine thousand selectors in the complete training set. Once the rules have been generated, the knowledge-based system is able to recognize a digitized document in 9 s.

The complete discussion of the experimental results is reported by Esposito et al. (1990b).

Table 6. Confusion matrix for the integrated method (training set 121 cases; testing set 110 cases)

	Olivetti	Sifi	Sims	Sogea	PAMI	Spectrum	Computer	Reject
Olivetti	96							4
Sifi		100						
Sims			100					
Sogea				100				
PAMI					95		5	
Spectrum						100		
Computer							100	

Values are percentages.

DOCUMENT UNDERSTANDING

The success of the machine learning approach to document classification induced us to investigate the possibility of adopting the same approach for the problem of document understanding, that is, for recognizing logical components of a document once it has been classified. Given a set of documents whose layouts have already been analyzed, and assuming that some layout components have been correctly labeled according to their meaning—for instance, as a sender or receiver block in a letter—then the problem is that of learning some rules that allow for correctly labeling the layout components in previously unseen documents (Malerba, 1993).

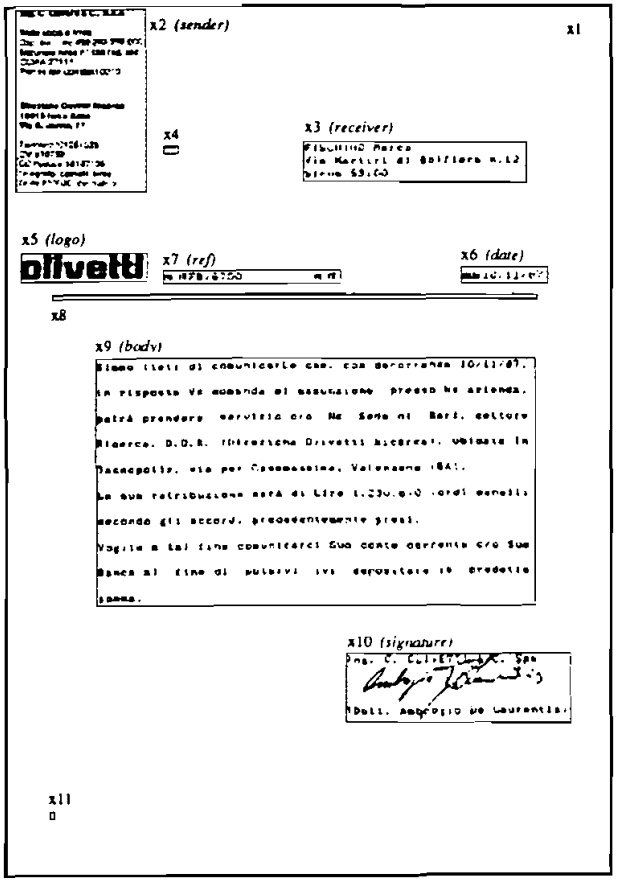
The classes do not concern the whole document anymore, but the logical objects within a document of a given class. The examples are descriptions in terms of geometrical characteristics of the logical objects. The descriptors used in this phase are

width(block), height(block), type(block), on_top(block1, block2), to_right(block1, block2), aligned(block1, block2),

which are defined in Table 1, plus the predicate *part_of(doc,block)*, and the nominal descriptor *position(block)*, which replace the descriptor *contain_in_pos(doc,block)* shown in Table 1, and the nominal descriptor *logic_type(block)* describing the logical class of a layout block. For example, in the class of Olivetti letters the domain of *logic_type* is *{sender, receiver, date, logo, ref, body, signature, unspecified}* (see Figure 11). It is worthwhile to note that a single document may generate a variable number of examples for each logical class. This is due to the fact that more logical objects of the same logical class may actually be in a document. Consider, for instance, the existence of multiple receivers in a circular or the fragmentation problem caused by the layout analysis (Esposito et al., 1993b).

As pointed out in the introduction, the problem of document understanding can be greatly simplified when the membership class of a document has already been identified. Indeed, in this case we can define logical components more easily for each class of documents, and we significantly reduce the variability of training instances for this new learning problem. In spite of such shrewdness, the problem of learning rules for document understanding is still more complex than the problem of learning recognition rules for classifying documents.

The main difficulty is that now concepts to be learned refer to a part of a document rather than to the whole document, and since parts of documents may be related to each other according to logical-logical relationships, this leads to the problem of *learning contextual rules*. Most of the studies on supervised inductive learning presented in the machine learning literature make the implicit assumption that concepts are independent (*independence assumption*) and, consequently, that



[part_of(x1,x2)]
[part_of(x1,x3)]
[part_of(x1,x4)]
[part_of(x1,x5)]
[part_of(x1,x6)]
[part_of(x1,x7)]
[part_of(x1,x8)]
[part_of(x1,x9)]
[part_of(x1,x10)]
[part_of(x1,x11)]
[position(x2)=north_west]
[position(x3)=north]
[position(x4)=north_west]
[position(x5)=north_west]
[position(x6)=north_east]
[position(x7)=north]
[position(x8)=center]
[position(x9)=center]
[position(x10)=south_east]
[position(x11)=south_west]
[width(x2)=medium]
[width(x3)=medium_large]
[width(x4)=smallest]
[width(x5)=medium]
[width(x6)=medium_small]
[width(x7)=medium_large]
[width(x8)=very_very_large]
[width(x9)=very_very_large]
[width(x10)=medium_large]
[width(x11)=smallest]
[height(x2)=medium_large]
[height(x3)=small]
[height(x4)=smallest]
[height(x5)=very_small]
[height(x6)=very_very_small]
[height(x7)=very_very_small]
[height(x8)=smallest]
[height(x9)=large]
[height(x10)=medium_small]
[height(x11)=smallest]
[type(x2)=text]
[type(x3)=text]
[type(x4)=text]
[type(x5)=picture]
[type(x6)=text]
[type(x7)=text]
[type(x8)=text]
[type(x9)=text]
[type(x10)=mixture]
[type(x11)=text]
[on_top(x5,x8)]
[on_top(x6,x8)]
[on_top(x7,x8)]
[on_top(x9,x10)]
[to_right(x2,x4)]
[to_right(x5,x7)]
[aligned(x2,x5)=both_columns]
[aligned(x5,x7)=ending_row]
[aligned(x4,x7)=starting_col]
[aligned(x7,x6)=both_rows]
[aligned(x8,x9)=ending_col]
[aligned(x4,x3)=starting_row]
[aligned(x8,x11)=starting_col]

Figure 11. Page layout of an Olivetti letter and its symbolic description.

training instances are independent. Henceforth, we will denote learning algorithms based on this assumption as *traditional* learning algorithms. Of course, it is possible to use such algorithms in the problem of document understanding by simply neglecting logical-logical relationships, but it is our opinion that this is not the correct way to solve the problem. Indeed, we believe that by taking into account concept dependencies, it is possible to generate more accurate and simpler rules, since the learning paradigm is a better approximation of reality.

Another difficulty with the problem of learning rules for document understanding is the necessity to change the representation language. Indeed, VL_{21}, the representation language adopted by INDUBI, does not allow variables to be introduced in the action part of the decision rules, since it is implicitly assumed that the condition part describes an observation and the action part defines the class to which *the whole* observation belongs.

This means that by using VL_{21}, we cannot represent rules, like those necessary for document understanding, in which the action part refers to a part of an observation rather than to the whole observation. By allowing variables and constants to be introduced in the referee of the action part of a VL_{21} decision rule, we get a new representation language, which has the same expressive power of Horn clauses (Genesereth & Nilsson, 1987). Henceforth, we will indicate such decision rules as VL_{21} *definite clauses*. In the next subsection we will provide a brief introduction to the new representation language for document understanding.

From VL_{21} Decision Rules to VL_{21} Definite Clauses

A VL_{21} *literal* is a selector or the negation of a selector. Each VL_{21} *expression* may be obtained from a set of literals by applying different operators, such as $\wedge$, $\vee$, and $\leftarrow$ (logical implication). In particular, we will define VL_{21} *definite clause* as any VL_{21} expression of the kind

$$[f(t_1,t_2, ..., t_n) = \lambda_i] \leftarrow \phi(t_1,t_2, ..., t_m)$$

where f is a function symbol, t_i may be constants or variables, λ_i is a value of the fs domain, and $\phi(t_1,t_2, ..., t_m)$ is a conjunction of VL_{21} literals, whose referee's arguments contain the terms $t_1,t_2, ..., t_m$. The selector to the left side of the $\leftarrow$ operator is called head, while the conjunction to the right is called body [for a translation of VL_{21} clauses into Horn clauses, see Malerba (1993)].

Both training/test examples and hypotheses are expressed as VL_{21} clauses, but there are some differences:

(1) Each training/test example represents just one ground clause, that is, a clause without variables.

(2) Each hypothesis is expressed as a set of constant-free clauses having the same head.

(3) The reference of each selector in any training/testing example is made up of exactly one value.

(4) In the examples, false predicates are omitted, while all the attributes are specified.

(5) In the hypotheses, every omitted selector is assumed to be a function taking any value of its domain.

(6) Variables with different names denote distinct objects (*object identity*) (Esposito, et al., 1993f; Semeraro et al., 1993).

These points describe the main differences between the language of observation L_O and the languages L_B and L_H for the background knowledge and the hypotheses, respectively.

VL_{21} clauses are an extension of the original definition of the VL_{21} language that was adopted for the problem of document classification. In fact, it is always possible to translate VL_{21} decision rules into VL_{21} clauses with very little effort. For instance, for the application of document classification, the constant $x1$ was always used to represent the whole training/testing document; therefore the decision rule in Figure 7 can be equivalently rewritten as

$$[class(x1) = icomp] \leftarrow [contain\_in\_pos(x1,x2) = north\_west] \\ \ldots [sd\_p(x1) = 893.429]$$

Moreover, VL_{21} clauses can represent concepts that could not be represented with VL_{21} decision rules, as in the case of document understanding. For instance, let us consider the document layout depicted in Figure 11, in which $x2$ is a block *sender*, $x3$ is a block *receiver*, $x5$ is a block *logotype*, and so on (not all blocks have an associated label). Then, the description of the block *sender* is

$$[logic\_type(x2) = sender] \leftarrow [part\_of(x1,x2)]\ [part\_of(x1,x3)] \ldots \\ [position(x2) = north\_west] \ldots$$

while the description of the block *receiver* is

$$[logic\_type(x3) = receiver] \leftarrow [part\_of(x1,x2)][part\_of(x1,x3)] \\ \ldots [position(x2) = north\_west] \ldots$$

The only differences between these two instances are in the referee's argument and reference value of the heads of the clauses. In the case in which we are not allowed to define an argument for the referee of the action part, as in the previous definition of VL_{21} decision rules, there is no way to distinguish the instance of *sender* from the instance of *receiver*.

Finally, it is necessary to extend the definition of VL_{21} connected formula given by Esposito et al. (1992a). For this reason, we give the following definition of a VL_{21} *linked* clause:

> A VL_{21} clause is linked if all its literals are. A literal is linked if at least one of its referee's arguments is. An argument of a literal is linked if either the literal is the head of the clause or another argument in the same literal's referee is linked.

An example of a linked clause is the following:

$$[logic\_type(X1) = logo] \leftarrow [type(X1) = picture]\ [on\_top(X2,X1)]$$

while

$$[logic\_type(X1) = logo] \leftarrow [type(X3) = picture]\ [on\_top(X2,X1)]$$

is an example of a nonlinked VL_{21} clause. It is worthwhile to note that nonlinked clauses are generally meaningless, as in this case; therefore, in the learning process, only the space defined by linked clauses should be explored by search strategies. INDUBI/H (Esposito et al., 1993e), an evolution of INDUBI that can deal with VL_{21} clauses, searches for the best hypothesis by adding a literal to the (initially empty) body of the clause to be learned only when the constraint on linkedness is not violated.

Learning Contextual Rules

Although inductive learning systems generally differ for the representation language, the search strategy, and the amount and type of background knowledge, they share the common assumption that concepts are mutually independent. Problems caused by such an independence assumption are particularly evident in at least three situations:

(1) learning multiple attributes in propositional calculus
(2) learning multiple predicates in first-order logic
(3) learning classification rules for labeling problems

As to the first situation, let us consider a relational database in which there are *multiple attributes* to be learned; then training instances can be represented as follows:

$$<a_1,a_2, \ldots, a_n, c_1, c_2, \ldots, c_m>$$

where a_is are *known* attributes and c_is are *target* attributes to be learned. When c_i has a nominal domain, it is possible to interpret c_i as a *class* attribute, and the problem of learning c_i can be cast as the problem of learning a classifier K_i. Dependency

between class attributes c_i and c_j can be expressed by allowing K_j to depend on the outcome of the classifier K_i and the known attributes. This means that while instances of c_i are not described by means of the attribute c_j, the language used to describe instances of c_j will include c_i. Therefore, the learning task becomes more difficult, since in addition to the problem of learning K_j, it is necessary to consider the possibility of changing the description language. When concept dependencies are intrinsically acyclic, we can use *dependency hierarchies*, that is, directed acyclic graphs whose nodes are concepts to be learned, to represent them (see Figure 12). The order in which concepts should be learned is completely defined by a dependency hierarchy. In particular, the concepts in the lowest level of a dependency hierarchy have to be learned first, since their definition does not depend on other concepts (*minimally dependent* concepts). When a dependency hierarchy is provided, it is possible to use traditional learning systems, that is, systems based on the independence assumption, in order to learn contextual rules. On the contrary, when the dependency hierarchy is not known a priori, we may try to discover concept dependencies first. Some studies in the field of causal inference and concerning zeroth-order languages (Esposito et al., 1993c) can already address this problem.

No easy solution is yet available for the case of first-order languages. The problem of multiple-predicate learning, which challenges researchers of inductive logic programming (ILP) (Muggleton, 1992), is strictly connected to the problems of intensional evaluation of predicates (De Raedt et al., 1993).

When learning multiple-predicate definitions in first-order logic, it may happen that some predicate definitions have to use others, which implies that the order in which clauses for different predicates are learned is important. De Raedt et al. (1993) claims that extensional systems, such as GOLEM (Muggleton & Feng, 1990) and FOIL (Quinlan, 1990), suffer from the problem that learned hypotheses might not be intensionally complete and consistent. Further problems come from the fact that Horn clauses allow recursion to be represented and there is no mechanical method that guarantees the termination of a recursive logic program. Moreover, extensional learning methods generally adopt θ-subsumption as a generalization model, which is incomplete for recursive clauses (Plotkin, 1971).

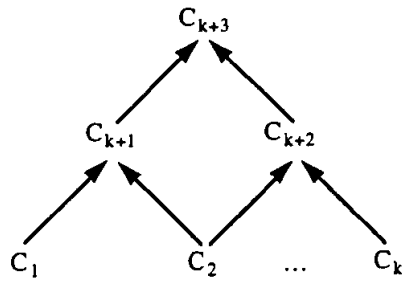


Figure 12. An example of dependency hierarchy between concepts.

As to labeling problems in pattern recognition (Haralick & Shapiro, 1979), the aim is that of learning rules that allow for correctly labeling subparts of a previously unseen structured object, provided that a set of instances of structured objects, whose subparts have already been labeled, is available. Such a learning problem can be called *whole-to-part generalization*, since it is complementary to Dietterich and Michalski's (1986) definition of *part-to-whole generalization*, in which the learning task is that of hypothesizing a description of the whole object, given selected parts of it. Both in the case of learning multiple predicates in first-order logic and in the case of labeling problems, it is possible to use traditional learning algorithms, in order to learn contextual rules when the concept dependencies are intrinsically acyclic and the dependency hierarchy is known a priori. This can be done by learning a concept at a time according to the order defined by the dependency hierarchy and then by changing the observation language L_O after all concepts at the same hierarchical level have been learned.

With reference to Figure 12, C_{k+1} will be learned only after C_1 and C_2 are learned. Of course, we expect that the rule generated for C_{k+1} depends on the predicates denoting C_1 and C_2. Such an expectation is explained by the fact that concepts C_1 and C_2 are useful to explain or predict C_{k+1}. However, it may happen that the rule generated for C_{k+1} does not depend on either C_1 or C_2. This means that either the evidence in the observations is not enough to detect those concept dependencies, or the bias of the search strategy does not allow a system to discover them, or the dependency hierarchy does not express real concept dependencies.

The influence of the bias on the learnability of contextual rules induced us to select carefully the traditional learning system. In the next section, we present INDUBI/H, which implements a beam-search strategy for building clauses. Its search bias is weaker than that of FOIL and FOCL (Pazzani and Kibler, 1992). This is the reason for which we chose it for performing experiments on the problem of document understanding (Esposito et al., 1993e).

INDUBI/H

At the high level, there is no difference between INDUBI and INDUBI/H; thus Algorithm 1 is still valid. At the low level, INDUBI/H proceeds top-down, specializing the unit clause

$$[f(t_1,t_2, ..., t_n) = \lambda] \leftarrow$$

by adding one of the selectors in the positive example e^+ and turning constants into variables. Only a subset of n selectors among all selectors in the example e^+ is considered. Such selectors are chosen according to the cost associated with each function symbol in the referee of the selector and according to the arity of the function symbol, so that the greater the arity, the better. The former criterion offers

the user a way to express a sort of preference for some literals, while the latter criterion guarantees that relations are not treated unfairly. Obviously, selectors that make the partial clause unlinked are not considered at all. All specialized VL_{21} clauses, which cover e^+ and possibly other positive examples, are reordered according to another LEF, which takes into account the number of positive/negative examples covered, the complexity, as well as the total cost. The first p generalizations are selected and stored in a set *PS*. The consistent ones are removed from *PS*, extended-against, and stored into *MQ* (see Algorithm 3).

The extension-against generalization rule was introduced for the first time in AQ11 (Michalski & Larson, 1983) and has been formally presented by Michalski (1983). Here, we provide its extension to VL_{21} linked clauses. Briefly, given a VL_{21} generalization

$$[f(t_1,t_2, ..., t_p) = \lambda_i] \leftarrow \phi\ (t_1, t_2, ..., t_q)$$

its body can be factorized as a logical product of a conjunction $\phi_U(t_{11}, t_{12},...,t_{1r})$ of selectors involving unary functions and a conjunction $\phi_R(t_{21}, t_{22}, ..., t_{2s})$ of selectors involving n-ary functions with $n > 1$. The aim of the extension-against rule is that of generalizing $\phi_U(t_{11}, t_{12}, ..., t_{1r})$, while preserving the original consistency property of the generalization. This is done by possibly extending the references of the selectors in $\phi_U(t_{11}, t_{12}, ..., t_{1r})$.

Algorithm 3: Low-level beam search strategy for learning a single VL_{21} clause

```
procedure Beam_Search_for_consistent_hypotheses(e+, E+, E-, m)
PS := {[f(t1,t2, ..., tp)=λi] ← }
OldPS := Ø
MQ := Ø
while PS ≠ OldPS and |MQ| ≤ m do
      OldPS := PS
      PS := Ø
      foreach VL21 generalization G in OldPS do
            S := choose_best_linked_selectors(e+, G, n)
            PS := PS ∪ specialize_G_by_adding_a_selector_in_S(G,S)
      endfor
      CONS := consistent(PS)
      PS := PS - CONS
      PS := select_best_p_generalizations(PS)
      foreach VL21 generalization G in CONS do
            MQ := MQ ∪ extension_against(G, E+, E-)
      endfor
endwhile
return MQ
endproc
```

Let $\phi^*_U(t_{11}, t_{12}, ..., t_{1r})$ be the conjunctive formula $\phi_U(t_{11}, t_{12}, ..., t_{1r})$, whose references have been extended to the whole domain. We define two sets of conjunctive ground formulae, C^+ and C^-, as follows: if σ_i (τ_i) is a substitution, according to which

$$\phi_R(t_{21}, t_{22}, ..., t_{2s}) \wedge \phi^*_U(t_{11}, t_{12}, ..., t_{1r})$$

matches against a positive (negative) example e^+ (e^-), then C^+ (C^-) is a set of conjunctions of selectors in e^+ (e^-) that unifies with $\phi^*_U(t_{11}, t_{12}, ..., t_{1r})\sigma_i$ $[\phi^*_U(t_{11}, t_{12}, ..., t_{1r})\tau_i]$. The extension-against generalizing operator produces a conjunction of selectors that matches all formulae in C^+ and no formulae in C.

For instance, let us consider the following generalization:

$$[f(X) = \lambda_1] \leftarrow [on(X,Y)], [color(X) = black], [shape(Y) = triangle]$$

then

$$\phi_R(X,Y) \equiv [on(X,Y)]$$

$$\phi_U(X,Y) \equiv [color(X) = black]\ [shape(Y) = triangle]$$

and

$$\phi^*_U(X,Y) \equiv [color(X) = \Delta_{color}][shape(Y) = \Delta_{shape}]$$

where $\Delta_{color} = \{black, white\}$ and $\Delta_{shape} = \{triangle, square, circle\}$ are the domains of the functions color and shape respectively. If

e_1^+: $[f(t) = \lambda_1] \leftarrow [on(t,u)], [color(t) = black], [color(u) = black],$
$[shape(t) = triangle], [shape(u) = square]$
e_2^+: $[f(v) = \lambda_1] \leftarrow [on(v,w)], [color(v) = black], [color(w) = black],$
$[shape(v) = square], [shape(w) = triangle]$
e^-: $[f(x) = \lambda_2] \leftarrow [on(x,y)], [on(x,z)], [color(x) = black], [color(y) = white],$
$[color(z) = white], [shape(x) = circle], [shape(y) = circle], [shape(z) = circle]$

then

$$C^+ = \{[color(t) = black][shape(u) = square];$$
$$[color(v) = black][shape(w) = triangle]\}$$

$$C^- = \{[color(x) = black][shape(y) = circle]; [color(x) = black][shape(z) = circle]\}$$

An extension-against generalization is

$$[color(X) = black, white][shape(Y) = triangle,square]$$

Thus, the final rule is

$$[f(X) = \lambda_1] \leftarrow [on(X,Y)], [shape(Y) = triangle,square]$$

which is complete and consistent. The selector $[color(X) = black,white]$ has been dropped, since it has no discriminatory power.

Experimental Results

Several experiments have been organized in order to verify if learning contextual rules leads to better results than learning under the independence assumption. What do we expect from this experimentation?

(1) A decrease in learning time. This is counterintuitive, since for some concepts the search space is wider and not smaller, due to the introduction of new predicates. Our guess is that when concepts are really dependent, this approach should lead to better and not worse results, that is, information on the context should help faster learning.

(2) An increase in accuracy. This is counterintuitive as well, since when a prediction of a rule depends on the prediction made by another rule, it may happen that an error in the first rule to fire is propagated to the second (dependent) rule. This problem does not occur when rules are learned independently of each other. Once again, our guess is that when concepts are really dependent, this error propagation effect should not occur frequently, so that the global accuracy should not decrease but possibly increase.

In our experiments, we considered a set of thirty single-page documents, namely, copies of letters sent by Olivetti. In each experiment, six different replications were made by randomly splitting the set of documents into two subsets, according to the following criterion:

- twenty documents for the training set
- ten documents for the test set

Seven concepts had to be learned, namely, *sender* of the letter, *receiver, logotype, reference number, date, body,* and *signature.* Obviously, not all blocks were instances of one of these concepts, that is, there were some unlabeled blocks that we were not interested in classifying. Moreover, in some cases there was more than one block with the same label in a document, since some logical components were fragmented

into several layout blocks, which the layout analysis had not been able to group together.

A preliminary experiment was intended to prove that the presence of a domain theory could improve the accuracy of the recognition rules (Esposito et al., 1993d). However, a later experiment with FOCL confirmed that the use of background knowledge alone did not significantly improve the classifier performance when concept independence was assumed [for a detailed description, see Esposito et al., (1993b)]. Here, we present only the summary of the experiments made with INDUBI/H (1) under the independence assumption, (2) considering a user-defined dependency graph, and (3) using a dependency order defined by a statistical technique.

Results in Table 7 were obtained under the independence assumption. Entries *m/n* in columns 1–6 report the number of commission (*m*) and omission (*n*) errors of each rule for each replication. The last column of the table reports the average of the error rates of each rule, computed as the sum of commission and omission errors, divided by the number of logical components in the test set, for a given class. The average error of each rule is calculated as the average of the total errors made by the rule in each experiment, divided by the number of logical components (including unlabeled blocks). Finally, entries in the lowest row report the total number of commission/omission errors made in each experiment by the whole set of rules, and the total average error.

As we can see, there are two logical objects that can be easily recognized in a letter, namely, *logotype* and *signature*, since the former corresponds to the only layout object of type *picture*, while the latter corresponds to a layout block, which is generally classified as *graphic* by the document analysis. On the contrary, other logical objects, such as *sender, receiver, date* and *body*, cannot be easily recognized without looking at the relationships with other logical objects. An example of a rule generated in the sixth replication for labeling the *date* is shown in Figure 13. It is quite plain that in the recognition rule for *date*, relationships with neighboring layout

Table 7. Experimental results with INDUBI/H: independence assumption

Rule/replication	1	2	3	4	5	6	Average error, %
Logotype	0/0	0/0	0/0	0/0	0/0	0/0	0.0
Sender	0/0	0/1	0/1	0/1	0/9	0/2	16.9
Ref	1/0	1/3	1/0	1/1	1/4	0/2	14.3
Date	0/4	1/4	0/2	0/4	0/2	0/4	28.8
Receiver	2/1	0/0	0/0	0/3	0/5	0/2	17.9
Signature	0/1	0/2	0/2	0/0	0/0	0/0	8.9
Body	1/0	2/0	2/1	0/2	0/1	1/2	20
Total	4/6	4/10	3/6	1/11	1/21	1/12	11

[logic_type(S1)=date] ← [width(S1)=medium_small, medium] [height(S1)=very_very_small] [aligned(S2,S1)=both_rows]

[logic_type(S1)=date] ← [position(S1)=north_east] [to_right(S3,S4)] [aligned(S2,S1)=both_rows] [aligned(S3,S4)=ending_row] [aligned(S4,S2)=both_rows]

[logic_type(S1)=date] ← [width(S1)=medium_small, medium] [on_top(S1,S3)] [on_top(S2,S1)] [to_right(S4,S2)] [aligned(S2,S1)=starting_col] [aligned(S2,S5)=ending_col]

[logic_type(S1)=date] ← [width(S1)=medium_small] [on_top(S3,S4)] [on_top(S2,S1)] [to_right(S3,S2)] [aligned(S3,S2)=ending_row] [aligned(S2,S1)=ending_row] [aligned(S5,S3)=starting_col]

[logic_type(S1)=date] ← [position(S1)=north_east] [on_top(S5,S2)] [on_top(S1,S2)] [on_top(S3,S4)] [to_right(S4,S2)] [to_right(S3,S6)]

Figure 13. Noncontextual rule for *date* generated in the sixth replication.

objects are very important. However, neighboring objects have a logical class associated with them; thus, we expect that contextual rules should perform better.

Table 8 shows results produced by INDUBI/H when the following linear dependency order

$$logo \rightarrow signature \rightarrow body \rightarrow sender \rightarrow receiver \rightarrow ref \rightarrow date$$

is provided. Such an order is user defined and can be partly explained in terms of spatial reasoning. In fact, when the *logotype* and the *signature* have been recognized, the recognition of the contiguous blocks, namely, *sender* for the *logotype* and *body* for the *signature*, should be easier. Following the same line of reasoning, we expect that the identification of the *sender* should help to identify the *receiver*, which together with the identification of *logo*, should help to identify the *ref* and, finally, the *date*. By comparing data in Tables 7 and 8, it transpires that there is a uniform decrease in the error rate for all the logical classes when the independence assump-

Table 8. Experimental results with INDUBI/H: contextual rules with predefined order

Rule/replication	1	2	3	4	5	6	Average error, %
Logotype	0/0	0/0	0/0	0/0	0/0	0/0	0.0
Sender	0/0	0/1	0/1	0/1	0/9	0/1	15.6
Ref	2/0	0/1	0/1	0/1	1/1	0/2	10.6
Date	0/3	0/0	1/0	0/5	0/6	1/2	23.5
Receiver	0/1	0/0	0/0	0/2	0/5	0/2	13.5
Signature	1/0	0/2	0/2	0/0	0/0	0/0	8.9
Body	0/1	2/0	2/1	0/2	0/1	1/1	18.3
Total	3/5	2/4	3/5	0/11	1/22	2/8	9.0

tion is dropped. Globally, the total average error is 2.0% lower for contextual than for noncontextual rules. Moreover, we did not observe any increase in the learning time for this latter experiment. An example of a contextual rule produced for the logical class *date* is shown in Figure 14.

Since the dependency order is linear, we have also tried to learn the order by means of statistics. The idea is the following: the classification accuracy of a classifier that takes into account only the characteristics of each block can provide information on how easily a logical component can be recognized without looking at other logical components in the context. We are conscious of the fact that the dependency order defined in this way may be just a rule of thumb, but as we will show later, it can help to identify at least the minimally dependent concepts. The classifier that we adopted for the third experiment is Fisher's (1936) linear discriminant analysis. Obviously, discriminant analysis cannot be applied to symbolic features like those induced by INDUBI/H; therefore, we used a different set of statistical features, describing each single block and resulting from the layout analysis. Some of them are height, width, coordinates of the centroid of a block, eccentricity, number of black pixels per block, and so on. For each logical class C_i, we computed the following coefficient:

$$\kappa_i = (\text{no. positive examples covered} + \text{no. negative examples not covered})/2$$

and we ordered the logical classes according to κ_i. When two classes had the same coefficient, the one with the greatest number of examples was preferred. In the following, we list the different orderings obtained in the six replications:

1. *logo* → *ref* → *date* → *body* → *signature* → *sender* → *receiver*
2. *logo* → *ref* → *signature* → *date* → *sender* → *body* → *receiver*
3. *logo* → *signature* → *ref* → *body* → *date* → *sender* → *receiver*

[logic_type(S1)=date] ←	[logic_type(S2)=ref] [width(S1)=small ... medium] [height(S1)=smallest,very_very_small] [aligned(S2,S1)=both_rows]
[logic_type(S1)=date] ←	[logic_type(S2)=receiver] [logic_type(S4)=receiver] [on_top(S2,S1)] [on_top(S2,S3)] [to_right(S4,S2)] [aligned(S2,S1)=starting_col]
[logic_type(S1)=date] ←	[logic_type(S2)=ref] [logic_type(S3)=logo] [width(S1)=medium_small] [to_right(S3,S2)] [aligned(S2,S1)=ending_row]
[logic_type(S1)=date] ←	[position(S1)=north_east] [on_top(S3,S2)] [on_top(S1,S2)][to_right(S4,S2)]

Figure 14. Contextual rule for *date* generated in the sixth replication.

4. *logo* → *ref* → *date* → *signature* → *sender* → *body* → *receiver*

5. *logo* → *sender* → *date* → *signature* → *ref* → *body* → *receiver*

6. *logo* → *sender* → *ref* → *date* → *signature* → *body* → *receiver*

Table 9 shows the experimental results obtained by exploiting the ordering defined by the discriminant analysis. The idea of using statistical methods in order to define the linear dependency order seems to work well, since we obtained even a slightly lower average error rate than that obtained with the predefined order. Only in two experiments did the results deteriorate, but globally, the number of commission errors decreased. These encouraging results spur us to investigate further the possibility of adopting a multistrategy learning approach to document understanding, in order to check if the integrated method applied to contextual rules provides better results.

FURTHER DEVELOPMENTS

It is possible to improve the performance of the processes of document classification and understanding by defining a better representation of the page layout. In fact, coarse descriptions may lead to inconsistencies between instances of different logical components, since there may be no way to discriminate between them, while descriptions too detailed may cause a high increase in time for the learning process. Furthermore, not all learning systems can manage quantitative information; thus, a qualitative spatial model seems more adequate (Mukerjee and Joe, 1990).

A way to represent a page layout could be based on interrelationships between projection intervals of each block along one axis. For instance, the interrelationships between the blocks in Figure 15 can be described as follows:

Table 9. Experimental results with INDUBI/H: contextual rules with order defined by discriminant analysis

Rule/replication	1	2	3	4	5	6	Average error, %
Logotype	0/0	0/0	0/0	0/0	0/0	0/0	0.0
Sender	0/0	0/1	0/1	0/1	0/9	0/2	16.9
Ref	1/0	0/1	0/1	0/2	1/0	0/2	9.2
Date	0/2	0/0	0/1	0/5	0/2	0/4	16.4
Receiver	0/1	0/0	0/0	0/3	0/4	0/2	13.5
Signature	1/0	0/2	0/2	0/0	0/0	0/0	8.9
Body	0/1	2/0	2/1	1/2	0/1	0/2	20
Total	2/4	2/4	2/6	1/13	1/16	0/12	8.7

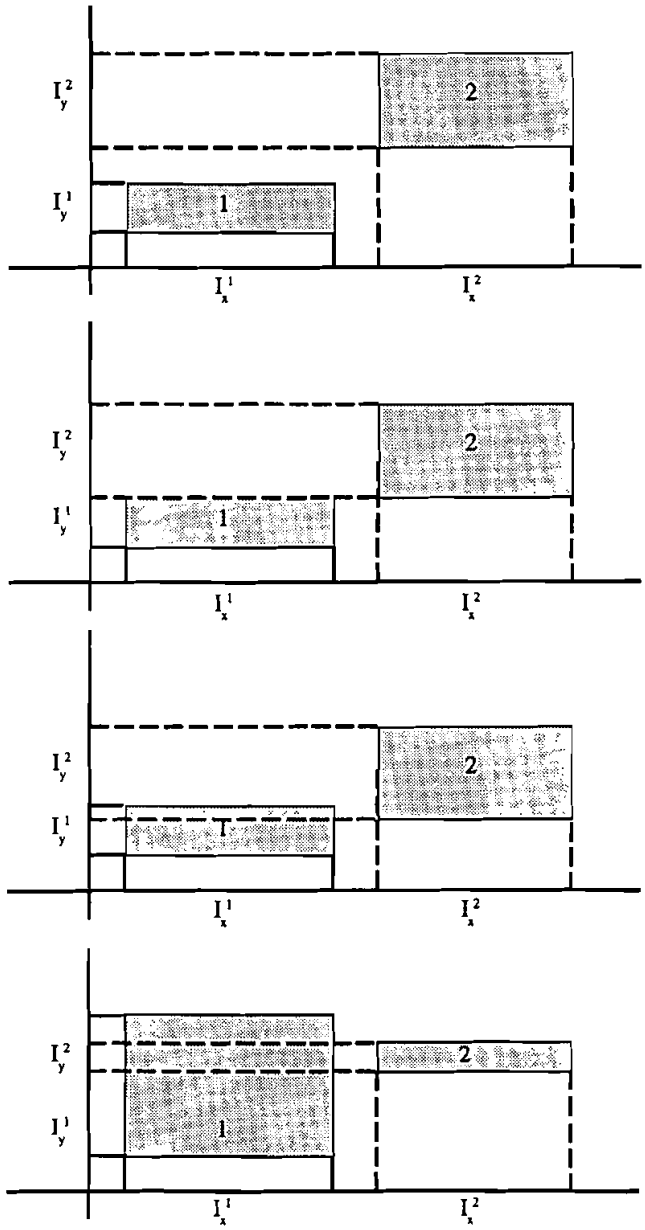


Figure 15. Interrelationships between blocks.

(1) $I_y^1 < I_y^2 \wedge I_x^2 < I_x^1$
(2) $I_y^1 \leq I_y^2 \wedge I_x^2 < I_x^1$
(3) $I_y^1 \cap I_y^2 \wedge I_x^2 < I_x^1$
(4) $I_y^1 \subseteq I_y^2 \wedge I_x^2 < I_x^1$

Currently, only the spatial relation (3) is described by DDOCUM, provided that the two blocks are not too far apart. Therefore, it is possible to improve the qualitative description of a page layout by introducing two descriptors (or equivalently, two predicates):

- *proj-y*, for the relationships between interval projections on axis y
- *proj-x*, for the relationships between interval projections on axis x

It is worthwhile to note that it is not necessary to describe relationships between each couple of interval projections, but it is enough, for our purposes, to describe relationships between subsequent or overlapping projection intervals along each direction.

Another possible improvement concerns the shift of language during the document understanding phase. In particular, we are planning to apply a more sophisticated learning algorithm, in order to discover complex dependency graphs without any prior knowledge. This involves the parallel learning of multiple predicates and the capability of evaluating the appropriateness of the shift of language, when a clause has already been learned.

As to the nominal descriptors *position* or *contain_in_pos*, it would be better to extend their domain by introducing new values, such as

top_horiz_band, central_horiz_band, bottom_horiz_band, left_vert_band, central_vert_band, right_vert_band

in order to express the fact that two layout blocks are in the same band (a band is a set of three contiguous areas in a page). However, we should be able to express also the partially ordered relation that can be defined on values of such an extended domain. This requires the introduction of a new kind of domain, called *dag-structured domain*, in which values can be represented as nodes of a directed acyclic graph.

Further limitations in PLRS concern the document analysis process. In particular, we observed that RLSA, the segmentation algorithm used by PLRS, is not adequate for some forms, in which a portion of content is contained within a frame (invoice or order forms are classic examples). Indeed, in this case, the algorithm is not able to separate text regions from the graphics represented by horizontal and vertical lines of the frame. Detecting such lines and removing them before segmenting the document may be a solution to this kind of problem.

As to the segmentation process, it is possible to improve the performance of the parametric classifier, which assigns types to layout blocks. Indeed, in some cases, we observed bad classifications of some blocks, due to an inappropriate choice of parameters. However, it would be difficult to make these changes by hand; thus, we aim at exploiting supervised learning techniques, such as decision trees, in order to obtain a more reliable set of classification rules.

Similar problems can be observed in the layout analysis, when some layout objects are grouped wrongly or are not grouped at all. In this case, it is difficult to change parameters of the grouping criteria embedded in the code without affecting the global results. We are planning to build an intelligent system devoted to layout analysis, which exploits knowledge on human criteria used while grouping layout objects. In this way, it should be easier to understand why the layout analysis produces unexpected results and, consequently, to revise the knowledge base. Once again, the knowledge base of the intelligent system can be automatically acquired and refined by machine learning techniques.

CONCLUSIONS

PLRS is an innovative system from several points of view. First, it applies machine learning techniques in order to generate recognition rules, which constitute the knowledge base of an expert system, devoted to document classification and understanding. This allows the document processing system to be easily customized, according to the needs of the end-user, as well as to deal with the variety of formats exhibited by office documents. Moreover, the process of recognition is layout based, that is, PLRS recognizes the class of a document and its logical structure by simply using automatically detected geometrical characteristics. This means that it is not necessary to read the whole document by OCR in order to process it. On the contrary, once the logical objects have been identified, an OCR is used to read those parts relevant for retrieval purposes. Other novelties concern the learning strategies adopted by the inductive learning system of PLRS. In particular, in the phase of document classification, we used a multistrategy learning system, called RES, that integrates a parametric method with a conceptual one, in order to deal with noisy symbolic/numeric data. As to the phase of document understanding, we used a system that learns VL_{21} clauses, called INDUBI/H; by shifting the document description language, it is possible to obtain contextual rules, which are more accurate than those obtained under the independence assumption. Some experimental results have shown that it is convenient to adopt the same multistrategy learning approach even for the problem of document understanding, at least for deriving a linear dependency order between logical classes.

RES and INDUBI/H have been implemented in C and embedded in PLRS, which is a module of IBIsys, a software environment for office automation distributed by Olivetti.

REFERENCES

Ciardiello, G., G. Scafuro, M. T. Degrandi, M. R. Spada, and M. P. Roccotelli. 1988. An experimental system for office document handling and text recognition. In *Proceedings of ninth international conference on pattern recognition*, 739–743. Los Alamitos: IEEE Computer Society.

Cooley, W. W., and P. R. Lohnes. 1971. *Multivariate data analysis*. New York: Wiley.

Dengel, A., and G. Barth. 1988. High level document analysis guided by geometric aspects. *Int. J. Pattern Recognition Artificial Intel.* 2(4).

De Raedt, L., N. Lavrac, and S. Dzeroski. 1993. Multiple predicate learning. In *Proceedings of the third international workshop on inductive logic programming*, 221–240.

Dietterich, T. G., and R. S. Michalski. 1986. Learning to predict sequences. In Vol. 2, *Machine learning: an artificial intelligence approach*, eds. R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, 63–106. Los Altos, Calif.: Morgan Kaufmann.

Duda, R. O., and P. E. Hart. 1973. *Pattern classification and scene analysis*. New York: Wiley.

Eirund, H., and K. Kreplin. 1988. Knowledge based document classification supporting integrated document handling. In *Proceedings of the ACM conference on office information systems*, 189–196. New York: Association for Computing Machinery.

Esposito, F. 1990. Automated acquisition of production rules by empirical supervised learning methods. In *Knowledge, data and computer-assisted decisions*, eds. M. Schader and W. Gaul, 35–48. Berlin: Springer-Verlag.

Esposito, F., D. Malerba, G. Semeraro, E. Annese, and G. Scafuro. 1990a. Empirical learning methods for digitized document recognition: an integrated approach to inductive generalization. In *Proceedings of the sixth conference on artificial intelligence applications*, 37–45. Los Alamitos: IEEE Computer Society.

Esposito, F., D. Malerba, G. Semeraro, E. Annese, and G. Scafuro. 1990b. An experimental page layout recognition system for office document automatic classification: an integrated approach for inductive generalization. In *Proceedings of the tenth IEEE international conference on pattern recognition*, 557–562. Los Alamitos: IEEE Computer Society.

Esposito, F., D. Malerba, and G. Semeraro. 1991a. A distance measure for decision making in uncertain domain. In *Uncertainty in knowledge bases*. Lecture Notes in Computer Science no. 521, eds. B. Bouchon-Meunier, R. R. Yager, and L. A. Zadeh, 538–547, Berlin: Springer-Verlag.

Esposito, F., D. Malerba, and G. Semeraro. 1991b. Flexible matching for noisy structural descriptions. *Proceedings of the twelfth international joint conference on artificial intelligence*, 658–664. San Mateo: Morgan Kaufmann.

Esposito, F., D. Malerba, and G. Semeraro. 1991c. Classification of incomplete structural descriptions using a probabilistic distance measure. In *Symbolic-numeric data analysis and learning*, eds. E. Diday and Y. Lechevallier, 469–482. New York: Nova Science Publishers.

Esposito, F., D. Malerba, and G. Semeraro. 1992a. Classification in noisy environments using a distance measure between structural symbolic descriptions. *IEEE Trans. Pattern Anal. Mach. Intel.* PAMI-14(3):390–402.

Esposito, F., G. Semeraro, and D. Malerba. 1992b. A syntactic distance for partially matching learned concepts against noisy structural object descriptions. *Int. J. Expert Syst.* 4(4):409–451.

Esposito, F., D. Malerba, and G. Semeraro. 1993a. Incorporating statistical techniques into empirical symbolic learning systems. In *Artificial intelligence frontiers in statistics*, ed. D. J. Hand, 168–181. London: Chapman and Hall.

Esposito, F., D. Malerba, G. Semeraro, and M. Pazzani. 1993b. A machine learning approach to document understanding. In *Proceedings of the second international workshop on multistrategy learning*, 276–292.

Esposito, F., D. Malerba, and G. Semeraro. 1993c. A comparison of statistical methods for inferring causation. In *Proceedings of the fourth international workshop on artificial intelligence and statistics*, 127–140.

Esposito, F., D. Malerba, and G. Semeraro. 1993d. Machine learning techniques for knowledge acquisition and refinement. In *Proceedings of the fifth international conference on software engineering and knowledge engineering*, 319–323.

Esposito, F., D. Malerba, and G. Semeraro. 1993e. Learning contextual rules in first order logic. Paper presented at *the fourth Italian Workshop on Machine Learning GAA93*, June 11–12, Milan.

Esposito, F., D. Malerba, and G. Semeraro. 1993f. Negation as a specializing operator. In *Advances in artificial intelligence*, Lecture notes in artificial intelligence 728, ed. P. Torasso, 166–177. Berlin: Springer-Verlag.

INTREPID Project. 1993. *Final report. Part 1: technical achievements*. Esprit Project 5203.

Fisher, J. L., S. C. Hinds, and D. P. D'Amato. 1990. A rule-based system for document image segmentation. In *Proceedings of the tenth IEEE international conference on pattern recognition*, 557–562. Los Alamitos: IEEE Computer Society.

Fisher, R. A. 1936. The use of multiple measurements in taxonomic problems. *Ann. Eugenics* 7:179–188.

Fletcher, L. A., and R. Kasturi. 1988. A robust algorithm for text string separation from mixed text/graphics images. *IEEE Trans. Pattern Anal. Mach. Intel.* PAMI-10(6):910–918.

Genesereth, M. R., and N. J. Nilsson. 1987. *Logical foundations of artificial intelligence*. Palo Alto, Calif.: Morgan Kaufmann.

Hand, D. J. 1981. *Discrimination and classification*. London: Wiley.

Haralick, R. M., and L. G. Shapiro. 1979. The consistent labelling problem: part I. *IEEE Trans. Pattern Anal. Mach. Intel.* PAMI-1(2):173–184.

Horak, W. 1985. Office document architecture and office document interchange formats: current status of international standardization. *IEEE Comput.* (October):50–60.

Kubota, K., O. Iwaki, and H. Arakawa. 1984. Document understanding system. In *Proceedings of the seventh international conference on pattern recognition*, 612–614. Los Alamitos: IEEE Computer Society.

Lachenbruch, P. A. 1975. *Discriminant analysis*. New York: Hafner Press.

Larson, J. B. 1977. Inductive inference in the variable valued predicate logic system VL_{21}: methodology and computer implementation. Ph.D. thesis, Department of Computer Science, University of Illinois, Urbana.

Malerba, D. 1993. *Document understanding: a machine learning approach*. Technical report, Esprit Project 5203 INTREPID.

Michalski, R. S. 1980. Pattern recognition as rule-guided inductive inference. *IEEE Trans. Pattern Anal. and Mach. Intel.* PAMI-2(4):349–361.

Michalski, R. S. 1983. A theory and methodology of inductive learning. In *Machine learning: an artificial intelligence approach*, eds. R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, 83–134. Palo Alto, Calif.: Tioga Publishing.

Michalski, R. S., and J. B. Larson. 1983. Incremental generation of VL_1 hypotheses: the underlying methodology and the description of program AQ11. Revised by K. Chen. Technical report UIUCDCS-F-83-905, Department of Computer Science, University of Illinois at Urbana-Champaign.

Muggleton, S. 1992. Inductive logic programming. In *Inductive logic programming*, ed. S. Muggleton, 3–27. London: Academic.

Muggleton, S., and C. Feng. 1990. Efficient induction of logic programs. In *Inductive logic programming*, ed. S. Muggleton, 281–298. San Diego, Calif.: Academic.

Mukerjee, A., and G. Joe. 1990. A qualitative model for space. In *Proceedings of AAAI 90*, 721–727. Menlo Park, Calif.: AAAI Press.

Nagy, G., J. Kanai, and M. Krishnamoorthy. 1988. Two complementary techniques for digitized document analysis. Paper presented at the *ACM Conference on Document Processing Systems*, Santa Fe, N. Mex., Dec. 1988.

Nagy, G., S. C. Seth, and S. D. Stoddard. 1986. Document analysis with an expert system. In *Pattern recognition in practice II*, eds. E. S. Gelsema and L. N. Kanal. North Holland: Elsevier Science Publishers.

O'Gorman, L., and R. Kasturi. 1992. Document image analysis systems. *IEEE Comput.* (July):5–8.

Pagurek, B., N. Dawes, G. Bourassa, G. Evans, and P. Smithers. 1990. Letter pattern recognition. In *Proceedings of the sixth IEEE conference on artificial intelligence applications*, 312–319. Los Alamitos: IEEE Computer Society.

Pazzani, M., and D. Kibler. 1992. The utility of knowledge in inductive learning. *Mach. Learning* 9(1):57–94.

Plotkin, G. D. 1971. Automatic methods of inductive inference. PhD thesis, Edinburgh University.

Quinlan, J. R. 1990. Learning logical definitions from relations. *Mach. Learning* 5(3):239–266.

Semeraro, G. 1987. Un sistema per l'apprendimento induttivo concettuale da esempi con logica variable-valued. Laurea dissertation, Università di Bari. (Also available as LACAM Technical Report, Dipartimento di Informatica, Università di Bari.)

Semeraro, G., C. A. Brunk, and M. J. Pazzani. 1993. *Traps and pitfalls when learning logical theories: a case study with FOIL and FOCL*. Technical Report 93-33, Department of Information and Computer Science, University of California, Irvine.

Srihari, S. N. 1987. Document image analysis: an overview of algorithms. In *Advanced printing of the symposium summaries, SPSE's 40th annual conference and symposium on hybrid imaging systems*, 28–31.

Tang, Y. Y., C. Y. Suen, C. D. Yan, and M. Cheriet. 1991. Document analysis and understanding: a brief survey. In *Proceedings of the first international conference on document analysis and recognition*, 17–31.

Trunk, G. V. 1979. A problem of dimensionality: a simple example. *IEEE Trans. Pattern Anal. Mach. Intel.* PAMI-1:306–307.

Tsujimoto, S., and H. Asada. 1990. Understanding multi-article documents. In *Proceedings of the tenth international conference on pattern recognition*, 551–556. Los Alamitos: IEEE Computer Society.

Wong, K. Y., R. G. Casey, and F. M. Wahl. 1982. Document analysis system. *IBM J. Res. Develop.* 26(6):647–656.